BỘ GIAO THÔNG VẬN TẢI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ GIAO THÔNG VẬN TẢI

LÊ CHÍ LUẬN (Chủ biên)
LÊ TRUNG KIÊN - LÊ THỊ CHI
PHẠM THỊ THUẬN - NGUYỄN THỊ KIM HUỆ

GIÁO TRÌNH

NGÔN NGỮ LẬP TRÌNH C

NHÀ XUẤT BẢN KHOA HỌC TỰ NHIÊN VÀ CÔNG NGHỆ
HÀ NỘI - 2020

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC HÌNH

LỜI NÓI ĐẦU

Năm 2016, C là ngôn ngữ lập trình hoạt động mạnh thứ 8 trên GitHub xếp sau các ngôn ngữ như Javascript, Java, Python, Ruby, PHP, C#, C++. Tuy nhiên, GitHub chưa phải là hình ảnh chính xác nhất để đánh giá bởi GitHub luôn thiên vị với những ngôn ngữ mã nguồn mở và hợp thời. Trên thực tế, tạp chí IEEE Spectrum xếp ngôn ngữ C như là ngôn ngữ hàng đầu trong năm 2017 trước cả Java, C # và Javascript. Có một số lý do để các bạn sinh viên nên bắt đầu với ngôn ngữ lập trình C.

Thứ nhất, C là ngôn ngữ cấp thấp so với ngôn ngữ trìu tượng khác. Bạn có thể viết mã gần gũi với phần cứng và trực tiếp điều khiển bộ nhớ. Mặc dù đây là một trong những phần khiến người học khó học C nhưng đó cũng chính là lý do tại sao các lập trình viên ngôn ngữ này có xu hướng phù hợp hơn với cách hoạt động của máy tính. Để viết mã C tốt, bạn cần suy nghĩ như máy tính.

Thứ hai, nhiều lập trình viên nói rằng: Khi đã biết một ngôn ngữ lập trình, bạn sẽ biết tất cả những ngôn ngữ khác nữa. Câu nói này có vẻ đúng, đặc biệt là đối với ngôn ngữ C. Ví dụ, việc học từ ngôn ngữ cấp thấp như C sang một ngôn ngữ cấp cao hơn như Python, Ruby khá dễ dàng nhưng để học ngược lại thì không hề đơn giản. Bằng cách học C, bạn chủ yếu sẽ học các nền tảng của ngôn ngữ lập trình hiện đại. Nếu bạn thực sự hiểu C, bạn sẽ có thể học bất cứ ngôn ngữ nào khác một cách dễ dàng bởi hầu hết các ngôn ngữ hiện đại thường cao cấp hơn C.

Lý do thứ ba, hầu hết các ngôn ngữ hiện đại đều được sinh ra để đáp ứng những thiếu sót của ngôn ngữ khác: ngôn ngữ C++ ra đời đáp ứng những thiếu sót của ngôn ngữ C, tiếp theo sự ra đời của Java đáp ứng những điểm chưa hoàn thiện của C++ và C# ra đời để cải thiện Java v.v. Vì thế bằng cách học C, bạn có thể hiểu rõ lý do tại sao một số ngôn ngữ được thiết kế theo cách đó và có khả năng đánh giá tốt hơn về sự tiện lợi mà các ngôn ngữ cao hơn cung cấp.

Lý do thứ tư, hầu hết các ngôn ngữ lập trình hiện đại đều được sử dụng cho ba mục đích, đó là các ứng dụng kinh doanh, các ứng dụng web và di động, cuối cùng là phân tích dữ liệu. Tuy nhiên, nếu bạn muốn phát triển phần mềm được kết nối trực tiếp với phần cứng, bạn sẽ cần một ngôn ngữ thấp hơn - và C là ngôn ngữ được sử dụng nhiều nhất. Các ứng dụng đáng chú ý bao gồm hệ điều hành, ngôn ngữ lập trình, trình biên

dịch, các hệ thống nhúng, v.v. Chẳng hạn, Linux kernel được viết bằng ngôn ngữ C và Assembly. Các ngôn ngữ phổ biến như Python, PHP, Perl và Ruby đều được thực hiện trong C. Thậm chí, C cũng được viết bằng chính nó. Bởi nhiều hệ thống nhúng có giới hạn tài nguyên nghiêm ngặt nên C thường là ngôn ngữ được lựa chọn nhiều nhất vì tốn ít chi phí.

Trong chương trình môn học của ngành Hệ thống thông tin và mới nhất là ngành Công nghệ thông tin, C là môn học được đưa vào sớm nhất để giảng dạy nền tảng lập trình cho sinh viên của khoa Công nghệ thông tin, Trường Đại học Công nghệ Giao thông vận tải.

Vì những lý do đó, nhóm tác giả với sự nhiệt huyết của mình, với kinh nghiệm đã giảng dạy môn học nhiều năm cho các thế hệ sinh viên từ K64 đến K68 ngành Hệ thống thông tin đã biên soạn cuốn "Giáo trình Ngôn ngữ lập trình C". Giáo trình nhằm cung cấp những kiến thức nền tảng của một ngôn ngữ đã có từ lâu đời, là nền tảng, cội nguồn của các ngôn ngữ hiện đại ngày nay, để từ đây nhóm tác giả hy vọng các bạn sinh viên có thể lĩnh hội và làm chủ kiến thức về một ngôn ngữ lập trình nào đó mà mình theo đuổi, là hành trang trên giảng đường đại học và lập nghiệp sau này.

Giáo trình gồm 8 chương:

CHƯƠNG 1: CÁC KHÁI NIỆM CƠ BẢN CỦA NGÔN NGỮ C

Một số vấn đề trọng tâm được trình bày trong Chương 1 bao gồm: tập ký tự được sử dụng trong ngôn ngữ C, tập 32 từ khóa dành riêng trong C mà người lập trình có thể sử dụng trong chương trình tùy theo ý nghĩa của từng từ. Ngoài ra, Chương 1 còn giới thiệu cấu trúc, thành phần của một chương trình viết bằng C, trình tự các bước từ lúc soạn thảo đến lúc biên dịch và chạy chương trình, các kiểu dữ liệu được sử dụng cho các biến, các hằng, các toán tử, toán hạng được sử dụng trong các biểu thức, v.v. Cuối Chương 1 giới thiệu hai công cụ được sử dụng để lập trình C, bao gồm Turbo C, Dev C++.

CHƯƠNG 2: CÁC LỆNH NHẬP - XUẤT DỮ LIỆU TRONG C

Hai hoạt động chủ yếu của bất kỳ ngôn ngữ lập trình, đó là đưa dữ liệu vào chương trình để xử lý và đưa dữ liệu sau khi xử lý ra ngoài. Chương 2, đề cập đến các lệnh được sử dụng để đưa dữ liệu từ bàn phím hoặc tệp vào chương trình và các lệnh để đưa dữ liệu ra màn hình hoặc ghi vào tệp. Dữ liệu được đưa vào hoặc đưa ra có thể là một ký tự, một chuỗi ký tự, có thể là dữ liệu dạng số nguyên, số thực, v.v. Vì vậy một nội

dung khá quan trọng của Chương 2, giới thiệu các mã định dạng, để đưa dữ liệu vào hoặc đưa dữ liệu ra đúng quy chuẩn, định dạng.

CHƯƠNG 3: CÁC CẤU TRÚC PHÂN NHÁNH VÀ VÒNG LẶP

Hầu hết các ngôn ngữ lập trình đều cố gắng diễn đạt các hoạt động của con người thành các cấu trúc lệnh. Có ba cấu trúc lệnh chủ yếu, thứ nhất là cấu trúc tuần tự, các lệnh được thực hiện liên tiếp nhau, câu lệnh nào viết trước thì được xử lý trước, thứ hai là cấu trúc điều kiện, hay còn gọi là cấu trúc phân nhánh, thực hiện các công việc dựa vào một điều kiện đúng hay sai, hoặc dựa trên các giá trị có thể nhận được của một biểu thức mà thực hiện các nhánh tương ứng. Cấu trúc thứ ba, diễn đạt một công việc nào đó được lặp đi lặp lại, có thể vô hạn, dựa vào một biểu thức điều kiện.

Chương 3, giới thiệu các câu lệnh điều kiện if, cấu trúc lựa chọn switch, cấu trúc lặp với số lần xác định for và lặp với số lần không xác định với hai câu lệnh do… while và while. Với từng cấu trúc của các lệnh này, các bạn sinh viên nên tìm hiểu kỹ từng thành phần, cú pháp và chức năng hoạt động thông qua các ví dụ minh họa được trình bày khá chi tiết trong giáo trình.

CHƯƠNG 4: HÀM

Một trong những yếu điểm của các sinh viên khi học các ngôn ngữ lập trình, là xây dựng một chương trình gọn gàng với số dòng mã ít, tối ưu. Làm thế nào để phân chia chương trình hợp lý nhất, sử dụng code như thế nào cho hợp lý, tránh trường hợp một đoạn code xuất hiện nhiều lần trong một chương trình được trình bày trong nội dung Chương 4.

Nội dung Chương 4, trình bày khái niệm về hàm, cách khai báo hàm, cách truyền tham số và gọi hàm trong chương trình. Khác với các ngôn ngữ khác, chẳng hạn như C++, ngôn ngữ C chỉ có hai cách truyền tham số: đó là truyền tham trị và truyền con trỏ. Xác định đúng thì mới xây dựng được một chương trình tường minh, theo đúng tư tưởng của ngôn ngữ thuần C. Có rất nhiều ví dụ được đưa ra trong Chương 4, giúp cách bạn xác định được khi nào nên dùng hàm và dùng hàm như thế nào cho đúng cách.

CHƯƠNG 5: MẢNG VÀ CON TRỎ

Ngoài các kiểu dữ liệu cơ bản như ký tự, số nguyên, số thực, v.v. ngôn ngữ lập trình C cho phép lập trình viên tự định nghĩa các các kiểu dữ liệu phù hợp với nhu cầu xử lý thực tế trong các bài toán cụ thể.

Mảng, chuỗi ký tự là những kiểu dữ liệu như vậy. Chương 5, trình bày khái niệm về mảng, chuỗi ký tự và các bài toán xử lý trên mảng, chuỗi ký tự. Các bài toán này có ý nghĩa thực tế, chẳng hạn tìm kiếm một phần tử trong tập hợp các phần tử, sắp xếp một mảng, chỉ ra phần tử lớn nhất, phần tử nhỏ nhất, liệt kê các phần tử thỏa mãn một tính chất nào đó, v.v.

Ngoài ra, Chương 5 còn trình bày một vấn đề, có thể nói là khá khó với hầu hết những người mới làm quen với ngôn ngữ lập trình, đó là con trỏ. Thực chất con trỏ là một biến được sử dụng để lưu địa chỉ của một biến, nhưng thông qua biến con trỏ có thể xử lý các đối tượng mà con trỏ đó đang trỏ tới. Sử dụng con trỏ cho phép lập trình viên có thể cấp phát động bộ nhớ cho một mảng gồm nhiều phần tử. Các bạn sinh viên có thể hiểu rõ bản chất của con trỏ qua những ví dụ khá điển hình trong Chương 5.

CHƯƠNG 6: KIỂU CẤU TRÚC

Nội dung của Chương 6, trình bày về một kiểu dữ liệu do người dùng định nghĩa, kiểu bản ghi (kiểu cấu trúc), phản ánh cho sự đa dạng của các thành phần dữ liệu trong một phần tử. Hiểu nôm na, giống như việc chúng ta loay hoay tìm một kiểu dữ liệu phù hợp cho một đối tượng thực tế trong đời sống như sinh viên, công nhân, hay một cuốn sách, một bộ phim, v.v. Rõ ràng các đối tượng này có rất nhiều thuộc tính, chẳng hạn như cuốn sách có các thuộc tính như mã sách, tên sách, tác giả của cuốn sách, năm xuất bản, v.v. Vậy đối tượng này khi đưa vào để xử lý phải làm như thế nào? Kiểu cấu trúc cho phép lập trình viên định nghĩa ra một cấu trúc có tên là Sach bao gồm các thuộc tính vừa kể và mọi chuyện đã trở nên khá đơn giản, khi lập trình xử lý các biến thuộc kiểu Sach giống như các biến thông thường.

Trong Chương 6, đề cập đến định nghĩa kiểu cấu trúc, cách thức khai báo kiểu cấu trúc, khai báo các biến kiểu cấu trúc và khai báo một mảng mà mỗi phần tử có kiểu cấu trúc, mối liên hệ giữa kiểu cấu trúc và con trỏ. Có rất nhiều ví dụ minh họa, được trình bày khá chi tiết để các bạn sinh viên có thể tham khảo và để hiểu rõ hơn, tại sao ngôn ngữ C là ngôn ngữ lập trình cấu trúc, phân biệt với ngôn ngữ lập trình hướng đối tượng như C++ hoặc Java. Chẳng hạn với lập trình hướng đối tượng, khi khai báo một lớp (lớp là tập hợp của các đối tượng), thường phải chỉ rõ 2 thành phần: các thuộc tính và các phương thức (tức là các hàm), trong khi với ngôn ngữ C, khi khai báo kiểu cấu trúc (phản ánh cho một đối tượng) chỉ phải khai báo các thuộc tính của các đối tượng, các hàm tác động lên kiểu cấu trúc đó được xây dựng hoàn toàn độc lập, v.v.

CHƯƠNG 7: TỆP

Trong đề cương của môn học đối với sinh viên ngành Hệ thống thông tin và sinh viên ngành Công nghệ thông tin không có phần này, tuy nhiên, đối với một lập trình viên, không thể không biết cách làm việc và xử lý trên tệp, vì đó là xương sống của hầu hết các ứng dụng. Chính vì thế trong nội dung của cuốn giáo trình này, nhóm tác giả đã mạnh dạn đưa thêm Chương 7, là một nội dung để các bạn tham khảo và hơn cả là làm chủ kiến thức về tệp.

Nội dung của Chương 7, trình bày khái niệm về tệp, cách thức khai báo tệp, phân loại tệp, các thao tác xử lý với tệp bao gồm mở tệp, đọc và ghi tệp, đóng tệp. Nắm được nguyên tắc cơ bản này, trong hầu hết các ứng dụng, các bạn đều phải tuần tự thực hiện việc tạo tệp, dùng lệnh để mở tệp, sau đó là xử lý trên tệp, bao gồm hai hoạt động đọc nội dung từ một tệp hoặc ghi nội dung vào tệp. Sau khi kết thúc quá trình làm việc với tệp, cần đóng tệp để tránh mất dữ liệu. Có nhiều tham số, chẳng hạn mở tệp chỉ đọc, mở tệp cho phép đọc ghi, v.v các bạn nên tham khảo qua các ví dụ của Chương để tìm hiểu thêm.

CHƯƠNG 8: ĐỒ HỌA

Nội dung của Chương 8, giới thiệu những kiến thức cơ bản về đồ họa, cách thức vẽ các hình khối cơ bản, phối kết hợp giữa các hình khối để tạo ra những hình ảnh chuyển động. Chương 8, thực sự mới chỉ mang đến cho các bạn cái nhìn tổng quan, nhóm tác giả hy vọng trong những lần tái bản sau sẽ đem đến một nội dung đầy đủ và phong phú hơn.

Cuối mỗi chương đều xây dựng hệ thống câu hỏi và bài tập để sinh viên ôn tập, rèn luyện và củng cố lý thuyết môn học. “Giáo trình Ngôn ngữ lập trình C” được biên soạn công phu, nghiêm túc dựa trên kinh nghiệm giảng dạy môn học cho sinh viên từ khóa 64 đến khóa 68 tại Trường Đại học Công nghệ Giao thông vận tải của các giảng viên bộ môn Hệ thống thông tin. Khi biên soạn giáo trình này, nhóm tác giả có tham khảo một số giáo trình viết về ngôn ngữ C của một số trường đại học có uy tín ở Việt Nam, một số chuyên gia nổi tiếng và một số tài liệu uy tín của nước ngoài, được sự tham gia và đóng góp ý kiến của nhiều thầy cô đã và đang giảng dạy môn Ngôn ngữ lập trình C ở các trường đại học trên địa bàn Hà Nội. Nhóm tác giả chân thành cảm ơn NCS.ThS. Trần Nguyên Hương, giảng viên Trường Đại học Kinh doanh và Công nghệ Hà Nội, đã góp nhiều ý kiến quý báu, chia sẻ nhiều kinh nghiệm trong quá trình nhóm biên soạn cuốn giáo trình này.

Trong quá trình biên soạn, nhóm tác giả đã rất cố gắng để có được một tài liệu tham khảo có chất lượng, tuy nhiên không thể tránh khỏi những thiếu sót. Rất mong được sự đóng góp ý kiến để giáo trình được cập nhật và hoàn thiện hơn.

Hà Nội, tháng 7 năm 2020.

Nhóm tác giả

Chương 1
CÁC KHÁI NIỆM CƠ BẢN CỦA NGÔN NGỮ C

Nội dung của Chương 1 đề cập đến những khái niệm ban đầu của ngôn ngữ lập trình C, một ngôn ngữ đã ra đời từ khá lâu, là nền tảng cho rất nhiều ngôn ngữ lập trình hiện đại ngày nay. Có rất nhiều vấn đề mà một người mới bắt đầu làm quen với một ngôn ngữ lập trình gặp phải, chẳng hạn như ngôn ngữ đó sử dụng tập ký hiệu nào là hợp lệ, quy cách viết một chương trình theo đúng cấu trúc của ngôn ngữ, cần bao nhiêu đại lượng như biến và hằng trong một chương trình và kiểu dữ liệu của các biến, hằng đó như thế nào? Giữa các biến, hằng và các toán tử có một mối quan hệ ra sao trong một biểu thức, cần sử dụng công cụ (tool) nào để soạn thảo và biên dịch chương trình. Những vấn đề này sẽ được giới thiệu và giải đáp chi tiết trong nội dung của Chương 1, các khái niệm cơ bản của ngôn ngữ C.

1.1. GIỚI THIỆU NGÔN NGỮ C

1.1.1. Lịch sử ngôn ngữ lập trình C

Lập trình cấu trúc là phương pháp tổ chức, phân chia chương trình thành các hàm, thủ tục, chúng được dùng để xử lý dữ liệu nhưng lại tách rời các cấu trúc dữ liệu. Thông qua các ngôn ngữ Foxpro, Pascal, C, v.v. đa số những người làm Tin học đã khá quen biết với phương pháp lập trình này.

Năm 1972, tại phòng thí nghiệm Bell, lập trình viên Dennis Ritchie đã tạo ra một ngôn ngữ mới có tên C. (Được dùng để thay thế ngôn ngữ lập trình cũ mà anh ta đang sử dụng là ngôn ngữ B. C có nguồn gốc từ ngôn ngữ BCPL do Martin Richards phát triển, BCPL sau đó đã được Ken Thompson phát triển thành ngôn ngữ B. Trong khi BCPL và B không hỗ trợ kiểu dữ liệu, thì C đã có nhiều kiểu dữ liệu khác nhau. Những kiểu dữ liệu chính gồm: kiểu ký tự (character), kiểu số nguyên (interger) và kiểu số thực (float)).

Ngôn ngữ C được thiết kế với một mục tiêu: Sử dụng để viết hệ điều hành. (Ban đầu, ngôn ngữ C được phát triển để sử dụng trong hệ điều hành UNIX. Nó kế thừa nhiều tính năng của các ngôn ngữ trước đây như B và BCPL). Ngôn ngữ cực kỳ đơn giản và linh hoạt và sớm được sử dụng cho nhiều loại chương trình khác nhau. Nó nhanh chóng trở thành một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới.

Sự phổ biến của C là do hai yếu tố chính. Đầu tiên là ngôn ngữ không cản trở lập trình viên. Anh ta có thể làm bất cứ điều gì bằng cách sử dụng cấu trúc C thích hợp. Lý do thứ hai mà C phổ biến là trình biên dịch C có sẵn cho hầu hết các máy tính. Do đó, mọi người có thể đính kèm trình biên dịch C cho máy của họ một cách dễ dàng và ít chi phí. Các mã lệnh viết bằng C trên máy này có thể được biên dịch và chạy trên một máy khác chỉ cần thay đổi rất ít hoặc không phải thay đổi gì cả.

Năm 1980, Bjarne Stroustrup bắt đầu làm việc với một ngôn ngữ mới, được gọi là "C++". Ngôn ngữ này được cải thiện và tăng cường trên C bằng cách thêm một số tính năng mới.

Một trong những ngôn ngữ mới nhất, Java, dựa trên C++, Java là được thiết kế để trở thành "C++ với các lỗi đã được sửa".

1.1.2. Các tính chất của ngôn ngữ C

C là ngôn ngữ được sử dụng rộng rãi. C có những tính chất sau đây:

- Đơn giản

C là một ngôn ngữ đơn giản theo nghĩa nó cung cấp một cách tiếp cận có cấu trúc (để giải quyết vấn đề thành nhiều phần), tập hợp phong phú các hàm thư viện, kiểu dữ liệu, v.v.

- Máy độc lập và linh hoạt

Không giống như ngôn ngữ Assembly, các chương trình C có thể được thực thi trên các máy khác nhau với một số thay đổi cụ thể của máy. Do đó, C là một ngôn ngữ độc lập máy.

- Ngôn ngữ lập trình bậc trung

Ban đầu, C được dự định để làm lập trình cấp thấp. Nó được sử dụng để phát triển các ứng dụng hệ thống như kernel, trình điều khiển, v.v. Nó cũng hỗ trợ các tính năng của ngôn ngữ cấp cao. Đó là lý do tại sao nó được gọi là ngôn ngữ bậc trung.

- Ngôn ngữ lập trình cấu trúc

C là một ngôn ngữ lập trình cấu trúc theo nghĩa chúng ta có thể chia chương trình thành các phần bằng các hàm. Vì vậy, nó rất dễ hiểu và sửa đổi. Các hàm cũng cung cấp khả năng sử dụng lại mã.

- Thư viện phong phú

C cung cấp rất nhiều hàm sẵn có giúp phát triển nhanh chương trình.

- Quản lý bộ nhớ

C hỗ trợ tính năng phân bổ bộ nhớ động. Trong ngôn ngữ C, chúng ta có thể giải phóng bộ nhớ được phân bổ bất cứ lúc nào bằng cách gọi hàm free().

- Tốc độ nhanh

Thời gian biên dịch và thực thi nhanh do ngôn ngữ C cung cấp sẵn các thư viện.

- Con trỏ

C cung cấp các tính năng của con trỏ. Chúng ta có thể tương tác trực tiếp với bộ nhớ bằng cách sử dụng các con trỏ. Chúng ta có thể sử dụng các con trỏ cho bộ nhớ, cấu trúc, hàm, mảng, v.v.

- Đệ quy

Trong C, chúng ta có thể gọi hàm trong hàm. Nó cung cấp khả năng sử dụng lại mã cho mọi hàm. Đệ quy cho phép lập trình viên sử dụng phương pháp quay lui.

- Mở rộng

Một lý do nữa cho việc C được sử dụng rộng rãi và hiệu quả là do các trình dịch, các thư viện và các phần mềm thông dịch của các ngôn ngữ bậc cao khác lại thường được tạo nên từ C.

1.2. TẬP KÝ TỰ HỢP LỆ TRONG C

1.2.1. Tập ký tự

Mỗi ngôn ngữ lập trình đều được xây dựng từ một bộ ký tự riêng của nó. Các ký tự được nhóm lại theo nhiều cách khác nhau để tạo nên các từ. Sau đó các từ lại được liên kết theo một nguyên tắc nào đó để tạo thành các câu lệnh. Một chương trình bao gồm nhiều câu lệnh để diễn đạt một thuật toán nào đó.

Ngôn ngữ C được xây dựng trên bộ ký tự gồm:

- 26 chữ cái latinh: a, b, c,…, x, y, z hoặc chữ cái hoa: A, B, C,..., X, Y, Z.

- 10 chữ số: 0, 1, 2,…, 8, 9.

- Các ký tự toán học: +, - , *, /, =, (,), ...

- Ký tự gạch nối _

- Các ký tự đặc biệt khác như: . , : ; [ ] { } ? ! \ & | % # $, v.v.

Lưu ý:

- Dấu cách thực sự là một khoảng cách dùng để tách các từ.

🕮 Ví dụ 1.1

"Ha noi" chứa 6 ký tự, bao gồm các ký tự H, a, dấu cách, n, o và i.

"Hanoi" chứa 5 ký tự, bao gồm các ký tự H, a, n, o, i.

1.2.2. Từ khóa

Trong ngôn ngữ C, từ khóa được xác định trước, là các từ dành riêng được sử dụng trong lập trình, có ý nghĩa đặc biệt đối với trình biên dịch. Từ khóa là một phần của cú pháp và chúng không thể được sử dụng làm định danh.

🕮 Ví dụ 1.2

Cho khai báo: int bien_dem;

Trong đó: int là từ khóa để chỉ ra rằng bien_dem là một biến kiểu int (kiểu số nguyên).

Ngôn ngữ C phân biệt giữa hoa và chữ thường, vì vậy tất cả các từ khóa phải được viết bằng chữ thường. Sau đây là danh sách các từ khóa được sử dụng trong C:

auto	double	int	struct
reak	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned

1.2.3. Định danh

Khái niệm định danh rất quan trọng trong quá trình lập trình, nó không những thể hiện rõ ý nghĩa trong chương trình mà còn dùng để xác định các đại lượng khác nhau khi thực hiện chương trình. Định danh thường được đặt cho hằng, biến, mảng, con trỏ, nhãn, v.v. Chiều dài tối đa của định danh là 32 ký tự.

Quy tắc đặt định danh:

- Định danh phải là duy nhất.

- Một định danh hợp lệ chứa các chữ cái (cả chữ hoa và chữ thường), chữ số, dấu gạch dưới.

- Ký tự đầu tiên của định danh phải là chữ cái hoặc dấu gạch dưới.
- Không sử dụng từ khóa làm định danh.
- Chiều dài tối đa của định danh là 32 ký tự.

Lưu ý: Nên đặt định danh có ý nghĩa trong các bài toán cụ thể.

🕮 Ví dụ 1.3

Một số định danh đúng: delta, a_1, x, x1, x2, Num_ODD;

Một số định danh sai:

3a_1	(ký tự đầu là số)
a-1	(sử dụng dấu gạch ngang)
int	(đặt tên trùng với từ khóa)
del ta	(có khoảng trắng)
g(x)	(có dấu ngoặc tròn)

Lưu ý:

Trong C, định danh được phân biệt bởi chữ hoa và chữ thường.

Ví dụ number khác Number;

case khác Case (case là từ khóa, do đó bạn đặt tên là Case vẫn đúng).

1.3. MỘT SỐ QUY ƯỚC KHI VIẾT CHƯƠNG TRÌNH

1.3.1. Chương trình đầu tiên

Một ví dụ kinh điển được sử dụng để giới thiệu trong hầu hết các ngôn ngữ lập trình, in ra màn hình console dòng "Hello, World".

```
1   #include <stdio.h>
2   int main()
3   {
4   /* Chuong trinh dau tien trong C */
5   printf("Hello, World!\n");
6   return 0;
7   }
```

- Sử dụng DevC, gõ các dòng code trên;
- Lưu chương trình với tên hello.c
- Nhấn F9 để biên dịch.
- Nhấn F10 để chạy chương trình, dòng chữ *"Hello, World"* sẽ hiển thị trên màn hình console.

Giải thích:

Dòng 1: #include <stdio.h>: bộ lệnh tiền xử lý, thông báo cho trình biên dịch C đã thêm tệp tiêu đề *stdio.h* trong chương trình trước khi biên dịch mã nguồn, stdio là các chuẩn đầu vào/đầu ra, cho phép sử dụng các lệnh được chứa trong tệp *stdio.h*

Dòng 2: int main(): hàm chính, nơi bắt đầu thực thi chương trình.

Dòng 4: /* Chuong trinh dau tien trong C */: sẽ được trình biên dịch bỏ qua, được sử dụng để thêm các chú thích trong chương trình.

Dòng 5: printf("Hello, World! \n");: là một chức năng có sẵn trong C, hiển thị ra màn hình dòng Hello, World.

Dòng 6: return 0; : chấm dứt hàm main() và trả về giá trị 0.

1.3.2. Cấu trúc chương trình viết trong C

Một chương trình trong C được phân chia thành nhiều phần. Xem xét Ví dụ 1.4 sau đây:

🕮 Ví dụ 1.4

Viết chương trình tính tổng hai số nguyên a, b.

1	/*		Phần tài liệu - Mô tả tên tệp. - Tác giả. - Tóm tắt nội dung của chương trình.
2	* Vi_du_1_4.c - Le Thi Chi - 15.07.2019		
3	*Tinh tong cua hai so nguyen a va b		
4	*/		
5	#include <stdio.h>		Phần liên kết. Phần liên kết bao gồm các tệp tiêu đề của các hàm thư viện chuẩn có thể được sử dụng trong chương trình.
6	#include <conio.h>		
7	int tinhtong(int, int);		Phần khai báo nguyên mẫu hàm. Khai báo các nguyên mẫu hàm được sử dụng trong chương trình.
8	main()		Phần hàm chính.

9	{		Phần hàm main() là phần quan trọng nhất của bất kỳ chương trình C nào. Trình biên dịch bắt đầu thực hiện chương trình C từ hàm main(). Hàm main() là bắt buộc trong lập trình C. Nó có hai phần: Phần khai báo: Tất cả các biến được sử dụng sau này trong phần thi hành được khai báo trong phần này. Phần thực thi: Phần này chứa các câu lệnh sẽ được thực hiện bởi trình biên dịch.
10	int x, y, tong;		
11	printf("Nhap x = ");		
12	scanf("%d", &x);		
13	printf("Nhap y = ");		
14	scanf("%d", &y);		
15	tong = tinhtong(x,y);		
16	printf("Tong = %d ", tong);		
17	getch();		
18	}		
19	int tinhtong(int a, int b)		Phần chương trình con: Chứa tất cả các hàm do người dùng định nghĩa. Khi biên dịch chương trình, căn cứ vào các lời gọi hàm, mà các mã chương trình của các hàm tương ứng được thực thi.
20	{		
21	int t;		
22	t = a + b ;		
23	return t;		
24	}		

Như vậy, một chương trình bao gồm các thành phần cơ bản sau đây

1	[Phần tài liệu] Mô tả tệp, tác giả, ngày viết, tóm tắt chương trình. Phần này có thể có hoặc không có.
2	Phần liên kết Chỉ ra sự liên kết các thư viện chuẩn trong C. Phần này bắt buộc phải có.

3	[Phần định nghĩa marco] Xác định các hằng tượng trưng trong chương trình. Hằng tượng trưng là các đại lượng có giá trị không thay đổi trong chương trình. Phần này không bắt buộc phải có.
4	[Phần khai báo nguyên hàm mẫu] Khai báo các nguyên mẫu hàm. Phần này không bắt buộc, trong trường hợp phần định nghĩa các hàm được đặt ở vị trí này thì bỏ qua các nguyên hàm mẫu.
5	[Phần định nghĩa các cấu trúc] Phần này được sử dụng khi cần định nghĩa các kiểu dữ liệu do người dùng định nghĩa. Không bắt buộc phải có.

6	<kiểu trả về> main(tham số)
	{
	Thân hàm main(): chứa các khai báo và các lệnh thực thi, chứa các lời gọi hàm;
	}
	Đây là phần bắt buộc phải có. Chương trình khi biên dịch được thực hiện bắt đầu từ đây.

7	[Định nghĩa hàm của người dùng] Nếu có lời gọi hàm thì phần này bắt buộc phải có. Có hai vị trí có thể đặt các định nghĩa hàm. Vị trí 1: Trước hàm main(), khi đó có thể bỏ qua các khai báo nguyên hàm mẫu. Vị trí 2: Sau hàm main(), như vậy bắt buộc phải có phần khai báo nguyên hàm mẫu trước hàm main().

1.3.2.1. Phần tài liệu

Đây là phần đầu tiên trong một chương trình viết bằng ngôn ngữ C. Nhiệm vụ của phần tài liệu cung cấp những thông tin cơ bản về chương trình: tác giả, ngày giờ, tên tệp chương trình và có thể thêm phần tóm tắt nội dung của chương trình.

Khi biên dịch, chương trình sẽ bỏ qua phần này, bởi vì tất cả những nội dung trong phần này, được chương trình hiểu là các chú thích.

Các chú thích trong C được thể hiện dưới hai dạng:

Ghi chú thích trên 1 dòng, sử dụng ký hiệu //:

// dòng chú thích

Ghi chú thích trên nhiều dòng, các dòng trong chú thích được đặt trong cặp /* và */ như sau:

/*

Nội dung chú thích

*/

🕮 Ví dụ 1.5

// Chuong trinh tinh tong hai so

/

Ten tep: tinhtong.c

Tac gia: Le Thi Chi

Ngay 20/7/2019

**/*

1.3.2.2. Phần liên kết (Bao hàm tệp)

Trong chương trình C (trong hàm main cũng như các hàm khác do người lập trình viết) có thể sử dụng các hàm, hằng, kiểu dữ liệu, v.v. (gọi chung là các thành phần) đã được định nghĩa trong thư viện của C. Để sử dụng các thành phần này chúng ta phải chỉ dẫn cho chương trình dịch biết các thông tin về các thành phần cần sử dụng, các thông tin đó được khai báo trong tệp gọi là tệp tiêu đề (có phần mở rộng là h - viết tắt của header). Và phần các bao hàm tệp là các chỉ dẫn để chương trình gộp các tệp này vào chương trình. Trong một chương trình, có thể không dùng hoặc dùng nhiều tệp tiêu đề.

Cú pháp của một dòng bao hàm tệp:

#include <tên_tệp> hoặc #include “tên_tệp”.

Trong đó: tên_tệp là tên có thể có cả đường dẫn của tệp tiêu đề (.h) mà chúng ta cần sử dụng, mỗi lệnh bao hàm tệp trên một dòng.

🕮 Ví dụ 1.6

Cho các khai báo bao hàm tệp sau đây:

#include<stdio.h>

#include <conio.h>

#include “phanso.h”

Sự khác nhau giữa cặp < > và “ ” bao quanh tên tệp là với cặp < >, chương trình dịch tìm tên tệp tiêu đề trong thư mục ngầm định xác định bởi đường dẫn trong mục Option/Directories, còn với cặp “ ” chương trình dịch tìm tệp trong thư mục hiện tại, nếu không có mới tìm trong thư mục các tệp tiêu đề ngầm định như trường hợp < >.

1.3.2.3. Phần định nghĩa marco

Khái niệm macro là gì? Giả sử trong chương trình có một nội dung (giá trị) nào đó và chúng ta muốn sử dụng nó nhiều lần trong chương trình, nhưng chúng ta không muốn viết trực tiếp nó vào chương trình lúc soạn thảo, vì một vài lý do nào đó (chẳng hạn như nó sẽ làm chương trình khó đọc, khó hiểu, hoặc khi thay đổi sẽ khó, v.v.). Lúc này chúng ta hãy gán cho nội dung đó một “tên” và bạn sử dụng “tên” đó để viết trong chương trình nguồn. Khi biên dịch chương trình, chương trình dịch sẽ tự động thay thế nội dung của “tên” vào đúng vị trí của “tên” đó. Thao tác này gọi là phép thế macro và chúng ta gọi “tên” là tên của macro và nội dung của nó được gọi là nội dung của macro.

Một macro được định nghĩa như sau:

#define tên_macro nội_dung

Trong đó, tên macro là một tên hợp lệ, nội dung (giá trị) của macro được coi thuần tuý là một xâu cần thay thế vào vị trí xuất hiện tên của macro tương ứng, giữa tên và nội dung cách nhau một hay nhiều khoảng trống (dấu cách). Nội dung của macro bắt đầu từ kí tự khác dấu trống đầu tiên sau tên macro cho tới hết dòng.

🕮 Ví dụ 1.7

#define SOCOT 20

#define max(a,b) (a > b ? a:b)

Với hai ví dụ trên, khi gặp tên SOCOT, chương trình dịch sẽ tự động thay thế bởi 20 và max(a, b) sẽ được thay thế bởi (a > b ? a : b);

Lưu ý:

- Phép thay thế macro đơn giản chỉ là thay nội dung macro vào vị trí tên của nó do vậy sẽ không có cơ chế kiểm tra kiểu.

- Khi định nghĩa các macro có “tham số” có thể sau khi thay thế biểu thức mới thu được có trật tự tính toán không như bạn mong muốn. Chẳng hạn, ta có macro tính bình phương một số như sau: #define bp(a) a*a và trong chương trình có câu lệnh bp(x + y) sẽ được thay là x + y * x + y và

kết quả không như chúng ta mong đợi. Trong trường hợp này chúng ta nên sử dụng dấu ngoặc cho các tham số của macro #define bp(a) (a)*(a).

1.3.2.4. Khai báo nguyên mẫu

Trong phần này chúng ta nêu một số thông tin về khai báo nguyên mẫu để giải thích cấu trúc chương trình chứ không có ý định trình bày về hàm, chi tiết về hàm sẽ được trình bày trong phần định nghĩa hàm.

Nguyên mẫu một hàm là dòng khai báo cung cấp các thông tin: tên hàm, kiểu hàm, số đối số và kiểu từng đối số của hàm.

Cú pháp khai báo nguyên mẫu

<kiểu_hàm> <tên_hàm> ([khai báo đối]);

🕮 Ví dụ 1.8

Cho các khai bao nguyên mẫu sau đây:

int min (int, int);

float binhphuong (float y);

float giatri(int, float);

Lưu ý:

Phần khai báo đối của nguyên mẫu, mục đích là xác định số đối số và kiểu của từng đối số, do vậy bạn có thể không viết tên của đối số nhưng kiểu của chúng thì phải có và bạn phải liệt kê đầy đủ kiểu của từng đối số.

1.3.2.5. Phần định nghĩa các cấu trúc

Ngoài những kiểu chuẩn đã được cung cấp sẵn của ngôn ngữ, người lập trình có thể định nghĩa ra các kiểu mới từ những kiểu đã có bằng cách sử dụng từ khoá typedef (*type define).*

Với cú pháp như sau:

typedef <mô_tả_kiểu> <tên_kiểu_mới>;

Trong đó <tên_kiểu_mới> là tên kiểu cần tạo do người lập trình đặt theo quy tắc về tên của ngôn ngữ và <mô_tả_kiểu> là phần chúng ta định nghĩa các thành phần cấu thành lên kiểu mới.

🕮 Ví dụ 1.9

Cho hai định nghĩa cấu trúc sau đây:

typedef unsigned char byte;

typedef long nguyen_dai;

Sau định nghĩa này các tên mới byte được dùng với ý nghĩa là tên kiểu dữ liệu nó tương tự như unsigned charvà nguyen_dai tương tự như long.

Chúng ta có thể định nghĩa biến a, b kiểu byte như sau:

byte a, b; (byte được sử dụng là kiểu dữ liệu vừa được định nghĩa unsigned char).

1.3.2.6. Định nghĩa hàm

Trong phần này chúng ta định nghĩa các hàm của người dùng, một định nghĩa hàm bao gồm dòng tiêu đề của hàm và thân của hàm, với cú pháp như sau:

<kiểu_hàm> <tên_hàm> ([khai báo đối])

{

< thân hàm >

}

🕮 Ví dụ 1.10

Xây dựng hàm tìm số bé nhất của hai số nguyên a, b.

```
int max( int a, int b)
{
        if (a ≥ b) return b;
        else return a;
}
```

1.3.2.7. Hàm main()

Đây là thành phần bắt buộc duy nhất trong một chương trình C, thân của hàm main() bắt đầu từ sau dấu mở móc mở ngoặc {cho tới dấu móc đóng ngoặc}. Không giống như chương trình của Pascal luôn có phần chương trình chính, chương trình trong C được phân thành các hàm độc lập các hàm có cú pháp như nhau và cùng mức và một hàm đảm nhiệm phần thân chính của chương trình, tức là chương trình sẽ bắt đầu được thực hiện từ dòng lệnh đầu tiên và kết thúc sau lệnh cuối cùng trong thân hàm main().

Trong định nghĩa một hàm nói chung đều có hai phần đó là tiêu đề của hàm, dòng này bao gồm các thông tin: Tên hàm, kiểu hàm (kiểu giá trị hàm trả về), các tham số hình thức (tên tham số và kiểu của chúng). Phần thứ hai là thân của hàm, đây là tập các lệnh (hoặc khai báo) thực hiện các thao tác theo yêu cầu về chức năng của hàm đó. Hàm main() cũng chỉ là một trường hợp riêng của hàm nhưng có tên cố định là main,

có thể có hoặc không có các đối số và có thể trả về giá trị, kiểu của giá trị này được xác định bởi <kiểu_hàm> (chi tiết về đối, kiểu của hàm main() sẽ được đề cập kỹ hơn trong các phần sau).

Thân hàm main() được bao bởi cặp dấu mở ngoặc { và dấu đóng ngoặc } có thể gồm các lệnh, các khai báo hoặc định nghĩa biến, hằng, kiểu, các thành phần này trở thành cục bộ trong hàm main().

Lưu ý:

- Các thành phần của chương trình mà chúng ta vừa nêu, trừ hàm main() là thành phần phải có và duy nhất trong một chương trình viết bằng ngôn ngữ C, còn các thành phần khác là tuỳ chọn, có thể không có hoặc có.

- Thứ tự các thành phần không bắt buộc theo trật tự như trên mà chúng có thể xuất hiện theo trật tự tuỳ ý nhưng phải đảm bảo yêu cầu mọi thành phần phải được khai báo hay định nghĩa trước khi sử dụng.

- Các biến, hằng khai báo ngoài mọi hàm có phạm vi sử dụng là toàn cục (tức là có thể sử dụng từ sau lệnh khai báo cho tới hết chương trình). Các hằng, biến khai báo trong một hàm (hoặc trong một khối) là thành phần cục bộ (có phạm vi sử dụng trong hàm hoặc trong khối đó mà thôi).

- Các hàm trong C là một mức (tức là trong hàm không chứa định nghĩa hàm khác).

1.3.2.8. Câu lệnh

Chương trình C được tổ chức thành nhiều câu lệnh, mỗi câu lệnh có thể thực hiện một công việc nào đó, cuối mỗi câu lệnh phải được kết thúc bằng dấu chấm phẩy (;). Trên một dòng có thể viết nhiều câu lệnh và một câu lệnh cũng có thể viết trên nhiều dòng.

Tại những vị trí của câu lệnh mà cho phép xuất hiện một hoặc nhiều dấu khoảng cách thì ta có thể ngắt phần còn lại của câu lệnh xuống dòng tiếp theo.

🕮 Ví dụ 1.11

Cho đoạn chương trình sau

```
1  void main()
2  {
3  int  x,y;
4  x = 7; y = 8;
5  if
```

```
6  (x > y)
7  printf("x> y");
8  }
```

Trong ví dụ trên, ở dòng lệnh thứ hai, gồm có hai câu lệnh. Câu lệnh if được viết trên ba dòng thứ 5, 6 và 7.

Trong trư7ụ trên, ác câu lư7ụ trên, ở dòng lệnặc câu { và } thì đưì } lư7ụ trên, ở dòng lệnh thứ hai, gồm có hai câu lệnh. Câu lệnh if đượ

Trong cú pháp của các câu lệnh, tại những vị trí chỉ cho phép xuất hiện một câu lệnh mà ta muốn thực hiện nhiều câu lệnh thì ta phải đặt những câu lệnh đó trong một khối lệnh.

🕮 Ví dụ 1.12

Cho đoạn chương trình sau:

```
1   void main()
2   {
3      int a = 16, b = 8, c = 4;
4   if((a > b) && ( b > c))
5               {
6                 max = a;
7                 printf("\n phan tu lon nhat = %d", max);
8               }
9     …
10  }
```

Trong Ví dụ 1.12, đoạn chương trình

```
{
        max = a;
        printf("\n phan tu lon nhat = %d", max);
}
```

được gọi là một khối lệnh;

1.3.2.9. Chú thích

Trong khi lập trình cần phải ghi chú để giải thích các biến, hằng, các thao tác xử lý giúp cho chương trình rõ ràng dễ hiểu, dễ nhớ, dễ sửa chữa và để người khác đọc vào dễ hiểu. Trong C có các kiểu ghi chú sau:

- // nội dung ghi chú. Khi đó nội dung ghi chú được ghi trên một dòng hoặc phần còn lại của một dòng.

- /* nội dung ghi chú */. Khi đó nội dung ghi chú có thể ghi trên một dòng, trên nhiều dòng hoặc trên phần còn lại của một dòng.

🕮 Ví dụ 1.13

Viết chương trình in ra số lớn nhất giữa 2 số a và b.

```
1  void  main()
2  {
3  int a, b;  //khai báo biến a,b kiểu int
4  a = 11;   //gán 11 cho a
5  b = -13;  //gán -13 cho b
6  /* thuat toan tim so lon nhat giua 2 so a va b, neu a lon hon b thi a
   lon nhat nguoc lai b lon nhat */
7  if (a > b) printf("max =: %d", a);
8  else printf("max =: %d", b);
9  }
```

Khi biên dịch chương trình, C gặp cặp dấu ghi chú sẽ không dịch ra ngôn ngữ máy.

1.3.3. Trình tự các bước thực thi một chương trình viết bằng C

Chương trình C tuân theo nhiều bước trong thực thi. Để hiểu rõ về trình tự các bước của chương trình C, trước tiên chúng ta hãy xem một chương trình đơn giản sau đây:

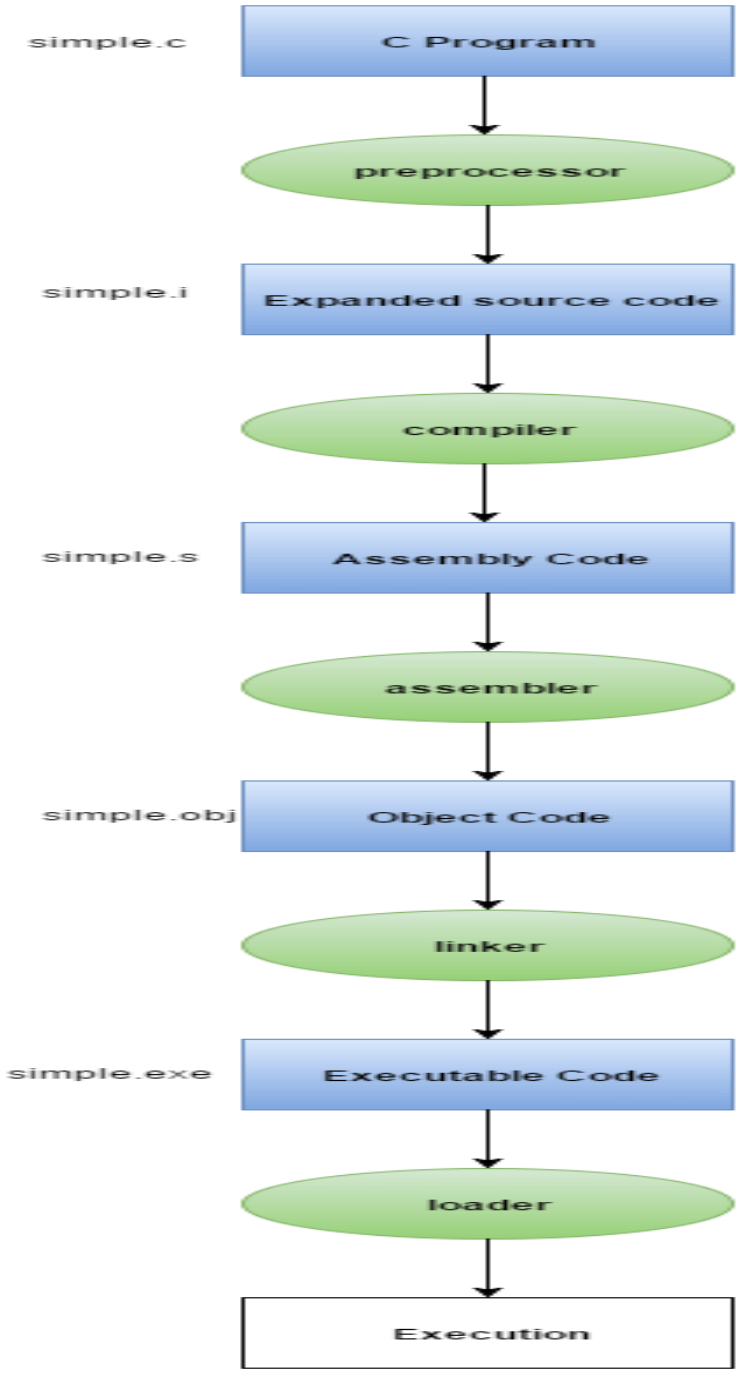
🕮 Ví dụ 1.14

Tệp: simple.c

```
1  #include <stdio.h>
2  int main(){
3  printf("Xin chao! Day la chuong trinh don gian!");
4  return 0;
5  }
```

Chúng ta hãy sẽ tìm hiểu quá trình thực thi của chương trình trên bằng Hình 1.1:

Bước 1: Mã nguồn của chương trình C (C program, tệp simple.c) được gửi đến bộ tiền xử lý (preprocessor). Bộ tiền xử lý có trách nhiệm chuyển đổi các chỉ thị tiền xử lý thành các giá trị tương ứng của chúng. Bộ tiền xử lý tạo ra một mã nguồn mở rộng (Expanded source code, tệp simple.i).



Hình 1.1. Quá trình thực thi chương trình

Bước 2: Mã nguồn mở rộng (tệp simple.i) được gửi đến trình biên dịch (compiler) để biên dịch mã và chuyển đổi nó thành mã assembly (mã máy, tệp simple.s).

Bước 3: Mã asembly (tệp simple.s) được gửi đến trình biên dịch mã (assembler) chuyển đổi simple.s thành mã đối tượng (Object code). Bây giờ một tệp tin simple.obj được tạo ra.

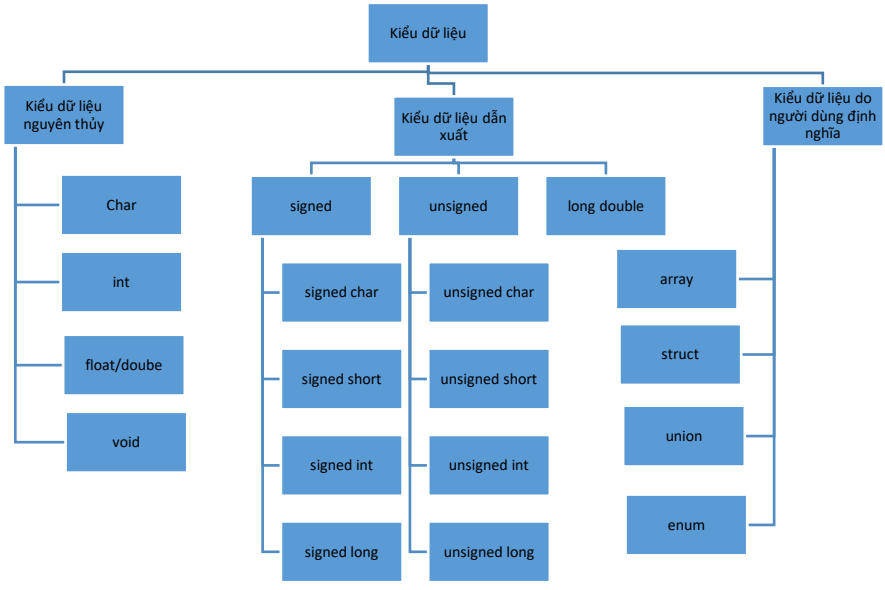
Bước 4: Mã đối tượng (tệp simple.obj) được gửi đến trình liên kết liên kết nó với thư viện, chẳng hạn như các tệp tiêu đề. Sau đó, nó được chuyển đổi thành mã thực thi (executable code). Một tập tin đơn simple.exe được tạo ra.

Bước 5: Mã thực thi (tệp simple.exe) được gửi đến trình tải (loader) sẽ tải vào bộ nhớ và sau đó nó được thực thi. Sau khi thực hiện, đầu ra được gửi đến bàn điều khiển.

1.4. KIỂU DỮ LIỆU TRONG C

Kiểu dữ liệu được hệ thống dùng để xác định các thuộc tính cơ bản khác nhau về dữ liệu được lưu trong bộ nhớ. Các thuộc tính đó bao gồm: loại dữ liệu, phạm vi của dữ liệu, số byte bị chiếm dụng, v.v.

Các kiểu dữ liệu trong C được phân chia thành 3 nhóm chính:



Hình 1.2. Các kiểu dữ liệu của ngôn ngữ C

Bảng 1.1. Các kiểu dữ liệu - kích thước và phạm vi

Kiểu dữ liệu	Kích thước	Phạm vi
char	1 byte	-128 đến 127
signed char		
unsigned char	1 byte	0 đến 255
short	2 bytes	-32,767 đến 32,767
signed short		
signed short int		
unsigned short	2 bytes	0 đến 65,535
unsigned short int		
int	2 hoặc 4 bytes	-32,768 đến 32,767 hoặc -2,147,483,648 đến 2,147,483,647
signed int		
unsigned int	2 hoặc 4 bytes	0 đến 65,535 hoặc 0 đến 4,294,967,295
long	4 bytes	-2,147,483,648 đến 2,147,483,647
signed long		
signed long int		
unsigned long	4 bytes	0 đến 4,294,967,295
unsigned long int		
long long	8 bytes	-9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807
long long int		
signed long long		
signed long long int		
unsigned long long	8 bytes	0 đến 18,446,744,073,709,551,615
unsigned long long int		
float	4 bytes	1.2E-38 đến 3.4E+38
double	8 bytes	2.3E-308 đến 1.7E+308
long double	12 bytes	3.4E-4932 đến 1.1E+4932

1.4.1. Kiểu dữ liệu nguyên thủy

Ngôn ngữ C hỗ trợ bốn kiểu dữ liệu nguyên thủy, bao gồm char, int, float và void. Các kiểu dữ liệu nguyên thủy còn được gọi là các kiểu dữ liệu cơ bản, được xác định trước.

Bảng 1.2. Kiểu dữ liệu nguyên thủy

Kiểu dữ liệu	Kích thước	Phạm vi
char	1 byte	[-128, 127]
int	2 byte hoặc 4 byte	[-32.768, 32.767] hoặc [-2.147.483.648 , 2.147.483.647]
float	4 byte	[1.2E-38, 3.4E+38]
void	1 byte	

Kích thước và phạm vi của một loại dữ liệu phụ thuộc vào máy và có thể khác nhau tùy theo trình biên dịch. Ngôn ngữ C chỉ yêu cầu kích thước tối thiểu được thực hiện bởi mọi trình biên dịch cho từng loại dữ liệu. Chẳng hạn, kích thước của kiểu int thay đổi từ trình biên dịch này sang trình biên dịch khác, nhưng nó phải có ít nhất 2 byte trên mỗi trình biên dịch.

1.4.1.1. Kiểu ký tự

Bất kỳ một ký tự đơn trong C đều được biểu diễn bằng kiểu char. Kích thước của kiểu char là 1 byte và có thể lưu trữ 128 ký tự.

🕮 Ví dụ 1.15

Cho khai báo sau:

char test = ‘A’;

1.4.1.2. Kiểu nguyên

Trong ngôn ngữ C, từ khóa int được sử dụng để xác định dữ liệu dạng số nguyên. Kích thước của int là 2 byte hoặc 4 byte (phụ thuộc vào trình biên dịch) và có thể lưu trữ các giá trị trong phạm vi [-32.768, 32.767] (2 byte) hoặc [-2.147.483.648, 2.147.483.647] (4 byte).

🕮 Ví dụ 1.16

int a = 15;

Khai báo một biến a kiểu nguyên, có giá trị trong phạm vi [-32.768, 32.767].

1.4.1.3. Kiểu thực

Trong ngôn ngữ C, giá trị kiểu thực được xác định bằng từ khóa float

hoặc double, trong đó từ khóa float xác định một số chấm động chính xác đơn trong phạm vi [1.2E-38, 3.4E+38], từ khóa double xác định một số chấm động chính xác kép trong phạm vi [2.3E-308, 1.7E+308]. Số chữ số có nghĩa sau dấu thập phân được gọi là độ chính xác, độ chính xác của kiểu float là 6 chữ số thập phân và double là 15 chữ số thập phân.

🕮 Ví dụ 1.17

Cho các khai báo sau đây:

```
float percentage = 95.67;
const float PI = 3.14159265359f;
double speed = 2.998e+8;
```

1.4.1.4. Kiểu void

Giống như tên, kiểu void không xác định giá trị. Từ khóa void được sử dụng để xác định hàm không trả về giá trị hoặc kiểu con trỏ.

🕮 Ví dụ 1.18

```
void * p; //Định nghĩa con trỏ p kiểu void.
```

1.4.2. Kiểu dữ liệu dẫn xuất

Các kiểu dữ liệu nguyên thủy được trình bày trong 1.4.1 được sử dụng cho việc khai báo các biến trong chương trình, tùy thuộc vào phạm vi sử dụng. Những kiểu dữ liệu này có thể được sửa đổi cho phù hợp với những tình huống khác nhau. Kết quả của việc sửa đổi đó, chúng ta có được những kiểu dữ liệu dẫn xuất từ các kiểu nguyên thủy này.

Một kiểu dữ liệu dẫn xuất được xác định bằng cách xác định các bổ từ chỉ kích thước hoặc dấu cùng với các kiểu dữ liệu nguyên thủy. Các kiểu dẫn xuất được tạo bằng các dữ liệu nguyên thủy cùng với các hành vi hay thuộc tính được sửa đổi. Các bổ từ được sử dụng để kết hợp trong C bao gồm: signed, unsigned, long và short. Các bổ từ trên được áp dụng với kiểu dữ liệu ký tự và kiểu dữ liệu số nguyên, bổ từ long cũng có thể được áp dụng với kiểu double.

Để khai báo kiểu dữ liệu dẫn xuất, ta đặt các bổ từ trước các kiểu dữ liệu nguyên thủy.

🕮 Ví dụ 1.19

```
signed int a;
unsigned int b;
signed char c;
```

unsigned char d;

short int e;

long int f;

long double g;

1.4.2.1. Bổ từ signed và unsigned

Khi khai báo một số nguyên, mặc định đó là số nguyên có dấu (âm, dương). Bổ từ signed và unsigned được sử dụng để thể hiện dấu của các kiểu dữ liệu ký tự hoặc số nguyên.

Có hai kiểu char và int có thể có dấu âm hoặc không. Theo mặc định thì mọi kiểu int là có dấu (nghĩa là chúng chấp nhận các số âm). Để dùng dưới dạng không có dấu (tức là kiểu nguyên chỉ chấp nhận các số không âm) thì từ khoá unsigned phải được dùng. Ngoài ra, thay vì khai báo đầy đủ trong dạng unsigned int, người ta có thể lược bỏ bớt từ khóa int (và nó được xem như hiểu ngầm - điều này chỉ dùng được cho kiểu int mà thôi).

🕮 Ví dụ 1.20

Khai báo unsigned int a và unsigned a được chương trình xem là tương đương nhau.

Kiểu unsigned chỉ rõ rằng một biến chỉ có thể nhận giá trị dương. Bổ từ này thường được sử dụng với kiểu int và trong một vài trường hợp có thể được sử dụng với kiểu dữ liệu float, nhưng điều này làm mất tính linh động của mã lệnh.

Với việc thêm bổ từ unsigned vào trước kiểu dữ liệu int, miền giá trị của các số dương này được tăng lên gấp đôi.

Chẳng hạn, nếu ta khai báo: unsigned int a; khi đó phạm vi của biến a sẽ là [0, 65535], thay vì [-32768, 32767] mà kiểu int hỗ trợ.

Đặc tả của C không xác định rõ ràng là kiểu char sẽ là loại có dấu hay không dấu; khi đó, dấu của kiểu này tùy thuộc vào quy định của nhà phát hành trình dịch. Như vậy, một cách để giảm sai sót khi làm việc trên nhiều loại trình dịch C khác nhau là khai báo rõ ràng bằng các định tính signed hay unsigned nếu dùng kiểu char để tính toán trên các con số. (Điều này thực sự sẽ không quá quan trọng nếu dùng kiểu char như là kiểu "ký tự".)

🕮 Ví dụ 1.21

unsigned char g;

signed char t;

Tiêu chuẩn chung yêu cầu char, signed char, unsigned char là các kiểu khác nhau. Ngoài ra, các hàm chuẩn về dãy các ký tự sử dụng các con trỏ chỉ tới kiểu char (không có định tính), nhiều trình dịch C sẽ bắt lỗi (hay cảnh cáo) nếu các kiểu ký tự khác được dùng như là dãy ký tự được chuyển vào các hàm này.

Bảng 1.3. Kiểu dữ liệu char

Kiểu	Phạm vi	Kích thước
[signed] char	-128 đến 127	1byte
unsigned char	0 đến 255	1byte

🕮 Ví dụ 1.22

Đoạn chương trình sau minh họa sự khác nhau giữa hai kiểu char

```
char ch1;
unsigned char ch2;
ch1 = 200; ch2 = 200;
```

Khi đó, nếu ta dùng câu lệnh in ra màn hình:

```
printf("ch1 = %d, ch2 = %d", ch1, ch2);
```

thì sẽ cho kết quả như sau ch1 = -56, ch2 = 200, nhưng cả ch1 và ch2 đều biểu diễn chung một ký tự có mã trong bảng mã ASCII là 200, tức là nếu ta dùng câu lệnh in printf("ch1 = %c, ch2 = %c", ch1, ch2) thì sẽ ra màn hình hai giá trị ký tự của ch1 và ch2 là giống nhau.

✍**Giải thích:** Số 200 được biểu diễn ở hệ nhị phân là 11001000. Vì ch2 là unsigned char (số không dấu) nên tất cả các bit trong dãy nhị phân 11001000 đều được dùng để tính giá trị vậy giá trị của ch2 là 200, còn ch1 là kiểu signed char (có dấu) nên bit đầu tiên trong dãy nhị phân là bit dấu, ở đây là bit 1 nên đó là số âm. Như vậy, dãy nhị phân trên là biểu diễn số âm ở dạng mã bù. Để chuyển về biểu diễn ở dạng mã ngược ta lấy dãy nhị phân trên trừ cho 1 được kết quả là 11000111. Từ mã ngược trên ta đảo ngược các bit sẽ được biểu diễn ở dạng mã thuận như sau 10111000. Giá trị của dãy nhị phân 10111000 là -56.

1.4.2.2. Bổ từ short và long

Hai bổ từ này được sử dụng khi muốn biểu diễn số nguyên ngắn hơn hoặc dài hơn chiều dài bình thường. Bổ từ short được sử dụng khi chiều dài yêu cầu ngắn hơn chiều dài bình thường và bổ từ long được dùng khi chiều dài yêu cầu dài hơn chiều dài số nguyên bình thường.

Bổ từ short được sử dụng với kiểu dữ liệu int. Nó sửa đổi kiểu dữ liệu int theo hướng chiếm ít vị trí bộ nhớ hơn. Như vậy, nếu kiểu dữ liệu

int cần 16 bit (2 byte) thì một biến kiểu short int (hoặc có thể viết ngắn gọn là short) chiếm giữ 8 bit (1 byte), cho phép biểu diễn các số trong phạm vi [-128, 127].

Bổ từ long được sử dụng với miền giá trị rộng hơn. Nó có thể được sử dụng với kiểu int và kiểu double. Khi sử dụng với kiểu int, phạm vi biến trong phạm vi từ -2.147.483.648 đến 2.147.483.647 và chiếm giữ 32 bit (8 byte) bộ nhớ. Tương tự, khi áp dụng với kiểu double, biến chiếm giữ 128 bit (16 byte) bộ nhớ.

1.4.3. Kiểu dữ liệu do người dùng định nghĩa

Mặc dù có một số loại cơ bản và dẫn xuất, ngôn ngữ C hỗ trợ tính năng để xác định loại tùy chỉnh dựa trên nhu cầu người dùng. Kiểu do người dùng xác định bao gồm mảng, con trỏ, cấu trúc, kiểu liệt kê, kiểu hợp nhất, v.v.

Các kiểu dữ liệu do người dùng định nghĩa sẽ được trình bày chi tiết ở các Chương 5, 6, 7 của cuốn giáo trình này.

1.5. BIẾN VÀ HẰNG

1.5.1. Biến

1.5.1.1. Định nghĩa

Biến được sử dụng để thể hiện cho đại lượng có giá trị thay đổi (đã biết hoặc chưa biết trong bộ nhớ). Biến là tên tham chiếu được đặt cho vị trí bộ nhớ chứa dữ liệu trong chương trình. Mỗi biến trong C có một kiểu dữ liệu cụ thể, xác định kích thước và phạm vi của các giá trị có thể được lưu trữ trong bộ nhớ đó, đồng thời xác định tập hợp các phép toán được áp dụng cho các biến đó. Tại thời điểm khai báo, phải xác định rõ ràng kiểu dữ liệu của biến.

Trong ngôn ngữ C, có hai cách truy cập dữ liệu được lưu trữ trong bộ nhớ, theo địa chỉ của bộ nhớ hoặc tên tham chiếu đến vùng bộ nhớ đó, tức là tên của biến. Tên biến là một định danh, phải tên thủ nghiêm ngặt các quy tắc đặt tên định danh. Tên của một biến có thể bao gồm các chữ cái, chữ số và ký tự gạch dưới. Nó phải bắt đầu bằng một chữ cái hoặc dấu gạch dưới. C phân biệt ký tự hoa và ký tự thường, do đó trình biên dịch C xử lý các biến chứa các ký tự viết hoa và các biến chứa các ký tự viết thường khác nhau. Chẳng hạn, num, Num, NUM, nUm, v.v. là các biến khác nhau trong chương trình. Chúng ta không thể có hai biến có cùng tên trong cùng một phạm vi.

1.5.1.2. Khai báo biến

Trong ngôn ngữ C, bạn phải khai báo biến trước khi sử dụng. Khai báo biến thông báo cho trình biên dịch về tên tham chiếu mà chúng ta sử dụng cho một vị trí trong bộ nhớ.

Cú pháp:

<data_type> <variable_names>;

Trong đó:

data_type: kiểu dữ liệu, có thể là kiểu dữ liệu nguyên thủy (char, int, float) hoặc có thể là kiểu dữ liệu dẫn xuất (signed char, unsigned int, long double, v.v.), cũng có thể là các kiểu dữ liệu do người dùng định nghĩa (array, struct, pointer, v.v.)

variable_names: danh sách các biến có cùng kiểu dữ liệu, các biến được ngăn cách bởi dấu phẩy (,), kết thúc câu lệnh khai báo biến là dấu chấm phẩy (;)

🕮 **Ví dụ 1.23**

int a, b, c; //khai báo ba biến a, b, c có kiểu nguyên int.

float delta ;//khai báo biến delta có kiểu thực chấm động đơn float.

char ch_1, ch_2; //khai báo hai biến ch_1, ch_2 có kiểu char.

unsigned int arr[100]; // khai báo một biến mảng arr, có tối đa 100 phần tử, mỗi phần tử có kiểu nguyên không âm.

int * ptr; // khai báo biến con trỏ *ptr, trỏ tới các phần tử có kiểu nguyên.

1.5.1.3. Khai báo và khởi tạo giá trị cho các biến

Cú pháp:

<data_type> <variable_name> = <value-or-expression>;

Trong đó:

data_type: kiểu dữ liệu, có thể là kiểu dữ liệu nguyên thủy (char, int, float) hoặc có thể là kiểu dữ liệu dẫn xuất (signed char, unsigned int, long double, v.v.), cũng có thể là các kiểu dữ liệu do người dùng định nghĩa (array, struct, pointer, v.v.)

variable_name: tên biến;

<value-or-expression>: giá trị hoặc một biểu thức;

🕮 Ví dụ 1.24

Khai bước trước, khởi tạo sau:

```
int a, b, c;
float d, si;
a = 10;   // gán giá trị sau khai báo a = 10
b = 5000;  // gán giá trị b = 5000
c = 10; // gán giá trị c = 10
d = 0.5; // gán giá trị d = 05.
si = (b * c * d) / 100; // gán si bởi biểu thức
```

🕮 Ví dụ 1.25

Vừa khai báo vừa khởi tạo giá trị:

```
int a = 10, b = 5000, c = 10;
float d = 0.5;
```

1.5.1.4. Phạm vi của biến

Khi lập trình, chúng ta phải nắm rõ phạm vi của biến. Nếu khai báo và sử dụng không đúng, không rõ ràng, sẽ dẫn đến sai sót khó kiểm soát được, vì vậy cần phải xác định đúng vị trí, phạm vi sử dụng biến trước khi sử dụng biến.

Khai báo biến ngoài (biến toàn cục): vị trí biến đặt bên ngoài tất cả các hàm, cấu trúc, v.v. Các biến này có ảnh hưởng đến toàn bộ chương trình. Chu trình sống của biến toàn cục là từ lúc bắt đầu chạy chương trình đến khi kết thúc chương trình.

Khai báo biến trong (biến cục bộ): Vị trí biến đặt bên trong hàm, cấu trúc, v.v. Biến cục bộ chỉ ảnh hưởng nội bộ bên trong hàm, cấu trúc đó, v.v. Chu trình sống của biến cục bộ từ lúc bắt đầu từ lúc hàm, cấu trúc, v.v. được gọi thực hiện đến khi thực hiện xong hàm, cấu trúc, v.v.

🕮 Ví dụ 1.26

Xét đoạn chương trình sau đây:

```
1   #include <stdio.h>
2   // Khai bao bien toan cuc:
3    int a, b;
4    int c;
5    float f;
```

```
6   int main () {
7     /* Khai bao bien cuc bo: */
8     int a, b;
9     int c;
10    float f;
11    /* Khoi tao gia tri */
12    a = 10;
13    b = 20;
14   // cau lenh gan
15    c = a + b;
16    printf("Gia tri cua c : %d \n", c);
17    f = 70.0/3.0;
18    printf("Gia tri cua f : %f \n", f);
19    return 0;
20  }
```

✍Kết quả chương trình:

```
Gia tri cua c : 30
Gia tri cua f : 23.333334
```

1.5.2. Hằng

1.5.2.1. Định nghĩa

Hằng là một đại lượng có giá trị không thay đổi trong quá trình thực thi chương trình.

Xem xét đoạn chương trình sau đây, để thấy sự cần thiết phải khai báo trong chương trình một số đại lượng có giá trị không thay đổi trong toàn bộ chương trình:

🕮 Ví dụ 1.27

Viết chương trình tính diện tính hình tròn, khi biết bán kính của đường tròn.

```
1  #include <stdio.h>
2  int main()
3  {
4      float r, pi, dt;
5      pi = 3.14159;
```

```
6       r = 12;
7       dt = pi * r * r;
8       printf("Dien tich = %f", dt);
9       return 0;
10  }
```

Trong ví dụ 1.27, chúng ta đã khai báo và sử dụng biến pi như một biến bình thường, chúng ta dễ dàng sửa đổi giá trị của pi ở bất kỳ đâu trong chương trình. Đối với những chương trình bé, số lượng dòng code ít (10 dòng), việc thay đổi giá trị pi có thể được kiểm soát và kiểm tra vị trí thay đổi. Nhưng với những chương trình lớn hơn, nếu ta vô tình thay đổi giá trị của pi, điều này sẽ ảnh hưởng khá nhiều đến các biểu thức phụ thuộc vào số pi, bởi vì pi là một hằng toán học có giá trị cố định.

Việc khai báo các hằng sẽ hạn chế người dùng thay đổi giá trị của nó. Trình biên dịch tìm kiếm sự thay đổi trong hằng và báo cáo lỗi nếu tìm thấy.

Chúng ta sẽ cải thiện chương trình ở ví dụ 1.27 bằng cách sử dụng một đại lượng có giá trị không thay đổi.

🕮 Ví dụ 1.28

```
1   #include <stdio.h>
2   int main()
3   {
4       const float PI = 3.14159;
5       float r, pi, dt;
6       PI = 3.14; // se gay ra loi khi bien dich
7       r = 12;
8       dt = PI * r * r;
9       printf("Dien tich = %f", dt);
10      return 0;
11  }
```

Tại thời điểm biên dịch sẽ phát hiện sự thay đổi của biến pi, chương trình sẽ báo lỗi *“Assignment of a read-only variable 'PI’”*.

Có thể định nghĩa hằng có bất kỳ kiểu dữ liệu nào. Các hằng hoạt động giống các biến thông thường, nhưng chúng chỉ được đọc ra, không thể sửa đổi. Ngôn ngữ C hỗ trợ hai kiểu định nghĩa hằng:

- Sử dụng từ khóa const.

- Sử dụng chỉ thị #define.

1.5.2.2. Khai báo hằng sử dụng từ khóa const

Một biến được khai báo với từ khóa const được ghi nhận là chỉ đọc. Trình biên dịch sẽ tìm kiếm sự thay đổi của biến này và báo cáo lỗi nếu tìm thấy.

Cú pháp:

const <data_type> <constant_name> = <constant_value>;

hoặc

<data_type> const <constant_name> = <constant_value>;

Trong đó:

data_type: kiểu dữ liệu của biến không thay đổi giá trị.

constant_name: tên biến không thay đổi giá trị.

constant_value: giá trị hằng.

const: từ khóa để khai báo biến không thay đổi giá trị.

🕮 Ví dụ 1.29

```
const float PI = 3.14159;
float const e = 2.71828;
```

1.5.2.3. Khai báo hằng sử dụng chỉ thị #define

#define là một chỉ thị tiền xử lý, được sử dụng để xác định các hằng và các marco. Nó xác định hằng số trong suốt thời gian biên dịch và đảm bảo 100 % là hằng số. Không giống như từ khóa const, chỉ thị #define không xác định bất kỳ biến nào và không tiêu tốn bộ nhớ. Thay vào đó, trong quá trình biên dịch, trình biên dịch sẽ thay thế tất cả các xuất hiện của hằng số bằng giá trị của nó.

Cú pháp:

#define <constant_name> <constant_value>

Trong đó: - #define: là chỉ thị tiền xử lý

- constant_name: Tên hằng

- constant_value: giá trị hằng

🕮 Ví dụ 1.30

Viết lại đoạn chương trình ở ví dụ 1.27, khai báo hằng.

```
1  #include <stdio.h>
2  #define float PI = 3.14159;
3  int main()
4  {
5     float r, pi, dt;
6     printf(“Nhap vao ban kinh hinh tron:”);
7     Scanf(“%f”, r);
8     dt = PI * r * r;
9     printf("Dien tich = %f", dt);
10    return 0;
11 }
```

#define PI 3.14159 xác định PI có giá trị không đổi, bằng 3.14159. Bộ xử lý thay thế tất cả các xuất hiện của PI trong chương trình bằng 3.14159 trước khi biên dịch.

Lưu ý: Tên hằng nên khai báo với tất cả các ký tự viết hoa.

1.5.2.4. Phân loại hằng

a. Hằng số nguyên

Là một số nguyên có giá trị trong khoảng từ -32768 đến 32767.

🕮 Ví dụ 1.31

-45,543 là các hằng số nguyên.

Lưu ý: Phân biệt cách viết 543 và 543.0. Số 543 là hằng số nguyên còn 543.0 là hằng số thực.

Một hằng số nguyên có thể được biểu diễn như sau:

- Theo cách thông thường: 2019, 12, v.v. tức là số được biểu diễn ở hệ cơ số 10;

- Theo cơ số 8: Thêm số 0 vào đằng trước số 017 (8) = 15 (10);

- Theo cơ số 16: Thêm 0x ở đầu số: 0x1A (16) = 26 (10);

- Kiểu char cũng được xem như là một hằng nguyên.

🕮 Ví dụ 1.32

Câu lệnh printf(“so = %d”, ‘A’ + 5); sẽ cho ra kết quả là so = 70.

Câu lệnh printf(“chu = %c”, ‘A’+ 2); sẽ cho ra kết quả là chu = ‘C’.

b. Hằng số thực

Hằng số thực được viết theo hai cách:

Cách 1: Theo dạng thập phân. Số thực gồm phần nguyên, dấu chấm thập phân và phần thập phân.

🕮 Ví dụ 1.33

Các hằng số thực: 214.55, 12.0

Cách 2: Theo dạng mũ. Số thực được tách gồm hai phần là định trị và bậc. Phần định trị là một số nguyên hoặc số thực được viết dưới dạng thập phân, phần bậc là một số nguyên biểu diễn số mũ của cơ số 10, hai phần này cách nhau bởi ký tự e hoặc E.

🕮 Ví dụ 1.34

0.12E3 biểu diễn giá trị 0.12*103 = 120.0.

1.23e - 2 biểu diễn giá trị 1.23 * 10^- 2 = 0.0123.

c. Hằng ký tự

Là một ký tự được viết trong hai dấu nháy đơn.

🕮 Ví dụ 1.35

‘a’ là một hằng ký tự. Giá trị của ‘a’ chính là mã của ký tự ‘a’ trong bảng mã ASCII. Như vậy giá trị của “a” là 97.

Đối với một số hằng ký tự đặc biệt, ta cần sử dụng cách viết sau (thêm dấu \ vào trước ký tự đó):

Bảng 1.4. Hằng ký tự đặc biệt

Cách viết	Ký tự
‘\’’	’
‘\”’	”
‘\\’	\
‘\n’	xuống dòng
‘\0’	\0 (null)
‘\t’	Tab
‘\b’	Backspace
‘\r’	Về đầu dòng
‘\f’	Sang trang

d. Hằng ký tự

Là một dãy ký tự bất kỳ đặt trong hai dấu nháy kép.

🕮 Ví dụ 1.36

Hằng chuỗi ký tự "Ha Noi";

" " /*Chuỗi rỗng*/.

Chuỗi ký tự được lưu trữ trong máy dưới dạng một mảng có các phần tử là các ký tự riêng biệt. Trình biên dịch tự động thêm ký tự '\0' vào cuối mỗi chuỗi (ký tự \0 được xem là dấu hiệu kết thúc của một chuỗi ký tự).

Lưu ý: Phân biệt giữa hằng ký tự và hằng chuỗi ký tự: 'A' là hằng ký tự khác với "A" là hằng chuỗi ký tự.

1.6. BIỂU THỨC VÀ PHÉP TOÁN

1.6.1. Toán tử gán

Làm thế nào để gán giá trị cho các biến? Ngôn ngữ C cung cấp một toán tử gán cho mục đích này, gán giá trị cho một biến bằng toán tử gán được gọi là một câu lệnh gán trong C. Chức năng của toán tử này là gán các giá trị hoặc giá trị trong các biến ở phía bên phải của một biểu thức cho các biến ở phía bên trái.

Cú pháp của biểu thức gán:

Variable = constant/variable/expression;

Trong đó:

Variable: biến

Constant: hằng, giá trị không thay đổi

Exprssion: biểu thức

Lưu ý:

Kiểu dữ liệu của biến ở phía bên trái phải khớp với kiểu dữ liệu của hằng/ biến / biểu thức ở phía bên phải với một vài ngoại lệ trong đó có thể chuyển đổi loại tự động.

🕮 Ví dụ 1.37

b = c; // biến b được gán giá trị của biến c;

a = 9; // biến a được gán giá trị bằng 9;

b = c + 5; // biến b được gán giá trị bằng biểu thức c + 5.

1.6.2. Ép kiểu dữ liệu

Trong C cũng như một số ngôn ngữ lập trình khác, trong một biểu thức thì các toán hạng phải cùng kiểu. Tuy nhiên, trong thực tế thì không

thể cứng nhắc như vậy, chẳng hạn cộng một số nguyên với một số thực rõ ràng là khác kiểu nhưng bạn vẫn có thể thực hiện được. Thực ra thì trước khi thực hiện toán tử cộng đó chương trình dịch đã thực hiện thao tác chuyển đổi kiểu của số nguyên thành số thực chúng ta gọi là phép chuyển kiểu (ép kiểu). Trong một số tình huống việc chuyển kiểu trong C có thể được chương trình dịch thực hiện tự động (gọi là ép kiểu tự động) hoặc được ép kiểu kiểu tường minh (người lập trình viết câu lệnh - toán tử chuyển kiểu).

Nói chung sự chuyển kiểu tự động xảy ra trong bốn trường hợp sau:

- Trong một biểu thức: các toán hạng có kiểu khác nhau.

- Gán một biểu thức vào một biến khác kiểu.

- Truyền tham số thực sự khác kiểu với tham số hình thức.

- Giá trị trả về của hàm sau câu lệnh return khác với kiểu hàm được khai báo.

Trong trường hợp thứ nhất quy tắc chuyển kiểu từ thấp lên cao được áp dụng, tức là toán hạng có kiểu thấp hơn sẽ được tự động chuyển thành kiểu của toán hạng cao hơn theo trật tự:

char → int → long → float → double

Trong ba trường hợp cuối kiểu của giá trị vế phải được chuyển theo kiểu của biến bên trái, kiểu các tham số thực sự được chuyển theo kiểu của tham số hình thức, kiểu giá trị trả về (sau return) phải chuyển thành kiểu của hàm.

Lưu ý: Chuyển kiểu giá trị tức thời của toán hạng rồi thực hiện phép toán chứ kiểu của bản thân toán hạng thì không thay đổi.

Trong một số yêu cầu chúng ta cần sự chuyển kiểu rõ ràng (ép kiểu tường minh) chứ không sử dụng quy tắc chuyển kiểu ngầm định, trong trường hợp này bạn có thể sử dụng toán tử chuyển kiểu theo cú pháp sau:

(new_data_type) (expression);

Trong đó: new_data_type: kiểu_dữ_liệu_mới là tên một kiểu hợp lệ nào đóvà giá trị của (expression) trả về bắt buộc phải chuyển thành (kiểu_dữ_liệu_mới).

🕮 Ví dụ 1.38

```
int a = 5, b  = 2;
float c;
```

c = (float) a /b; thì c có giá trị = 2.5.

Trong trường hợp không ép kiểu thì c = a/b ; cho kết quả c = 2.0

🕮 Ví dụ 1.39

float a = 7.0;

int b;

b = (int) (a/3);

Kết quả thu được b = 2.

1.6.3. Biểu thức

1.6.3.1. Định nghĩa biểu thức

Biểu thức là sự kết hợp giữa các phép toán và các toán hạng, dùng để diễn đạt một công thức toán học nào đó. Toán hạng có thể là hằng, biến, phần tử mảng, các hàm, v.v. Các phép toán bao gồm các phép toán số học (+, - ,*, /), phép toán so sánh (>, <, =, !=, ≥, ≤, v.v.) và phép toán logic (and, or, not, v.v.).

Biểu thức được phân loại theo kiểu giá trị: nguyên, thực và biểu thức logic. Khi viết biểu thức nên dùng các dấu ngoặc tròn để thể hiện đúng trình tự tính toán trong biểu thức.

Biểu thức thường được dùng trong:

- Vế phải của câu lệnh gán;

- Làm tham số thực sự của hàm;

- Làm chỉ số trong các câu lệnh if, switch, for, while, do…while.

🕮 Ví dụ 1.40

Biểu thức tính nửa chu vi và diện tích của tam giác

p = (a + b + c)/2;

s = sqrt((p-a)*(p-b)*p-c)); (trong đó a, b, c là 3 biến số thực).

1.6.3.2. Biểu thức số học

Biểu thức với các toán tử là phép toán số học gọi là biểu thức số học.

🕮 Ví dụ 1.41

Cho đoạn chương trình sau đây:

```
1  #include <stdio.h>
2  int main ()
3  { float a, b, c x, y, z;
```

```
4  a = 9;
5  b = 12;
6  c = 3;
7  x = a - b / 3 + c * 2 - 1; // tinh gia tri cua bien x
8  y = a - b / (3 + c) * (2 - 1); // tinh gia tri cua y
9  z = a - ( b / (3 + c) * 2) - 1; //tinh gia tri cua z
10 printf ("x = %.2f",x); // in ra gia cua x ra man hinh
11 printf ("y = %.2f",y); // in ra gia tri y ra man hinh
12 printf ("z = %.2f",z); // in gia tri cua z ra man hinh
13 return 0;
14 }
```

Kết quả chương trình:

```
x = 10.00
y = 7.00
z = 4.00
```

Giải thích:

Dòng lệnh 7: Vế trái là biến x, là biến đang cần được xác định giá trị. Vế phải là một biểu thức (a - b / 3 + c * 2 - 1). Tính giá trị của biểu thức dựa vào giá trị đã gán a = 9, b = 12, c = 3. Kết quả thu được, giá trị của biểu thức (a - b / 3 + c * 2 - 1) = 9 - 12/3 + 3*2 - 1 = 10.00. Như vậy, câu lệnh gán x = a - b / 3 + c * 2 - 1 cho kết quả x = 10.00.

Dòng lệnh 8: Vế trái là biến y, cần tính giá trị. Vế phải là biểu thức (a - b / (3 + c) * (2 - 1)). Biểu thức bên vế phải được tính giá trị với a = 9, b = 12, c = 3. Kết quả thu được, giá trị của biểu thức = 7.00. Do đó kết quả in ra màn hình ở câu lệnh 11 là y = 7.00

Tương tự với dòng lệnh 9: Tính giá trị biểu thức (a - (b / (3 + c) * 2) - 1) với a = 9, b = 12, c = 3, kết quả của biểu thức = 4.00 được gán cho biến z và in ra màn hình ở dòng lệnh 12 là z = 4.00.

1.6.3.3. Biểu thức logic

Biểu thức với các phép toán logic và phép toán so sánh được gọi là biểu thức logic.

🕮 Ví dụ 1.42

Biểu thức ((a + b > c) && (b + c > a) && (a + c > b) && (a >0) && (b > 0) && (c > 0)) là một biểu thức logic với phép && (and)

Biểu thức ((a == b) || (b == c) || (c == a)) là một biểu thức logic với phép || (or).

Các biểu thức logic này có thể được sử dụng trong câu lệnh if, câu lệnh do … while hoặc câu lệnh while … do.

1.6.3.4. Biểu thức điều kiện

Cú pháp:

Expression_1 ? expression_2: expression_3

Trong đó:

- expression_1: là biểu thức logic, nhận giá trị là T (True) hoặc F (False).

- expression_2, expression_3: là các biểu thức tương ứng nhận được khi expression_1 đúng hoặc sai.

Chức năng: Giá trị của biểu thức điều kiện bằng giá trị của biểu thức expression_2 nếu expression_1 nhận giá trị T (True) hoặc bằng giá trị của biểu thức expression_3 nếu expression_2 nhận giá trị F (False).

🕮 Ví dụ 1.43

Cho biểu thức điều kiện sau:

(a > b) ? a: b;

Biểu thức cho kết quả là số lớn nhất trong hai số a, b. Nếu a > b đúng thì kết quả nhận được là a, ngược lại nếu a > b sai, thì kết quả nhận được là b.

1.6.4. Các toán tử

Các toán tử trong ngôn ngữ C là các ký hiệu được sử dụng để thực hiện các thao tác toán học hoặc logic. Các toán tử tham gia vào một chương trình để thao tác dữ liệu, cùng với các biến, hằng và tạo thành một phần của các biểu thức toán học hoặc biểu thức logic.

Các kiểu toán tử trong C:

- Toán tử số học.
- Toán tử quan hệ.
- Toán tử logic.
- Toán tử so sánh bit.
- Toán tử gán.

- Toán tử hỗn hợp.

1.6.4.1. Toán tử số học

Bảng 1.5. Toán tử số học

Phép toán	Mô tả	Biểu diễn
+	Cộng	a + b
-	Trừ	a - b
*	Nhân	a * b
/	Chia lấy phần nguyên	a / b
%	Chia lấy dư	a % b

🕮 Ví dụ 1.44

Xem xét đoạn chương trình sau đây:

```
1  #include <stdio.h>
2  int main () {
3     /* Khai bao bien cuc bo: */
4     int a, b;
5     /* Khoi tao gia tri */
6     a = 20;
7     b = 7;
8     printf("% d + % d  = %d\n", a, b, a + b);
9     printf("% d - % d  = %d\n", a, b, a - b);
10    printf("% d * % d  = %d\n", a, b, a * b);
11    printf("% d / % d  = %.2f\n", a, b, (float) a / b);
12    printf("% d %% % d  = %d\n", a, b, a % b);
13    return 0;
14 }
```

✍Kết quả chương trình

```
20 +  7  = 27
20 -  7  = 13
20 *  7  = 140
20 /  7  = 2.86
20 %  7  = 6
```

Nhận xét:

- Có phép toán một ngôi -

Chẳng hạn, có - (a + b) nhưng không có phép toán + (a +b).

- Phép chia hai số nguyên sẽ cho kết quả bỏ đi phần thập phân.

Chẳng hạn, 11/3 = 3. Muốn có kết quả là số thực có cả phần thập phân, ta phải ép kiểu kết quả: (float) 11/3.

- Phép % cho phần dư của phép chia nguyên, không áp dụng cho các giá trị kiểu float và double.

1.6.4.2. Toán tử quan hệ

Kết quả trả về chỉ nhận một trong hai giá trị: T (True) hoặc F (False).

Bảng 1.6. Toán tử quan hệ

Toán tử	Mô tả	Biểu diễn
==	Kiểm tra giá trị của hai toán hạng có bằng nhau hay không? Nếu đúng, thì kết quả trả về là T.	A == B
!=	Kiểm tra giá trị của hai toán hạng có bằng nhau hay không? Nếu sai, thì kết quả trả về là F.	A!=B
>	Kiểm tra giá trị của toán hạng bên trái có lớn hơn giá trị của toán hạng bên phải hay không? Nếu có, điều kiện trở thành T.	A > B
<	Kiểm tra giá trị của toán hạng bên trái nhỏ hơn giá trị của toán hạng bên phải. Nếu có, điều kiện trở thành T.	A < B
>=	Kiểm tra giá trị của toán hạng bên trái có lớn hơn hoặc bằng giá trị của toán hạng bên phải hay không? Nếu có, điều kiện trở thành T.	A >= B
<=	Kiểm tra giá trị của toán hạng bên trái nhỏ hơn hoặc bằng giá trị của toán hạng bên phải. Nếu có, điều kiện trở thành T	A <= B

🕮 Ví dụ 1.45

Cho đoạn chương trình sau đây:

```
1  #include <stdio.h>
2  int main()
3  {
4    int a = 2, b = 2, c = 3;
5    //kq = 1: Dung, kq = 0: Sai
6    printf("1: Dung - 0: Sai\n");
7    printf("Phep so sanh %d == %d cho kq = %d \n", a, b, a == b);
8    printf("Phep so sanh %d == %d cho kq = %d \n", a, c, a == c);
9    printf("Phep so sanh %d > %d cho kq = %d \n", a, b, a > b);
10   printf("Phep so sanh %d > %d cho kq = %d \n", a, c, a > c);
11   printf("Phep so sanh %d < %d cho kq = %d \n", a, b, a < b);
12   printf("Phep so sanh %d < %d cho kq = %d \n", a, c, a < c);
13   printf("Phep so sanh %d != %d cho kq = %d \n", a, b, a != b);
14   printf("Phep so sanh %d != %d cho kq = %d \n", a, c, a != c);
15   printf("Phep so sanh %d >= %d cho kq = %d \n", a, b, a >= b);
16   printf("Phep so sanh %d >= %d cho kq = %d \n", a, c, a >= c);
17   printf("Phep so sanh %d <= %d cho kq = %d \n", a, b, a <= b);
18   printf("Phep so sanh %d <= %d cho kq = %d \n", a, c, a <= c);
19   return 0;
20 }
```

✍Kết quả chương trình:

```
1: Dung - 0: Sai
Phep so sanh 2 == 2 cho kq = 1
Phep so sanh 2 == 3 cho kq = 0
Phep so sanh 2 > 2 cho kq = 0
Phep so sanh 2 > 3 cho kq = 0
Phep so sanh 2 < 2 cho kq = 0
Phep so sanh 2 < 3 cho kq = 1
Phep so sanh 2 != 2 cho kq = 0
Phep so sanh 2 != 3 cho kq = 1
Phep so sanh 2 >= 2 cho kq = 1
```

Phep so sanh 2 >= 3 cho kq = 0
Phep so sanh 2 <= 2 cho kq = 1
Phep so sanh 2 <= 3 cho kq = 1

1.6.4.3. Toán tử logic

Một biểu thức chứa toán tử logic trả về 0 hoặc 1 tùy thuộc vào kết quả biểu thức đúng hay sai. Toán tử logic thường được sử dụng trong việc ra quyết định trong lập trình C.

Bảng 1.7. Toán tử logic

Toán tử	Mô tả	Biểu diễn
&&	Toán tử AND, cho kết quả T khi tất cả các toán hạng là T	A&&B
||	Toán tử OR, cho kết quả T khi một trong các toán hạng là T	A||B
!	Toán tử NOT, cho kết quả T khi toán hạng là F	NOT A

🕮 Ví dụ 1.46

Cho chương trình sau đây:

```
1   #include <stdio.h>
2   int main()
3   {
4    int a = 2, b = 2, c = 3, kq;
5    printf("a = %d  b = %d c = %d\n", a, b, c);
6    printf("1: Dung - 0: Sai\n");
7    kq = (a == b) && (c > b);
8    printf("Ket qua cua phep (a == b) && (c > b) la = %d \n", kq);
9    kq = (a == b) && (c < b);
10   printf("Ket qua cua phep (a == b) && (c < b) la %d \n", kq);
11   kq = (a == b) || (c < b);
12   printf("Ket qua cua phep (a == b) || (c < b) la %d \n", kq);
13   kq = (a != b) || (c < b);
14   printf("Ket qua cua phep (a != b) || (c < b) la %d \n", kq);
15   kq = !(a != b);
```

```
16    printf("Ket qua cua phep !(a == b) la %d \n", kq);
17    kq = !(a == b);
18    printf("Ket qua cua phep !(a == b) la %d \n", kq);
19    return 0;
20   }
```

✍Kết quả chương trình

```
a = 2  b = 2  c = 3
1: Dung - 0: Sai
Ket qua cua phep (a == b) && (c > b) la = 1
Ket qua cua phep (a == b) && (c < b) la 0
Ket qua cua phep (a == b) || (c < b) la 1
Ket qua cua phep (a != b) || (c < b) la 0
Ket qua cua phep !(a == b) la 1
Ket qua cua phep !(a == b) la 0
```

1.6.4.4. Toán tử tăng giảm

Có hai toán tử tăng ++ và giảm -- để thay đổi giá trị của toán hạng (hằng hoặc biến) 1 đơn vị.

Toán tử ++ tăng giá trị lên 1 trong khi toán tử -- giảm giá trị xuống 1.

Hai toán tử này là toán tử đơn nguyên, nghĩa là chúng chỉ hoạt động trên một toán hạng duy nhất.

Bảng 1.8. Toán tử tăng giảm

Toán tử	Mô tả
i++	i = i + 1, i được đưa vào thực hiện biểu thức trước, sau đó mới tăng i lên.
++i	i = i + 1, i được tăng trước, sau đó sẽ lấy kết quả để thực hiện biểu thức.
i--	i = i - 1, i được đưa vào thực hiện biểu thức trước, sau đó mới giảm i.
--i	i = i - 1, i được giảm trước, sau đó lấy kết quả để thực hiện biểu thức.

🕮 Ví dụ 1.47

Xem xét đoạn chương trình sau đây, để thấy sự khác nhau giữa ++i và i++, -- i và i--

✍Chương trình

```
1   #include <stdio.h>
2   int main()
3   {
4       int a = 10, b = 7;
5       int c = 12, d = 9;
6       printf("++a = %d \n", ++a);
7       printf("--b = %d \n", --b);
8       printf("c++ = %d \n", c++);
9       printf("d-- = %d \n", d--);
10      printf("Ket qua sau khi thuc hien tang, giam\n");
11      printf("a = %d, b = %d, c = %d, d = %d", a, b, c, d) ;
12      return 0;
13  }
```

✍Kết quả chương trình

```
++a = 11
--b = 6
c++ = 12
d-- = 9
Ket qua sau khi thuc hien tang, giam
a = 11, b = 6, c = 13, d = 8
```

✍Giải thích chương trình:

Dòng lệnh 6: printf("++a = %d \n", ++a); in ra giá trị của ++a; do đó giá trị được in ra là 11 (vì a được tăng trước).

Dòng lệnh 7: printf("--b = %d \n", --b); in ra giá trị của -b; do đó giá trị được in ra là 6 (vì b bị giảm trước).

Dòng lệnh 8: printf("c++ = %d \n", c++); in ra giá trị của c++, do đó giá trị được in ra là 12 (c được đưa vào thực hiện trước, sau đó mới tăng c, tại thời điểm của dòng lệnh 8, giá trị c chưa tăng).

Dòng lệnh 9: printf("d-- = %d \n", d--); in ra giá trị của d--, giá trị được in ra là 9 (d được đưa vào thực hiện trước, sau đó mới giảm d, tại thời điểm của dòng lệnh 9, giá trị d chưa giảm)

Dòng lệnh 11: printf("a = %d, b = %d, c = %d, d = %d", a, b, c, d); lúc này các biến a, b, c, d đã được tăng hoặc giảm giá trị, do đó, kết quả in ra là a = 11, b = 6, c = 13, d = 8.

1.6.4.5. Toán tử gán

Toán tử gán được sử dụng để gán giá trị cho một biến.

Bảng 1.9. Toán tử gán

Toán tử	Mô tả	Biểu diễn
=	a = b	a = b
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a*= b	a = a * b
/=	a /= b	a = a/b
%=	a %=b	a = a % b

🕮 Ví dụ 1.48

✍Cho đoạn chương trình sau đây:

```
1  #include <stdio.h>
2  int main()
3  {
4      int a = 10, c;
5      c = a;      // c = 10
6      printf("c = %d\n", c);
7      c += a;     // c = 20
8      printf("c = %d\n", c);
9      c -= a;     // c = 10
10     printf("c = %d\n", c);
11     c *= a;     // c = 100
12     printf("c = %d\n", c);
13     c /= a;     // c = 10
14     printf("c = %d\n", c);
```

```
15      c %= a;     // c = 0
16      printf("c = %d\n", c);
17      return 0;
18  }
```

1.6.4.6. Toán tử bit

Trong quá trình tính toán, các phép toán như: cộng, trừ, nhân, chia, v.v được chuyển thành cấp độ bit giúp xử lý nhanh hơn và tiết kiệm điện năng. Toán tử bit được sử dụng trong lập trình C để thực hiện các phép toán bit

Bảng 1.10. Các toán tử bit

Toán tử	Nghĩa của toán tử
&	AND bit
|	OR bit
⋀	XOR bit
~	Đảo bit
≪	Dịch trái
≫	Dịch phải

a. AND bit

Đầu ra của AND bit là 1 nếu các bit tương ứng của hai toán hạng là 1. Nếu một bit của toán hạng là 0, kết quả của bit tương ứng được ước tính thành 0.

```
1 & 1 = 1       1 & 0 = 0
0 & 1 = 0       0 & 0 = 0
```

🕮 Ví dụ 1.49

Chúng ta hãy tìm hiểu cách tính AND bit của hai số nguyên 9 và 15.

9_{10} = 00001001_2

15_{10} = 00001111_2

Toán tử AND bit của 16 và 15

9_{10} = 00001001_2

& 15_{10} = 00001111_2

00001001_2 = 9 (Trong hệ thập phân)

🕮 Ví dụ 1.50

Xét chương trình sau đây:

```
1    #include <stdio.h>
2    int main()
3    {
4        int a = 9, b = 15;
5        printf("Vi du ve AND bit:\n")
6        printf("Ket qua = %d", a&b);
7        return 0;
8    }
```

✍Kết quả chương trình

```
Vi du ve AND bit:
Ket qua = 9
```

b. OR bit

Đầu ra của OR bit là 1 nếu ít nhất một bit tương ứng của hai toán hạng là 1. Trong ngôn ngữ C, toán tử OR bit được ký hiệu là |.

1 | 1 = 1 1 | 0 = 1 0 | 1 = 1 0 | 0 = 0

🕮 Ví dụ 1.51

Chúng ta hãy tìm hiểu cách tính OR bit của hai số nguyên 9 và 15.

9_{10} = 00001001_2

15_{10} = 00001111_2

Toán tử OR bit của 16 và 30

9_{10} = 00001001_2

| 15_{10} = 00001111_2

00001111_2 = 15 (Trong hệ thập phân)

🕮 Ví dụ 1.52

Xét chương trình sau đây:

```
1    #include <stdio.h>
2    int main()
```

```
3   {
4       int a = 9, b = 15;
5       printf("Vi du ve OR bit:\n")
6       printf("Ket qua = %d", a|b);
7       return 0;
8   }
```

✍**Kết quả chương trình**

```
Vi du ve OR bit:
Ket qua = 15
```

c. XOR bit

Kết quả của toán tử XOR bit là 1 nếu các bit tương ứng của hai toán hạng ngược nhau. Nó được ký hiệu là ^.

```
1 ^ 1 = 0      1 ^ 0 = 1
0 ^ 1 = 1      0 ^ 0 = 0
```

🕮 **Ví dụ 1.53**

Chúng ta hãy tìm hiểu cách tính XOR bit của hai số nguyên 9 và 15.

9_{10} = 00001001_2

15_{10} = 00001111_2

Toán tử XOR bit của 16 và 30

9_{10} = 00001001_2

| 15_{10} = 00001111_2

00000110_2 = 15 (Trong hệ thập phân)

🕮 **Ví dụ 1.54**

✍Xét chương trình sau đây:

```
1   #include <stdio.h>
2   int main()
3   {
4       int a = 9, b = 15;
```

```
5       printf("Vi du ve XOR bit:\n")
6       printf("Ket qua = %d", a^b);
7       return 0;
8   }
```

✍**Kết quả chương trình**

```
Vi du ve XOR bit:
Ket qua = 6
```

d. Đảo bit ~

Toán tử đảo bit là toán tử đơn nguyên (chỉ hoạt động trên một toán hạng). Nó thay đổi 1 thành 0 và 0 thành 1 và được ký hiệu là ~.

```
~ 1 = 0          ~ 0 = 1
```

Lưu ý:

Với số nguyên n bất kỳ, ~n = - (n + 1).

Để có công thức này, chúng ta cần dựa trên phép toán bù 2. Bù 2 là một phép toán trên số nhị phân. Bù 2 của một số nguyên được tính bằng - (số đảo bit của số đó cộng với 1).

Bảng 1.11. Bù 2 của một số nguyên

Số thập phân	Nhị phân	Bù 2
0	00000000	- (1111111 +1) = -00000000 = - 0 (thập phân)
1	00000001	- (1111110 + 1) = -11111111 = -256 (thập phân)
12	00001100	- (11110011 +1) = -11110100 = -244 (thập phân)
220	11011100	- (00100011 +1) = -00100100 = -36 (thập phân)

🕮 **Ví dụ 1.55**

35 = 00100011 (Hệ nhị phân)

Xác định đảo bit của 35

~ 00100011

= 11011100 = 220 (Hệ thập phân) = - 36 (Bù 2)

🕮 **Ví dụ 1.56**

✍Cho chương trình sau đây:

```
1   #include <stdio.h>
```

```
2   int main()
3   {
4       printf("Ket qua ~35  = %d\n",~35);
5       printf("Ket qua ~-12 = %d\n",~-12);
6       return 0;
7   }
```

↘Kết quả chương trình

```
Ket qua ~35 = -36
Ket qua ~-12 = 11
```

e. Dịch bit

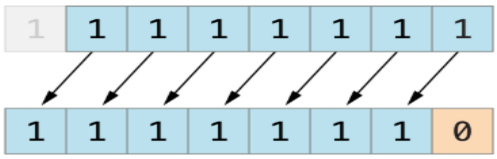
Có hai phép toán dịch bit trong C:

- Dịch trái bit;

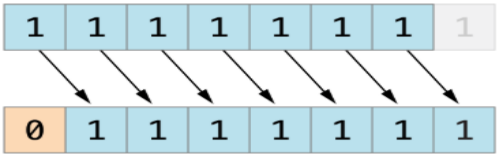
- Dịch phải bit.

Toán tử dịch trái dịch chuyển tất cả các bit về phía trái theo số bit nhất định. Nó được ký hiệu là <<.

Toán tử dịch phải dịch chuyển tất cả các bit về phía bên phải theo số bit nhất định. Nó được ký hiệu là >>.



Hình 1.3. Dịch trái 1 bit



Hình 1.4. Dịch phải 1 bit

🕮 Ví dụ 1.57

$212_{10} = 11010100_2$.

212>>2 = 00110101 [Dịch phải 2 bit].

212>>7 = 00000001 (Dịch phải 7 bit).

212>>8 = 00000000 (Dịch phải 8 bit).

212>>0 = 11010100 (Không dịch).

212_{10} = 11010100.

212<<1 = 110101000 [Dịch trái 1 bit].

212<<0 =11010100 (Không dịch, hoặc dịch bởi 0 bit).

212<<4 = 110101000000 = 3392 (Hệ thập phân).

🕮 Ví dụ 1.58

✍Cho đoạn chương trình sau đây:

```
1   #include <stdio.h>
2   int main()
3   {
4      int num=212, i;
5      for (i=0; i<=2; ++i)
6         printf("Dich phai %d bit: %d\n", i, num>>i);
7      printf("\n");
8      for (i=0; i<=2; ++i)
9         printf("Dich trai %d bit: %d\n", i, num<<i);
10     return 0;
11  }
```

✍Kết quả chương trình

```
Dich phai 0 bit: 212
Dich phai 1 bit: 106
Dich phai 2 bit: 53
Dich trai 0 bit: 212
Dich trai 1 bit: 424
Dich trai 2 bit: 848
```

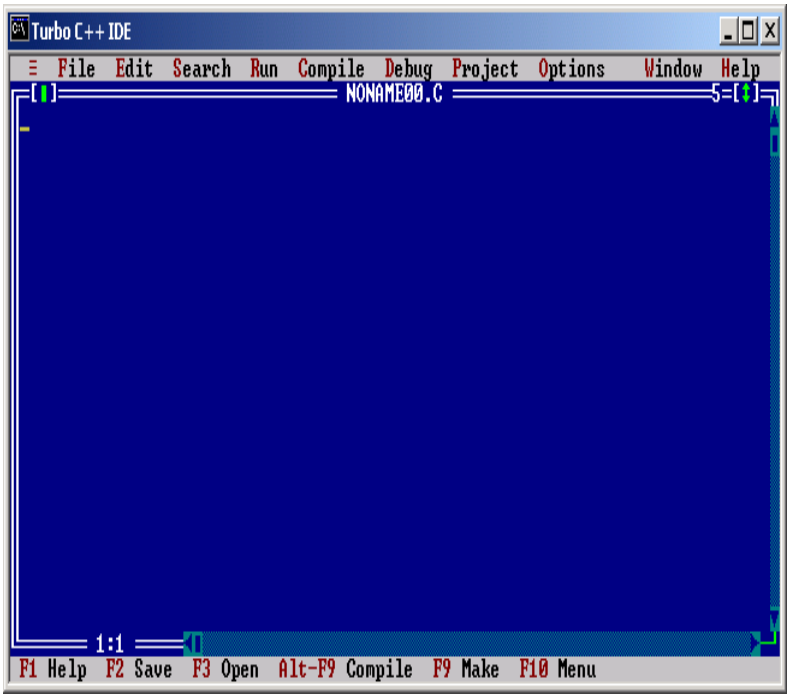
1.7. MỘT SỐ TOOL SỬ DỤNG

1.7.1. Turbo C

Turbo C là môi trường hỗ trợ lập trình C do hãng Borland cung cấp. Môi trường này cung cấp các chức năng như: soạn thảo chương trình, dịch, thực thi chương trình, v.v. Phiên bản được sử dụng ở đây là Turbo C 3.0.

1.7.1.1. Gọi Turbo C

Chạy Turbo C cũng giống như chạy các chương trình khác trong môi trường DOS hay Windows, màn hình sẽ xuất hiện menu của Turbo C có dạng như sau:



Hình 1.5. Giao diện màn hình của Turbo C

1.7.1.2. Soạn thảo chương trình mới

Muốn soạn thảo một chương trình mới ta chọn mục New trong menu File (File/New). Trên màn hình sẽ xuất hiện một vùng trống để cho ta soạn thảo nội dung của chương trình. Trong quá trình soạn thảo chương trình ta có thể sử dụng các phím sau:

Các phím xem thông tin trợ giúp:

- F1: Xem toàn bộ thông tin trong phần trợ giúp.

- Ctrl-F1: Trợ giúp theo ngữ cảnh (tức là khi con trỏ đang ở trong một từ nào đó, chẳng hạn int mà bạn gõ phím Ctrl-F1 thì bạn sẽ có được các thông tin về kiểu dữ liệu int).

Các phím di chuyển con trỏ trong vùng soạn thảo chương trình:

Bảng 1.12. Các phím di chuyển con trỏ

Phím	**Ý nghĩa**	**Phím tắt (tổ hợp phím)**
Enter	Đưa con trỏ xuống dòng	
Mũi tên đi lên	Đưa con trỏ lên hàng trước	Ctrl-E
Mũi tên đi xuống	Đưa con trỏ xuống hàng sau	Ctrl-X

Mũi tên sang trái	Đưa con trỏ sang trái một ký tự	Ctrl-S
Mũi tên sang phải	Đưa con trỏ sang phải một ký tự	Ctrl-D
End	Đưa con trỏ đến cuối dòng	
Home	Đưa con trỏ đến đầu dòng	
PgUp	Đưa con trỏ lên trang trước	Ctrl-R
PgDn	Đưa con trỏ xuống trang sau	Ctrl-C
	Đưa con trỏ sang từ bên trái	Ctrl-A
	Đưa con trỏ sang từ bên phải	Ctrl-F

Các phím xoá ký tự/dòng:

Bảng 1.13. Các phím xóa ký tự

Phím	Ý nghĩa	Phím tắt
Delete	Xoá ký tự tại ví trí con trỏ	Ctrl-G
BackSpace	Di chuyển sang trái đồng thời xoá ký tự đứng trước con trỏ	Ctrl-H
	Xoá một dòng chứa con trỏ	Ctrl-Y
	Xóa từ vị trí con trỏ đến cuối dòng	Ctrl-Q-Y
	Xóa ký tự bên phải con trỏ	Ctrl-T

Các phím chèn ký tự/dòng:

Bảng 1.14. Các phím chèn ký tự/dòng

Insert	Thay đổi viết xen hay viết chồng
Ctrl-N	Xen một dòng trống vào trước vị trí con trỏ

Sử dụng khối:

Bảng 1.15. Tổ hợp phím xử lý khối

Phím tắt	Ý nghĩa
Ctrl-K-B	Đánh dấu đầu khối
Ctrl-K-K	Đánh dấu cuối khối
Ctrl-K-C	Chép khối vào sau vị trí con trỏ
Ctrl-K-V	Chuyển khối tới sau vị trí con trỏ
Ctrl-K-Y	Xoá khối
Ctrl-K-W	Ghi khối vào đĩa như một tập tin
Ctrl-K-R	Đọc khối (tập tin) từ đĩa vào sau vị trí con trỏ
Ctrl-K-H	Tắt/mở khối
Ctrl-K-T	Đánh dấu từ chứa con trỏ
Ctrl-K-P	In một khối

Các phím, phím tắt thực hiện các thao tác khác:

Bảng 1.16. Một số phím khác

Phím	Ý nghĩa	Phím tắt
F10	Kích hoạt menu chính	Ctrl + K + D, Ctrl + K + Q
F2	Lưu chương trình đang soạn vào đĩa	Ctrl + K + S
F3	Tạo tập tin mới	
Tab	Di chuyển con trỏ một khoảng đồng thời đẩy dòng văn bản	Ctrl + I
ESC	Hủy bỏ thao tác lệnh	Ctrl + U
	Đóng tập tin hiện tại	Alt + F3
	Hiện hộp thoại tìm kiếm	Ctrl + Q +F
	Hiện hộp thoại tìm kiếm và thay thế	Ctrl + Q + A
	Tìm kiếm tiếp tục	Ctrl + L

🕮 Ví dụ 1.59

Soạn thảo đoạn chương trình sau, sau đó ghi chương trình với tên là CHAO.C

```
1   #include <stdio.h>
2   #include<conio.h>
3   int main ()
4   {
5   char ten[50];
6   printf("Xin cho biet ten cua ban !");
7   scanf("%s",ten);
8   printf("Xin chao ban %s",ten);
9   getch();
10  return 0;
11  }
```

1.7.1.3. Ghi chương trình đang soạn thảo vào đĩa

Sử dụng File/Save hoặc gõ phím F2. Có hai trường hợp xảy ra:

- Nếu chương trình chưa được ghi lần nào thì một hội thoại sẽ xuất hiện cho phép bạn xác định tên tập tin (FileName). Tên tập tin phải tuân

thủ quy cách đặt tên của DOS và không cần có phần mở rộng (sẽ tự động có phần mở rộng là .C hoặc .CPP). Sau đó gõ phím Enter.

- Nếu chương trình đã được ghi một lần rồi thì nó sẽ ghi những thay đổi bổ sung lên tập tin chương trình cũ.

Lưu ý:

Để đề phòng mất điện trong khi soạn thảo chương trình bạn nên gõ phím F2.

Quy tắc đặt tên tập tin của DOS:

Tên của tập tin gồm 2 phần: Phần tên và phần mở rộng.

Phần tên của tập tin phải bắt đầu là 1 ký tự từ a...z (không phân biệt hoa thường), theo sau có thể là các ký tự từ a…z, các ký số từ 0...9 hay dấu gạch dưới (_), phần này dài tối đa là 8 ký tự.

Phần mở rộng: phần này dài tối đa 3 ký tự.

1.7.1.4. Thực hiện chương trình

Để dịch chương trình ta dùng chức năng Compile trên menu Compile hoặc nhấn tổ hợp phím Alt + F9, chức năng này có tác dụng dịch chương trình gốc và liên kết với các hàm thư viện nhằm tạo ra tệp chương trình thực hiện có đuôi là EXE. Các sai sót (error) và cảnh báo (warning) nếu có sẽ được chỉ ra. Nếu có error hoặc warning thì ta cần trở lại màn hình soạn thảo để sửa chữa chương trình. Trường hợp biên dịch không còn lỗi, ta nhận được một chương trình đúng, nó đượcc ghi lên đĩa dưới dạng một tệp có tên là: CHAO.EXE. Đến đây quá trình dịch xem như được hoàn thành.

1.7.1.5. Chạy chương trình

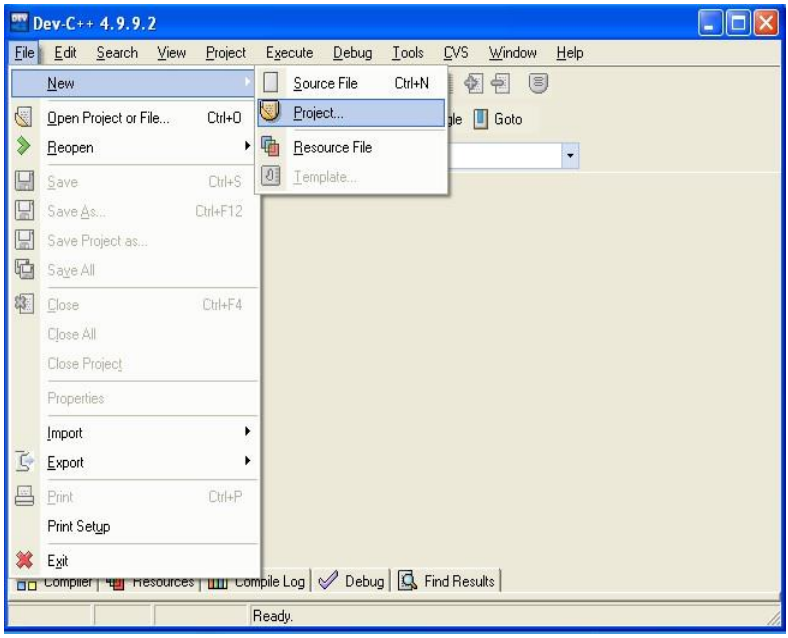
Để khởi động một tệp chương trình thực hiện ta sử dụng tên của nó. Chẳng hạn, để khởi động chương trình CHAO.EXE ta bấm CHAO và nhấn phím Enter (trong môi trường DOS) hoặc ta có thể chạy trong môi trường Turbo C bằng cách bấm đồng thời hai phím Ctrl + F9. Khi đó chương trình bắt đầu làm việc. Nếu nhập đủ các số liệu cần thiết chương trình sẽ tính toán và đưa ra kết quả cuối cùng trên máy tính.

1.7.2. DevC

1.7.2.1. Soạn thảo chương trình C bằng DevC++

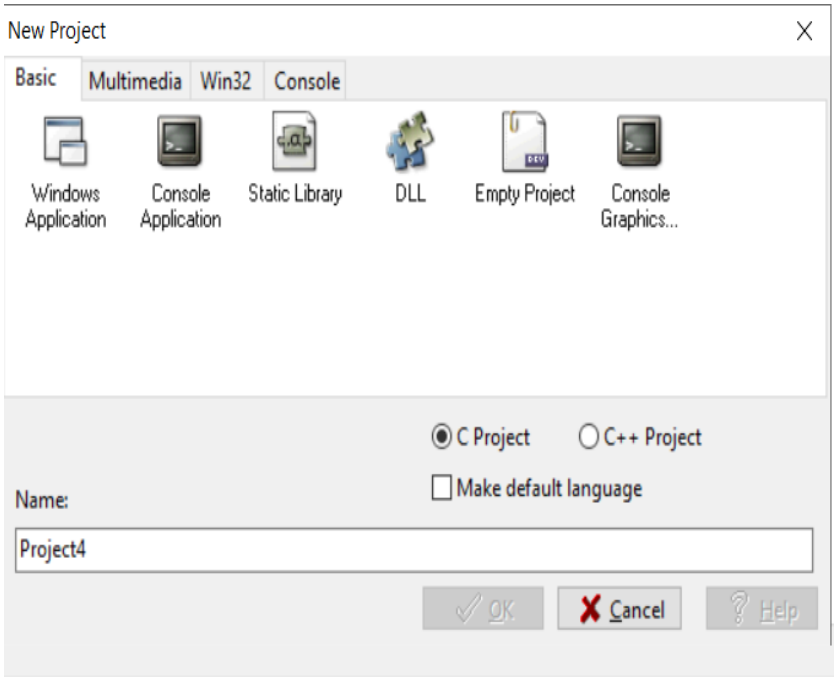
Bước 1: Chọn vào trình đơn File, chọn tiếp New.

Bước 2: Lựa chọn Project…



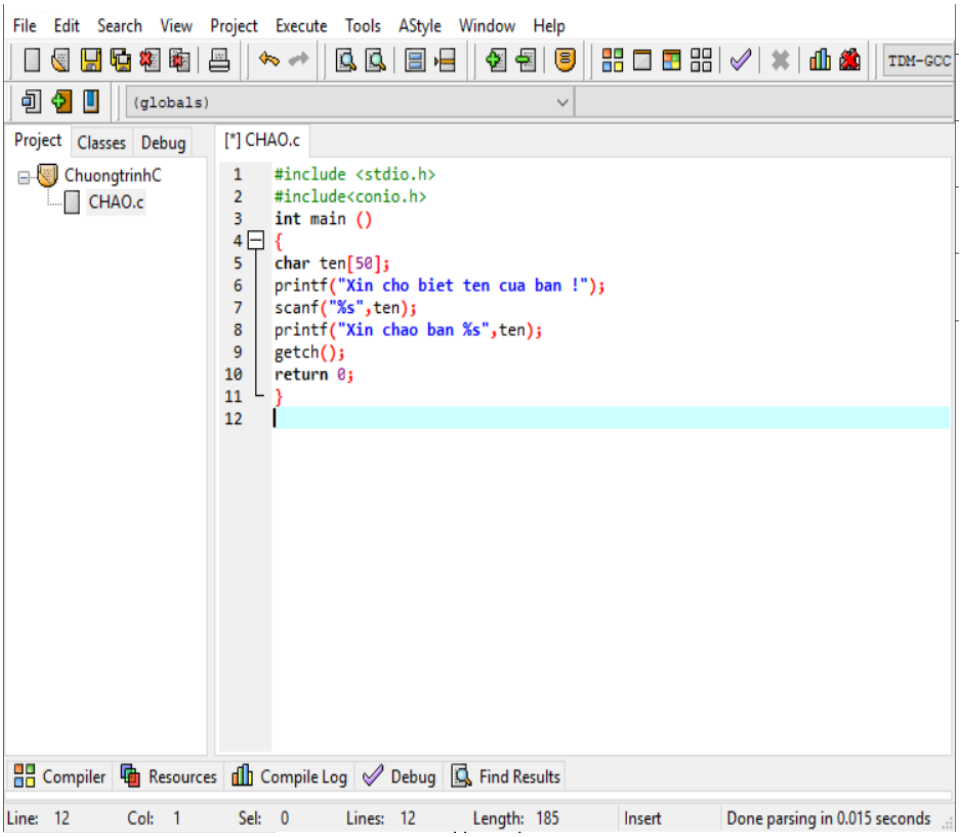
Hình 1.6. Khởi động soạn thảo chương trình C bằng DevC++

Bước 3: Chọn C Project



Hình 1.7. Màn hình New Project

Bước 4: Soạn thảo chương trình



Hình 1.8. Màn hình soạn thảo chương trình C bằng DevC++

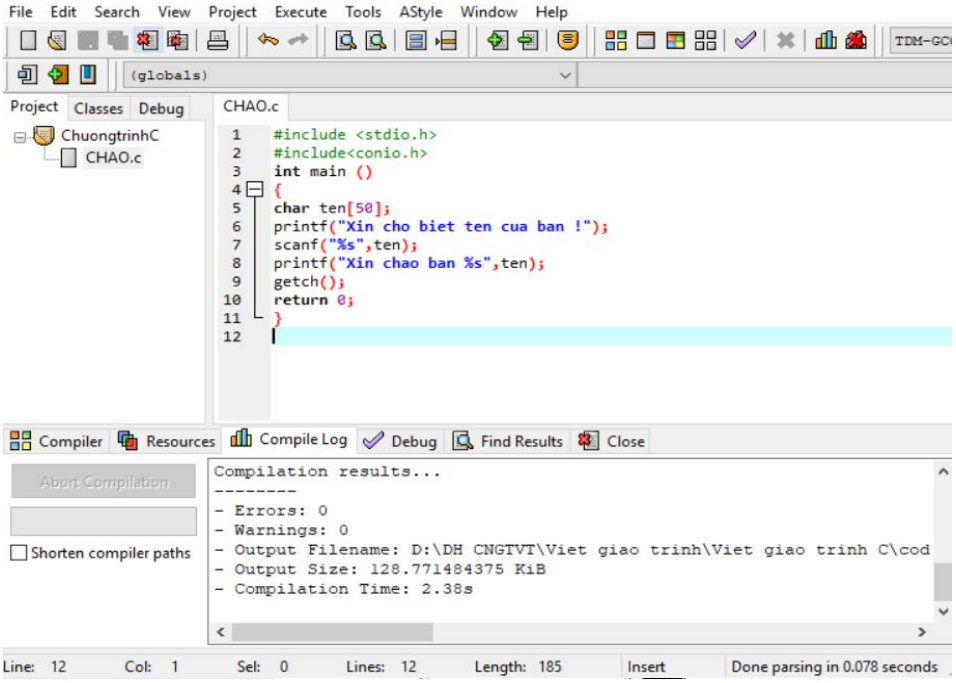
1.7.2.2. Thực hiện chương trình

Sau khi đã viết xong chương trình, để thực hiện chương trình, chúng ta làm như sau:

Bước 1: Vào menu Execute, chọn Compile (hoặc nhấn F9).

Bước 2: Nếu chương trình còn lỗi, thì sửa các dòng lỗi.

Bước 3: Nếu biên dịch không có lỗi thì sẽ trả về kết quả như sau:



Hình 1.9. Màn hình khi biên dịch thành công

Bước 4: Chạy chương trình.

Vào menu Execute và chọn Run (hoặc nhấn F10).

Hình 1.10. Màn hình kết quả

BÀI TẬP CHƯƠNG 1

A. BÀI TẬP CÓ LỜI GIẢI

🕮 Bài tập 1.1

Soạn thảo chương trình ở Ví dụ 1.59, quan sát trên màn hình để thấy kết quả của việc thực thi chương trình sau đó gõ phím bất kỳ để trở lại với Turbo, lưu chương trình với tên CHAO.c.

🕮 Bài tập 1.2

Mở một chương trình đã có trên đĩa của bài tập 1.1.

Với một chương trình đã có trên đĩa, ta có thể mở nó ra để thực hiện hoặc sửa chữa bổ sung. Để mở một chương trình ta dùng File/Open hoặc gõ phím F3. Sau đó gõ tên tập tin vào hộp File Name hoặc lựa chọn tập tin trong danh sách các tập tin rồi gõ Enter.

Mở tập tin CHAO.C, sau đó bổ sung để có chương trình mới như sau:

```
1  #include <stdio.h>
2  #include<conio.h>
3  int main()
4  {
5  char ten[50];
6  printf(“Xin cho biet ten cua ban !”);
7  scanf(“%s”,ten);
8  printf(“Xin chao ban %s\n ”,ten);
9  printf(“Chao mung ban den voi Ngon ngu lap trinh C”);
10 getch();
11 return 0;
12 }
```

✍Yêu cầu:

- Ghi lại chương trình này (F2) và cho thực hiện (Ctrl-F9). Hãy so sánh xem có gì khác trước?

🕮 Bài tập 1.3

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình sau đây:

```
1  # include “stdio.h”
```

```
2  # include “conio.h”
3  void void main()
4  {
5  printf(“TURBO C HAN HANH\n”);
6  printf(“LAM QUEN VOI BAN”);
7  getch();          // tạm dừng máy để xem kết quả
8  }
```

✍Yêu cầu:

- Chạy chương trình và cho biết kết quả?
- Giải thích kết quả đạt được của chương trình?
- Ý nghĩa của các thư viện stdio.h và conio.h?

Gợi ý:

- Kết quả của chương trình:

TURBO C HAN HANH

LÀM QUEN VỚI BẠN

- Ý nghĩa của thư viện stdio.h và conio.h

+ Thư viện stdio.h: Thư viện chứa các hàm vào/ra chuẩn (standard input/output). Gồm các hàm printf(), scanf(), getc(), putc(), gets(), puts(), fflush(), fopen(), fclose(), fread(), fwrite(), getchar(), putchar(), getw(), putw(),…

+ Thư viện conio.h. được viết tắt của từ Console Input/Output.header là thư viện được sử dụng trong trình biên dịch của các hệ điều hành cũ MS-DOS những năm 1980s với giao diện dòng lệnh. Thư viện conio.h hỗ trợ các hàm giúp bạn thực hiện các thao tác input hoặc output từ màn hình console, trong đó có 2 hàm bạn thường thấy sử dụng phổ biến nhất là:

* clrscr(): lệnh xóa các output đã có trên màn hình console (làm sạch màn hình console).

* getch(): lệnh lấy input từ màn hình console (do đó lập trình viên thường dùng điểm này của getch() để dừng màn hình console sau khi xuất kết quả).

🕮 Bài tập 1.4

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình sau đây:

```
1  #include<stdio.h>
2  #include<conio.h>
3  void main()
4  {
5    int a, b, tong, hieu, tich;
6    float thuong;
7       printf(“\n nhap a=”);  scanf(“%d”,&a);
8       printf(“\n nhap b=”);  scanf(“%d”,&b);
9       tong = a + b;
10      hieu = a - b;
11      tich = a * b;
12      thuong = (float) a/b;
13      printf(“\n tong = %d”, tong);
14      printf(“\n hieu = %d”, hieu);
15      printf(“\n tich = %d”, tich);
16      printf(“\n thuong = %3.2f”, thuong);
17 getch();
18 }
```

✍Yêu cầu:

- Biên dịch và chạy chương trình.

- Tìm hiểu về câu lệnh printf(), ý nghĩa của các ký hiệu %d, %3.2f.

- Mục đích của dòng lệnh thuong = (float) a/b; là gì?

Gợi ý:

- Tìm hiểu câu lệnh pritnf():

+ Hàm printf() được sử dụng cho đầu ra. Nó in ra lệnh đã cho vào bảng điều khiển (console).

+ Cú pháp của hàm printf() được đưa ra dưới đây:

```
printf("format string", argument_list);
```

+ Mã định dạng có thể là %d (số nguyên), %c (ký tự), %s (chuỗi), %f (float), v.v.

- **Mục đích của dòng lệnh** *thuong = (float) a/b;*: ép kiểu dữ liệu của biểu thức a/b về kiểu float.

🕮 Bài tập 1.5

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình sau đây:

```
1  #include <stdio.h>
2  #include <conio.h>
3  void main()
4  {
5  int a;
6  long int b;
7  float x;
8  char st[10];
9  printf("\n Vao du lieu tu ban phim ");
10 printf("\n a = "); scanf("%d",&a);
11 printf("\n b = "); scanf("%ld",&b);
12 printf("\n x = "); scanf("%f",&x);
13 printf("\n Nhap vao mot chuoi :");scanf("%s",st);
14 printf("\n a :%10d\n b :%10ld\n x :%10.2f",a,b,x);
15 printf("\n Chuoi da nhap :%s",st);
16 getch();
17 }
```

✍Yêu cầu:

- Biên dịch và chạy chương trình.

- Tìm hiểu về câu lệnh printf(), từ câu lệnh 9 đến câu lệnh 15?

Gợi ý: Câu lệnh prinft() từ 9 đến 13: in ra màn hình mời nhập các giá trị a, b, x, chuỗi.

Câu lệnh printf() từ 14 đến 15: in ra màn hình các biến a, b, x theo định dạng của số nguyên, số nguyên dàivà số thực.

🕮 Bài tập 1.6

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình sau đây:

```
1  #include<stdio.h>
2  #include<conio.h>
3  #define  pi   3.14
4  void  main()
5  {
```

```
6       int  r;
7             float  p, s;
8               printf(“\n nhap r=”); scanf(“%d”,&r);
9                 p = 2 * pi * r;
10                s = pi * r * r;
11              printf(“\n chu vi = %3.1f”, p);
12        printf(“\n dien tich = %3.1f”, s);
13              getch();
14            }
```

✍**Yêu cầu:**

- Biên dịch và chạy chương trình?
- Cho biết chương trình trên thực hiện công việc gì?
- Cho biết ý nghĩa của dòng lệnh #define pi 3.14?
- Dòng lệnh 9, 10 được gọi là gì?

Gợi ý:

- Chương trình tính diện tích và chu vi.
- Ý nghĩa của lệnh #define pi 3.14: khai báo hằng pi có giá trị là 3.14.

B. BÀI TẬP TỰ GIẢI

🕮 **Bài tập 1.7**

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình sau đây:

```
1  #include<stdio.h>
2  #include<conio.h>
3  #include<math.h>
4  void   main()
5  {
6  int x, n;
7  float   t, p, q;
8  printf(“\n nhap n va x= ”);
9  scanf(“%d%d”,&n,&x);
10 t = sqrt(x + n*3/2);
11 P = exp(x) + pow(x,n);
12 q = sin(x) + log(x)  + pow(10,n);
```

```
13 printf(“bieu thuc T = %3.1f”, t);
14 printf(“bieu thuc P = %3.1f”, p);
15 printf(“bieu thuc Q = %3.1f”, q);
16 getch();
17 }
```

✍**Yêu cầu:**

- Biên dịch và chạy chương trình.
- Cho biết chương trình trên tính giá trị của những biểu thức nào?
- Tìm hiểu về các hàm sqrt(), exp(), sin(), log(), pow().
- Cho biết thư viện math.h dùng để làm gì?

🕮 **Bài tập 1.8**

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình sau đây:

```
1  # include “stdio.h”
2  # include “conio.h”
3  void main()
4  {
5  int dai, rong;   //khai báo các biến float cv, dt;
6  //In ra màn hình thông báo nhap chiều dai, rong
7  printf(“Nhap chieu dai, rong:”);
8  //Nhập số liệu cho các biến
9  scanf(“%d %d”, &dai, &rong);
10 cv = (float)(dai+rong)*2;
11 // Tính chu vi
12 dt = (float)(dai*rong); // Tính diện tích
13 printf(“\nchu vi = %d”,cv);    // In ra màn hình giá trị chu vi
14 printf(“\ndien tich = %d”,dt); // In ra màn hình diện tích
15 getch(); //Dừng màn hình để xem kết quả
16 }
```

✍**Yêu cầu:**

- Biên dịch và chạy chương trình, chương trình thông báo lỗi gì? Cách khắc phục lỗi chương trình thông báo?

- Cho biết ý nghĩa của dòng lệnh 1, dòng lệnh 2. Phân biệt giữa cách viết “stdio.h” và <stdio.h>.

- Dòng lệnh 10, cv = (float)(dai+rong)*2; cho kết quả in ra như thế nào?

🕮 Bài tập 1.9

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình minh họa cho việc khai báo và khởi gán giá trị cho các biến.

```
1  #include<stdio.h>
2  #include<conio.h>
3  void main()
4  {
5  int a = 20; /* Khai bao va khoi dau cac bien */
6  int b = 15;
7  float x = 25.678;
8  clrscr();
9  printf("\n1:%d %f\n",a,x);
10 printf("2:%4d %10f\n",b,x);
11 printf("3:%2d %3f\n",a,x);
12 printf("4:%10.3f %10d\n",x,b);
13 printf("5:%-5d %f\n",a,x);
14 printf("6:%*d\n",b,b);
15 printf("7:%*.*f\n",12,5,x);
16 printf("8:%x :%8x :\n",a,a);
17 printf("9:%o :%8o :\n",a,a);
18 getch();
19 }
```

✍ Yêu cầu:

- Biên dịch và chạy chương trình.

- Giải thích kết quả thu được.

- Tìm hiểu về các ký tự định dạng của câu lệnh printf().

🕮 Bài tập 1.10

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình sau đây:

```
1  #include<stdio.h>
2  #include<conio.h>
3  void main()
4  {
5  int m = 3,p = 5;
6  int a1,a2,a3,a4,a5;
7  float x1,x2,x3,x4;
8  clrscr();
9  printf("\n Tim gia tri gan cho cac bien");
10 a1 = m<p;
11 a2 = m == p;
12 a3 = p%m + p>m;
13 a4 = m*(p>m ? m:p);
14 a5 = m*(p<m ? p:p);
15 x1 = p/m;
16 x2 = (float)p/m;
17 x3 = (p +0.5)/m;
18 x4 = (int)(p+0.5)/m;
19 printf("\n a1 = %d ",a1);
20 printf("\n a2 = %d ",a2);
21 printf("\n a3 = %d ",a3);
22 printf("\n a4 = %d ",a4);
23 printf("\n a5 = %d ",a5);
24 printf("\n x1 = %10.3f ",x1);
25 printf("\n x2 = %10.3f ",x2);
26 printf("\n x3 = %10.3f ",x3);
27 printf("\n x4 = %10.3f ",x4);
28 getch();
29 }
```

✍ **Yêu cầu:**

- Biên dịch và chạy chương trình? Chương trình có lỗi? Hãy sửa lỗi của chương trình?

- Nếu không còn lỗi, hãy cho biết kết quả của chương trình và giải thích?

🕮 Bài tập 1.11

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình sau đây:

```
1  #include <stdio.h>
2  #include <conio.h>
3  void main()
4  {
5  int a = 10, b = 5,c = 10, d;
6  clrscr();
7  printf("\n Minh hoa phep toan tang giam \n");
8  d = a == (b=c);
9  printf(" A :a =%d b =%d c =%d d =%d\n",a,b,c,d);
10 a = b = c = 5;
11 a += b += c;
12 printf(" B :a =%d b =%d c =%d \n",a,b,c);
13 c = a < b ? a++ : b++;
14 printf(" C :a =%d b =%d c =%d \n",a,b,c);
15 c = a > b ? a++ : b++;
16 printf(" D :a =%d b =%d c =%d \n",a,b,c);
17 getch();
18 }
```

✍Yêu cầu:

- Biên dịch và chạy chương trình? Chương trình có lỗi? Hãy sửa lỗi của chương trình?

- Nếu không còn lỗi, hãy cho biết kết quả của chương trình và giải thích?

- Giải thích dòng lệnh 8.

- Giải thích dòng lệnh 11.

- Giải thích dòng lệnh 13.

- Giải thích dòng lệnh 15.

🕮 Bài tập 1.12

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình sau đây:

```
1   #include<stdio.h>
2   #include<conio.h>
3   void main()
4   {
5   int a,b,c;
6   clrscr();
7   printf(" \n Chuong trinh minh hoa toan tu logic \n ");
8   a = 5; b = 2; /* Truong hop 1 */
9   c = (a++ > b ) || ( b++ != 3);
10  printf("A : a = %d b = %d c = %d\n",a,b,c);
11  a = 5; b = 2 ; /* Truong hop 2 */
12  printf(" B : a = %d b = %d c = %d\n",a,b,c);
13  a = 5; b = 2 ; /* Truong hop 3 */
14  c = (++a == 3)&&( ++b == 3);
15  printf(" C : a = %d b = %d c = %d\n",a,b,c);
16  a = 5; b = 2; /* Truong hop 4 */
17  c = (++a == 6)&& ( ++b == 3);
18  printf(" D : a = %d b = %d c = %d\n",a,b,c);
19  getch();
20  }
```

✍Yêu cầu:

- Biên dịch và chạy chương trình? Chương trình có lỗi? Hãy sửa lỗi của chương trình?

- Nếu không còn lỗi, hãy cho biết kết quả của chương trình và giải thích?

- Giải thích dòng lệnh 9.

- Giải thích dòng lệnh 14.

- Giải thích dòng lệnh 17.

🕮 Bài tập 1.13

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình sau đây:

```
1  #include <stdio.h>
2  #include <conio.h>
```

```
 3  void main()
 4  {
 5  float x,y,z,max,min;
 6  clrscr();
 7  printf("\nNhap vao 3 so ");
 8  scanf("%f%f%f",&x,&y,&z);
 9  Max = (x > y)?x:y;
10 Max = (max > z)?max:z;
11 Min = (x > y)?y:x;
12 Min = (min > z)?z:min;
13 printf("\nSo lon nhat la %f",max);
14 printf("\nSo nho nhat la %f",min);
15 printf("\nDay la 5 tieng chuong !\a\a\a\a\a");
16 getch();
17 }
```

✍Yêu cầu:

- Biên dịch và chạy chương trình? Chương trình có lỗi? Hãy sửa lỗi của chương trình?

- Nếu không còn lỗi, hãy cho biết kết quả của chương trình và giải thích?

- Giải thích dòng lệnh 9, 10.

- Giải thích dòng lệnh 11, 12.

- Giải thích dòng lệnh 13, 14, 15.

🕮 Bài tập 1.14

Sử dụng Turbo C hoặc Dev C++, soạn thảo chương trình sau đây:

```
1   #include <stdio.h>
2   main()
3   {
4           int i,j;
5           clrscr();
```

```
6           printf("\nNhap so thu nhat = ");
7           scanf("%d",&i);
8           printf("\nNhap so thu hai = ");
9           scanf("%d",&j);
10          printf("\nSo %d co NOT %d : %d ",i,i,!i);
11          printf("\nSo %d co NOT %d : %d ",j,j,!j);
12          printf("\nSo %d AND %d : %d ",i,j,i&&j);
13          printf("\nSo %d OR %d : %d ",i,j,i||j);
14          getch();
15  }
```

✍**Yêu cầu:**

- Biên dịch và chạy chương trình?
- Hãy cho biết kết quả của chương trình và giải thích?
- Giải thích dòng lệnh 10, 11, 12, 13.

Chương 2
CÁC LỆNH NHẬP - XUẤT DỮ LIỆU TRONG C

Nội dung của Chương 2, đề cập đến các câu lệnh được sử dụng để nhập xuất các dữ liệu trên màn hình Console, hoặc đọc ghi trên các tệp. Dữ liệu nhập vào hoặc xuất ra có thể là một ký tự, một đoạn văn bản, một con số (số nguyên, số thực), v.v. Tùy theo dữ liệu nhập vào hoặc xuất ra mà chúng ta có thể sử dụng những câu lệnh nhập xuất hợp lý, hiệu quả với các mã định dạng để có kết quả như mong muốn.

2.1. NHẬP XUẤT DỮ LIỆU

Như chúng ta đều biết ba chức năng thiết yếu của máy tính là đọc, xử lý và ghi dữ liệu. Phần lớn các chương trình lấy dữ liệu làm đầu vào và sau đó sau khi xử lý dữ liệu đã xử lý được hiển thị được gọi là thông tin. Trong lập trình C, chúng ta có thể sử dụng hàm scanf() và printf() được xác định trước để đọc và in dữ liệu.

2.1.1. Hàm kết xuất dữ liệu printf()

Hàm printf() được viết tắt của cụm từ "print formated", là một hàm dựng sẵn trong thư viện "stdio.h", có khả năng chuyển tạo khuôn mẫu đầu ra trên màn hình được cung cấp với chuỗi có mã định dạng phù hợp tương ứng với các kiểu dữ liệu và theo trình tự xuất hiện của các đối số.

Cú pháp

printf("format string", [argument_1, argument_2,…, argument_n]);

Trong đó:

printf: tên hàm, phải viết bằng chữ thường.

format string: chuỗi định dạng, được đặt trong cặp dấu nháy kép " ", bao gồm một trong các thành phần sau đây:

- Chuỗi ký tự: in ra màn hình chuỗi ký tự được đưa vào.
- Đối với những ký tự chuyển đổi dạng thức cho phép kết xuất giá trị của các đối số ra màn hình (được gọi là mã định dạng). Mã định dạng còn được gọi là các đặc tả, bao gồm dấu % và theo sau là các ký tự c, d, f, u, s, v.v. tương ứng với các kiểu dữ liệu của các đối số:

+ %c: char (ký tự đơn);

+ %s: string (chuỗi ký tự);

+ %d hoặc %i: int (số nguyên dạng thập phân có dấu);

+ %u: unsigned int (số nguyên dạng thập phân không dấu);

+ %f: float (số thực - số chấm động);

+ %e hoặc %E: float or double exponential format (số chấm động dạng số mũ, chẳng hạn 1,2E-8, nghĩa là 1,2 × 10^{-8});

+ %o: octal (số nguyên dạng bát phân);

+ %x hoặc %X: hexadecimal (số thập lục phân (cơ số 16) không dấu viết thường và viết hoa tương ứng);

+ %ld, %li, %lu, %lx: cho các số nguyên dài;

+ %lf, %le, %lE: cho kiểu double, thay l bằng L cho kiểu long double.

+ Với số thực, quy định độ rộng và độ chính xác được ghi sau dấu % có dạng wid.pre với wid là độ rộng của số thực, pre là phần lẻ sau dấu thập phân. Ví dụ %5.2f nghĩa là độ rộng ít nhất 5 ký tự (thêm các khoảng trống vào nếu thiếu) và nhiều nhất là 2 ký tự cho phần lẻ sau dấu chấm thập phân.

- Các ký hiệu điều khiển và các ký tự đặc biệt:

+ \n: new line (xuống dòng);

+ \t: tab (ký tự tiếp theo cách 1 tab);

+ \r: carrier return (nhảy về đầu dòng, không xuống dòng);

+ \a: tiếng kêu bip;

+ \\: in ra dấu \;

+ \": in ra dấu ";

+ \': in ra dấu ';

+ %%: in ra dấu %;

argument_1, argument_2,…, argument_n: danh sách các đối số được phân cách với nhau bằng dấu chấm phẩy theo đúng trình tự, nếu không đúng trình tự dẫn đến kết quả đầu ra sai.

Lưu ý: Khi sử dụng hàm printf(), bắt buộc phải khai báo tiền xử lý #include <stdio.h>

🕮 Ví dụ 2.1

Câu lệnh printf("Bai hoc C dau tien. \n"); In lên màn hình dòng thông báo "Bai hoc C dau tien". Sau đó con trỏ nhảy xuống dòng.

✍Chương trình

```
1  #include <stdio.h>
2  #include <conio.h>
3  main()
4  {
```

```
5  printf("\Bai hoc C dau tien. \n");
6  getch();
7  }
```

✍Kết quả chương trình

```
Bai hoc C dau tien.
```

🕮 Ví dụ 2.2

In ra màn hình giá trị của a = 5 (a là số nguyên)

✍Chương trình

```
8  #include <stdio.h>
9  #include <conio.h>
10 main()
11 {
12 int a = 5;
13 printf("\n gia tri cua a = %d", a);
14 getch();
15 }
```

✍Kết quả chương trình

```
gia tri cua a = 5
```

✍Giải thích:

Dòng 5: Khai báo biến nguyên a, khởi tạo giá trị bằng 5.

Dòng 6: Trong câu lệnh printf(), khi biên dịch gặp mã định dạng %d, tương ứng với đối số a, chương trình sẽ in ra giá trị a, tức là in số 5 ra màn hình.

🕮 Ví dụ 2.3

Giả sử biến a có giá trị = 7 và b có giá trị = 4. Viết chương trình tính và n tổng của 2 số ra màn hình.

✍Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
```

```
3   main()
4   {
5   int a = 7;
6   int b = 4;
7   printf("Tong cua 2 so %d va %d la %d . \n", a, b, a + b);
8   getch();
9   }
```

✍Kết quả chương trình

```
Tong cua 2 so 7 va 4 la 11.
```

✍Giải thích:

Dòng 5 và dòng 6: Khai báo hai biến số nguyên a và b, khởi tạo giá trị a = 7 và b = 4.

Dòng 7: Trong câu lệnh printf() xuất hiện ba mã định dạng %d, tương ứng với ba đối số a, b và a + b. Do đó chương trình sẽ in ra "Tong cua 2 so 7 va 4 la 11."

🕮 Ví dụ 2.4

Viết chương trình in ra chuỗi ký tự "Truong Dai hoc Cong nghe GTVT".

✍Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
3   main()
4   {
5   char st[50]= "Truong Dai hoc Cong nghe GTVT";
6   printf("%s", st);
7   getch();
8   }
```

✍Giải thích:

Dòng 5: Khai báo 1 chuỗi ký tự st, có độ dài tối đa 50 ký tự, được khởi tạo bằng câu "Truong Dai hoc Cong nghe GTVT".

Dòng 6: Trong câu lệnh printf(), xuất hiện mã định dạng %s, do đó

chương trình sẽ dò tìm đối số tương ứng là st. Vì vậy chuỗi "Truong Dai hoc Cong nghe GTVT" được in ra màn hình.

🕮 Ví dụ 2.5

Viết chương trình in ra màn hình giá trị của số thực pi = 3.1415.

✍Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
3   main()
4   {
5   float pi = 3.1415;
6   printf("In ra man hinh so thuc pi = %f \n", pi);
7   printf("In ra man hinh so thuc pi = %.2f \n", pi);
8   getch();
9   }
```

✍Kết quả chương trình

```
In ra man hinh so thuc pi = 3.141500
In ra man hinh so thuc pi = 3.14
```

✍Giải thích:

Dòng 5: Khai báo một biến pi kiểu thực, khởi tạo giá trị pi = 3.1415;

Dòng 6: Trong câu lệnh printf() có một mã định dạng %f, do đó chương trình sẽ dò tìm đối số tương ứng là pi, in ra màn hình số thực chấm động.

Dòng 7: Trong câu lệnh printf() có một mã định dạng %.2f, do đó chương trình sẽ dò tìm đối số tương ứng là pi, tuy nhiên số thực được in ra định dạng với phần thập phân sau dấu chấm có 2 ký tự. Vì vậy, kết quả số pi được in ra là 3.14.

🕮 Ví dụ 2.6

Cho đoạn chương trình sau đây:

```
1   #include <stdio.h>
2   int main()
3   {
```

```
    int a = 6;
    float b = 6.7;
    char c = 'A';
    long d = 8866;
char s[] = "Truong Dai hoc Cong nghe GTVT"; // Khai bao va khoi tao s
    printf("%6d %5.3f %.3f \n", a, b, a+b);
    printf("%-5d %5ld %5ld \n", a, d, a*d);
    printf("%5c \n", c);
    printf("%30s \n", s);
    return 0;
}
```

✍**Kết quả chương trình**:

```
6 6.700 12.700
6     8866 53196
    A
 Truong Dai hoc Cong nghe GTVT
```

Ý nghĩa của các ký tự định dạng trong chương trình:

- %5c: Xuất ký tự có bề rộng 5;
- %6d: Số nguyên có bề rộng 6;
- %30s: Xuất chuỗi có bề rộng 30;
- %5.3f: Xuất số thực có bề rộng 5 trong đó có 3 số sau dấu phẩy;
- %-5d: Số nguyên có bề rộng 5 nhưng căn lề trái.

🕮 **Ví dụ 2.7**

Đoạn chương trình sau đây thể hiện các mã định dạng với các kiểu dữ liệu khác nhau:

✍**Chương trình:**

```
#include <stdio.h>
int main()
{
```

```
4     int n = 255;
5     int l = 0;
6     float a = 3.14567;
7     printf("Xin chao");
8     printf("Xin chao\nBan co khoe khong?");
9     printf("Xin chao \"ban\", Ban co khoe khong?\n");
10    printf("Toi da hoan thanh 84.20%%  bai tap ve nha\n");
11    printf("n dinh dang he bat phan: %o\n", n);
12    printf("n dinh dang o he hecxa (thuong) : %x\n", n);
13    printf("n dinh dang o he hecxa (Hoa) : %X\n", n);
14    printf("Tep duoc chua tai c:\\thumuc\\tep_tep\n");
15    l = printf("Xin chao\n");
16    printf("Do dai: %d\n", l);
17    printf("n(dem): %05d\n", n);
18    printf("str1=\"%20s\", str2=\"%-20s\"\n", "Xin chao", "Ban");
19    printf("a = %.2f\n", a);
20    printf("n = %i \n", n);
21    printf("Dia chi cua n: %p\n", &n);
22    return0;
23 }
```

Kết quả chương trình

```
Xin chao
Ban co khoe khong?Xin chao "ban", Ban co khoe khong?
Toi da hoan thanh 84.20%  bai tap ve nha
n dinh dang he bat phan: 377
n  dinh dang o he hecxa (thuong): ff
n dinh dang o he hecxa (Hoa): FF
Tep duoc chua tai c:\thumuc\tep_tep
Xin chao
```

Do dai: 9

n(dem): 00255

str1 = "Xin chao", str2 = "Ban"

a = 3.15

n = 255

Dia chi cua n: 000000000062FE14

✍Yêu cầu:

Sinh viên dựa vào các dòng lệnh của chương trình để giải thích kết quả thu được.

2.1.2. Hàm nhập liệu scanf()

Hàm scanf() được viết tắt của cụm từ "scan formatted", là hàm được khai báo trong thư viện *"stdio.h"*. Có tác dụng đọc thông tin từ thiết bị vào chuẩn (bàn phím), chuyển dịch chúng (thành số nguyên, số thực,…) và lưu trữ vào bộ nhớ theo các địa chỉ xác định.

Cú pháp:

scanf("format specifiers", argument_1, argument_2,…, argument_n);

Trong đó:

- scanf: tên hàm, phải viết bằng chữ thường;

- format specifiers: chuỗi định dạng, bao gồm các mã định dạng, được đặt trong cặp dấu nháy kép (""), tương tự như hàm printf, xác định có bao nhiêu đối số và có những kiểu dữ liệu nào. Các mã định dạng còn được gọi là đặc tả. Mỗi đặc tả bắt đầu bằng ký hiệu %, có dạng như sau: *[wid[.pre]]<ký_tự_định_dạng>*, trong đó:

+ *wid[.pre]* với wid quy định độ rộng, pre là phần lẻ sau dấu thập phân (áp dụng với số thực chấm động).

+ *<ký_tự_định_dạng>:* là ký tự quy định kiểu dữ liệu cần nhập, danh sách các ký tự định dạng được thể hiện ở Bảng 2.1 dưới đây;

- argument_1, argument_2,…, argument_n: danh sách các đối số, được cách nhau bởi dấu phẩy, mỗi đối số sẽ tiếp nhận giá trị nhập vào. Danh sách các đối số tương ứng với số mã định dạng trong chuỗi định dạng.

Sau đây là bảng danh sách của mã định dạng được sử dụng trong câu lệnh scanf():

Bảng 2.1. Mã định dạng tương ứng với các kiểu dữ liệu

Kiểu dữ liệu	Mã định dạng
int	%d
char	%c
float	%f
double	%lf
short int	%hd
unsigned int	%u
long int	%li
long long int	%lli
unsigned long int	%lu
unsigned long long int	%llu
signed char	%c
unsigned char	%c
long double	%Lf

🕮 Ví dụ 2.8

Nhập vào 1 số nguyên từ bàn phím và in giá trị của số này ra màn hình.

✍Chương trình

```
1    #include <stdio.h>
2    #include <conio.h>
3    main()
4    {
5    int a;
6    printf("Nhap vao gia tri cua a:");
7    scanf("%d", &a);
8    printf("Gia tri cua a = %d", a);
9    getch();
10   }
```

✍Kết quả chương trình

```
Nhap vao gia tri cua a:12
Gia tri cua a = 12
```

Giải thích chương trình:

Dòng 7: scanf("%d", &a); chương trình chờ người dùng nhập một giá trị là số nguyên từ bàn phím và gán giá trị đó cho biến a.

Dòng 8: Trong câu lệnh printf(), chương trình sẽ đọc và gặp %d, khi đó chương trình sẽ tìm đối số tương ứng và in giá trị của biến a ra màn hình.

🕮 Ví dụ 2.9

Viết chương trình nhập vào giá trị của hai số nguyên a và b. In giá trị của số a và b ra màn hình.

✍Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
3   main()
4   {
5   int a, b, tong;
6   printf("Nhap vao gia tri cua a va b:");
7   scanf("%d %d", &a, &b);
8   printf("Gia tri cua a = %d, b = %d\n", a, b);
9   tong = a + b;
10  printf("Tong = %d", tong);
11  getch();
12  }
```

✍Kết quả chương trình

```
Nhap vao gia tri cua a va b:12 34
Gia tri cua a = 12, b = 34
Tong = 46
```

✍Giải thích chương trình:

Dòng 7: scanf("%d %d", &a, &b); Chương trình khi đọc đến dòng lệnh này, sẽ kiểm tra các mã định dạng. Có hai mã định dạng %d và %d, tương ứng với hai biến a và biến b, chương trình chờ người dùng nhập hai giá trị và gán hai giá trị này cho hai biến a, b.

Dòng 8: Hàm printf() sẽ tự động kiểm tra mã định dạng trong chuỗi định dạng của hàm printf và in giá trị của hai biến a, b ra màn hình.

Dòng 9: Thực hiện việc gán tổng bằng a + b.

Dòng 10: Câu lệnh prinf() dò tìm mã định dạng trong chuỗi định dạng, có một mã định dạng, tương ứng với một đối số là biến tổng. Do đó giá trị của biến tổng được in ra màn hình.

🕮 Ví dụ 2.10

Viết chương trình nhập giá trị cho 3 biến, trong đó biến a và b có kiểu nguyên, biến x có kiểu thực. In giá trị của các biến ra màn hình.

✍Chương trình

```
1  #include <stdio.h>
2  #include <conio.h>
3  main()
4  {
5  int a, b;
6  float x;
7  printf("Nhap vao gia tri cua a, b va x: ");
8  scanf("%d %d %f", &a, &b, &x);
9  printf("Gia tri cua a = %d, b = %d, x = %f\n", a, b, x);
10 getch();
11 }
```

✍Kết quả chương trình

```
Nhap vao gia tri cua a, b va x: 12 24 8
Gia tri cua a = 12, b = 24, x = 8.000000
```

✍Giải thích:

Dòng 8: Trong câu lệnh scanf(), có ba mã định dạng %d %d %f, được gán tương ứng với 3 đối số của câu lệnh a, b và x. Như vậy chương trình ngầm định a, b là các biến nhận giá trị nguyên, x là biến nhận giá trị thực như đã khai báo.

Dòng 9: Câu lệnh prinft() in giá trị của các biến a, b, x, thông qua việc nhận dạng 3 mã định dạng %d, %d và %f tương ứng.

Cải tiến chương trình: Nếu thay thế câu lệnh:

```
printf("Gia tri cua a = %d, b = %d, x = %f\n", a, b, x);
```

bằng câu lệnh:

printf("Gia tri cua a = %d, b = %d, x = %.2f\n", a, b, x);

thì kết quả in ra màn hình sẽ là:

Giá trị của a = 12, b = 24, x = 8.00

Giá trị của biến x khi in ra có hai chữ số thập phân sau dấu chấm.

🕮 Ví dụ 2.11

Viết chương trình nhập vào ngày, tháng, năm theo định dạng ngay/thang/nam.

✍Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
3   main()
4   {
5   int ngay, thang,nam;
6   printf("Nhap vao du lieu kieu ngay/thang/nam: ");
7   scanf("%d/%d/%d", &ngay, &thang, &nam);
8   printf("%d/%d/%d", ngay, thang, nam);
9   getch();
10  }
```

✍Kết quả chương trình

```
Nhap vao du lieu kieu ngay/thang/nam: 17/7/2019
17/7/2019
```

🕮 Ví dụ 2.12

Viết chương trình nhập vào ngày, tháng, năm với dấu phân cách /, - v.v., ngoại trừ số.

✍Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
3   main()
4   {
```

```
5   int ngay, thang,nam;
6   printf("Nhap vao du lieu kieu ngay-thang-nam: ");
7   scanf("%d%*c%d%*c%d", &ngay, &thang, &nam);
8   printf("%d/%d/%d", ngay, thang, nam);
9   getch();
10  }
```

✍Kết quả chương trình

```
Nhap vao du lieu kieu ngay-thang-nam: 12-7-2019
12/7/2019
```

🕮 Ví dụ 2.13

Viết chương trình nhập vào ngày tháng năm theo định dạng dd/mm/yyyy và in ra màn hình theo định dạng dd/mm/yyyy.

✍Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
3   main()
4   {
5   int ngay, thang,nam;
6   printf("Nhap vao du lieu kieu dd/mm/yyyy: ");
7   scanf("%2d%2d%4d", &ngay, &thang, &nam);
8   printf("%2d/%2d/%4d", ngay, thang, nam);
9   getch();
10  }
```

✍Kết quả chương trình

```
Nhap vao du lieu kieu dd/mm/yyyy: 12 12 2019
12/12/2019
```

🕮 Ví dụ 2.14

Viết chương trình nhập vào một ký tự, in ký tự đó ra màn hình và in giá trị trong bảng mã ASCII của ký tự đó.

✍Chương trình

```
1   #include <stdio.h>
2   int main()
3   {
4     char chr;
5     printf("Nhap vao mot ky tu: ");
6     scanf("%c", &chr);
7     printf("Ban vua nhap ky tu %c.\n",chr);
8     printf("Gia tri cua ky tu trong bang ma ASCII: %d.", chr);
9     return 0;
10  }
```

✍Kết quả chương trình

```
Nhap vao mot ky tu: a
Ban vua nhap ky tu a.
Gia tri cua ky tu trong bang ma ASCII: 97.
```

2.1.3. Hàm đọc và ghi tệp trong C

2.1.3.1. Hàm fprintf()

Cú pháp:

```
int fprintf(FILE *stream, const char *format [, argument, ...])
```

Chức năng: Hàm fprintf() trong C được sử dụng để ghi các ký tự vào tệp.

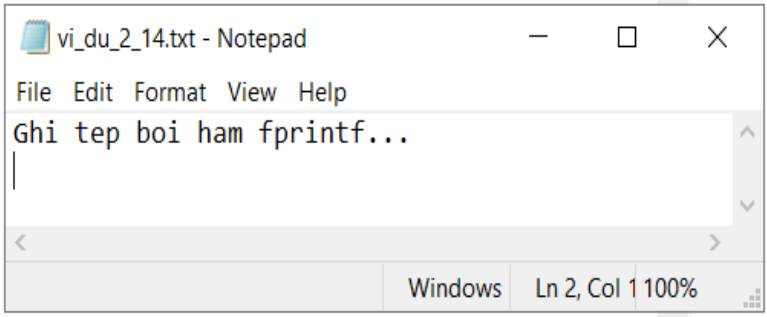
🕮 Ví dụ 2.15

Sử dụng hàm fprintf() ghi vào tệp nội dung "Ghi tep boi ham fprintf…"

✍Chương trình

```
1  #include <stdio.h>
2  main(){
3    FILE *fp;
4    fp = fopen("vi_du_2_14.txt", "w");//mo tep
5    fprintf(fp, "Ghi tep boi ham fprintf...\n");//ghi du lieu vao tep
6    fclose(fp);//doc tep
7  }
```

✍Nội dung trong tệp vi_du_2_14.txt



2.1.3.2. Hàm fscanf()

Cú pháp:

```
int fscanf(FILE *stream, const char *format [,argument, ...])
```

Chức năng: Hàm fscanf() được sử dụng để đọc tập hợp các ký tự từ tệp. Nó đọc một từ trong tệp và trả về EOF ở cuối tệp.

🕮 Ví dụ 2.16

Đọc lại nội dung tệp vi_du_2_14.txt

✍Chương trình:

```
1  #include <stdio.h>
2  main(){
3     FILE *fp;
4     char a[255];//mang cac ky tu chua noi dung cua tep
5     fp = fopen("vi_du_2_14.txt", "r");
6     while(fscanf(fp, "%s", a)!=EOF){
7     printf("%s ", a );
8     }
9     fclose(fp);
10 }
```

✍Kết quả chương trình:

```
Ghi tep boi ham fprintf...
```

Lưu ý: Nội dung của đọc và ghi tệp sẽ được trình bày kỹ hơn ở Chương 7_ Tệp.

2.2. HÀM NHẬP/XUẤT KÍ TỰ VÀ DÒNG VÀO STDIN

Input: là đưa một số dữ liệu vào một chương trình. Một đầu vào có thể được đưa ra dưới dạng một tập tin hoặc từ dòng lệnh. C cung cấp một tập hợp các hàm dựng sẵn để đọc đầu vào đã cho và đưa nó vào chương trình theo yêu cầu.

Output: là hiển thị một số dữ liệu trên màn hình, máy in hoặc trong bất kỳ tệp nào, C cung cấp một tập hợp các hàm tích hợp để xuất dữ liệu trên màn hình máy tính cũng như lưu nó trong tệp văn bản hoặc tệp nhị phân.

Các tệp tin tiêu chuẩn

Các tập tin tiêu chuẩn của ngôn ngữ C xem tất cả các thiết bị là tập tin. Vì vậy, các thiết bị như màn hình được xử lý giống như các tệp và ba tệp sau sẽ tự động được mở khi chương trình thực thi để cung cấp quyền truy cập vào bàn phím và màn hình.

Bảng 2.2. Danh sách các tệp tiêu chuẩn

Tệp tin tiêu chuẩn	Con trỏ tệp	Thiết bị
standard input	stdin	Bàn phím Mặc định là bàn phím, nhưng có thể được xem là tệp ảo chỉ đọc và có thể định nghĩa lại là một tệp trên đĩa cứng.
standard output	stdout	Màn hình Mặc định là màn hình console, nhưng có thể được xem là tệp ảo chỉ ghi và có thể định nghĩa lại là một tệp trên đĩa học máy in.
standard error	stderr	Tương tự như stdout, nhưng thường được dùng để ghi các dòng lỗi gặp phải trong chương trình.

Các con trỏ tệp là phương tiện để truy cập tệp cho mục đích đọc và viết. Phần tiếp theo sẽ giải thích cách đọc các giá trị từ màn hình và cách in kết quả trên màn hình.

2.2.1. Dòng vào stdin

Trong C, dòng vào tiêu chuẩn là stdin. Các hàm như scanf(), gets(), getchar(), v.v. đều nhận dữ liệu từ stdin. Khi trên stdin không còn dữ liệu, các hàm nhập dữ liệu sẽ yêu cầu người dùng nhập vào

từ bàn phím. Các hàm trên chỉ nhận đủ dữ liệu mà chúng yêu cầu (trong trường hợp này, scanf chỉ nhận chuỗi không có khoảng trắng), do đó một phần dữ liệu còn sót lại trên stdin, có thể là ký tự '\n' hoặc phần dữ liệu sau khoảng trắng. Điều này ảnh hưởng đến các hàm nhập dữ liệu phía sau.

2.2.1.1. Hàm gets()

Hàm gets() được sử dụng để đọc một dòng văn bản từ một thiết bị đầu vào chuẩn (stdin) và lưu vào một biến chuỗi. Khi chương trình đọc vào một dòng mới thì hàm gets sẽ bị chấm dứt. Nội dung của 2.2.1.1 sẽ giới thiệu cách đọc dữ liệu chuỗi từ console bằng cách sử dụng hàm gets() trong C và chỉ ra sự khác biệt giữa hàm gets() và hàm scanf().

Cú pháp:

char *gets(char *str)

Trong đó:

- str là con trỏ kiểu char trỏ tới vùng nhớ chứa dãy ký tự.

Chức năng: Hàm gets() ghi nhận chuỗi ký tự từ stdin, cho đến khi gặp ký tự "\n". Ký tự "\n" bị loại khỏi stdin, nhưng không được đặt vào chuỗi. Chuỗi được bổ sung thêm "\0" và đặt vào vùng nhớ do con trỏ str trỏ tới. Hàm trả về địa chỉ của chuỗi nhận được.

Có thể sử dụng cú pháp sau: gets(<variable_name>); với **variable_name:** biến ghi nhận chuỗi ký tự nhập vào.

Lưu ý:

Trước khi sử dụng hàm gets(), phải khai báo thư viện <stdio.h>

#include <stdio.h>

🕮 Ví dụ 2.17

Viết chương trình nhập vào tên trường và hiển thị ra màn hình.

✍Chương trình

```
1   // Vi du ve ham gets()
2   #include <stdio.h>
3   int main()
4   {
5           char st[50];
6           printf("\nNhap ten truong: \n");
7           gets(st);
```

```
8           printf("=============\n");
9           printf("%s", st);
10          return 0;
11  }
```

✍Kết quả chương trình:

```
Nhap ten truong:
Truong Dai hoc Cong nghe GTVT
=============
Truong Dai hoc Cong nghe GTVT
```

Lưu ý:

- Hàm scanf() cũng được sử dụng để nhập dữ liệu từ thiết bị chuẩn, tuy nhiên giữa hàm scanf() và hàm gets() khác nhau:

- Nếu sử dụng hàm scanf() để nhập vào một chuỗi thì nó sẽ kết thúc khi gặp một khoảng trắng, dòng mới hoặc End Of File (EOF), vì vậy hàm scanf() rất khó lưu chuỗi có chứa khoảng trắng. Đối với hàm gets() khi nhập chuỗi chỉ kết thúc khi ta nhấn Enter, có nghĩa là hàm gets() dễ dàng lưu được một chuỗi mà có chứa khoảng trắng.

- Hàm scanf() có thể đọc nhiều giá trị của các kiểu dữ liệu khác nhau, trong khi hàm gets() chỉ nhận kiểu dữ liệu chuỗi ký tự.

🕮 Ví dụ 2.18

Ví dụ sau đây so sánh sự khác nhau giữa hàm gets() và hàm scanf().

```
1   #include <stdio.h>
2   int main()
3   {
4       char str[20];
5       printf("Nhap vao mot chuoi su dung ham gets(): \n");
6       gets(str);
7     printf("Chuoi ky tu vua nhap su dung ham gets() : %s\n", str);
8   printf("Nhap vao mot chuoi su dung ham scanf(): \n");
9   scanf("%s",str);
10  printf("Chuoi ky tu vua nhap su dung ham scanf() : %s\n", str);
11      return 0;
12  }
```

✍Kết quả chương trình:

```
Nhap vao mot chuoi su dung ham gets():
Truong Dai hoc Cong nghe GTVT
Chuoi ky tu vua nhap su dung ham gets(): Truong Dai hoc Cong nghe GTVT
Nhap vao mot chuoi su dung ham scanf():
Truong Dai hoc Cong nghe GTVT
Chuoi ky tu vua nhap su dung ham scanf(): Truong
```

2.2.1.2. Hàm getchar()

Cú pháp:

```
int getchar(void);
```

Chức năng: Hàm getchar() đọc một ký tự đầu vào stdin (từ bàn phím). Hàm trả về ký tự đọc được, nếu thành công, trả về EOF nếu thất bại.

Lưu ý:

Muốn sử dụng hàm getchar(), phải khai báo thư viện “stdio.h”.

🕮 Ví dụ 2.19

Vi dụ 2.19n trình nhình2.19ng hàm getchar(), phải khai báo thư viện “s

✍Chương trình

```
1  #include <stdio.h>
2  int main () {
3     char s;
4     printf("Nhap vao mot ky tu: ");
5     s = getchar();
6     printf("Ky tu vua nhap: ");
7     printf("%c",s);
8     return(0);
9  }
```

✍Kết quả chương trình:

```
Nhap vao mot ky tu: a
Ky tu vua nhap: a
```

🕮 Ví dụ 2.20

Viết chương trình đọc vào từ bàn phím một ký tự, thoát khi nhấn phím "q".

✍Chương trình

```
1  #include <conio.h>
2  #include <stdio.h>
3  int main()
4  {
5     char ch;
6     printf("Nhap vao 1 ky tu:");
7     // nhap vao 1 ky tu den khi ky tu nhap vao la 'q'
8     while((ch = getchar())!= 'q')
9     {
10       fflush(stdin);
11       printf("ky tu vua nhap: %c\n", ch);
12    }
13   printf("Thoat chuong trinh!");
14   return 0;
15 }
```

✍Kết quả chương trình

```
Nhap vao 1 ky tu: a
ky tu vua nhap: a
b
ky tu vua nhap: b
c
ky tu vua nhap: c
q
Thoat chuong trinh!
```

Lưu ý:

- Xét câu lệnh ch = getchar(); Nếu bấm phím A và nhấn phím Enter thì ch = 'A', ký tự "\n" vẫn ở lại trên stdin và nó sẽ làm trôi các hàm getchar() hoặc hàm gets() sau đó. Nếu chỉ nhấn phím Enter thì ch = '\n' và '\n' bị loại khỏi stdin. Để tránh bị trôi ký tự, trong đoạn chương trình đã sử dụng hàm fflush(stdin);

- Hàm scanf() cũng để lại ký tự '\n' trên stdin. Ký tự này sẽ làm trôi hàm gets() và getchar() sau đó. Để các hàm này hoạt động đúng thì phải khử ký tự '\n' trong hàm scanf() bằng cách thêm đặc tả %*c vào cuối chuỗi điều khiển như sau:

🕮 Ví dụ 2.21

Cho đoạn chương trình sau:

```
1   #include <conio.h>
2   #include <stdio.h>
3   int main()
4    {
5    char ht[25];
6    int  t;
7    printf("Nhap t = ");
8    scanf("%d",&t);
9    printf("Nhap vao chuoi ht: ");
10   gets(ht);
11   printf("\nChuoi vua nhap %s", ht);
12   return 0;
13   }
```

✍Kết quả chương trình:

Nhap t = 12

Nhap vao chuoi ht:

Chuoi vua nhap:

Để nhận dữ liệu đúng đắn bằng cách nhập vào từ bàn phím thì trước khi thực hiện scanf(), gets(), getchar(), ta nên *"làm sạch"* stdin bằng hàm fflush(stdin). Dùng hàm này sẽ tránh được mọi hậu quả của các thao tác nhập số liệu trước đó. Với ví dụ 2.21 trên ta có thể viết lại thành ví dụ 2.22 như sau:

🕮 Ví dụ 2.22

```
1  #include "conio.h"
2  #include <stdio.h>
3   int main()
4   {
```

```
 5   char ht[25];
 6   int  t;
 7   printf("Nhap t = ");
 8   scanf("%d",&t);
 9   printf("Nhap vào chuoi ht: ");
10   fflush(stdin);
11   gets(ht);
12   printf("Chuoi vua nhap %s", ht);
13   return 0;
14  }
```

✍Kết quả chương trình

```
Nhap t = 12
Nhap vao chuoi ht: Truong DH CNGTVT
Chuoi vua nhap: Truong DH CNGTVT
```

2.2.2. Các hàm xuất ký tự puts(), putchar()

Các hàm này được khai báo trong thư viện <stdio.h>. Các hàm printf(), puts(), putchar() đều có tác dụng đưa dữ liệu lên dòng ra chuẩn stdout (màn hình).

2.2.2.1. Hàm puts()

Cú pháp:

```
int puts(const char *s);
```

Trong đó:

s là con trỏ kiểu char, trỏ tới vùng nhớ chứa chuỗi ký tự cần xuất ra stdout.

Chức năng: Đưa chuỗi s và đưa thêm ký tự "\n" lên stdout. Nếu thành công, hàm trả về ký tự cuối cùng được đưa ra (chính là '\n'). Ngược lại, nếu có lỗi hàm trả về EOF.

Lưu ý:

Phải khai báo thư viện <stdio.h> trước khi sử dụng hàm puts().

```
#include <stdio.h>
```

🕮 Ví dụ 2.23

Viết chương trình nhập vào 1 chuỗi ký tự, sử dụng hàm puts() in chuỗi vừa nhập ra màn hình.

Chương trình:

```
1  #include <stdio.h>
2  int main()
3  {
4  char ch[100];
5  printf("Nhap vao chuoi ky tu: \n");
6  //Nhap chuoi su dung ham gets()
7  gets(ch);
8  printf("Chuoi vua duoc nhap la - ");
9  //In chuoi ra man hinh
10 puts(ch);
11 return 0;
12 }
```

Kết quả chương trình:

```
Nhap vao chuoi ky tu:
Ha Noi!
Chuoi vua duoc nhap la - Ha Noi!
```

Lưu ý:

- Hàm puts() được ưu tiên để in một chuỗi vì nó đơn giản và ít rắc rối hơn hàm printf() (việc thực hiện hàm puts() thường đơn giản hơn printf() và nếu chuỗi có các ký tự định dạng như '%' thì hàm printf() sẽ cho những kết quả khác). Ngoài ra, nếu ch là mỗi chuỗi đầu vào của người dùng, thì việc dùng hàm printf() có thể gây ra các sự cố bảo mật.

- Hàm puts() di chuyển con trỏ đến dòng tiếp theo, nếu không muốn di chuyển con trỏ tới dòng tiếp theo, chúng ta có thể sử dụng hàm puts() ở dạng biến thể khác như sau:

```
fputs(str, stdout)
```

2.2.2.2. Hàm putchar()

Cú pháp:

```
int putchar(int ch);
```

Trong đó: ch là mã ASCII của ký tự muốn in ra màn hình.

Chức năng:

Hàm trả về ký tự có mã ASCII lên stdout, nếu thành công. Nếu gặp lỗi hàm trả về EOF.

🕮 Ví dụ 2.24

```
1  #include <stdio.h>
2  int main()
3  {
4   // Khoi tao ky tu de ghi
5  char ch = 'G';
6  // ghi ra stdout
7  putchar(ch);
8  return (0);
9  }
```

2.2.3. Các hàm vào ra trên màn hình, bàn phím

2.2.3.1. Hàm getch()

Cú pháp:

int getch(void);

Trả về giá trị: hàm getch() trả về ký tự đọc từ bàn phím.

Chức năng:

Hàm getch() được sử dụng để bắt một ký tự từ bàn phím. Hàm getch() đọc một ký tự từ bàn phím nhưng không hiển thị trên màn hình. Người dùng có thể sử dụng hàm getch() để giữ cửa sổ đầu ra cho đến khi nhấn bất kỳ phím nào từ bàn phím.

Lưu ý: Muốn sử dụng hàm getch(), bắt buộc phải khai báo thư viện <conio.h>

#include <conio.h>

🕮 Ví dụ 2.25

```
1  #include <conio.h>
2  #include <stdio.h>
3  main()
4  {
```

```
5  int c;
6  printf("Nhan mot phim bat ky:\n");
7  c = getch();
8  if (c)
9     printf("Mot phim dang duoc nhan tu ban phim");
10   else
11    printf("Xay ra loi!");
12 getch();
13 }
```

✍Kết quả chương trình:

Nhan mot phim bat ky

(Người dùng nhấn 1 phím trên bàn phím, nhưng không hiển thị trên màn hình)

Mot phim dang duoc nhan tu ban phim

2.2.3.2. Hàm getche()

Cú pháp:

```
int  getche(void);
```

Chức năng: giống như hàm getch(), getche() là hàm trong thư viện <conio.h>. Hàm getche() đọc một ký tự từ bàn phím và hiển thị ngay lập tức ký tự vừa nhập lên màn hình mà không cần đợi nhấn phím Enter.

🕮 Ví dụ 2.26

Xem xét ví dụ sau đây:

```
1  #include <stdio.h>
2  #include <conio.h>
3  int main()
4  {
5     char ok;
6     printf("Xin chao!\n");
7     printf("Ban muon tiep tuc, nhan y hoac n: ");
8     ok = getche(); // Cho nhan ban phim
9     if (ok == 'y')
10    {
```

```
11        printf("\nBan vua nhan Yes");
12     }
13     else
14     {
15        printf("\nBan vua nhan No");
16     }
17     return 0;
18  }
```

✍Kết quả chương trình:

```
Xin chao!
Ban muon tiep tuc, nhan y hoac n: y
Ban vua nhan Yes
```

2.2.3.3. Hàm putch()

Cú pháp:

```
putch(character); hoặc putch(character_variable);
```

Chức năng: Hàm putch() được sử dụng để in ký tự ra màn hình tại vị trí con trỏ hiện tại.

Để sử dụng hàm putch() cần khai báo thư viện <conio.h>

🕮 Ví dụ 2.27

Xem xét ví dụ sau đây:

```
1  #include<stdio.h>
2  #include<conio.h>
3  int main()
4  {
5    char ch;
6    printf("Nhan mot phim bat ky: ");
7    ch = getch();
8    printf("\nPhim vua nhan la: ");
9    putch(ch);
10   getch();
11 }
```

✍Kết quả chương trình:

```
Nhan mot phim bat ky:
Phim vua nhan la: a
```

2.2.3.4. Hàm kbhit()

Cú pháp:

```
int kbhit(void);
```

Chức năng: Hàm kbhit() được sử dụng để xác định một phím đã được nhấn hay chưa. Nếu một phím được nhấn thì hàm trả về kết quả khác không, ngược lại hàm trả về kết quả bằng 0.

Lưu ý: Để sử dụng hàm kbhit() cần khai báo thư viện <conio.h>

🕮 Ví dụ 2.28

```
1  #include <stdio.h>
2  #include <conio.h>
3   main()
4  {
5     while (!kbhit())
6        printf("Ban da khong nhan mot phim nao ca!.\n");
7     printf("Ban da nhan mot phim nao do!");
8     return 0;
9  }
```

Đoạn chương trình trên sẽ in ra màn hinh “Ban da khong nhan mot phim nao ca!” cho đến khi người dùng nhấn một phím bất kỳ và dòng “Ban da nhan mot phim nao do!” được in ra màn hình.

2.3. MỘT SỐ VÍ DỤ

🕮 Ví dụ 2.29

Viết chương trình đọc tên, họ và năm sinh và hiển thị tên và năm của người vừa nhập.

✍Chương trình:

```
1  #include <stdio.h>
2  int main()
3  {
```

```
4     char firstname[20], lastname[20];
5     int bir_year;
6     printf("Nhap vao ho: ");
7     scanf("%s", firstname);
8     printf("Nhap vao ten: ");
9     scanf("%s", lastname);
10    printf("Nhap vao nam sinh: ");
11    scanf("%d", &bir_year);
12    printf("%s %s %d\n", firstname, lastname, bir_year);
13    return 0;
14 }
```

✍Kết quả chương trình:

```
Nhap vao ho: Le
Nhap vao ten: Kien
Nhap vao nam sinh: 1980
Le Kien 1980
```

🕮 Ví dụ 2.30

Viết chương trình nhập vào thông tin của một sinh viên gồm: masv, hoten, diemtoan, diemanh.

Yêu cầu:

a. Tính tongdiem = diemtoan + diemanh.

b. In thông tin của sinh viên đó lên màn hình gồm: masv, hoten, tongdiem.

✍Phác thảo lời giải:

- Xác định các biến đầu vào: masv (Kiểu char), hoten (Kiểu char), diemtoan (Kiểu float), diemanh (Kiểu float);

- Sử dụng câu lệnh nhập, xuất, tính tổng điểm;

- In thông tin của các sinh viên.

✍Chương trình:

```
1  #include <stdio.h>
2  #include <conio.h>
```

```
3  #include <string.h>
4  main()
5  {
6    char masv[5], ht[30];
7    float d_toan, d_anh, tong_diem;
8    printf("Nhap ma sinh vien: ");
9    gets(masv);
10   puts("Nhap ho ten: "); gets(ht);
11   fflush(stdin);
12   printf("Nhap diem toan: ");
13   scanf("%f",&d_toan);
14   printf("Nhap diem tieng anh: ");
15   scanf("%f", &d_anh);
16   tong_diem = d_toan + d_anh;
17   puts("Ma Sv: "); puts(masv);
18   puts("Ho ten: "); puts(ht);
19   printf("Tong diem: = %3.2f",tong_diem);
20   getch();
21 }
```

↘Kết quả chương trình

```
Nhap ma sinh vien: sv01
Nhap ho ten:
Kien
Nhap diem toan: 9
Nhap diem tieng anh: 8
Ma Sv:
sv01
Ho ten:
Kien
Tong diem: = 17.00
```

BÀI TẬP CHƯƠNG 2

A. BÀI TẬP CÓ LỜI GIẢI

🕮 Bài tập 2.1

Viết chương trình đổi từ độ C sang độ F.

Thang đo độ C, còn được gọi là thang đo độ Celsius, được phát triển bởi nhà thiên văn học người Thụy Điển, ông Andres Celsius. Trong thang đo độ C, nước đóng băng ở 0 độ và sôi ở 100 độ.

Thang đo Fahrenheit được phát triển bởi nhà vật lý người Đức Daniel Gabriel Fahrenheit. Trong thang đo Fahrenheit, nước đóng băng ở 32 độ và sôi ở 212 độ.

Công thức chuyển đổi từ centigrade sang Fahrenheit là:

$$C = (\frac{5}{9}) * (F - 32)$$

Công thức:

$$℃ = (\frac{5}{9}) * (°F - 32) \qquad °F = \frac{9℃}{5} + 32$$

✍Phác thảo lời giải

- Nhập vào nhiệt độ ở độ C;
- Sử dụng công thức tính $°F = \frac{9℃}{5} + 32$ để chuyển sang độ F;
- In kết quả ra màn hình, với định dạng chỉ gồm 2 chữ số thập phân.

✍Chương trình

```
1  #include <stdio.h>
2  int main()
3  {
4          float do_f;
5     float do_c;
6          printf("Nhap nhiet do (do C): ");
7          scanf("%f", &do_c);
8          do_f = ((9.0 / 5.0) * do_c) + 32.0;
9          printf("Do F tuong ung la: %2.2f .\n", do_f);
10         return(0);
11 }
```

✍Kết quả chương trình:

```
Nhap nhiet do (do C): 33
Do F tuong ung la: 91.40.
```

🕮 Bài tập 2.2

Viết chương trình tính thể tích hình cầu, khi biết bán kính của hình cầu.

Công thức tính thể tích hình cầu:

$$V = \frac{4}{3} * \pi * r^3$$

✍Phác thảo lời giải:

- Khai báo hằng PI.
- Nhập bán kinh của hình cầu.
- Sử dụng công thức tính $V = \frac{4}{3} * \pi * r^3$ để tính thể tích hình cầu.
- In kết quả ra màn hình, với định dạng chỉ gồm 2 chữ số thập phân.

✍Chương trình

```
1  #include <stdio.h>
2  #define PI  3.141592653589793238
3  int main()
4  {
5          float r; // ban kinh cua hinh cau
6          float V; // the tich cua hinh cau
7          printf("Nhap vao ban kinh cua hinh cau: ");
8          scanf("%f",&r);
9          V = (4.0 / 3.0) * PI * (r *r *r);
10         printf("The tich cua hinh cau ban kinh %.2f la %5.2f.\n", r,
   V);
11         return(0);
12 }
```

✍Kết quả của chương trình

```
Nhap vao ban kinh cua hinh cau: 6
The tich cua hinh cau ban kinh 6.00 la 904.78
```

🕮 Bài tập 2.3

Viết chương trình tính chu vi và diện tích của hình chữ nhật, khi biết chiều dài và chiều rộng.

Chu vi của hình chữ nhật CV = 2*(a + b) với a, b là chiều dài và chiều rộng của hình chữ nhật;

Diện tích của hình chữ nhật S = a * b.

✍Phác thảo lời giải

- Nhập vào chiều dài và chiều rộng của hình chữ nhật;
- Sử dụng công thức tính chu vi và diện tính hình chữ nhật để tính.
- In kết quả ra màn hình.

✍Chương trình

```
1   #include <stdio.h>
2   int main()
3   {
4           float chieu_dai;
5      float chieu_rong;
6           float cv, S;
7           printf("Nhap chieu dai cua hinh chu nhat: ");
8           scanf("%f", &chieu_dai);
9           printf("Nhap chieu rong cua hinh chu nhat: ");
10          scanf("%f", &chieu_rong);
11          cv = 2.0 * (chieu_dai + chieu_rong);
12          S = chieu_dai * chieu_rong;
13          printf("Chu vi cua hinh chu nhat la % 4.2f\n", cv);
14          printf("Dien tich cua hinh chu nhat la  %4.2f\n", S);
15          return(0);
16  }
```

✍Kết quả chương trình

```
Nhap chieu dai cua hinh chu nhat: 8.2
Nhap chieu rong cua hinh chu nhat: 6.4
Chu vi cua hinh chu nhat la  29.20
Dien tich cua hinh chu nhat la  52.48
```

🕮 Bài tập 2.4

Viết chương trình nhập vào thời gian di chuyển của các phương tiện (tính theo giờ và phút). Quy đổi thời gian di chuyển sang phút.

Dữ liệu test: 4h37 phút.

Kết quả mong muốn: 277 phút.

✍Phác thảo lời giải

- Nhập vào số giờ di chuyển số phút di chuyển.
- Sử dụng công thức chuyển đổi: số phút (tổng) = Số giờ * 60 + số phút
- In kết quả ra màn hình.

✍Chương trình

```
1  #include <stdio.h>
2  #define M  60   // so phut cua mot gio
3  int main()
4  {
5  int hrs;    // so gio
6  int mins;   // so phut
7  int tot_mins;    //tong so phut
8          printf("Nhap so gio: ");
9          scanf("%d", &hrs);
10         printf("Nhap so phut: ");
11         scanf("%d", &mins);
12         tot_mins = mins + (hrs * M);
13         printf("Tong cong co: %d phut.\n", tot_mins);
14         return(0);
15 }
```

✍Kết quả chương trình

```
Nhap so gio: 4
Nhap so phut: 37
Tong cong co: 277 phut.
```

🕮 Bài tập 2.5

Viết chương trình thực hiện phép cộng, trừ, nhân, chia hai số, chia lấy dư.

✍Phác thảo chương trình:

- Nhập vào hai số (kiểu nguyên hoặc kiểu thực).

- Sử dụng các công thức tính tổng, tính hiệu, tính tích, tính thương của hai số.

- In các kết quả thu được ra màn hình.

✍Chương trình:

```
1  #include <stdio.h>
2  #include <conio.h>
3  int main()
4  {
5      int a, b;
6      int tong, hieu, tich, mod;
7      float thuong;
8      printf("Nhap vao hai so cach nhau dau phay : ");
9      scanf("%d, %d", &a, &b);
10     tong = a + b;
11     hieu = a - b;
12     tich = a * b;
13     thuong = (float)a / b;
14     mod = a % b;
15     printf("Tong cua hai so = %d\n", tong);  // in tong
16     printf("Hieu cua hai so = %d\n", hieu);  // in hieu
17     printf("Tich cua hai so =  %d\n", tich);  //in tich
18     printf("Thuong cua hai so=  %4.2f\n", thuong);  // in thuong
19     printf("Du cua phep chia = %d\n", mod);  // chia lay du
20     return 0;
21 }
```

✍Kết quả thực hiện

```
Nhap vao hai so cach nhau dau phay : 12, 6
Tong cua hai so = 18
Hieu cua hai so = 6
Tich cua hai so =  72
Thuong cua hai so=  2.00
Du cua phep chia = 0
```

B. BÀI TẬP TỰ GIẢI

🕮 Bài tập 2.6

Viết chương trình in lên màn hình dòng chữ "CHAO MUNG DEN VOI NGON NGU LAP TRINH C".

Với điều kiện nếu không bấm phím hoặc bấm một phím khác C và P thì dòng chữ trên tiếp tục hiện ra. Nếu bấm C thì chương trình in ra dòng chữ TURBO C. Nếu bấm P thì chương trình in ra dòng chữ TURBO PASCAL. Bấm tiếp phím bất kỳ thì chương trình kết thúc.

🕮 Bài tập 2.7

Viết chương trình tạo hình sau và in lên màn hình:

```
* * * * * * * * * * * *

* * * * *     * * * * *

* * * *           * * * *

* * *                 * * *

* *                       * *

* * * * * * * * * * * * *
```

🕮 Bài tập 2.8

Viết chương trình nhập phần thực x và phần ảo y của một số phức, sau đó in ra theo mẫu: (x, y) với các yêu cầu:

- Số in ra có hai chữ số sau dấu chấm thập phân;

- Kết quả in ra tạo thành một dãy ký tự liên tiếp nhau (không có khoảng trống);

🕮 Bài tập 2.9

Viết chương trình nhập vào ba số thỏa mãn điều kiện là ba cạnh của một tam giác. Sau đó tính chu vi và diện tích của tam giác và in các giá trị vừa tính được ra màn hình.

🕮 Bài tập 2.10

Viết chương trình nhập vào bán kính của hình tròn. Tính chu vi và diện tích của một hình tròn sau đó in các giá trị vừa tính được ra màn hình.

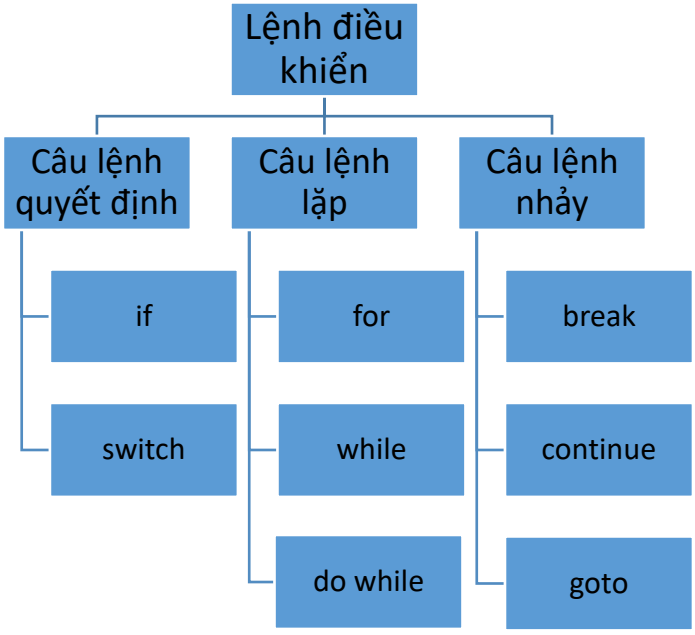
Chương 3

CÁC CẤU TRÚC PHÂN NHÁNH VÀ VÒNG LẶP

Các câu lệnh điều khiển là trái tim của bất kỳ ngôn ngữ lập trình nào. Trong hầu hết các chương trình, 60-70 % các câu lệnh của chương trình là câu lệnh điều khiển. Để làm chủ hoàn toàn chương trình của mình, chúng ta cần kiểm soát tốt chương trình, cần nắm vững các cấu trúc của các lệnh điều khiển.

Mỗi một chương trình, theo mặc định được thực hiện tuần tự. Từng câu lệnh được thực thi lần lượt. Tuy nhiên, có những tình huống xảy ra khi chúng ta cần thực thi một hoặc nhiều câu lệnh dựa trên các điều kiện, chẳng hạn hiển thị các tệp tin bí mật nếu người dùng có thông tin xác thực. Cũng có nhiều tình huống, chúng ta cần thực hiện lặp đi lặp lại một số câu lệnh nào đó, chẳng hạn gửi email đề nghị cho tất cả các khách hàng, v.v. Các câu lệnh điều khiển trong lập trình xử lý các tình huống như vậy.

Tập hợp các câu lệnh điều khiển trong ngôn ngữ lập trình C được phân chia thành ba loại như sơ đồ dưới đây:



Hình 3.1. Phân cấp các câu lệnh trong chương trình C

3.1. CẤU TRÚC PHÂN NHÁNH

Trong ngôn ngữ lập trình C, các câu lệnh quyết định được sử dụng để thực hiện các hoạt động dựa trên cơ sở điều kiện và khi chương trình gặp phải tình huống phải lựa chọn thực hiện một câu lệnh cụ thể trong số nhiều câu lệnh.

Các câu lệnh quyết định cho phép chúng ta đánh giá một hoặc nhiều điều kiện và sau đó ra quyết định có thực thi một tập hợp các câu lệnh hay không. Bằng cách sử dụng các câu lệnh quyết định, chúng ta có thể thực thi các câu lệnh hoặc chọn một số câu lệnh sẽ được thực thi dựa trên kết quả của điều kiện (cho dù đúng hay sai).

Trong ngôn ngữ lập trình C, có ba cấu trúc của câu lệnh quyết định thường được sử dụng, bao gồm:

- Cấu trúc điều kiện if;
- Cấu trúc lựa chọn switch;
- Câu lệnh goto.

3.1.1. Câu lệnh điều kiện if

Trong quá trình viết chương trình, lập trình viên luôn muốn kiểm soát luồng chương trình, cho phép chương trình đưa ra các quyết định về một đoạn mã chương trình nào đó được thực thi, có giá trị. Câu lệnh if cho phép chúng ta kiểm soát đoạn mã chương trình được thực thi dựa trên một điều kiện đã cho được kiểm tra là đúng hay sai. Một trong những chức năng quan trọng của câu lệnh if cho phép chương trình chọn một hành động thực hiện dựa trên điều kiện đầu vào của người dùng. Chẳng hạn, bằng cách sử dụng câu lệnh if để kiểm tra mật khẩu do người dùng nhập, chương trình quyết định liệu người dùng có được phép truy cập chương trình hay không.

Có bốn dạng câu lệnh điều kiện:

- Câu lệnh điều kiện dạng khuyết;
- Câu lệnh điều kiện dạng đầy đủ;
- Câu lệnh else if;
- Câu lệnh if lồng nhau.

3.1.1.1. Câu lệnh điều kiện dạng khuyết

a. Cú pháp:

```
if (expression)
<statement>;
```

Trong đó:

- **expression:** là biểu thức điều kiện (hay còn gọi là biểu thức logic)
- **statement:** là câu lệnh được thực hiện tùy theo điều kiện, có thể là câu lệnh đơn (một câu lệnh) hoặc câu lệnh ghép (từ hai câu lệnh trở lên, được đặt trong dấu {}).

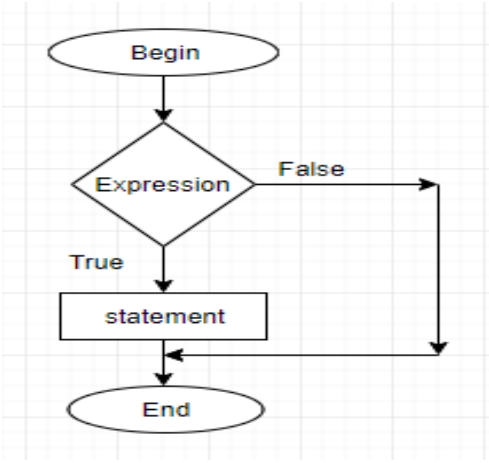
✍ Chú ý: Không đặt dấu chấm phẩy sau câu lệnh if

if (expression);

Trình biên dịch không báo lỗi nhưng khối lệnh không được thực hiện cho dù điều kiện đúng hay sai.

b. Hoạt động: Nếu biểu thức điều kiện (expression) đúng thì thực hiện câu lệnh (statement). Nếu biểu thức điều kiện sai thì máy bỏ qua câu lệnh (statement), thực hiện các câu lệnh tiếp theo trong chương trình.

c. Sơ đồ khối



Hình 3.2. Sơ đồ khối biểu diễn câu lệnh if dạng khuyết

🕮 Ví dụ 3.1

Viết chương trình nhập vào hai số nguyên a, b. Tìm số lớn nhất của hai số và in giá trị lớn nhất đó ra màn hình.

✍ Phác họa lời giải:

Trước tiên ta cho a là giá trị lớn nhất, bằng cách gán a cho max, sau đó so sánh b với max, nếu b lớn hơn max ta gán b cho max và kết quả số lớn nhất là max.

✍ Mô tả bằng ngôn ngữ tự nhiên

- Khai báo 3 biến a, b, max;
- Nhập vào a;
- Nhập vào b;

- Gán a cho max;
- Nếu b > max thì gán b cho max;
- In kết quả max.

✍ **Chương trình**

```
1  #include "stdio.h"
2  main()
3  {
4          int a, b, max;
5          printf("Nhap a= "); scanf("%d", &a);
6          printf("Nhap b= "); scanf("%d", &b);
7          max = a;
8          if (b > max) max = b;
9          printf("Gia tri lon nhat la: %d",max);
10 }
```

✍**Kết quả in ra màn hình:**

Nhap a = 3

Nhap b = 6

Gia tri lon nhat la: 6

🕮 **Ví dụ 3.2**

Viết chương trình nhập vào tuổi của một người, nếu tuổi người đó > 18 tuổi thì in ra màn hình 2 dòng:

Dòng 1: Tuổi của bạn là 18+;

Dòng 2: Bạn đủ tuổi để bỏ phiếu.

Nếu tuổi người đó < 18 tuổi thì không thực hiện gì cả.

✍**Phác họa lời giải:**
- Nhập vào tuổi của một người.
- Kiểm tra: nếu tuổi ≥ 18 thì in ra 2 dòng như trên.
- Nếu tuổi < 18 thì không in gì cả.

✍**Chương trình:**

```
1  #include <stdio.h>
2  #include <conio.h>
3  main()
4  {
```

```
5   int tuoi;
6   char ten[10];
7   printf("Nhap vao ten cua ban:");
8   fflush(stdin);
9   gets(ten);
10  printf("Nhap vao tuoi cua ban %s: ", ten);
11  scanf("%d", &tuoi);
12  if (tuoi >= 18)
13          {
14              printf("Tuoi cua ban %s la 18+ \n",ten);
15              printf("Ban %s  du tuoi de bo phieu",ten);
16          }
17  }
```

Kết quả chương trình

```
Nhap vao ten cua ban: Kien
Nhap vao tuoi cua ban Kien: 21
Tuoi cua ban Kien la 18+
Ban Kien  du tuoi de bo phieu
```

3.1.1.2. Câu lệnh điều kiện đầy đủ

Câu lệnh điều kiện dạng khuyết chỉ thực hiện khi điều kiện đúng, trong trường hợp điều kiện sai, khối lệnh if sẽ được bỏ qua, thực hiện các câu lệnh khác trong chương trình. Trong hầu hết các chương trình, chúng ta đều mong muốn tùy thuộc vào giá trị của điều kiện, chương trình đều thực hiện một hoặc nhiều câu lệnh nào đó.

a. Cú pháp

if (expression)
<statement block 1>;
else
<statement block 2>;

Trong đó:

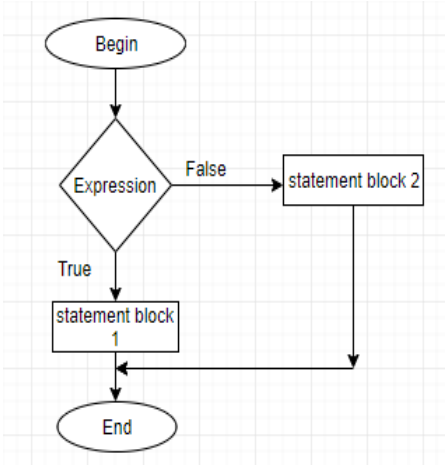
expression: điều kiện, là một biểu thức logic

statement block 1, statement block 2: khối các câu lệnh được thực thi.

b. Hoạt động

Dựa vào biểu thức điều kiện đúng hoặc sai mà chương trình sẽ thực hiện khối lệnh tương ứng. Nếu biểu thức điều kiện (expression) đúng thì thực hiện khối câu lệnh 1 (statement block 1), nếu biểu thức điều kiện sai thì thực hiện khối câu lệnh 2 (statement block 2).

c. Sơ đồ khối



Hình 3.3. Sơ đồ khối biểu diễn câu lệnh if đầy đủ

🕮 Ví dụ 3.3

Viết chương trình nhập vào kí tự c. Kiểm tra xem nếu kí tự nhập vào là kí tự thường trong khoảng từ 'a' đến 'z' thì đổi sang chữ in hoa và in ra, ngược lại in ra thông báo "Kí tự bạn vừa nhập là: c".

✍Phác họa lời giải:

Trước tiên chúng ta phải kiểm tra xem nếu kí tự c thuộc khoảng 'a' và 'z' thì c là kí tự thường. Ta đổi kí tự c thành chữ in hoa bằng cách lấy kí tự c - 32 rồi gán lại cho chính nó (c = c - 32) (vì giữa kí tự thường và in hoa trong bảng mã ASCII cách nhau 32, chẳng hạn mã ASCII của ký tự ‘A’ trong bảng mã là 65, mã ASCII của ký tự ‘B’ là 66, trong khi, mã ASCII của ký tự ‘a’ là 97, mã ASCII của ký tự b là 98, v.v.), sau khi đổi xong bạn in kí tự c ra. Ngược lại, in câu thông báo "Kí tự bạn vừa nhập là: c".

✍Chương trình:

```
1  /* Chuong trinh nhap vao ky tu c, neu c la chu thuong in ra chu IN HOA */
2  #include <stdio.h>
```

```
#include <conio.h>
 main()
{
 char c;
 printf("Nhap vao 1 ki tu: ");
 scanf("%c", &c);
 if(c >= 97 && c <= 122)
        {
                c = c- 32;
                printf("Ki tu hoa la: %c\n", c);
        }
else
   printf("Ki tu ban vua nhap la: %c.\n", c);

 /* Cach 2:
if (c >= 'a' && c <= 'z')
 {
                 c = c- 32;
                 printf("Ki tu hoa la: %c\n", c);
 }
else
   printf("Ki tu ban vua nhap la: %c.\n", c);
*/
   getch();
}
```

Kết quả chương trình

```
Nhap vao 1 ki tu: a
Ki tu hoa la: A
```

3.1.1.3. Câu lệnh điều kiện else… if

Cấu trúc else if thực hiện một trong n nhánh công việc (có n lựa chọn).

a. Cú pháp

```
if (expression 1)
<statement block 1>;
```

else if (expression 2)

<statement block 2>;

…

else if (expression n-1)

<statement block n-1>;

else

<statement block n>;

Trong đó:

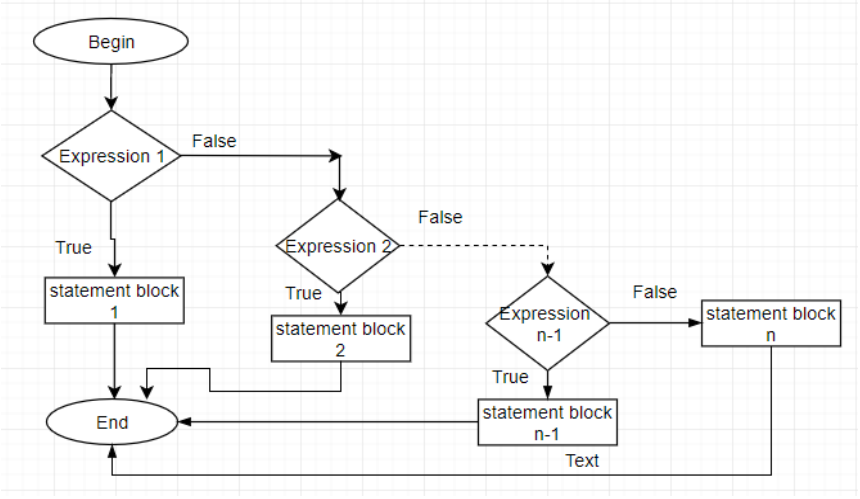
- **expression 1, expression 2,…, expression n**: là các điều kiện;
- **statement block 1, statement block 2,…, statement block n**: Khối lệnh thực hiện.

b. Hoạt động

Nếu biểu thức điều kiện 1 (expression 1) đúng thì chương trình thực hiện khối lệnh 1 (statement block 1) và thoát khỏi cấu trúc if. Ngược lại nếu biểu thức điều kiện 2 (expression 2) đúng thì chương trình thực hiện khối lệnh 2 (statement block 2) và thoát khỏi cấu trúc if, v.v.

Tương tự như vậy, nếu biểu thức điều kiện n - 1 (expression n - 1) đúng thì chương trình thực hiện khối lệnh n - 1 (statement block n - 1) đồng thời thoát khỏi cấu trúc if, ngược lại thực hiện khối lệnh n (statement block n).

c. Sơ đồ khối



Hình 3.4. Sơ đồ khối biểu diễn câu lệnh else if

🕮 Ví dụ 3.4

Viết chương trình giải phương trình bậc 2: $ax^2 + bx + c = 0$.

✍ Phác họa lời giải:

Sử dụng biến delta để xác định biến của chương trình:

delta = b*b - 4 * a * c;

Xét delta

+ Nếu delta > 0 thì phương trình có 2 nghiệm phân biệt:

x1 = (-b + sqrt(delta))/(2*a)

x2 = (-b - sqrt(delta))/(2*a)

+ Nếu delta = 0 thì phương trình có 1 nghiệm kép.

x1 = x2 = -b/(2*a)

+ Nếu delta < 0 thì phương trình vô nghiệm.

✍ Chương trình:

```
1  #include<stdio.h>
2  #include<math.h>
3  int main()
4  {
5     float delta,a,b,c,x1,x2;
6     printf("a = "); scanf("%f",&a);
7     printf("b = "); scanf("%f",&b);
8     printf("c = "); scanf("%f",&c);
9     delta = b * b - 4 * a * c;
10    if (delta < 0)
11       printf("ptvn\n");
12    else if (delta == 0)
13                    {
14                    printf("\nPhuong trinh co nghiem kep:\n");
15                      printf("x1 = x2 = %8.2f",-b/(2*a));
16                    }
17        else
```

```
18          {
19    printf("\nPhuong trinh co 2 nghiem phan biet:\n ");
20              x1 = (-b + sqrt(delta))/(2*a);
21                    x2 = (-b - sqrt(delta))/(2*a);
22              printf("\nx1 = %8.2f",x1);
23              printf("\nx2 = %8.2f",x2);
24          }
25  }
```

✍ Kết quả chương trình:

```
a = 1
b = 3
c = 2
Phuong trinh co 2 nghiem phan biet:
x1 = -1.00
x2 = -2.00
```

🕮 Ví dụ 3.5

Viết chương trình xếp loại kết quả học tập của sinh viên dựa vào điểm trung bình khoá học (dtb) theo tiêu chí sau:

① Nếu $0.0 \leq dtb < 4.0$: Xếp loại Yếu;

② Nếu $4.0 \leq dtb < 5.5$: Xếp loại Trung bình yếu;

③ Nếu $5.5 \leq dtb < 7.0$: Xếp loại Trung bình;

④ Nếu $7.0 \leq dtb < 8.5$: Xếp loại Khá;

⑤ Nếu $dtb \geq 8.5$: Xếp loại Giỏi.

✍ Phác họa lời giải:

- Chương trình yêu cầu nhập điểm trung bình của sinh viên theo khóa học;
- Căn cứ vào điểm trung bình để xếp loại:
+ Nếu điểm trung bình < 4.0: Xếp loại Yếu;
+ Ngược lại, nếu điểm trung bình < 5.5: Xếp loại Trung bình yếu;
+ Nếu điểm trung bình < 7: Xếp loại Trung bình;
+ Nếu điểm trung bình < 8.5: Xếp loại Khá;
+ Nếu điểm trung bình ≥ 8.5: Xếp loại Giỏi.

- Chương trình sử dụng 4 vòng lặp if … else để xếp loại.

Chương trình:

```
#include <stdio.h>
#include <math.h>
int main()
{
float dtb;
printf("\nNhap diem tb: "); scanf("%f",&dtb);
if(dtb < 4)
        {
    printf("\nLoai yeu");
        }
else if(dtb < 5.5)
        {
         printf("\n Loai trung binh yeu");
        }
else if(dtb < 7)
        {
         printf("\n Loai trung binh");
        }
else if(dtb < 8.5)
        {
    printf("\nLoai kha");
        }
        else
    printf("\nLoai gioi");
}
```

Kết quả chương trình:

```
Nhap diem tb: 7
Loai kha
```

3.1.1.4. Câu lệnh điều kiện lồng nhau

Khi một câu lệnh *if...else* xuất hiện bên trong một câu lệnh "if" hoặc "else" thì được gọi là câu lệnh điều kiện lồng nhau.

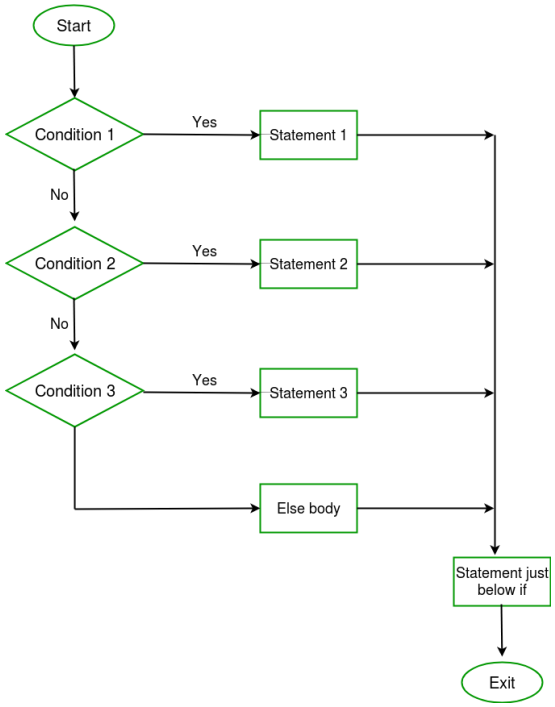
Khi một loạt các quyết định được yêu cầu, cấu trúc if else lồng nhau sẽ được sử dụng.

a. Cú pháp:

```
if (expression 1)
   statement block 1;
else if (expression 2)
   statement block 2;
…
else
   statement  block n;
```

b. Hoạt động: Câu lệnh if else lồng nhau quyết định chọn một trong nhiều lựa chọn. Trình tự thực hiện được thực hiện từ trên xuống dưới. Ngay khi một trong các điều kiện trong khối if else được kiểm soát là đúng, câu lệnh liên quan đến điều kiện đó được thực thi và phần còn lại của khối if else lồng sẽ bị bỏ qua. Nếu không có điều kiện nào là đúng, câu lệnh sau else cuối cùng sẽ được thực hiện.

c. Sơ đồ khối



Hình 3.5. Sơ đồ khối biểu diễn câu lệnh if else lồng nhau

Lưu ý:

Khi số từ khóa if bằng số từ khóa else thì ta dễ dàng xác định được từng cặp *if - else* tương ứng. Trong trường hợp số từ khóa else ít hơn số từ khóa if thì else sẽ được gắn với if không có else gần nhất trước đó. Vì vậy, khi gặp những lệnh if không có else, chúng ta phải đặt chúng trong những khối lệnh rõ ràng để tránh bị hiểu sai câu lệnh.

🕮 Ví dụ 3.6

```
if (n > 0)
   if (a > b)
    x = a;
 else
   x = b;
  …
```

Mặc dù chúng ta viết lệnh else thẳng hàng với if (n > 0), nhưng lệnh else ở đây được hiểu đi kèm với if (a > b), vì nó nằm gần với if (a > b) nhất và if (a > b) chưa có else. Để dễ nhìn và dễ hiểu hơn chúng ta viết lại như sau:

```
if (n > 0)
   if (a > b)
    x = a;
   else
    x = b;
  …
```

Còn nếu chúng ta muốn lệnh else là của if (n > 0) thì chúng ta phải đặt if (a > b) x = a trong một khối lệnh, viết lại như sau:

```
if (n > 0)
  {
   if (a > b)
    x = a;
  }
else
   x = b;
```

🕮 Ví dụ 3.7

Viết chương trình nhập vào điểm trung bình khóa học của sinh viên (0≤ dtb ≤10) theo tiêu chí sau:

① Nếu 0.0 ≤ dtb ≤ 4.0: Xếp loại Yếu;

② Nếu 4.0 ≤ dtb ≤ 5.5: Xếp loại Trung bình - Yếu;

③ Nếu 5.5 ≤ dtb ≤ 7.0: Xếp loại Trung bình;

④ Nếu 7.0 ≤ dtb ≤ 8.5: Xếp loại Khá;

⑤ Nếu dtb ≥ 8.5: Xếp loại Giỏi.

✍Phác họa lời giải:

Điểm số nhập vào nếu hợp lệ (0 ≤ đtb ≤ 10), bạn tiếp tục công việc xếp loại, ngược lại thông báo *"Nhập điểm không hợp lệ*". Việc xếp loại bạn sử dụng cấu trúc else if như ví dụ 3.5.

✍Chương trình

```
#include<stdio.h>
#include<math.h>
int main()
{
float dtb;
printf("\nNhap diem tb: "); scanf("%f",&dtb);
if (dtb >= 0 && dtb <= 10)
 if(dtb < 4)
                {
    printf("\nLoai yeu");
                }
 else if(dtb < 5.5)
                {
                 printf("\n Loai trung binh yeu");
                }
 else if(dtb < 7)
                {
                 printf("\n Loai trung binh");
                }
 else if(dtb < 8.5)
```

```
21              {
22      printf("\nLoai kha");
23              }
24              else
25      printf("\nLoai gioi");
26  else
27    printf("\nNhap diem khong hop le");
28  }
```

✍Kết quả chương trình:

Trường hợp 1:

Nhap diem tb: 11

Nhap diem khong hop le

Trường hợp 2:

Nhap diem tb: 8

Loai gioi

✍Giải thích chương trình:

Trong chương trình trên cấu trúc else if được lồng vào trong cấu trúc dạng 2 (từ dòng 7 đến 26), trong cấu trúc else if ta không cần đặt trong khối vì tất cả các if trong cấu trúc này đều có else, nên câu lệnh sau else cuối cùng: printf("\nNhap diem khong hop le") thuộc về khối lệnh: if (dtb >= 0 && dtb <= 10).

Giả sử trong cấu trúc else if không có dòng 25 (else printf("\nLoai gioi")) thì khi đó dòng 27 (else printf("Nhap diem khong hop le.\n")) được chương trình hiểu nằm trong cấu trúc else if, mà không thuộc về câu lệnh if (dtb>=0 && dtb<= 10). Trong trường hợp đó bạn cần phải đặt cấu trúc else if vào trong {}, thì khi đó dòng else printf("\nNhap diem khong hop le") mới được hiểu là phần sau của câu lệnh if (dtb >= 0 && dtb <= 10).

🕮 Ví dụ 3.8

Viết chương trình giải và biện luận phương trình: $ax^2 + bx + c = 0$.

✍Phác họa lời giải:

Nếu a = = 0: Phương trình trở thành phương trình bậc nhất bx + c = 0.

Nếu b = = 0: Phương trình được rút gọn c = 0, có hai khả năng:

Nếu c = = 0: Phương trình vô số nghiệm;

Nếu c != 0: Phương trình vô nghiệm;

Nếu b != 0: Phương trình có một nghiệm x = $\frac{-c}{b}$;

Nếu a ! = 0: Giải phương trình bậc hai:

Tính delta = b * b - 4 * a * c.

Xét delta

Nếu delta > 0: Phương trình có hai nghiệm phân biệt:

x1 = (-b + sqrt(delta))/(2*a)

x2 = (-b - sqrt(delta))/(2*a)

Nếu delta = =0: Phương trình có một nghiệm kép

x1 = x2 = $\frac{-b}{2*a}$

Nếu delta < 0: Phương trình vô nghiệm.

✍Chương trình:

```
1    #include<stdio.h>
2    #include<math.h>
3    int main()
4    {
5      float delta,a,b,c,x1,x2;
6      printf("a = ");scanf("%f",&a);
7      printf("b = ");scanf("%f",&b);
8      printf("c = ");scanf("%f",&c);
9      if (a == 0)
10           {
11           if (b == 0)
12                  {
13           if (c == 0) printf("\nPhuong trinh vo so nghiem");
14                  else printf("\nPhuong trinh vo nghiem");
15                  }
16           else
17                  {
18                   float x = -c/b;
```

```
19                    printf("\nPhuong trinh co 1 nghiem x =
    %.1f",x);
20                   }
21            }
22      else
23            {
24              delta = b * b - 4 * a * c;
25              if (delta < 0)
26           printf("ptvn\n");
27         else if (delta == 0)
28                              {
29                    printf("\nPhuong trinh co nghiem kep:\n");
30                               printf("x1 = x2 = %.2f",-b/(2*a));
31                              }
32              else
33              {
34         printf("\nPhuong trinh co 2 nghiem phan biet:\n ");
35               x1 = (-b + sqrt(delta))/(2*a);
36                     x2 = (-b - sqrt(delta))/(2*a);
37               printf("x1 = %.2f\n",x1);
38               printf("x2 = %.2f",x2);
39              }
40       }
41    }
```

Kết quả chương trình:

Trường hợp 1:

a = 0

b = 0

c = 1

Phuong trinh vo nghiem

Trường hợp 2:

a = 1

b = 2

c = 1

Phuong trinh co nghiem kep:

x1 = x2 = -1

Trường hợp 3:

a = 1

b = 3

c = 2

Phuong trinh co hai nghiem phan biet:

x1 = -1

x2 = -2

🕮 Ví dụ 3.9

Viết chương trình thực hiện các yêu cầu sau:

Nhập vào 3 số thực a, b, c từ bàn phím.

a. Kiểm tra a, b, c có lập thành ba cạnh của tam giác không?

b. Nếu có hãy kiểm tra đây là tam giác cân, đều, vuông hay tam giác thường. Tính chu vi và diện tích của tam giác. In kết quả ra màn hình.

✍Phác họa lời giải:

a, b, c là 3 cạnh tam giác nếu a, b, c > 0 và tổng hai cạnh luôn lớn hơn một cạnh (a + b > c và b + c > a và a + c > b).

✍Chương trình:

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <math.h>
4   main()
5   {
6   float a, b,c;
7           printf("\nNhap canh a= "); scanf("%f",&a);
8           printf("\nNhap canh b= "); scanf("%f",&b);
9           printf("\nNhap canh c= "); scanf("%f",&c);
10  if (a + b > c && b + c > a && c + a > b && a > 0 && b > 0 && c
    >0)
11  {
12          if (a == b && b == c) printf("\nTam giac deu");
```

```
13          else if (a == b ||b == c|| c == a) printf(" \nTam giac can");
14          else if (a * a + b * b == c * c ||a * a + c * c == b * b||c * c +
    b * b == a * a) printf("\nTam giac vuong");
15          else printf("\nTam giac thuong");
16          float p = (a + b + c)/2;
17          float s = sqrt(p * (p - a) * (p - b) * (p - c));
18          printf("\nChu vi tam giac: %.2f",2 * p);
19          printf("\nDien tich tam giac %.2f",s);
20  }
21  else printf("\n khong la tam giac");
22  }
```

✍Kết quả chương trình:

```
Nhap canh a= 3
Nhap canh b= 4
Nhap canh c= 5
Tam giac vuong
Chu vi tam giac: 12.00
Dien tich tam giac 6.00
```

🕮 Ví dụ 3.10

Nhập vào số nguyên dương n (n > 0), kiểm tra xem n có phải là số chính phương không?

Số chính phương là số mà căn bậc hai của nó là một số nguyên, chẳng hạn các số 4, 9, 16, v.v. là các số chính phương.

✍Chương trình:

```
1   #include <stdio.h>
2   #include <math.h>
3   int main()
4   {
5   unsigned n, m;
6   printf("Nhap n: ");
7   scanf("%d", &n);
```

```
 8      m=(unsigned)sqrt(n);
 9      if (n == m * m) printf("%u la so chinh phuong",n);
10      else printf("%u khong la so chinh phuong",n);
11      return 0;
12      }
```

Kết quả chương trình:

```
Nhap n: 16
16 la so chinh phuong
```

3.1.2. Cấu trúc switch

Lệnh switch cũng giống cấu trúc else if, thực hiện một trong n nhánh công việc (có n lựa chọn), nhưng switch mềm dẻo hơn và linh động, dễ nhìn, chương trình sáng sủa hơn. Tuy nhiên, câu lệnh switch cũng có mặt hạn chế là kết quả của biểu thức phải là giá trị hằng nguyên hoặc kí tự. Một bài toán sử dụng lệnh switch thì cũng có thể sử dụng if, nhưng ngược lại còn tùy thuộc vào giải thuật của bài toán.

a. Cú pháp:

```
switch (Variable/Expression)
{
   case  <value 1>: statement 1; break;
   case  <value 2>: statement 2; break;
   …
   case  <value n>: statement n; break;
   [default: statement n+1; break;]
}
```

Trong đó:

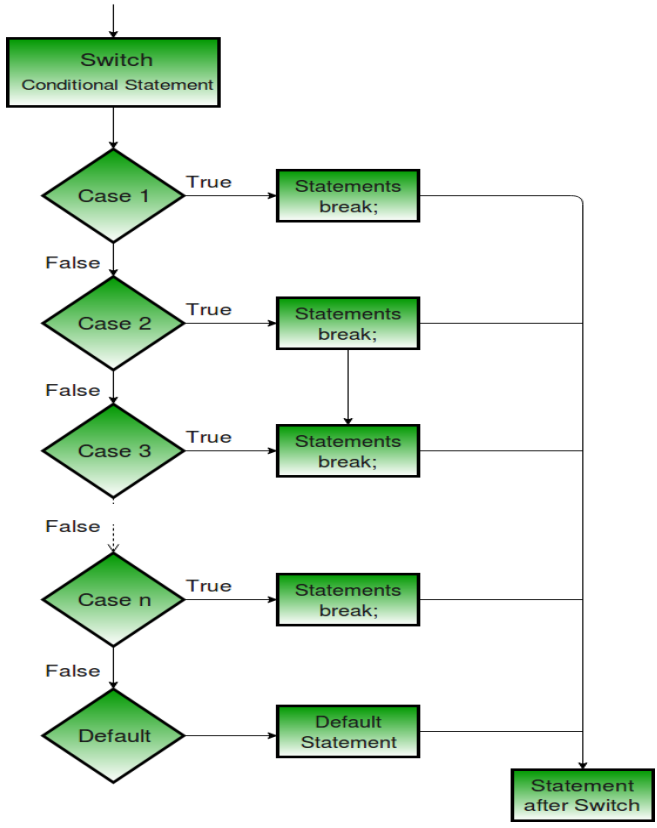
- Các từ khóa switch, case, break là viết thường.
- **Variable/Expression:** Biến/biểu thức, phải trả về là hằng nguyên hoặc kí tự;
- **value 1, value 2,…, value n:** giá trị 1, giá trị 2,… giá trị n, là các hằng nguyên hoặc kí tự, là các giá trị khác nhau của biến/biểu thức;
- **statement 1, statement 2,.., statement n:** Các khối lệnh 1, khối lệnh 2, …, khối lệnh n.

b. Hoạt động

Máy sẽ lần lượt so sánh biến/biểu thức với các giá trị 1, giá trị 2, giá trị 3,…, giá trị n.

- Nếu giá trị của biến/biểu thức bằng value i thì máy sẽ nhảy tới thực hiện lệnh statement thứ i (i = 1, 2,.., n), sau đó thực hiện lệnh break để thoát khỏi câu lệnh switch.
- Giá trị của biến/biểu thức khác tất cả các value i (i = 1, 2, 3,…, n) thì:
  - Nếu có từ khóa default thì máy sẽ thực hiện lệnh sau default (lệnh n + 1).
  - Nếu không có default thì máy sẽ ra khỏi cấu trúc switch.

c. Sơ đồ khối



Hình 3.6. Sơ đồ khối biểu diễn câu lệnh switch

Một số chú ý:

① Cấu trúc switch có thể lồng vào nhau

🕮 Ví dụ 3.11

```
1   #include <stdio.h>
2   #include <conio.h>
3   main()
4   {
5           int a, b;
6           printf("\n Nhap a = "); scanf("%d",&a);
7           printf("\n Nhap b = "); scanf("%d",&b);
8   switch (a)
9           {
10                  case 1 : printf("Mot"); break;
11                  case 2 : switch (b)
12                          {
13                                  case 1 : printf("A"); break;
14                                  case 2 : printf("B"); break;
15                          } break;
16                  case 3 : printf("Ba"); break;
17          }
18  }
```

☺ Các giá trị trong mỗi trường hợp phải khác nhau

🕮 Ví dụ 3.12

```
1     #include <stdio.h>
2     #include <conio.h>
3     main()
4     {
5             int a;
6             printf("\n Nhap a= "); scanf("%d",&a);
7     switch (a)
8     {
9             case 1 : printf("Mot"); break;
10            case 1 : printf("MOT"); break;
```

```
11              case 2 : printf("Hai"); break;
12              case 3 : printf("Ba"); break;
13              case 1 : printf("1"); break;
14              case 1 : printf("mot"); break;
15      }
```

✍Kết quả chương trình: báo lỗi ở ngay dòng 9, 10, 13, 14

③ Nếu các lệnh giống nhau thì viết gộp:

🕮 Ví dụ 3.13

```
1       #include <stdio.h>
2       #include <conio.h>
3       main()
4       {
5               int a;
6               printf("\n Nhap a= "); scanf("%d",&a);
7       switch (a)
8       {
9               case 1 : printf("\n Hello"); break;
10              case 2 : printf("\n Hello"); break;
11              case 3 : printf("\n Good bye"); break;
12              case 4 : printf("\n Good bye"); break;
13              case 5 : printf("\n Good bye"); break;
14              case 6 : printf("\n Good bye"); break;
15      }
16      }
```

✍ Chương trình trên ở dòng 9, 10 cùng in ra một dòng Hello. Các dòng 11, 12, 13, 14 cùng in ra dòng Good bye, do đó các case có các lệnh giống nhau thì ta bỏ trống không viết gì hoặc chỉ viết dấu chấm phẩy ";". Câu lệnh sẽ được viết ở case cuối có lệnh giống.

Chương trình trên sẽ được viết lại như sau:

```
1       #include <stdio.h>
2       #include <conio.h>
3       main()
4       {
```

```
5               int a;
6               printf("\n Nhap a= "); scanf("%d",&a);
7       switch (a)
8       {
9               case 1 : ;
10              case 2 : printf("\n Hello"); break;
11              case 3 :
12              case 4 :
13              case 5 : ;
14              case 6 : printf("\n Good bye"); break;
15      }
16      }
```

🕮 Ví dụ 3.14

Viết chương trình nhập vào 1 biến ký tự color rồi in ra thông báo như sau:

- RED, nếu color = 'R' hoặc color = 'r'.
- GREEN, nếu color = ‘G' hoặc color = 'g'.
- BLUE, nếu color = 'B' hoặc color = 'b'.
- ORANGE, nếu color = ‘O’ hoặc color = ‘o’.
- PURPLE, nếu color = ‘P’ hoặc color = ‘p’.
- BLACK, nếu color có giá trị khác.

✍Phác họa lời giải

Bài toán có 6 lựa chọn, chúng ta có thể sử dụng cấu trúc else if và so sánh với cấu trúc switch.

✍Chương trình

Cách 1: Sử dụng cấu trúc else if

```
1   #include <stdio.h>
2   #include <conio.h>
3   int main()
4   {
5       char color;
```

```
6       printf("nhap vao mot ky tu: ");  scanf("%c",&color);
7       if(color == 'R'||color == 'r') printf("RED");
8       else if (color == 'G'||color == 'g')
9                       printf("GREEN");
10      else if (color == 'B'||color == 'b')
11                      printf("BLUE");
12      else if (color == 'O'||color == 'o')
13                      printf("ORANGE");
14      else if (color == 'P'||color == 'p')
15                      printf("PURPLE");
16              else printf("BLACK");
17      getch();
18  }
```

Cách 2: Sử dụng cấu trúc swtich

```
1   #include <stdio.h>
2   #include <conio.h>
3   int main()
4   {
5       char color;
6       printf("nhap vao mot ky tu: ");  scanf("%c",&color);
7       switch(color)
8       {  case 'R': case ‘r': printf("RED"); break;
9          case 'G': case ‘g’: printf("GREEN");break;
10         case 'B': case 'b': printf("BLUE");break;
11         case 'O': case 'o': printf("ORANGE");break;
12         case 'P': case 'p': printf("PURPLE");break;
13         default : printf("BLACK");break;
14             }
15      getch();
16  }
```

✍Nhận xét:

Qua hai chương trình của ví dụ 3.14 ở trên ta thấy nếu một bài toán có nhiều lựa chọn thì việc lựa chọn cấu trúc switch sẽ dễ nhìn và linh động hơn.

🕮 Ví dụ 3.15

Viết chương trình nhập vào một ngày/tháng/năm, in ra màn hình ngày kế tiếp (ngày/tháng/năm) của ngày vừa nhập.

✍Phác họa lời giải

- *Đầu tiên nhập vào một ngày/tháng/năm*

- *Kiểm tra tháng:*

+ Các tháng 1, 3, 5, 7, 8, 10, 12 có số ngày = 31 ngày.

+ Các tháng 4, 6, 9, 11 có số ngày = 30 ngày.

+ Tháng 2 và năm nhuận có số ngày = 29 ngày.

+ Tháng 2 và không là năm nhuận có số ngày = 28 ngày.

(Năm nhuận là năm chia hết cho 4 và không chia hết cho 100 hoặc chia hết cho 400).

- *Nếu ngày nhập vào < ngày của tháng thì tăng ngày lên.*

+ Nếu ngày nhập vào ≥ ngày của tháng, đưa ra yêu cầu kiểm tra tháng.

+ Nếu tháng < 12, thực hiện gán ngày = 1 và tăng tháng lên 1 đơn vị.

+ Nếu tháng >= 12, thực hiện gán ngày = tháng = 1, tăng năm lên 1 đơn vị.

✍Chương trình

```
1   #include<stdio.h>
2   #include<conio.h>
3   int main()
4   {
5           int ng,th,n,d;
6           printf("\n Nhap nam: "); scanf("%d",&n);
7           printf("\n Nhap thang: "); scanf("%d",&th);
8           printf("\n Nhap ngay: "); scanf("%d",&ng);
9           switch(th)
10          {
```

```
11  case 1: case 3: case 5: case 7: case 8:  case 10: case 12: d = 31;
12              break;
13              case 9: case 4: case 6: d = 30; break;
14              case 2:
15               {
16               if(n % 4 == 0 && n % 100!= 0||n % 400 == 0)
17                 d = 29;
18               else d = 28;
19               };break;
20              }
21              if (ng < d)
22              {
23                      ng++;
24              }
25              else if (th < 12)
26              {
27                      ng = 1; th++;
28              }
29              else
30              {
31                      ng = th = 1; n++;
32              }
33              printf("\nNgay ke tiep: %d- %d - %d", ng,th,n);
34  getch();
35  return 0;
36  }
```

Kết quả chương trình

```
Nhap nam: 2019
Nhap thang: 3
Nhap ngay: 31
Ngay ke tiep: 1- 4 - 2019
```

3.1.3. Toán tử goto và nhãn

3.1.3.1. Cú pháp

goto label;

label: statement;

Trong đó:

label: tên nhãn do người dùng tự đặt;
statement: câu lệnh thực hiện.

3.1.3.2. Hoạt động

Khi gặp lệnh goto chương trình sẽ nhảy đến lệnh nơi chứa từ nhãn và thực hiện lệnh sau từ khóa nhãn.

✍Chú ý:

- Khi gặp lệnh goto chương trình sẽ nhảy đến lệnh nơi chứa từ nhãn và thực hiện lệnh sau từ khóa nhãn.

- Câu lệnh goto và nhãn cần nằm trong một hàm, có nghĩa là toán tử goto chỉ cho phép nhảy từ vị trí này đến vị trí khác trong thân một hàm và không thể dùng để nhảy từ một hàm này sang một hàm khác.

- Không cho phép dùng toán tử goto để nhảy từ ngoài vào trong một khối lệnh. Tuy nhiên, việc nhảy từ trong một khối lệnh ra ngoài là hoàn toàn hợp lệ.

🕮 Ví dụ 3.16

Xem xét lại ví dụ 3.15:

Viết chương trình nhập vào một ngày/tháng/năm (với ràng buộc cho ngày chỉ nhập từ 1 đến ngày theo tháng (nhận một trong các giá trị 28, 29, 30, 31), tháng nhập giá trị từ 1 đến 12, năm chỉ nhập năm có bốn chữ số). Tìm ngày kế tiếp (ngày/tháng/năm) của ngày vừa nhập.

✍Phác họa lời giải

- Nhập vào tháng (kiểm soát giá trị từ 1 đến 12).
- Nhập vào năm (kiểm soát giá trị từ 1000 đến 9999).
- Kiểm tra tháng để trả ngày theo tháng:

+ Các tháng 1, 3, 5, 7, 8, 10, 12: Số ngày d = 31 ngày.

+ Các tháng 4, 6, 9, 11: Số ngày d = 30 ngày.

+ Tháng là tháng 2 và năm nhuận thì có d = 29 ngày.

+ Tháng là tháng 2 và không là năm nhuận có số ngày d = 28 ngày.

(Năm nhuận là năm chia hết cho 4 và không chia hết cho 100 hoặc chia hết cho 400).

- Nhập vào ngày ($1 \leq$ số ngày $\leq$ d)

+ Nếu ngày nhập vào < d thì tăng ngày lên.

+ Nếu ngày nhập vào $\geq$ d, kiểm tra lại tháng:

* Nếu tháng < 12, thì thiết lập ngày = 1 và tăng tháng lên một đơn vị.

* Nếu tháng $\geq$12, thì ngày = tháng = 1, tăng năm lên một đơn vị.

Chương trình

```
#include <stdio.h>
#include <conio.h>
int main()
{
        int ng, th, n, d;
        //Nhãn tn
        tn:
        printf("\n Nhap nam: "); scanf("%d",&n);
        if (n < 1000 || n > 9999)
        {
        printf("\n Nhap lai nam: ");
        //Nhay tu trong ra ngoai
        goto tn;
        }
        tt:
        printf("\n Nhap thang: "); scanf("%d",&th);
        if (th < 1||th > 12)
        {
        printf("\n Nhap lai thang: ");
        goto tt;
        }
        switch(th)
        {
```

```
24          case 1: case 3: case 5: case 7: case 8: case 10: case 12: d = 31;
25                  break;
26                  case 9: case 4: case 6: d = 30; break;
27                  case 2:
28                   {
29                  if(n % 4 == 0 && n % 100!= 0||n % 400 == 0)
30                              d = 29;
31                          else d = 28;
32                   };break;
33          }
34          tng:
35          printf("\n Nhap ngay: "); scanf("%d",&ng);
36          if (ng < 1||ng > d)
37          {
38                  printf("\n Nhap lai ngay");
39                  goto tng;
40          }
41          printf("\n Ngay vua nhap %d-%d-%d\n",ng,th,n);
42          if (ng < d)
43          {
44                  ng++;
45          }
46          else if (th < 12)
47          {
48                  ng = 1; th++;
49          }
50          else
51          {
52                  ng = th = 1; n++;
53          }
54          printf("\nNgay ke tiep: %d- %d - %d", ng,th,n);
55  getch();
56  return 0;
57  }
```

✍ Kết quả chương trình:

Nhap nam: 200

Nhap lai nam:

Nhap nam: 2019

Nhap thang: 14

Nhap lai thang:

Nhap thang: 3

Nhap ngay: 26

Ngay vua nhap 26-3-2019

Ngay ke tiep: 27- 3 - 2019

🕮 Ví dụ 3.17

Tính tổng s = 1 + 2 + 3 +.... + 10

✍Phác họa lời giải

Khai báo biến tổng s, khởi tạo giá trị ban đầu s = 0.

Để tính tổng các số từ 1 đến 10, ta sử dụng một biến i. Ban đầu gán i = 0, cho i chạy từ 1 đến 10, ứng với mỗi i ta cộng vào s như sau:

Với i = 0, $s_0 = 0$, tăng i = i + 1 = 1.

Với i = 1, $s_1 = 0 + 1 = s_0 + 1 = s_0 + i$, tăng i = i + 1 = 2;

Với i = 2, $s_2 = 0 + 1+ 2 = s_1 + 2 = s_1 + i$, tăng i = i + 1 =3.

…

Với i = 10, $s_{10} = (0 + 1 + 2 + 3 + … + 9) + 10 = s_9 + 10 = s_9 + i$, tăng i = 11, dừng chương trình.

Như vậy:

+ Với $i \leq 10$ thì thực hiện tính tổng, nếu i > 10 thì dừng chương trình.

+ Với mỗi giá trị biến i ta có hai biểu thức giống nhau: i + 1 và s + i. Ta dùng toán tử goto để lặp cho mỗi giá trị i cho đến khi i > 10 thì dừng.

✍Chương trình

```
1    #include<stdio.h>
2    #include<conio.h>
3    main()
```

```
{
    int s, i;
    i = s = 0;
    tong:
    ++i;
    s = s + i;
    if (i < 10) goto tong;
    printf("\n Tong s = %d",s);
}
```

✍**Kết quả chương trình**

Tong s = 55

3.2. CẤU TRÚC VÒNG LẶP (FOR, WHILE, DO... WHILE)

Xem xét ví dụ sau đây:

📖 **Ví dụ 3.18**

Viết chương trình in ra 10 dòng *"Chao cac ban sinh vien K70, Dai học Cong nghe GTVT"* ra màn hình.

✍**Phác họa lời giải**

Với mỗi dòng được in ra màn hình, sử dụng một câu lệnh printf(). Như vậy để in ra 10 dòng *"Chao cac ban sinh vien K70, Dai học Cong nghe GTVT"*, chúng ta cần sử dụng 10 câu lệnh printf() như sau:

```
Printf("Chao cac ban sinh vien K70, Dai học Cong nghe GTVT");
Printf("Chao cac ban sinh vien K70, Dai học Cong nghe GTVT");
Printf("Chao cac ban sinh vien K70, Dai học Cong nghe GTVT");
Printf("Chao cac ban sinh vien K70, Dai học Cong nghe GTVT");
Printf("Chao cac ban sinh vien K70, Dai học Cong nghe GTVT");
Printf("Chao cac ban sinh vien K70, Dai học Cong nghe GTVT");
Printf("Chao cac ban sinh vien K70, Dai học Cong nghe GTVT");
Printf("Chao cac ban sinh vien K70, Dai học Cong nghe GTVT");
Printf("Chao cac ban sinh vien K70, Dai học Cong nghe GTVT");
Printf("Chao cac ban sinh vien K70, Dai học Cong nghe GTVT");
```

Nhược điểm:

Nếu in ra màn hình 10 dòng, mọi việc trong tầm kiểm soát, nhưng nếu cần in ra 1000 dòng, sẽ gây ra sự khó chịu nhất định, bởi do số dòng code lặp đi lặp lại.

Chương trình sẽ dài dòng, những lệnh giống nhau bị viết lặp đi lặp lại nhiều lần sẽ không rõ ràng chương trình, khó tái sử dụng code, rườm rà.

Giải pháp đưa ra là sử dụng vòng lặp. Vòng lặp có tác dụng giúp chương trình chạy nhanh, gọnvà tái sử dụng được code.

Trong một vòng lặp, một chương trình thực hiện chuỗi các câu lệnh nhiều lần cho đến điều kiện bị sai. Một vòng lặp bao gồm hai phần, phần thân của vòng lặp và câu lệnh điều khiển. Câu lệnh điều khiển là sự kết hợp của một số điều kiện, điều khiển sự thực thi của phần thân vòng lặp cho đến khi điều kiện được chỉ định trở thành sai.

Cấu trúc lặp bao gồm:

- Vòng lặp for;
- Vòng lặp while;
- Vòng lặp do... while.

3.2.1. Cấu trúc vòng lặp For

3.2.1.1. Cú pháp

for (Exp1; Exp2; Exp3)

statement;

Trong đó:

- **Exp1**: Biểu thức khởi tạo cho biến;
- **Exp2**: Biểu thức điều kiện - điều kiện lặp;
- **Exp3**: Bước nhảy - thay đổi giá trị của biến vòng lặp.
- **statement**: câu lệnh được thực thi, có thể là câu lệnh đơn, hoặc câu lệnh ghép.

Exp1, Exp2, Exp3 có thể vắng mặt trong câu lệnh nhưng các dấu chấm phẩy phải xuất hiện trong cấu trúc.

3.2.1.2. Hoạt động

Bước 1: Thực hiện exp1.

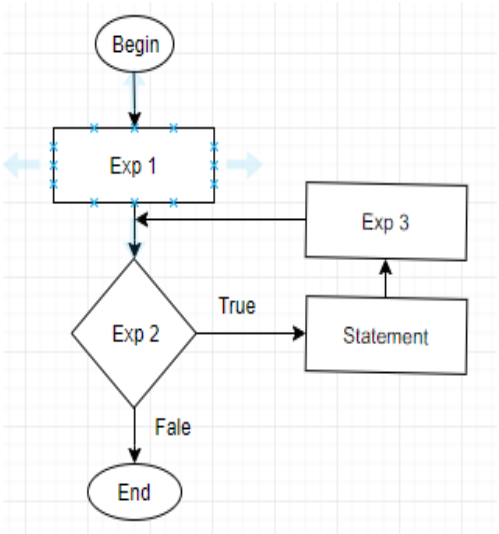
Bước 2: Kiểm tra điều kiện exp2.

- Nếu điều kiện đúng thì thực hiện lệnh statement và sang bước 3.
- Nếu điều kiện sai sang bước 4.

Bước 3: Thực hiện exp3 và quay lại bước 2.

Bước 4: Thoát khỏi vòng lặp.

3.2.1.3. Sơ đồ khối



Hình 3.7. Sơ đồ khối biểu diễn vòng lặp for

🕮 Ví dụ 3.18

Tính tổng s = 1 + 2 + 3 + ... + 10

✍Phác họa lời giải

Khai báo biến tổng s, khởi tạo giá trị ban đầu s = 0.

Để tính tổng các số từ 1 đến 10, ta sử dụng một biến i. Ban đầu gán i = 0, cho i chạy từ 1 đến 10, ứng với mỗi i ta cộng vào s như sau:

Với i = 0, $s_0 = 0$, tăng i = i + 1 = 1.

Với i = 1, $s_1 = 0 + 1 = s_0 + 1 = s_0 + i$, tăng i = i + 1 = 2;

Với i = 2, $s_2 = 0 + 1 + 2 = s_1 + 2 = s_1 + i$, tăng i = i + 1 = 3.

…

Với i = 10, $s_{10} = (0 + 1 + 2 + 3 + … + 9) + 10 = s_9 + 10 = s_9 + i$, tăng i = 11, dừng chương trình.

Như vậy:

+ Với $i \leq 10$ thì thực hiện tính tổng, nếu i > 10 thì dừng chương trình.

+ Với mỗi giá trị biến i ta có hai biểu thức giống nhau: i + 1 và s + i.

✍Chương trình:

```
1       #include<stdio.h>
2           #include<conio.h>
3           main()
4           {
5              int s = 0,i;
6              for(i = 1;i <= 10; i++)
7               s = s + i;
8              printf("\n Tong s = %d",s);
9           }
```

✍Kết quả chương trình

Tong s = 55

✍Chú ý:

① Cấu trúc for có thể lồng vào nhau

🕮 Ví dụ 3.19

Viết chương trình in ra hình chữ nhật sao như sau:

```
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
```

✍Phác họa lời giải

Hình chữ nhật sao có 5 hàng và 10 cột là ma trận 5 × 10.

Đối với 5 hàng: ta dùng biến chạy i từ 0 đến 4.

Với i = 0, hàng đầu thì phải in ra được 10 cột *, ta dùng biến chạy j, duyệt cột từ 0 đến 9.

Với i = 1 hàng thứ hai cũng phải in ra được 10 cột *, ta dùng biến chạy j, duyệt cột từ 0 đến 9.

…

Với i = 4, hàng thứ 5 cũng phải in ra được 10 cột *, ta dùng biến chạy j, duyệt cột từ 0 đến 9.

Ta thấy i duyệt từ 0 đến 4, lặp đi lặp lại công việc biến j duyệt từ 0 đến 9 và in ra dấu *.

Dùng một vòng for cho biến chạy i, ứng với mỗi i dùng một vòng for cho biến chạy j.

✍Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
3   int main()
4   {
5           int i,j;
6           for(i = 0; i < 5; i++)
7           {
8                   for(j = 0; j < 10; j++)
9                   printf("* ");
10                  printf("\n");
11          }
12          return 0;
13  }
```

✍Kết quả chương trình

```
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
* * * * * * * * * *
```

🕮 Ví dụ 3.20

Viết chương trình in ra trên màn hình một ma trận có n dòng m cột như sau:

```
1 2 3 4 5 6 7
2 3 4 5 6 7 8
3 4 5 6 7 8 9
…
```

✍Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
3   int main ()
4   {   unsigned int dong, cot, n, m;
5   printf("\n Nhap vao so dong va so cot :");
6   scanf("%d%d", &n, &m);
7   for (dong = 0; dong < n; dong++)
8   {
9   printf("\n");
10  for (cot = 1; cot <= m; cot++)
11  printf("%d\t", dong + cot);
12  }  getch();
13  return 0;
14  }
```

✍Kết quả chương trình

```
 Nhap vao so dong va so cot :7 8
1   2   3   4   5   6   7   8
2   3   4   5   6   7   8   9
3   4   5   6   7   8   9   10
4   5   6   7   8   9   10   11
5   6   7   8   9   10   11   12
6   7   8   9   10   11   12   13
7   8   9   10   11   12   13   14
```

② Trong câu lệnh for, có thể sẽ không có phần Exp1 (phần khởi tạo)

```
int i;
i = 0;
for (; i < 10; i++)
   printf("%d\n", i);
```

③ Trong câu lệnh for, có thể sẽ không có phần Exp2 (điều kiện lặp)

```
for (i = 0; ; i++)
{
```

```
    if (i <= 10)
    printf(“%d\n”, i);
}
```

④ Trong câu lệnh for, có thể sẽ không có phần Exp3 (bước nhảy)

```
int i;
for (i = 0; i < 10; )
    {
    printf(“%d\n”, i);
    i = i + 1;
    }
```

⑤ Trong câu lệnh for, có thể khuyết cả 3

```
int i;
i = 0;
for (; ; )
{
    if(i < 10)
    printf("%d\n", i);
    i++;
    }
}
```

⑥ Không được thêm; ngay sau lệnh for

```
for (i = 0; i < 10; i++);
{
    printf(“%d”, i);
    printf(“\n”);
}
for (i = 0; i < 10; i++)
{
};
```

⇨ Máy sẽ không báo lỗi nhưng thực hiện câu lệnh không đúng

⑦ Các thành phần Exp1, Exp2, Exp3 cách nhau bằng dấu “;”. Nếu có nhiều thành phần trong mỗi phần thì được cách nhau bằng dấu phẩy “,”

```
for (i = 1, j = 2; i + j < 10; i++, j += 2)
```

printf(“%d\n”, i + j);

🕮 Ví dụ 3.21

Viết chương trình in ra tam giác vuông cân đặc n (n > 0 nhập từ bàn phím) như sau:

n = 4

*

**

***

****

✍Phác họa lời giải:

- Kiểm tra điều kiện n được nhập vào đảm bảo lớn hơn 0.
- Với mỗi dòng in ra số dấu * bằng đúng thứ tự dòng.

✍Chương trình

```
1   #include<conio.h>
2   #include<stdio.h>
3   int main()
4   {
5      int n, i, j;
6      printf("Nhap n: ");
7      scanf("%d", &n);
8      // Kiem tra dieu kien cua n
9      for(; n <= 0;)
10     {
11        printf("Phai nhap n > 0 \nNhap lai n: ");
12        scanf("%d",&n);
13     }
14     // In ra tam giac vuong co chieu cao n
15     for(i = 1; i <= n; i++)
16     {
17        for(j = 1; j <= i;j++)
18        {
```

```
19              printf("*");
20          }
21          printf("\n");
22      }
23  }
```

✍Kết quả chương trình:

```
Nhap n: -2
Phai nhap n > 0
Nhap lai n: 4
*
**
***
****
```

🕮 Ví dụ 3.22

Nhập vào một số n (n > 0), kiểm tra xem số đó có phải là số hoàn hảo không?

In ra các số hoàn hảo nhỏ hơn n (số hoàn hảo là số có tổng các ước = chính nó, chẳng hạn 6 có các ước 1, 2, 3 và 1 + 2 + 3 = 6).

✍Phác họa lời giải:

- Kiểm tra số nhập vào n có là số hoàn hảo. Muốn kiểm tra một số có là số hoản hảo hơn không, chúng ta xác định các ước của n (trừ ước là n) và tính tổng của các ước này. Nếu tổng các ước bằng chính số vừa nhập, thì kết luận nó là số hoàn hảo, ngược lại kết luận n không phải số hoàn hảo.
- Để liệt kê tất cả các số hoàn hảo nhỏ hơn n, chúng ta kiểm tra từng số trong phạm vi từ 0 đến n, theo bước 1. Nếu là số hoàn hảo thì in số đó ra, ngược lại bỏ qua, kiểm tra đến số tiếp theo.

✍Chương trình

```
1  #include <stdio.h>
2  int main() {
3  unsigned n, i, j, sum;
4  sum=0;
```

```
5   printf( "Nhap n: " );
6   scanf( "%u", &n );
7   for(i = 1; i < n;i++)
8   if (n % i == 0) sum+ = i;
9   if (sum == n) printf("%u la so hoan hao",n);
10  else printf("%u khong la so hoan hao",n);
11  printf( "\nCac so hoan hao nho hon %u: ", n );
12  for ( i = 1; i < n; ++i )
13  {
14   sum=0;
15   for ( j = 1; j < i; ++j )
16      if ( i % j == 0 ) sum += j;
17   if ( sum == i ) printf( "%u ", i );
18  }
19  return 0;
20  }
```

✍Kết quả chương trình

```
Nhap n: 50
50 khong la so hoan hao
Cac so hoan hao nho hon 50: 6 28
```

🕮 Ví dụ 3.23

Nhập một số nguyên dương n. Xuất ra số ngược lại.

Dữ liệu test: Nhập 1706.

Kết quả mong đợi: 6071.

✍Phác họa lời giải

Liệp tiếp thực hiện phép chia của số vừa nhập cho 10, kết quả thu được tiếp tục chia cho 10, với lấy phần dư. In các số dư thu được của các phép chia ra màn hình.

✍Chương trình

```
1   #include <stdio.h>
```

```
2  int main()
3  {
4  int n, donvi;
5  printf("Nhap n: ");
6  scanf("%d", &n);
7  printf("So dao cua %d la ", n);
8  for(;n>0;)
9          {
10                 donvi = n % 10;
11                 n = n / 10;
12                 printf("%d", donvi);
13         }
14 return 0;
15 }
```

✍Kết quả chương trình

Nhap n: 1706

So dao cua 1706 la 6071

🕮 Ví dụ 3.24

Viết chương trình tính căn số liên tục sau:

$$s = \sqrt[n+1]{n + \sqrt[n]{n-1 + \sqrt[n-1]{n-2 + \cdots + \sqrt[3]{2+\sqrt{1}}}}}$$

✍ Phác họa lời giải

Dễ dàng nhận thấy công thức truy hồi của căn liên tục trên:

Điều kiện đầu: $s_1 = \sqrt{1} = 1$

$$s_2 = \sqrt[2+1]{2 + s_1}$$

$$s_3 = \sqrt[3+1]{3 + s_2}$$

….

$$s_k = \sqrt[k+1]{k + s_{k-1}}, k = 1, \dots, n$$

Như vậy, ta cần thực hiện một vòng lặp với biến đếm k chạy từ 2 (vì s_1 ứng với k = 1 đã được tính) đến n. Biểu thức trong thân vòng lặp:

$$s_k = \sqrt[k+1]{k + s_{k-1}} = (k + s_{k-1})^{1/k+1} = pow(k + s_{k-1}, \frac{1.0}{k+1})$$

Chương trình

```
1  #include <stdio.h>
2  #include <math.h>
3  int main() {
4    unsigned n, k;
5    double S = 1.0;
6    printf( "Nhap n: " );
7    scanf( "%u", &n );
8    for ( k = 2; k <= n; ++k )
9      S = pow( k + S, 1.0 / ( k + 1 ) );
10   printf( "Ket qua: %lf\n", S );
11   return 0;
12 }
```

Kết quả chương trình

```
Nhap n: 2
Ket qua: 1.44225
```

3.2.2. Cấu trúc vòng lặp while

3.2.2.1. Cú pháp

while (expression)

statement;

Trong đó:

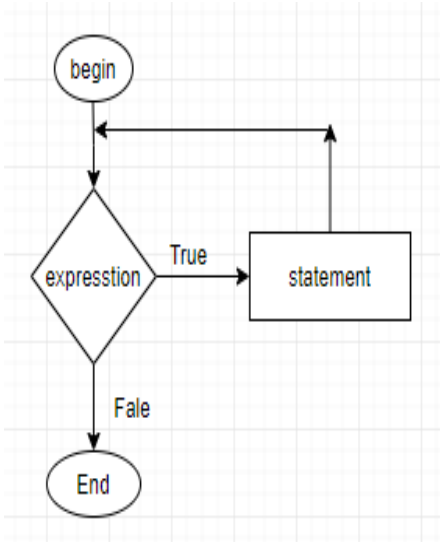
- expression: Biểu thức điều kiện

- statement: khối lệnh (là câu lệnh đơn hoặc câu lệnh ghép), gọi là thân vòng lặp.

3.2.2.2. Hoạt động

Chừng nào biểu thức điều kiện còn đúng thì thực hiện lệnh, biểu thức điều kiện sai thì thoát khỏi vòng lặp.

3.2.2.3. Sơ đồ khối



Hình 3.8. Sơ đồ khối biểu diễn vòng lặp while

Lưu ý:

Câu lệnh while có thể lồng vào nhau:

```
if (n < 10 && m < 20)
{
   while (n >= 1)
   {
            while (m >= 1)
            {
                     printf(“%d”, m);
                     m--;
            }
            n--;
   }
}
```

🕮 Ví dụ 3.25

Xem xét ví dụ 3.23.

Nhập một số nguyên dương n. Xuất ra số ngược lại.

Dữ liệu test: 1706.

Kết quả mong đợi: 6071.

✍Phác họa lời giải

Ở ví dụ 3.23, ta giải bài này bằng cách dùng vòng lặp for, nhưng trong vòng lặp for khuyết đi 2 biểu thức Exp1 và Exp3 vì không cần biến chạy và bước nhảy. Chỉ cần xét điều kiện dừng. Bài toán này không biết trước số lần lặp trước.

Và chừng nào điều kiện còn đúng n khác 0 thì thực hiện phép lấy số đơn vị (n%10) và bỏ số đơn vị (n/10).

Dùng while xử lý như sau:

✍Chương trình

```
1   #include <stdio.h>
2   int main()
3   {
4   int n, donvi;
5   printf("Nhap n: ");
6   scanf("%d", &n);
7   printf("So dao cua %d la ", n);
8   while (n > 0)
9           {
10                  donvi = n % 10;
11                  n = n / 10;
12                  printf("%d", donvi);
13          }
14  return 0;
15  }
```

✍Kết quả chương trình

```
Nhap n: 3265
So dao cua 3265 la 5623
```

✍Nhận xét

Qua ví dụ 3.23 và 3.25, ta thấy bài toán này dùng vòng lặp while gọn và rõ ràng hơn.

⇨ Vậy những bài toán biết trước số lần lặp ta dùng for;
⇨ Bài toán không biết trước số lần lặp ta dùng while.

🕮 Ví dụ 3.26

Tính $S = 1^3 + 2^3 + \ldots + n^3$

✍Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
3   int main()
4   {
5           int n, s,i;
6           s = 0;
7           i = 1;
8           printf("\nNhap n= "); scanf("%d",&n);
9           while (i <= n)
10          {
11                  s = s + i * i * i;
12                  i++;
13          }
14          printf("\nTong s= %d",s);
15          return 0;
16  }
```

✍Kết quả chương trình

```
Nhap n = 3
Tong s = 36
```

🕮 Ví dụ 3.27

Viết chương trình chuyển số nhị phân sang hệ thập phân

✍Phác họa lời giải

Đổi hệ nhị phân sang hệ thập phân:

$10111_2 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$

Ta dùng một biến donvi để lần lượt lấy chữ số hàng đơn vị của số nhị phân.

Biến decimal để tính số thập phân.

Biến num để lấy phần nguyên sau khi bỏ hàng đơn vị.

Biến base là cơ số 2, ban đầu base = 1 = 2^0.

Chừng nào biến num còn lớn hơn 0 thì thực hiện các thao tác trên.

✍Chương trình

```
1  #include "stdio.h"
2  #include "conio.h"
3  main()
4  {
5     int num, binary, decimal = 0, base = 1, donvi;
6     printf("\nNhap so nhi phan(1 & 0): ");
7     scanf("%d", &num);
8     binary = num;
9     while (num > 0)
10    {
11       donvi = num % 10;
12       decimal = decimal + donvi * base;
13       num = num / 10 ;
14       base = base * 2;
15    }
16    printf("So nhi phan = %d \n", binary);
17    printf("Gia tri he thap phan = %d \n", decimal);
18    getch();
19 }
```

✍Kết quả chương trình

```
Nhap so nhi phan(1 & 0): 1011
So nhi phan = 1011
Gia tri he thap phan = 11
```

🕮 Ví dụ 3.28

Viết chương trình đổi hệ thập phân sang hệ nhị phân.

✍Phác họa lời giải

Đổi hệ thập phân sang hệ nhị phân:

Ta lấy số thập phân chia dần cho 2 chừng nào thương = 0 thì phép chia dừng, cách chuyển đổi được lấy từ dưới lên.

$15_{10} = 1111_2 = 1*1000 + 1*100 + 1*10 + 1*1$

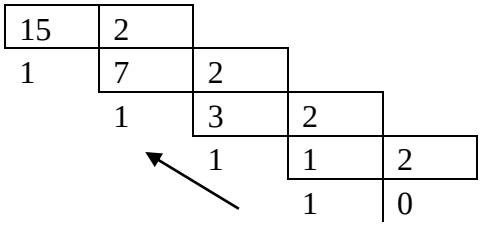
Biến ng để lưu phần dư của phép chia decimal cho 2.

Biến base =1 mỗi lần lặp tăng dần lên 10 lần.

Biến num để lấy phần nguyên sau khi decimal chia cho 2.

Biến binary để lưu các số dư của phép chia decimal cho 2.

Chừng nào biến num còn lớn hơn 0 thì thực hiện các thao tác trên.



Hình 3.9. Chuyển số thập phân sang số nhị phân

✍Chương trình

```
1   #include "stdio.h"
2   #include "conio.h"
3   main()
4   {
5      unsigned long num, binary = 0, decimal, base = 1, ng;
6      printf("\nNhap so nguyen he thap phan: ");
7      scanf("%ld", &num);
8      decimal = num;
9      while (num > 0)
10     {
11        ng = num % 2;
12        binary = binary + ng * base;
13        num = num / 2;
14        base = base * 10;
15     }
16     printf("So thap phan = %d \n", decimal);
17     printf("So nhi phan = %d \n", binary);
```

```
18        getch();
19    }
```

✍Kết quả chương trình

```
Nhap so nguyen he thap phan: 125
So thap phan = 125
So nhi phan = 1111101
```

3.2.3. Cấu trúc vòng lặp do … while

3.2.3.1. Cú pháp

```
do
{ statement;
} while (expression);
```

Trong đó:

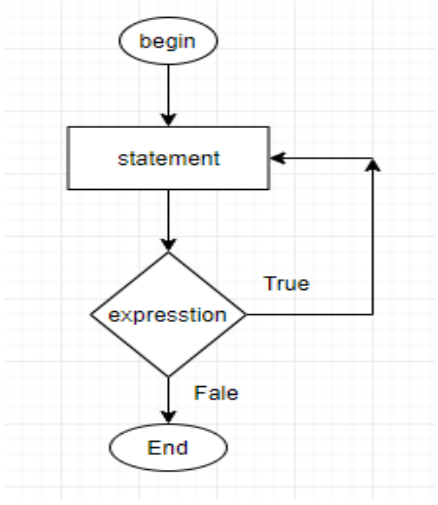
- **expression:** Biểu thức điều kiện, là một biểu thức logic.
- **statement:** Câu lệnh trong vòng lặp.

3.2.3.2. Hoạt động

Bước 1: Thực hiện các lệnh bên trong thân vòng lặp do…while.

Bước 2: Kiểm tra điều kiện, chừng nào điều kiện còn đúng thì thực hiện lệnh, điều kiện sai thì thoát khỏi vòng lặp.

3.2.3.3. Sơ đồ khối



Hình 3.10. Sơ đồ khối biểu diễn vòng lặp do… while

✍Lưu ý:

① Câu lệnh do… while có thể lồng nhau.

```
do
{
   b = 1;
   do
   {
            printf("%d\n", a + b);
            b = b + 2;
   }
   while (b < 20);
   a++;
}
while (a < 20);
```

② Câu lệnh do… while sẽ được thực hiện ít nhất 1 lần.

```
do
   {
            printf("Nhap n: ");
            scanf("%d", &n);
   }
   while (n < 1 || n > 100);
```

🕮 Ví dụ 3.29

Viết chương trình thực hiện các yêu cầu sau:

Nhập vào 3 số thực a, b, c từ bàn phím.

a. Kiểm tra a, b, c có lập thành ba cạnh của tam giác không?

b. Nếu có hãy kiểm tra đây là tam giác cân, đều, vuông hay tam giác thường. Tính chu vi và diện tích của tam giác. In kết quả ra màn hình.

Dữ liệu test: 1 4 7.

Kết quả mong đợi: Không là ba cạnh tam giác.

Dữ liệu test: 2 4 5.

Kết quả mong đợi: Tam giác thường.

✍Phác họa lời giải

Đây là ví dụ 3.8, nhưng có một hạn chế là không cho bắt lỗi khi nhập a, b, c nhỏ hơn 0. Ở đây, ta sử dụng cấu trúc *do…while* lệnh được

thực hiện trước rồi kiểm tra điều kiện, ta viết ràng buộc khi nhập a, b, c như sau:

Chương trình

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
float a, b,c;
do
{
        printf("\nNhap canh a= "); scanf("%f",&a);
}while (a <= 0);
do
{
        printf("\nNhap canh b= "); scanf("%f",&b);
}while (b <= 0);
do
{
        printf("\nNhap canh c= "); scanf("%f",&c);
}while (c <= 0);
if (a + b > c && b + c >a && c + a > b)
{
        if (a == b && b == c) printf("\nTam giac deu");
        else if (a == b ||b == c|| c == a) printf("\nTam giac can");
        else if (a * a + b * b == c * c ||a * a + c * c == b * b||c * c + b * b == a * a) printf("\nTam giac vuong");
        else printf("\nTam giac thuong");
        float p = (a + b + c)/2;
        float s = sqrt(p*(p-a)*(p-b)*(p-c));
        printf("\nChu vi tam giac: %.2f",2*p);
        printf("\nDien tich tam giac %.2f",s);
}
```

```
30  else printf("\n khong la tam giac");
31  }
```

✍Kết quả chương trình

```
Nhap canh a= 0
Nhap canh a= 2
Nhap canh b= 0
Nhap canh b= 2
Nhap canh c= 2
Tam giac deu
Chu vi tam giac: 6.00
Dien tich tam giac 1.73
```

✍Nhận xét

So sánh 2 chương trình ở ví dụ 3.8 và 3.29, ở ví dụ 3.29 từ dòng 7 đến dòng 18 ràng buộc điều kiện chỉ cho nhập a, b, c > 0, khi xét điều kiện a, b, c là 3 cạnh tam giác không phải xét điều kiện a, b, c lớn hơn 0 nữa. Còn ở ví dụ 3.8 không đưa ràng buộc khi nhập vào giá trị a, b, c, do đó người nhập vẫn có thể nhập giá trị âm, sau đó mới xét điều kiện để ba số là 3 cạnh của một tam giác.

🕮 Ví dụ 3.30

Viết chương trình nhập vào một ngày, tháng, năm (ràng buộc cho ngày chỉ nhập từ một đến ngày theo tháng, tháng nhập từ tháng một đến tháng mười hai, năm chỉ nhập năm có bốn chữ số). Tìm ngày kế tiếp (ngày, tháng, năm) của ngày vừa nhập.

Dữ liệu test: 22/12/2018.

Kết quả mong đợi: 23/12/2018.

✍Phác họa lời giải

Ví dụ này đã được giải quyết ở ví dụ 3.16, nhưng chúng ta kiểm tra ràng buộc điều kiện nhập cho ngày, tháng, năm bằng lệnh goto phức tạp hơn nhiều nếu ta dùng do…while:

- Nhập vào tháng, $1 \leq$ tháng ≤ 12.
- Nhập vào năm ($1000 \leq$ năm ≤ 9999).
- Kiểm tra tháng (d) để trả ngày theo tháng:
  + Tháng 1, 3, 5, 7, 8, 10, 12 có số ngày d = 31 ngày.

+ Tháng 4, 6, 9, 11 có số ngày d = 30 ngày.
+ Tháng 2 năm nhuận có số ngày d = 29 ngày.
+ Tháng 2, không là năm nhuận có số ngày d = 28 ngày.

(Năm nhuận là năm chia hết cho 4 và không chia hết cho 100 hoặc chia hết cho 400).

- Nhập vào ngày (1 ≤ ngày ≤ d).
  + Nếu ngày nhập vào < ngày của tháng (d) thì tăng ngày lên.
  + Nếu ngày nhập vào ≥ ngày của tháng (d), thực hiện kiểm tháng nhập vào:
  + Nếu tháng < 12, thiết lập ngày = 1 và tăng tháng lên 1.
  + Nếu tháng ≥ 12, thiết lập ngày = tháng = 1, tăng năm lên 1.

Chương trình

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int ng,th,n,d;
        do
        {
        printf("\n Nhap nam: "); scanf("%d",&n);
        }while(n < 1000 ||n > 9999);
        Do
        {
        printf("\n Nhap thang: "); scanf("%d",&th);
        }while(th < 1||th > 12);
        switch(th)
        {
        case 1: case 3: case 5: case 7: case 8: case 10: case 12: d = 31;
                break;
                case 9: case 4: case 6: d = 30; break;
                case 2:
                 {
                 if(n % 4 == 0 && n %100!= 0||n % 400 == 0)
                                d = 29;
```

```
23                          else d = 28;
24                  };break;
25          }
26          do
27          {
28          printf("\n Nhap ngay: "); scanf("%d",&ng);
29          }while(ng < 1||ng > d);
30          printf("\n Ngay vua nhap %d-%d-%d\n", ng, th, n);
31          if (ng < d)
32          {
33                  ng++;
34          }
35          else if (th < 12)
36          {
37                  ng = 1; th++;
38          }
39          else
40          {
41                  ng = th = 1; n++;
42          }
43          printf("\nNgay ke tiep: %d- %d - %d", ng,th,n);
44  getch();
45  return 0;
46  }
```

✍Kết quả chương trình

Nhap nam: 215

Nhap nam: 2015

Nhap thang: 15

Nhap thang: 12

Nhap ngay: 23

Ngay vua nhap 23-12-2015

Ngay ke tiep: 24- 12 - 2015

✍Nhận xét

Ở ví dụ 3.16, chưa học cấu trúc *do…while* nên ta dùng lệnh goto để ràng buộc điều kiện cho nhập ngày, tháng, năm.

Ở ví dụ 3.30 này chúng ta dùng cấu trúc *do…while* để ràng buộc điều kiện cho nhập ngày, tháng, năm, chương trình được viết gọn hơn, rõ ràng, dễ hiểu hơn.

✍So sánh sự khác nhau của các vòng lặp: For, while, do…while

① Đều có khả năng lặp lại nhiều hành động

```
//Vòng lặp for
int n = 10, i;
   for (i = 1; i <= n; i++)
            printf("%d\n", i);
   //Vòng lặp while
   int i = 1;
   while (i <= n)
   {
            printf("%d\n", i); i++;
   }
//Vòng lặp do…while
   int i = 1;
   do {
            printf("%d\n", i); i++;
   } while (i < n);
```

② Vòng lặp for thường sử dụng khi biết được số lần lặp xác định. Vòng lặp while, do…while sử dụng khi không biết rõ số lần lặp.

```
   //Vòng lặp for biết trước số lần lặp là n = 10.
      int n = 10,i;
      for (i = 1; i <= n; i++)
        …;
      //Vòng lặp while, do…while phụ thuộc thân vòng lặp mới biết được số lần lặp
      int i = 1;
      while (i <= n)
      {
        …;
```

```
}
int i = 1;
do {
  …;
} while (i > n);
```

③ while có thể không thực hiện lần nào, do…while sẽ được thực hiện ít nhất 1 lần.

```
int n = 100;
while (n < 10)
{
  …;
}
…
do
{
  printf(“Nhap n: ”);
  scanf(“%d”, &n);
}
while (n > 10);
```

🕮 Ví dụ 3.31

Viết chương trình in ra tam giác cân đặc và rỗng, tạo từ các dấu sao (*), có độ cao là n nhập từ bàn phím.

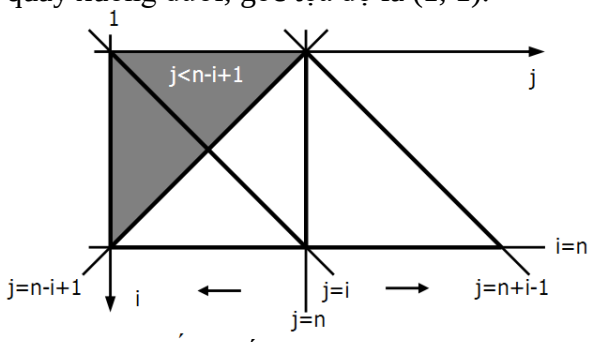
Chẳng hạn, nhập n: 4

```
            *
         *  *  *
      *  *  *  *  *
   *  *  *  *  *  *  *

            *
         *     *
      *           *
   *  *  *  *  *  *  *
```

✍Phác họa lời giải:

Bản chất của yêu cầu trên là vẽ các điểm trên màn hình theo tọa độ. Bạn hãy xem mô tả đồ thị của bài trên trong hình dưới, chú ý là trục tung i trên màn hình quay xuống dưới, gốc tọa độ là (1, 1).



Đường j = n + i - 1 là tịnh tiến của đường j = n và đường j = n - i + 1 là đối xứng qua trục j = n của đường j = n + i - 1.

Để vẽ cả dòng và cột bạn dùng 2 vòng lặp lồng vào nhau: vòng lặp ngoài tính dòng (biến đếm i), với mỗi dòng, vòng lặp trong vẽ các cột (biến đếm j). Hai vòng lặp này quét tất cả các vị trí phải vẽ (ký tự * hoặc ký tự space).

Khi vẽ tam giác đặc, bạn quét toàn bộ miền:

$$\begin{cases} \mathbf{1} \le \boldsymbol{i} \le \boldsymbol{n} \\ \mathbf{1} \le \boldsymbol{j} \le \boldsymbol{n} + \boldsymbol{i} - \mathbf{1} \end{cases}$$

trong đó: $1 \le \boldsymbol{j} \le \boldsymbol{n} - \boldsymbol{i} + \mathbf{1}$ ta vẽ dấu space

$\boldsymbol{n} - \boldsymbol{i} + \mathbf{1} \le \boldsymbol{j} \le \boldsymbol{n} + \boldsymbol{i} - \mathbf{1}$, phần còn lại ta vẽ dấu *

✍Chương trình:

```
1  #include <stdio.h>
2  int main() {
3    unsigned n, i, j;
4    printf( "Nhap n: " );
5    scanf( "%u", &n );
6   //tam giac can dac
7    for ( i = 1; i <= n; ++i)
8    {
9      for ( j = 1; j <= n + i - 1; ++j )
10       printf( ( j < n - i + 1 )? "  " : "* " );
```

```
11      printf("\n");
12    }
13   //Tam giac can rong
14    for (i = 1; i <= n; ++i)
15    {
16      for (j = 1; j <= n + i - 1; ++j)
17    if (j == n - i + 1 || j == n + i - 1 || i == n)
printf("*");
18    else printf("  ");
19      printf("\n");
20    }
21    return 0;
22  }
```

✍Kết quả chương trình:

Nhap n: 4

```
                    *
                *   *   *
            *   *   *   *   *
        *   *   *   *   *   *   *

                    *
                *       *
            *               *
        *   *   *   *   *   *   *
```

🕮 Ví dụ 3.32

Tìm ước số lẻ lớn nhất của số nguyên dương n.

Dữ liệu test: n = 100.

Kết quả mong đợi: ước lẻ lớn nhất là 25.

✍Chương trình

```
1     #include<stdio.h>
```

```
2   #include<conio.h>
3   int main()
4   {
5           int i, n, max;
6           do
7           {
8           printf("\nNhap n (n > 0): ");
9           scanf("%d", &n);
10            if(n <= 0)
11            {
12            printf("\n N phai > 0. Xin nhap lai !");
13            }
14          }while(n <= 0);
15      i = 1;
16          max = 1;
17          printf("\nCac uoc so le cua so %d la: ",n);
18          while(i <= n)
19          {
20                  if((n % i == 0) && (i % 2 == 1))
21                  {
22                          if(i > max)
23                          {
24                          max = i;
25                          }
26                          printf("%4d", i);
27                  }
28                          i++;
29          }
30          printf("\nUoc so le lon nhat la %d", max);
31          getch();
32          return 0;
33  }
```

✍Kết quả chương trình

```
Nhap n(n > 0): 0
n phai > 0. Xin nhap lai !
Nhap n(n > 0): 25
Cac uoc so le cua so 25 la:    1   5  25
Uoc so le lon nhat la 25
```

3.3. CÂU LỆNH BREAK, CONTINUE

3.3.1. Câu lệnh break

Cú pháp: **break;**

Chức năng: Lệnh break cho phép thoát ra khỏi vòng lặp chứa nó ngay lập tức.

Lưu ý:

- Câu lệnh break dùng trong vòng lặp và câu lệnh switch.

🕮 Ví dụ 3.33

Giả sử mật khẩu là 111. Đọc vào một mật khẩu người dùng tối đa số lần nhập là 4. Nếu nhập đúng số 111 thì thoát không phải kiểm tra hết n lần. Nếu nhập không đúng số 111 báo nhập sai. Nếu nhập quá số lần quy định thì in ra dòng thông báo *"So lan nhap da het, may se khoa trong 5 phut"*.

Dữ liệu test: 111

Kết quả mong đợi: thoát.

Dữ liệu test: 222

Kết quả mong đợi: Ban da nhap sai.

✍Chương trình

```
1   #include<stdio.h>
2   #include<conio.h>
3   main()
4   {
5   int pass, i;
6   for (i=0; i<4; i++)
7   {
```

```
8       printf(“Nhap pass= ”); scanf(“%d”, &pass);
9       if (pass == 111)        break;
10      printf(“Ban nhap sai\n”);
11  }
12  printf(“\nSo lan nhap da het, may se khoa trong 5 phut”);
13  getch();
14  }
```

✍Kết quả chương trình

```
Nhap pass= 23
Ban nhap sai
Nhap pass= 162
Ban nhap sai
Nhap pass= 25
Ban nhap sai
Nhap pass= 452
Ban nhap sai
So lan nhap da het, may se khoa trong 5 phut
```

🕮 Ví dụ 3.34

Viết chương trình nhập vào số n, sau đó in ra màn hình n là số nguyên tố hay không?

Dữ liệu test1: 50.

Kết quả mong đợi: 50 không phải là số nguyên tố.

Dữ liệu test2: 29.

Kết quả mong đợi: 29 là số nguyên tố.

✍Chương trình

```
1   #include “stdio.h”
2   #include “conio.h”
3   main()
4   {
```

```
5   int  n, i, kt;
6   printf("Nhap n:");
7   scanf("%d",&n);
8   kt = 1;
9   for (i = 2; i < n; i++)
10  {
11      if  (n % i == 0)
12        {
13          kt = 0;
14                      break;
15        }
16  }
17  if (kt == 1)   printf("%d la so nguyen to", n);
18  else   printf("%d khong la so nguyen to", n);
19  getch();
20  }
```

✍Kết quả chương trình

```
Nhap n:23
23 la so nguyen to
```

3.3.2. Câu lệnh continue

Cú pháp: continue;

Chức năng: Khi gặp lệnh continue, chương trình sẽ bỏ qua các lệnh còn lại trong thân vòng lặp của lần lặp hiện tại và quay lại thực hiện lần lặp tiếp theo của vòng lặp.

✍Lưu ý:

Lệnh continue chỉ áp dụng cho cấu trúc lặp, không áp dụng cho cấu trúc switch, if.

🕮 Ví dụ 3.35

In ra màn hình các tháng trong năm trừ tháng 5.

Chương trình

```
1       #include<stdio.h>
2       #include<conio.h>
3       int main()
4       {
        int i;
5       for (i = 1; i <= 12; i++ )
6       {       if (i == 5) continue;
7               printf("Day la thang %d\n", i);
8       }
9       return 0;
10      }
```

Kết quả chương trình

```
Day la thang 1
Day la thang 2
Day la thang 3
Day la thang 4
Day la thang 6
Day la thang 7
Day la thang 8
Day la thang 9
Day la thang 10
Day la thang 11
Day la thang 12
```

BÀI TẬP CHƯƠNG 3

A. BÀI TẬP CÓ LỜI GIẢI

🕮 Bài tập 3.1

Viết chương trình, nhập vào một số, in ra màn hình số vừa nhập là số chẵn hay số lẻ?

✍Phác thảo lời giải

Số chẵn là số chia hết cho hai (dư của phép chia = 0), số lẻ là số không chia hết cho 2 (dư của phép chia = 1).

✍Chương trình

```
1   #include <stdio.h>
2   int main()
3   {
4       int so;
5       printf("Nhap vao mot so: ");
6       scanf("%d", &so;
7       // Kiem tra so vua nhap la so chan hay so le
8       if(so % 2 == 0)
9           printf("%d la so chan.", so);
10      else
11          printf("%d la so le.", so);
12      return 0;
13  }
```

✍Kết quả chương trình:

```
Nhap vao mot so: 12
12 la so chan.
```

🕮 Bài tập 3.2

Viết chương trình nhập vào 3 số nguyên a, b, c. In ra màn hình số lớn nhất.

✍Chương trình

```
1  #include <stdio.h>
2  int main()
3  {
```

```
4     int s1, s2, s3;
5     printf("Nhap vao 3 so nguyen: ");
6     scanf("%d %d %d", &s1, &s2, &s3);
7     printf("So thu nhat = %d,\tSo thu hai = %d,\tSo thu ba = %d\n",
    s1, s2, s3);
8     if (s1 > s2)
9     {
10       if (s1 > s3)
11       {
12          printf("So thu nhat la so lon nhat, %d. \n",s1);
13       }
14       else
15       {
16          printf("So thu ba la so lon nhat, %d. \n", s3);
17       }
18    }
19    else if (s2 > s3)
20       printf("So thu hai la so lon nhat, %d \n", s2);
21    else
22       printf("So thu ba la so lon nhat, %d", s3);
23    return 0;
24 }
```

✍Kết quả chương trình

```
Nhap vao 3 so nguyen: 12 45 33
So thu nhat = 12,  So thu hai = 45,   So thu ba = 33
So thu hai la so lon nhat, 45
```

🕮 Bài tập 3.3

Viết chương trình đọc vào nhiệt độ theo độ C và hiển thị thông báo phù hợp theo trạng thái nhiệt độ sau đây:

Nhiệt độ < 0^oC: Đóng băng.

Nhiệt độ 0^oC - 10^oC: Rất lạnh.

Nhiệt độ 10^oC - 20^oC: Lạnh.

Nhiệt độ 20^oC - 30^oC: Bình thường

Nhiệt độ 30^{o}C - 40^{o}C: Nóng

Nhiệt độ > 40^{o}C: Rất nóng.

Dữ liệu Test: 35.

Kết quả mong đợi: Nóng.

✍Chương trình

```
#include <stdio.h>
int main()
{
 int t;
   printf("Nhap vao nhiet do trong ngay: ");
   scanf("%d",&t);
  if(t < 0)
        printf("Dong bang.\n");
  else if( t < 10)
      printf("Rat lanh.\n");
      else if( t < 20)
           printf("Lanh.\n");
          else if( t < 30)
                printf("Binh thuong.\n");
              else if(t < 40)
                printf("Nong.\n");
                Else
                    printf("Rat nong.\n");
}
```

✍Kết quả chương trình:

Nhap vao nhiet do trong ngay: 22

Binh thuong.

🕮 Bài tập 3.4

Chỉ số khối cơ thể (viết tắt là BMI - Body Mass Index) được dùng để đánh giá mức độ gầy hay béo của một người. Chỉ số này có thể giúp xác định một người bị bệnh béo phì hay bị bệnh suy dinh dưỡng.

Công thức tính chỉ số: BMI = $\frac{W}{H^2}$ được làm tròn kết quả đến 2 chữ số thập phân.

trong đó W là khối lượng của một người (tính bằng kg) và H là chiều cao của người đó (tính bằng m).

Căn cứ vào chỉ số BMI, đưa ra các nhận xét và yêu cầu sau đây:

- BMI < 18: Người gầy, cần bồi dưỡng thêm.
- BMI = 18 - 24.9: Thân hình hoàn toàn bình thường.
- BMI = 25 - 29.9: Người béo phì độ I, cần có chế độ ăn uống hợp lý.
- BMI = 30 - 34.9: Người béo phì độ II, hãy thận trọng.
- BMI > 35: Người béo phì độ III, cảnh báo nguy cơ béo phì.

Viết chương trình tính chỉ số BMI và đưa ra nhận xét, yêu cầu.

✍Phác họa lời giải

- Nhập vào chiều cao và cân nặng.
- Tính chỉ số BMI dựa vào chiều cao và cân nặng.
- Dựa vào chỉ số BMI, sử dụng câu lệnh if else đưa ra các nhận xét và lời khuyên.
- Khai báo biến ketQua lưu lời khuyên, sử dụng hàm strcpy() để sao chép đoạn lời khuyên cho biến ketQua.
- Hiển thị ketQua ra màn hình.

✍Chương trình

```
1   #include <stdio.h>
2   #include <string.h>
3   int main()
4   {
5       float chieuCao, canNang, chiSoBMI;
6       char ketQua[200];
7           printf("Nhap vao can nang: ");
8       scanf("%f",&canNang);
9        printf("Nhap vao chieu cao: ");
10      scanf("%f",&chieuCao);
11      // tinh chi so BMI
12      chiSoBMI = canNang / (chieuCao * chieuCao);
13      printf("Chi so BMI = %2.2f\n", chiSoBMI);
14      // phân loai dua vao chi so BMI
15      if (chiSoBMI < 18)
16      {
17              strcpy(ketQua, "Ban hoi gay, can boi duong them!!");
```

```
18      }
19              else if (chiSoBMI <= 24.9)
20             {
21          strcpy(ketQua, "Than hinh hoan toan binh thuong");
22      }
23                  else if (chiSoBMI <= 29.9)
24                                  {
25                          strcpy(ketQua, "Ban bi beo phi cap do 1, can co che
    do an uong hop ly");
26                           }
27                                      else if (chiSoBMI <= 34.9)
28                                      {
29                      strcpy(ketQua, "Ban beo phi cap do 2, hay than trong");
30                      }
31                                              else
32                                                      {
33                          strcpy(ketQua, "Ban beo phi cap do 3, canh bao
    nguy co beo phi!");
34                      }
35      printf("%s",ketQua);
36  }
```

✍Kết quả chương trình:

Nhap vao can nang: 78

Nhap vao chieu cao: 1.74

Chi so BMI = 25.76

Ban bi beo phi cap do 1, can co che do an uong hop ly

🕮 Bài tập 3.5

Viết chương trình nhập vào nhập vào một số trong phạm vi 10, hiển thị từ tiếng Anh tương ứng.

Dữ liệu Test_1: 5.

Kết quả mong muốn: Five.

Dữ liệu Test_2: 9.

Kết quả mong muốn: Nine.

Chương trình:

```
1       #include <stdio.h>
2       int main()
3       {
4         int so;
5         printf("Nhap vao mot so(0-9) : ");
6         scanf("%d",&so);
7         switch(so)
8         {
9                 case 0:
10                    printf("Zero\n");
11                    break;
12                case 1:
13                    printf("one\n");
14                    break;
15              case 2:
16                    printf("Two\n");
17                    break;
18              case 3:
19                    printf("Three\n");
20                    break;
21              case 4:
22                    printf("Four\n");
23                    break;
24              case 5:
25                    printf("Five\n");
26                    break;
27              case 6:
28                    printf("Six\n");
29                    break;
30              case 7:
31                    printf("Seven\n");
32                    break;
33              case 8:
34                    printf("Eight\n");
```

```
35                  break;
36              case 9:
37                  printf("Nine\n");
38                  break;
39              default:
40                printf("So khong hop le. \nHay thu lai ....\n");
41                  break;
42          }
43      }
```

✍Kết quả chương trình:

```
Nhap vao mot so(0-9) : 2
Two
```

🕮 Bài tập 3.6

Viết chương trình nhập vào một năm dương lịch, cho biết năm âm lịch tương ứng.

✍Phác họa lời giải

Cách tính Lịch Âm Lịch = Can + Chi.

Bước 1: Xác định Can

Tính Can dựa vào số dư (Ca) của phép chia năm dương lịch cho 10 rồi tra theo bảng sau đây:

Ca = namDL % 10

Ca	0	1	2	3	4	5	6	7	8	9
Can	Canh	Tân	Nhâm	Quý	Giáp	Ất	Bính	Đinh	Mậu	Kỷ

Bước 2: Xác định Chi

Tính Chi dựa vào số dư (Ci) của phép chia năm dương lịch cho 12 rồi tra theo bảng sau đây:

Ci = namdL % 12

Ci	0	1	2	3	4	5	6	7	8	9	10	11
Chi	Thân	Dậu	Tuất	Hợi	Tý	Sửu	Dần	Mão	Thìn	Tỵ	Ngọ	Mùi

- Nhập vào năm dương lịch.
- Sử dụng các biến Ca (dư của phép chia năm dương lịch cho 10) và biến Ci (dư của phép chia năm dương lịch cho 12) để xác định Can và Chi tương ứng.

- Ghép Can + Chi = Năm âm lịch.

Dữ liệu Test: 1980.

Kết quả mong muốn: Canh Thân.

✍Chương trình:

```
#include <stdio.h>
#include <string.h>
int main()
{
 int year;
 char can[20] = " ";
 char chi[20] = " ";
 printf("Nhap vao nam duong lich: ");
      scanf("%d", &year);
// Xac dinh Can
      switch (year % 10) {
         case 0:
            strcpy(can,"Canh");
            break;
         case 1:
           strcpy(can,"Tan");
            break;
         case 2:
            strcpy(can, "Nham");
            break;
         case 3:
            strcpy(can, "Quy");
            break;
         case 4:
            strcpy(can,"Giap");
            break;
         case 5:
            strcpy(can,  "At");
            break;
```

```
30              case 6:
31                  strcpy(can, "Binh");
32                  break;
33              case 7:
34                  strcpy(can, "Dinh");
35                  break;
36              case 8:
37                  strcpy(can, "Mau");
38                  break;
39              case 9:
40                  strcpy(can, "Ky");
41                  break;
42          }
43          // Xac dinh Chi
44          switch (year % 12) {
45              case 0:
46                 strcpy(chi, "Than");
47                  break;
48              case 1:
49                  strcpy(chi, "Dau");
50                  break;
51              case 2:
52                  strcpy(chi , "Tuat");
53                  break;
54              case 3:
55                  strcpy(chi, "Hoi");
56                  break;
57              case 4:
58                  strcpy(chi, "Ty");
59                  break;
60              case 5:
61                  strcpy(chi, "Suu");
62                  break;
```

```
63              case 6:
64                  strcpy(chi ,"Dan");
65                  break;
66              case 7:
67                  strcpy(chi , "Mao");
68                  break;
69              case 8:
70                  strcpy(chi, "Thin");
71                  break;
72              case 9:
73                  strcpy(chi, "Ty");
74                  break;
75              case 10:
76                  strcpy(chi, "Ngo");
77                  break;
78              case 11:
79                  strcpy(chi, "Mui");
80                  break;
81          }
82          printf("Nam am lich tuong ung voi nam duong lich %d la
        %s %s ", year, can, chi);
83      }
```

✍Kết quả chương trình:

```
Nhap vao nam duong lich: 1980
Nam am lich tuong ung voi nam duong lich 1980 la Canh Than
```

🕮 Bài tập 3.7

Viết chương trình bằng C để hiển thị bảng đa biến theo chiều dọc từ 1 đến n?

```
1   #include <stdio.h>
2   int main()
3   {
4     int j, i, n;
```

```
5       printf("Nhap so toi da bang bat dau tu 1 : ");
6       scanf("%d",&n);
7       printf("Bang nhan tu 1 den %d \n",n);
8       for(i = 1;i <= 10;i++)
9       {
10        for(j = 1;j <= n;j++)
11        {
12          if (j <= n - 1)
13              printf("%d x %d = %d \t",j,i,i*j);
14            else
15                  printf("%d x %d = %d",j,i,i*j);
16         }
17        printf("\n");
18       }
19   }
```

✍Kết quả chương trình:

```
Nhap so toi da bang bat dau tu 1 : 5
Bang nhan tu 1 den 5
1 x 1 = 1    2 x 1 = 2    3 x 1 = 3    4 x 1 = 4    5 x 1 = 5
1 x 2 = 2    2 x 2 = 4    3 x 2 = 6    4 x 2 = 8    5 x 2 = 10
1 x 3 = 3    2 x 3 = 6    3 x 3 = 9    4 x 3 = 12    5 x 3 = 15
1 x 4 = 4    2 x 4 = 8    3 x 4 = 12    4 x 4 = 16    5 x 4 = 20
1 x 5 = 5    2 x 5 = 10    3 x 5 = 15    4 x 5 = 20    5 x 5 = 25
1 x 6 = 6    2 x 6 = 12    3 x 6 = 18    4 x 6 = 24    5 x 6 = 30
1 x 7 = 7    2 x 7 = 14    3 x 7 = 21    4 x 7 = 28    5 x 7 = 35
1 x 8 = 8    2 x 8 = 16    3 x 8 = 24    4 x 8 = 32    5 x 8 = 40
1 x 9 = 9    2 x 9 = 18    3 x 9 = 27    4 x 9 = 36    5 x 9 = 45
1 x 10 = 10   2 x 10 = 20   3 x 10 = 30   4 x 10 = 40   5 x 10 = 50
```

🕮 Bài tập 3.8

Viết chương trình in ra hình sau đây:

```
**********
```

****

****

****

******

*****

****

***

**

*

Chương trình:

```
#include <stdio.h>
int main() {
    int i, j;
    for (i=0; i<10; i++){
        for (j=0; j<10-i; j++){
            printf("*");
        }
        printf("\n");
    }
    return (0);
}
```

Bài tập 3.9

Rút tiền tại cây ATM. Giả sử tại cây ATM của một ngân hàng X chỉ chứa các loại tiền mệnh giá 20000, 50000, 100000, 200000. Một khách hàng một rút một số tiền a nào đó. Số tiền a cần rút phải là bội số của 20000. Hãy in ra màn hình tất cả các phương án mà cây ATM có thể "nhả" tiền ra cho khách hàng? Tổng có bao nhiêu phương án rút tiền?

Phác thảo lời giải

- Khai báo các hằng m1, m2, m3, m4 ghi nhận các mệnh giá 20000, 50000, 100000, 200000.
- Nhập và kiểm tra số tiền nhập vào, đảm bảo là bội số của 20000.
- Xác định số tờ tối đa của từng mệnh giá: số tờ tối đa của từng mệnh giá bằng số tiền chia cho mệnh giá.

- Ở mỗi phương án rút tiền, nếu tổng tiền từng mệnh giá = số tiền thì in các phương án ra.

Chương trình:

```
#include <stdio.h>
#include <conio.h>
#define m1 20000
#define m2 50000
#define m3 100000
#define m4 200000
 main()
{
// khai bao so tien can rut
long sotien;
// khai bao so to  toi da cua cac menh gia 20000, 50000, 100000, 200000
int t1, t2, t3, t4;
// khai bao cac bien vong lap, la so to thuc te cua cac menh gia
// 20000, 50000, 100000, 200000
int i, j, k, l;
// dem cac phuong an
int dem =0;
// nhap va kiem tra so tien can rut la boi so cua 20000
printf("Nhap vao so tien can rut:");
do{
  scanf("%ld",&sotien);
  if (sotien % m1 !=0) printf("Nhap lai so tien! So tien la bo cua %d: ",m1);
} while (sotien % m1 !=0);
t1 = (int)sotien/m1;
t2 = (int)sotien/m2;
t3 = (int)sotien/m3;
t4 = (int)sotien/m4;
printf("In cac phuong an:\n");
printf("  P.A  20000 50000 100000 200000");
```

```
30   for (i=0; i <= t1; i++)
31   for (j=0; j <= t2; j++)
32   for (k=0; k <= t3; k++)
33   for (l=0; l <= t4; l++)
34   if ((i*m1 + j*m2 + k*m3 + l*m4 ) == sotien)
35   printf("\n%5d : %5d %5d %5d %5d", ++dem, i, j, k, l );
36   printf("\nTong so phuong an rut tien: %d", dem);
37   getch();
38   }
```

✍Kết quả chương trình:

```
Nhap vao so tien can rut: 50
Nhap lai so tien! So tien la bo cua 20000: 500000
In cac phuong an:
 P.A  20000 50000 100000 200000
  1 :     0     0      1      2
  2 :     0     0      3      1
  3 :     0     0      5      0
  4 :     0     2      0      2
  5 :     0     2      2      1
  6 :     0     2      4      0
  7 :     0     4      1      1
  8 :     0     4      3      0
  9 :     0     6      0      1
 10 :     0     6      2      0
 11 :     0     8      1      0
 12 :     0    10      0      0
 13 :     5     0      0      2
 14 :     5     0      2      1
 15 :     5     0      4      0
 16 :     5     2      1      1
 17 :     5     2      3      0
 18 :     5     4      0      1
 19 :     5     4      2      0
```

```
20 :  5  6  1  0
21 :  5  8  0  0
22 :  10  0  1  1
23 :  10  0  3  0
24 :  10  2  0  1
25 :  10  2  2  0
26 :  10  4  1  0
27 :  10  6  0  0
28 :  15  0  0  1
29 :  15  0  2  0
30 :  15  2  1  0
31 :  15  4  0  0
32 :  20  0  1  0
33 :  20  2  0  0
34 :  25  0  0  0
Tong so phuong an rut tien: 34
```

📖 Bài tập 3.10

Gửi tiền tiết kiệm

Viết chương trình nhập vào số tiền cần gửi a (a > 1000000) và số tiền sẽ nhận được b (ràng buộc điều kiện b > a). Yêu cầu in ra số năm cần gửi, biết mức lãi xuất 8 % /năm.

✍Phác thảo lời giải

- Tính theo công thức lãi kép, lãi nhập vốn:

+ Lãi = gốc*(1+lãi suất)^n - gốc.

+ Số tiền rút ra = gốc*(1+lãi suất kỳ)^n. (công thức của FV) với n là số năm gửi.

- Sử dụng hàm pow (x, y) để tính x^y (chú ý khai báo tệp tiêu đề #include <math.h>).

✍Chương trình

```
1  #include <conio.h>
2  #include <stdio.h>
3  #include <math.h>
4  int main()
```

```
5   {
6   float sotiengui, sotienrut;
7   int SoNam;
8   SoNam = 0 ;
9   printf("Nhap vao so tien can gui: ");
10  do
11  {
12          scanf("%f", &sotiengui);
13          if (sotiengui <= 1000000) printf("Nhap lai so tien can gui
    lon hon 1000000!: ");
14  } while (sotiengui < 1000000);
15  printf("Nhap vao so tien can rut:");
16  do
17  {
18          scanf("%f", &sotienrut);
19          if (sotienrut <= sotiengui) printf("Nhap lai so tien rut phai
    lon hon %f!",sotiengui);
20  } while (sotienrut < sotiengui);
21  while (sotiengui < sotienrut)
22  {
23  SoNam = SoNam + 1;
24  sotiengui = pow((1 + 0.08), SoNam) * sotiengui;
25  }
26  printf("\nSo nam  can gui la: %d",SoNam);
27  printf("\nSo tien rut duoc la: %2.1f",sotiengui);
28  return 0;
29  }
```

➢Kết quả chương trình:

```
Nhap vao so tien can gui: 500000
Nhap lai so tien can gui lon hon 1000000!: 20000000
Nhap vao so tien can rut:40000000
So nam  can gui la: 4
So tien rut duoc la: 43178500.0
```

B. BÀI TẬP TỰ GIẢI

🕮 Bài tập 3.11

Viết chương trình nhập vào ngày, tháng, năm.

a. Cho biết tháng đó có bao nhiêu ngày.

b. In ra ngày tháng năm trước đó.

Dữ liệu test: 12/12/2019.

Kết quả mong đợi: Tháng 12 có 30 ngày.

Ngày trước đó: 11/12/2019.

🕮 Bài tập 3.12

Viết chương trình nhập vào hai số nguyên a, b. Thực hiện việc lựa chọn công việc và in kết quả lên màn hình:

- Tính tổng hai số a và b.
- Tính hiệu hai số a và b.
- Tính tích hai số a và b.
- Tính thương của hai số a và b.

🕮 Bài tập 3.13

Viết chương trình nhập vào ba số. Kiểm tra ba số vừa nhập có thỏa mãn là độ đo ba góc của tam giác hay không?

Dữ liệu test: 40 50 90.

Kết quả mong đợi: là độ đo ba góc của tam giác.

🕮 Bài tập 3.14

Viết chương trình nhập vào một số nguyên có ba chữ số. In ra cách đọc của số nguyên này

Dữ liệu test: 345.

Kết quả mong đợi: ba trăm bốn mươi lăm.

🕮 Bài tập 3.15

Viết chương trình giải phương trình trùng phương: $ax^4 + bx^2 + c = 0$.

🕮 Bài tập 3.16

Viết chương trình giải hệ phương trình bậc nhất hai ẩn sau đây:

$$\begin{cases} ax + by = c \\ dx + ey = f \end{cases}$$

🕮 Bài tập 3.17

Viết chương trình tính tiền cước TAXI. Biết rằng:

- Tiền mở cửa 5000 VND.
- Km đầu tiên 10000 VND.
- Từ Km thứ hai đến Km thứ 30: 9000 VND.
- Từ Km thứ 31 trở đi: 8000 VND.

Hãy nhập số Km sau đó in ra số tiền phải trả.

🕮 Bài tập 3.18

Viết chương trình tính tiền điện. Biết rằng:

Bậc tính	Giá (VND)
Bậc 1: Cho kWh từ 0-50	1.678
Bậc 2: Cho kWh từ 51-100	1.734
Bậc 3: Cho kWh từ 101-200	2.014
Bậc 4: Cho kWh từ 201-300	2.536
Bậc 5: Cho kWh từ 301- 400	2.834
Bậc 6: Cho kWh từ 401 trở lên	2.927

Nhập số điện tiêu thụ, in số tiền cần thanh toán ra màn hình.

🕮 Bài tập 3.19

Viết chương trình thực hiện các yêu cầu sau:

- Nhập số nguyên dương n ở hệ m bất kỳ ($2 \le m \le 16$). Yêu cầu đổi n sang số ở hệ 10.
- In kết quả ra màn hình.

🕮 Bài tập 3.20

Viết chương trình thực hiện các yêu cầu sau:

- Nhập số nguyên dương n ở hệ 10. Yêu cầu đổi n sang số ở hệ m bất kỳ ($2 \le m \le 16$).
- In kết quả ra màn hình.

🕮 Bài tập 3.21

Viết chương trình thực hiện các yêu cầu sau:

- Thực hiện lặp lại việc nhập 1 số nguyên dương n từ bàn phím, tính n! (n giai thừa). Kết thúc chương trình khi nhập n = 0.
- In kết quả ra màn hình.

🕮 Bài tập 3.22

Viết chương trình in ra hình sau đây:

a.

```
**********
 *********
  ********
   *******
    ******
     *****
      ****
       ***
        **
         *
```

b.

```
********************
 ******************
  ****************
   **************
    ************
     **********
      ********
       ******
        ****
         **
```

c.

```
**********
 **********
  **********
   **********
    **********
     **********
      **********
       **********
        **********
         **********
```

🕮 Bài tập 3.23

Viết chương trình nhập vào số nguyên dương a. In ra màn hình các ước số của a.

Dữ liệu test: 12.

Kết quả mong đợi: 1 2 3 4 6 12.

🕮 Bài tập 3.24

Viết chương trình tính tổng các ước của số nguyên dương n (n nhập từ bàn phím).

Dữ liệu test: 12.

Kết quả mong đợi: 1 + 2 + 3 + 4 + 6 + 12 = 28.

🕮 Bài tập 3.25

a. Viết chương trình kiểm tra một số nhập vào n có là số nguyên tố hay không?

b. In các số nguyên tố nhỏ hơn n.

c. In các số nguyên tố trong phạm vi [a, b] với a và b được nhập từ bàn phím.

🕮 Bài tập 3.26

Viết chương trình giải quyết bài toán cổ sau đây, in ra màn hình các đáp án của bài toán.

"Vừa gà vừa chó

Bó lại cho tròn

Ba mươi sáu con

Một trăm chân chẵn"

🕮 Bài tập 3.27

Viết chương trình giải quyết bài toán cổ sau đây, in ra màn hình các đáp án của bài toán.

"Trăm trâu, trăm cỏ

Trâu đứng ăn năm

Trâu nằm ăn ba

Lụ khụ trâu già, ba con một bó"

Hỏi số trâu đứng, trâu nằm, trâu già?

🕮 Bài tập 3.28

Viết chương trình tìm các số có 3 chữ số, sao cho tổng các chữ số cộng lại bằng 9. In ra màn hình.

🕮 Bài tập 3.29

Viết chương trình xác định số nguyên dương n nhỏ nhất để tổng các số từ 1 đến n lớn hơn một giá trị S cho trước.

🕮 Bài tập 3.30

Viết chương trình tìm ước chung lớn nhất của 2 số nguyên dương A và B.

Trong chương trình yêu cầu người sử dụng phải nhập 2 số nguyên dương nếu nhập sai yêu cầu nhập lại đến khi nhập đúng thì tính.

🕮 Bài tập 3.31

Viết chương trình tính tổng các số chia hết cho 3 và có tận cùng là 6 trong khoảng từ 1 đến 100 bằng lệnh while.

🕮 Bài tập 3.32

Viết chương trình thực hiện các yêu cầu sau:

- Nhập vào một số nguyên n.

- Đếm xem số đó có bao nhiêu chữ số và tính tổng của chúng. In kết quả ra màn hình.

- Phân tích số đó thành tích của các số nguyên tố.

Dữ liệu test n = 12. Phân tích n = 2 * 2 * 3.

🕮 Bài tập 3.33

Xây dựng hàm tìm ước số chung lớn nhất (USCLN) và bội số chung nhỏ nhất (BSCNN) của 2 số nguyên dương bất kỳ. Áp dụng nhập vào 2 số nguyên từ bàn phím, in ra màn hình USCLN và BSCNN của 2 số đó.

🕮 Bài tập 3.34

Viết chương trình tính:

$$\boldsymbol{S = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}}$$

với n được nhập từ bàn phím.

🕮 Bài tập 3.35

Viết chương trình nhập vào số nguyên n, số thực x. Tính giá trị của biểu thức sau đây:

$$S_1 = 1 + x^1 + x^2 + \cdots + x^n$$

$$S_2 = \frac{1}{2^2} - \frac{1}{4^2} + \cdots + \frac{(-1)^{n+1}}{(2n)^2}$$

🕮 Bài tập 3.36

Viết chương trình nhập vào số nguyên n, số thực x. Tính giá trị của biểu thức sau đây:

$$S_1 = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

$$S_2 = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2}$$

🕮 Bài tập 3.37

Viết chương trình tính căn bậc hai của số a cho trước theo công thức lặp sau đây:

$x_0 = a;$

$$x_{n+1} = \frac{(xn + \frac{a}{x_n})}{2}$$

Quá trình lặp sẽ dừng khi $|x_{n+1} - x_n| < \varepsilon$, với ε đủ bé cho trước.

🕮 Bài tập 3.38

Viết chương trình in dãy số Fibonaci.

f1 = f0 =1;

fn = fn-1 + fn-2; (n > 1)

🕮 Bài tập 3.39

Viết chương trình in ra số fibonaci thứ n với n được nhập từ bàn phím.

🕮 Bài tập 3.40

Viết chương trình nhập vào một số tự nhiên n (n > 0)

a. Số tự nhiên n có bao nhiêu chữ số.

b. Hãy tìm chữ số cuối cùng của n.

c. Hãy tìm chữ số đầu tiên của n.

d. Tính tổng các chữ số của n.

e. Hãy tìm số đảo ngược của n.

Chương 4
HÀM

Tại sao chúng ta cần hàm? Có ba lý do sau đây được đưa ra giải thích cho lý do tại sao trong một chương trình nên có các hàm:

Thứ nhất, hàm giúp chúng ta trong việc giảm sự dư thừa mã chương trình. Nếu một đoạn chương trình được thực hiện ở nhiều nơi trong một phần mềm, thay vì lặp đi lặp lại viết cùng một mã, chúng ta sẽ tạo một hàm và gọi nó ở mọi nơi. Điều này cũng giúp bảo trì vì chúng ta chỉ phải thay đổi tại một nơi, thay vì phải sửa đổi tại nhiều vị trí khác nhau trong chương trình.

Thứ hai, hàm được sử dụng giống các modun. Hãy xem xét một tập tin lớn có nhiều dòng mã. Việc đọc và sử dụng mã trở nên thực sự đơn giản nếu mã được chia theo các hàm.

Thứ ba, hàm cung cấp sự trừu tượng. Chẳng hạn, chúng ta có thể sử dụng các hàm như các thư viện mà không phải lo lắng về hoạt động bên trong của nó.

Chương 4 của giáo trình, giới thiệu các khái niệm về hàm, cách thức khai báo hàm, truyền các tham số cho hàm và cách gọi hàm trong chương trình.

4.1. KHÁI NIỆM HÀM

4.1.1. Khái niệm

4.1.1.1. Định nghĩa

Hàm là một chương trình con thực hiện một khối công việc được lặp đi lặp lại nhiều lần trong khi chạy chương trình hoặc dùng tách một khối công việc cụ thể để chương trình đỡ phức tạp.

🕮 Ví dụ 4.1

Xem xét ví dụ sau đây:

Viết chương trình nhập vào ba số nguyên a, b, c. Lần lượt tính tổng a + b, a + c, b + c và in kết quả ra màn hình.

✍Chương trình 1

```
1  #include <stdio.h>
2  int main()
3  {
4      int a,b,c, tong1, tong2, tong3;
```

```
5     printf("Nhap vao ba so: ");
6     scanf("%d %d %d",&a,&b, &c);
7     tong1 = a + b;
8     tong2 = a + c;
9     tong3 = b + c;
10    printf("Tong %d + %d = %d\n", a , b , tong1);
11    printf("Tong %d + %d = %d\n", a , c , tong2);
12    printf("Tong %d + %d = %d\n", b , c , tong3);
13    return 0;
14 }
```

✍Kết quả chương trình

```
Nhap vao ba so: 12 23 43
Tong 12 + 23 = 35
Tong 12 + 43 = 55
Tong 23 + 43 = 66
```

✍Nhận xét:

Chương trình có những công việc được lặp lại, chẳng hạn các công việc ở dòng lệnh 7 đến dòng lệnh 9, dòng lệnh 10 đến dòng lệnh 12. Có thể xây dựng các hàm thể hiện các công việc đó như sau:

✍Chương trình 2

```
1  #include <stdio.h>
2  void tong(int x, int y);   // khai bao phuong thuc tong
3  int main()
4  {
5     int a, b, c;
6     printf("Nhap vao ba so: ");
7     scanf("%d %d %d", &a, &b, &c);
8     tong(a, b);
9     tong(a, c);
10    tong(b, c);
11    return 0;
```

```
12  }
13  void tong(int x,int y)    // dinh nghia ham tinh tong
14  {
15      int kq;
16      kq = x + y;
17      printf("Tong %d + %d = %d\n", x , y , kq);
18  }
```

↘Kết quả chương trình:

```
Nhap vao ba so: 12 23 43
Tong 12 + 23 = 35
Tong 12 + 43 = 55
Tong 23 + 43 = 66
```

4.1.1.2. Phân loại hàm

a. Thư viện chuẩn

Các hàm trong thư viện chuẩn là các hàm dựng sẵn trong ngôn ngữ C để xử lý các công việc như tính toán trong toán học, các hàm xử lý vào ra I/O, xử lý chuỗi, v.v.

Các hàm này được xác định trong các tệp tiêu đề. Khi khai báo các tệp tiêu đề này, các hàm có sẵn này được sử dụng. Chẳng hạn, hàm printf() là hàm thư viện chuẩn để gửi đầu ra được định dạng đến màn hình (hiển thị dữ liệu trên màn hình). Hàm này được định nghĩa trong tệp tiêu đề *"stdio.h"*.

Có rất nhiều hàm được định nghĩa trong cùng một thư viện, chẳng hạn trong thư viện "stdio.h", ngoài hàm printf(), một số hàm như scanf(), fprintf(), getchar(), v.v. sẽ được sử dụng khi đã khai báo thư viện *"stdio.h"*.

- Hàm có sẵn trong thư viện, được phân chia thành 2 loại:
  - Hàm có giá trị trả về: sqrt(), abs(), v.v.
  - Hàm không có giá trị trả về: exit(), getch(), v.v.

b. Hàm do người dùng định nghĩa

Ngôn ngữ C cho phép lập trình viên xác định các hàm. Hàm được tạo bởi người dùng được gọi là hàm do người dùng định nghĩa.

Hàm do người lập trình tự định nghĩa: có 2 loại:

- Hàm có giá trị trả về: tong(), nguyento(), v.v
- Hàm không có giá trị trả về: nhapmang(), xuatmang(), v.v.

4.1.1.3. Ưu điểm của hàm do người dùng định nghĩa

- Chương trình dễ hiểu, dễ duy trì và dễ gỡ lỗi.
- Tái sử dụng mã trong nhiều chương trình khác.
- Một chương trình lớn có thể được phân chia thành các mô đun nhỏ hơn. Do đó một dự án lớn có thể được chia cho nhiều lập trình viên cùng thực hiện.

4.1.1.4. Cách xác định một hàm

Một hàm cần phải xác định 3 thông tin:

- Tên hàm.
- Tham số của hàm (dữ liệu vào).
- Kiểu dữ liệu trả về (kiểu hàm).

4.1.2. Khai báo hàm

4.1.2.1. Cấu trúc chương trình chứa hàm

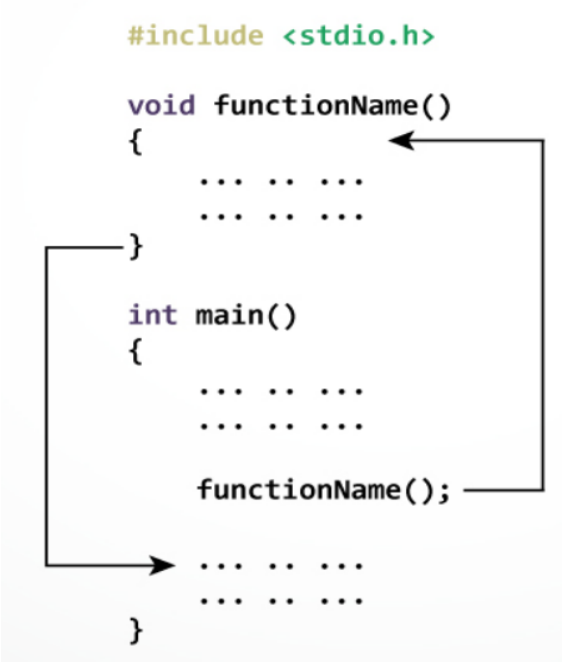
```
#include <stdio.h>
void functionName()
{
   ... .. ...
   ... .. ...
}

int main()
{
   ... .. ...
   ... .. ...

   functionName();
   ... .. ...
   ... .. ...
}
```

Trong đó: functionName(): tên của hàm

Việc thực hiện chương trình bắt đầu từ hàm main(). Khi trình biên dịch gặp hàm functionName() trong hàm main(), điều khiển chương trình nhảy đến hàm void functionName(). Lúc này trình biên dịch bắt đầu thực thi các mã bên trong hàm do người dùng định nghĩa. Khi tất cả các mã trong thân hàm void functionName() đã được thực thi, điều khiển chương trình nhảy về câu lệnh tiếp theo sau lời gọi hàm functionName() trong hàm main().



Hình 4.1. Minh họa cách thức thực thi hàm functionName()

4.1.2.2. Khai báo hàm

a. Khai báo nguyên mẫu hàm

Một nguyên mẫu hàm đơn giản là khai báo một hàm chỉ định tên, tham số và kiểu trả về của hàm, không chứa thân hàm. Một nguyên mẫu hàm cung cấp thông tin cho trình biên dịch biết hàm này có thể được sử dụng sau này trong chương trình.

Cú pháp của nguyên mẫu hàm

returnType functionName(type1 argument1, type2 argument2,...);

Trong Ví dụ 4.1, hàm void tong(int a, int b) là nguyên mẫu hàm cung cấp các thông tin cho trình biên dịch:

+ Tên của hàm là tong().

+ Kiểu trả về của hàm là void.

+ Hai đối số kiểu int được truyền cho hàm.

Nguyên mẫu hàm không cần thiết nếu hàm do người dùng định nghĩa được xác định trước hàm main().

b. Gọi hàm

Điều khiển chương trình nhảy đến hàm do người dùng định nghĩa bằng cách gọi hàm trong hàm main()

Cú pháp gọi hàm:

```
functionName(argument1, argument2, ...);
```

Trong Ví dụ 4.1, lời gọi hàm tong(a, b); đặt trong thân hàm main().

c. Định nghĩa hàm

Cú pháp

```
returnType functionName(type1 argument1, type2 argument2, ...)
{
 //body of the function
}
```

Trong đó:

- **returnType:** kiểu trả về của hàm, có thể là kiểu void, int, float, v.v.

+ Kiểu void: nếu hàm không có giá trị trả về, trong thân hàm main() không có lệnh return;

+ Kiểu int, float, char, bool, v.v.: trong thân hàm main() có lệnh return <giá trị> để trả về một giá trị trùng với kiểu dữ liệu đã khai báo.

functionName: tên của hàm, được đặt theo quy tắc đặt tên, định danh, nên đặt tên dễ gợi nhớ đến công việc cần thực hiện.

- type1, type2, …: Kiểu dữ liệu của các đối số.

- argument1, argument2,…: Các đối số của hàm, được gọi là tham số hình thức. Danh sách các đối số có thể là một, nhiều hoặc không có tham số nào. Nếu có nhiều đối số thì các đối số được ngăn cách với nhau bởi dấu (,), trong trường hợp không có tham số thì để trống, nhưng vẫn phải có dấu ().

🕮 Ví dụ 4.2

```
int max(int a, int b)
```

- Hàm này có tên hàm là max.
- Kiểu hàm là kiểu nguyên (int).
- Có 2 tham số hình thức là a, b có kiểu nguyên.

void Tinhtong(int a, int b)

- Hàm này có tên hàm là Tinhtong.
- Kiểu hàm là void (không trả về giá trị).
- Có 2 tham số hình thức là a, b có kiểu số nguyên.

🕮 Ví dụ 4.3

Viết chương trình tính tổng của hai số a, b nhập từ bàn phím (sử dụng hàm).

Cách 1: Không có kết quả trả về

```
1   #include <stdio.h>
2   #include <conio.h>
3   void tinhtong(int a, int b)
4   {
5   printf("tong= %d", a + b);
6   }
7   main()
8   {
9   int x, y;
10  printf("Nhap x = ");
11  scanf("%d", &x);
12  printf("Nhap y = ");
13  scanf("%d", &y);
14          tinhtong(x, y);
15          getch();
16  }
```

Cách 2: Hàm trả về giá trị

```
1  #include<stdio.h>
2  #include<conio.h>
3  int tinhtong(int a, int b)
4  {
5          return a + b;
6  }
7  main()
```

```
 8  {
 9          int x, y, z;
10          printf("Nhap x = ");
11  scanf("%d", &x);
12          printf("Nhap y = ");
13  scanf("%d", &y);
14          z = tinhtong(x, y);
15          printf("tong = %d", z);
16  getch();
17  }
```

✍**Nhận xét:**

So sánh	
Hàm có kiểu không trả về kết quả	Hàm có kiểu trả về kết quả
Xét dòng 3	
Kiểu hàm: void	Kiểu hàm: int
Thân hàm: dùng printf thông báo tổng	Thân hàm: dùng return trả về giá trị tổng
Trong main: ở dòng 12	
Gọi hàm như một lệnh: tinhtong(x, y)	Gọi hàm trong một biểu thức

4.2. SỬ DỤNG HÀM

4.2.1. Vị trí viết hàm

Cách 1:

#include<stdio.h>

<khai báo nguyên mẫu hàm>;

main()

{

… … …

Lời gọi hàm;

… … …

}

<định nghĩa hàm>

Cách 2:

#include<stdio.h>

<Định nghĩa hàm>

main()

{

--- --- ---

Lời gọi hàm;

… … …

}

✍Lưu ý:

Nguyên mẫu hàm là dòng khai báo cho chương trình dịch biết các thông tin về hàm bao gồm: tên hàm, kiểu hàm và kiểu các tham số (đầu vào) của hàm. Trong nguyên mẫu không bắt buộc phải có tên tham số nhưng kiểu của nó thì bắt buộc.

```
float fmax(float, int);
void nhapmang(int a[], int);
```

🕮 Ví dụ 4.4

Viết chương trình tính tổng của hai số a, b nhập từ bàn phím (sử dụng hàm).

Cách 1

```
#include<stdio.h>
#include<conio.h>
int addNumbers(int, int);
main()
{
int x, y,z;
printf("Nhap x = ");
scanf("%d",&x);
printf("Nhap y = ");
scanf("%d", &y);
z = addNumbers(x, y);
printf("tong = %d",z);
getch();
}
int tinhtong(int a, int b)
{
return a + b;
}
```

Cách 2

```
#include<stdio.h>
#include<conio.h>
int addNumbers (int a, int b)
{
    return a + b;
}
main()
{
    int x, y, sum;
    printf("Nhap x = ");
    scanf("%d", &x);
    printf("Nhap y = ");
    scanf("%d", &y);
    sum = addNumbers(x, y);
    printf("tong = %d", z);
    getch();
}
```

4.2.2. Lời gọi hàm và truyền tham số

4.2.2.1. Lời gọi hàm

Một hàm chỉ được thực thi khi có lời gọi đến nó. Khi hàm được gọi thì các tham số thực sự sẽ lần lượt truyền cho tham số hình thức và hàm được thực hiện bắt đầu từ dấu mở móc { và kết thúc khi gặp đóng móc }.

Cú pháp:

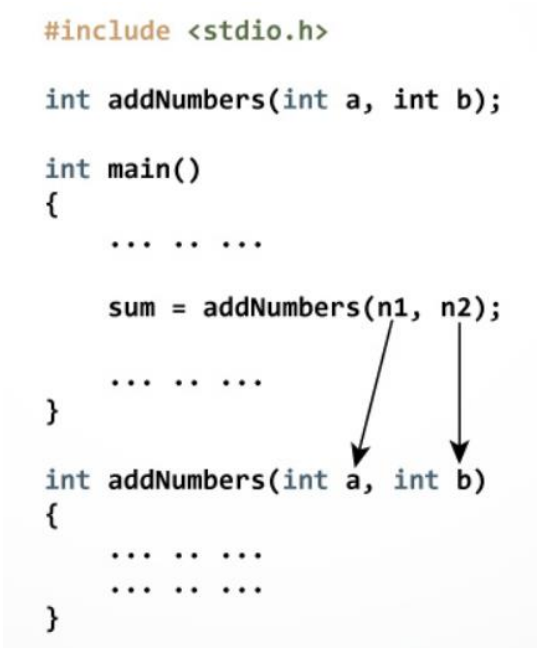
Hàm không có kiểu trả về kết quả	Hàm có kiểu trả về kết quả

- Gọi như một lệnh
<tên hàm> ([danh sách tham số thực sự]);
Ví dụ: tinhtong(x, y);
	- Gọi trong biểu thức
	C = <tên hàm>([danh sách tham số thực sự]);
	- Hoặc kết hợp lệnh in ra màn hình
	printf("%d", <tên hàm>(danh sách tham số thực sự);
	Ví dụ:
	z = tinhtong(x, y);
	printf("tong = d", tinhtong(x, y));

✍**Trong đó:**

- Số tham số thực sự phải bằng số tham số hình thức và có kiểu dữ liệu tương ứng.

- Tham số thực sự được truyền cho các tham số hình thức tuần tự từ trái sang phải, tham số thực sự thứ nhất truyền cho tham số hình thức thứ nhất, tham số thực sự thứ 2 truyền cho tham số hình thức thứ 2, v.v.



Hình 4.2. Minh họa cách truyền tham số

Khi gặp một lời gọi hàm ở một vị trí nào đó thì hàm bắt đầu được thực hiện, tức là máy sẽ tạm rời vị trí đó và chuyển đến hàm tương ứng. Quá trình đó sẽ diễn ra theo trình tự bốn bước sau:

1. Cấp phát bộ nhớ cho các tham số và các biến tự động.

2. Gán giá trị của các tham số thực sự cho các tham số hình thức (hay còn gọi là đối) tương ứng: Nếu đối không phải là con trỏ thì gán giá trị của tham số thực sự cho đối, nếu tham số hình thức là con trỏ thì gán địa chỉ của tham số thực sự cho tham số tương ứng

3. Thực hiện các câu lệnh trong thân hàm.

4. Khi gặp câu lệnh return hoặc dấu } cuối cùng của thân hàm thì máy sẽ giải phóng bộ nhớ cho các đối, các biến/mảng tự động, thoát khỏi hàm và trở về nơi gọi hàm đó.

✍Lưu ý:

- Đối và biến tự động được cấp phát bộ nhớ khi hàm được xét đến và sẽ bị giải phóng khỏi bộ nhớ trước khi ra khỏi hàm. Như vậy, không thể mang giá trị của đối ra khỏi hàm. Có nghĩa là không thể sử dụng đối để làm thay đổi giá trị của bất kỳ đại lượng nào bên ngoài hàm. Nhưng đối và biến tự động có thể trùng tên với bất kỳ đại lượng nào ở bên ngoài hàm mà không gây ra một nhầm lẫn nào.

- Khi một hàm được gọi, giá trị của các tham số thực sự được gán cho các đối không phải là con trỏ. Như vậy, các đối không phải là con trỏ chính là các bản sao của các tham số thực sự. Hàm chỉ làm việc trên các đối này, tức là chỉ làm việc trên các bản sao. Các đối này có thể bị biến đổi trong thân hàm, nhưng các tham số thực sự không bị thay đổi.

4.2.2.2. Truyền tham số

a. Phạm vi hoạt động của biến

Biến toàn cục:

+ Khai báo: Bên ngoài các hàm (cả hàm main). Thường sau khai báo #include.

+ Phạm vi: Toàn bộ chương trình.

+ Thời gian hoạt động: Đến khi kết thúc chương trình.

+ Ưu điểm: Sử dụng linh hoạt.

+ Nhược điểm: Tốn ô nhớ của máy tính.

Biến cục bộ:

+ Khai báo: Trong thân của hàm hoặc khối lệnh{ }.

+ Phạm vi: Chỉ trong bản thân hàm hoặc khối lệnh đó sử dụng được.

+ Thời gian: Biến cục bộ sẽ bị xóa khi ra khỏi thân hàm hoặc khối lệnh.
+ Ưu điểm: Tốn ít ô nhớ.
+ Nhược điểm: Sử dụng trong thân các hàm được khai báo.

🕮 Ví dụ 4.4

Cho đoạn chương trình sau, hãy cho biết kết quả chương trình?

```
1  #include <stdio.h>
2  #include <conio.h>
3  int a = 6;
4  void tinh1()
5  {
6          int a = 5;
7          a = a + 10;
8          printf("a = %d\n", a);
9          if (a > 3)
10         {
11                 int a = 2;
12                 a = a + 5;
13                 printf("a = %d\n", a);
14         }
15 }
16 void tinh2()
17 {
18         a = a + 2;
19         printf("a = %d\n",a);
20 }
21 main()
22 {
23         int a = 3;
24         tinh1();
25         tinh2();
26 printf("a = %d\n",a);
27 getch();
```

```
28  }
```

✍Kết quả chương trình

```
a = 15
a = 7
a = 8
a = 3
```

✍Nhận xét

Khi chạy chương trình máy sẽ thực hiện các lệnh trong main(), gọi đến tinh1() dòng 24.

Ở dòng 8: gặp lệnh printf("a= %d\n", a); in ra a = 15.

Từ dòng 9 đến dòng 13: Khi thực hiện khối lệnh của câu lệnh if, sử dụng biến cục bộ a = 2. Do đó, ở dòng 13 in ra a = 7.

Gọi hàm tinh2(): Trong hàm tinh2() không khai báo biến a, nên sử dụng a toàn cục a = 6. Do đó, ở dòng 19, khi in ra a = 8.

Gặp dòng 26, in ra a = 3.

b. Truyền tham số cho hàm

Nếu một hàm sử dụng các đối số, trong hàm main() phải khai báo các biến chấp nhận các giá trị của các đối số. Các biến này được gọi là tham số hình thức của hàm.

Các tham số hình thức hoạt động giống như các biến cục bộ khác trong hàm, được tạo khi nhập giá trị vào hàm và bị hủy khi thoát khỏi hàm.

Trong các lời gọi hàm, có hai cách để đối số có thể truyền cho hàm:

+ Gọi theo giá trị

Phương thức này sẽ sao chép giá trị thực của một đối số vào tham số hình thức của hàm. Trong trường hợp này, các thay đổi được thực hiện cho tham số bên trong hàm, không có tác dụng với đối số.

+ Gọi theo tham chiếu

Phương thức này sẽ sao chép địa chỉ của một đối số vào tham số hình thức. Bên trong hàm, địa chỉ được sử dụng để truy cập đối số thực tế được sử dụng trong các lời gọi hàm. Điều này có nghĩa là những thay đổi được thực hiện cho các tham số ảnh hưởng đến đối số.

✍**Khái niệm tham số:** Tham số là biến được khai báo trong hàm, dùng để cung cấp giá trị cho hàm và lưu kết quả tính toán cho hàm.

① **Truyền tham trị:** Khi một hàm được gọi, giá trị của các tham số thực sự được gán cho các tham số hình thức, các tham số hình thức này chính là các bản sao của các tham số thực sự. Các đối này được cấp phát bộ nhớ khi hàm được xét đến và sẽ bị giải phóng khỏi bộ nhớ trước khi ra khỏi hàm. Như vậy, không thể mang giá trị của đối ra khỏi hàm. Có nghĩa là không thể sử dụng đối để làm thay đổi giá trị của bất kỳ đại lượng nào bên ngoài hàm. Hàm chỉ làm việc trên các bản sao, các đối này có thể bị biến đổi trong thân hàm, nhưng các tham số thực sự không bị thay đổi.

🕮 **Ví dụ 4.5**

Nhập vào 2 số nguyên x, y. Hoán đổi vị trí hai số nguyên x và y.

✍**Chương trình**

```
1   #include<stdio.h>
2   void hoandoi(int a, int b)
3   {
4           int tg;
5           tg = a;
6           a = b;
7           b = tg;
8   printf("\nHoan doi trong ham a = %d va b = %d\n", a , b);
9   }
10  main()
11  {
12  int x = 5, y = 9;
13  printf("\nTruoc khi doi a = %d va b = %d\n", x , y);
14  hoandoi(x, y);
15  printf("\nSau khi doi a = %d va b = %d\n", x , y);
16  }
```

✍**Kết quả chương trình**

```
Truoc khi doi a = 5 va b = 9
Hoan doi trong ham a = 9 va b = 5
```

Sau khi doi a = 5 va b = 9

✍Nhận xét:

- Ở dòng 2 hàm hoandoi chúng ta sử dụng hai tham số hình thức là tham trị. Khi gọi đến hàm này ở dòng 14, hoandoi(x,y) máy tính sẽ cấp phát 2 ô nhớ a, b là bản sao của 2 biến x và y, 2 ô nhớ này có địa chỉ khác x và y nên khi thực hiện các lệnh trong hàm này nó chỉ thực hiện trên 2 bản sao này và nó chỉ thay đổi trong hàm này. Khi ra khỏi hàm sẽ bị giải phóng khỏi bộ nhớ. Như vậy, không thể mang giá trị của đối ra khỏi hàm. Nên các tham số thực sự x và y truyền vào cho a, b không bị thay đổi.

🕮 Ví dụ 4.6

Nhập vào 2 số nguyên x, y. Hoán đổi vị trí hai số nguyên x và y (ở Ví dụ 4.5), kiểm tra địa chỉ của các biến ở các hàm.

✍Chương trình

```
1  #include<stdio.h>
2  void hoandoi(int a, int b)
3  {
4  int tg;
5  tg = a;
6  a = b;
7  b = tg;
8  printf("Hoan doi trong ham a = %d va b = %d\n", a, b);
9  printf("Dia chi cua bien a,b trong ham hoan doi la:\na = %p\nb =
   %p\n", &a, &b);
10 }
11 main()
12 {
13 int x = 5, y = 9;
14 printf("Truoc khi hoan doi a = %d va b = %d\n", x, y);
15 hoandoi(x, y);
16 printf("Dia chi cua bien x,y trong main:\n x= %p\ny = %p\n", &x,
   &y);
17 printf("Sau doi a = %d va b = %d\n",x, y);
```

```
18 }
```

✍Kết quả chương trình:

```
Truoc khi hoan doi a = 5 va b = 9
Hoan doi trong ham a = 9 va b = 5
Dia chi cua bien a,b trong ham hoan doi la:
a = 000000000023FE20
b = 000000000023FE28
Dia chi cua bien x, y trong main:
X = 000000000023FE4C
Y = 000000000023FE48
Sau doi a = 5 va b= 9
```

✍Nhận xét:

Ở ví dụ 4.6, chúng ta thấy rõ được địa chỉ của tham biến trị ở dòng 2 void hoanvi(int a, int b), khi gọi đến hàm ở dòng 15, hoanvi(x,y) máy tính sẽ cấp phát 2 ô nhớ có địa chỉ như trên là a, b khác với địa chỉ của x, y ở trong main(). Nên mọi thao tác trong hàm không làm thay đổi giá trị của các biến x, y.

② Truyền tham chiếu theo con trỏ

Khi một hàm được gọi, giá trị của các tham số thực sự được gán cho các tham số hình thức là con trỏ, các đối là con trỏ chính là giá trị của các tham số thực sự. Như vậy, hai địa chỉ của tham số thực sự và tham số hình thức đều cùng một địa chỉ, vì vậy mọi thao tác trên tham số hình thức cũng chính là thao tác trên tham số thực sự. Có nghĩa là có thể sử dụng kết quả đối để làm thay đổi giá trị của bất kỳ đại lượng nào bên ngoài hàm. Như vậy, có thể mang giá trị của đối ra khỏi hàm.

🕮 Ví dụ 4.7

Nhập vào 2 số nguyên a, b. Hoán đổi vị trí 2 số nguyên vừa nhập.

✍Chương trình

```
1 #include<stdio.h>
2 void hoandoi(int *a, int *b)
3 {
```

```
4               int tg;
5               tg = *a;
6               *a = *b;
7               *b = tg;
8   printf("Hoan doi trong ham a = %d va b = %d\n",*a , *b);
9   }
10  main()
11  {
12              int x = 5, y = 9;
13  printf("Truoc khi hoan doi a = %d va b = %d\n", x , y);
14              hoandoi(&x, &y);
15              printf("Sau doi a = %d va b = %d\n", x , y);
16  }
```

✍Kết quả chương trình:

```
Truoc khi hoan doi a = 5 va b = 9
Hoan doi trong ham a = 9 va b = 5
Sau doi a = 9 va b = 5
```

🕮 Ví dụ 4.8

Nhập vào 2 số nguyên a, b. Hoán đổi vị trí 2 số nguyên vừa nhập.

Xét chương trình ở ví dụ 4.7. In ra địa chỉ của các biến ở chương trình con và địa chỉ của biến ở chương trình chính.

✍Chương trình:

```
1   #include<stdio.h>
2   void hoandoi(int *a, int *b)
3   {
4               int tg;
5               tg = *a;
6               *a = *b;
7               *b = tg;
8   printf("Trong hoandoi a = %d va b = %d\n",*a , *b);
```

```
9   printf("Dia chi cua bien a,b trong ham hoan doi:\na= %p\nb = %p\n", a , b);
10  }
11  main()
12  {
13          int x = 5, y = 9;
14  printf("Truoc khi doi a = %d va b = %d\n",x, y);
15          hoandoi(&x, &y);
16          printf("Dia chi cua bien x, y trong main:\nx = %p va\ny = %p\n", &x , &y);
17          printf("Sau doi a = %d va b = %d\n", x , y);
18  }
```

✍Kết quả chương trình:

```
Truoc khi doi a = 5 va b = 9
Trong hoandoi a = 9 va b = 5
Dia chi cua bien a,b trong ham hoan doi:
a = 000000000023FE4C
b = 000000000023FE48
Dia chi cua bien x, y trong main:
x = 000000000023FE4C va
Y = 000000000023FE48
Sau doi a = 9 va b = 5
```

🕮 Ví dụ 4.9

Dùng hàm giải phương trình bậc 2: $ax^2 + bx + c = 0$ (a <> 0)

a. Xây dựng hàm nhập các hệ số a, b, c;
b. Xây dựng hàm để tính nghiệm số;
c. Xây dựng hàm để in kết quả.

✍Phác họa chương trình

Xem xét đoạn chương trình của ví dụ 3.4:

```
1       #include <stdio.h>
2       #include <math.h>
```

```
3       int main()
4       {
5         float delta, a, b, c, x1, x2;
6         printf("a = "); scanf("%f", &a);
7         printf("b = "); scanf("%f", &b);
8         printf("c = "); scanf("%f", &c);
9         delta = b * b - 4 * a * c;
10        if (delta < 0)
11          printf("ptvn\n");
12        else if (delta == 0)
13                        {
14                          printf("\nPhuong trinh co nghiem kep:\n");
15                          printf("x1 = x2 = %8.2f",-b/(2*a));
16                        }
17            Else
18             {
19               printf("\nPhuong trinh co 2 nghiem phan biet:\n ");
20               x1 = (-b + sqrt(delta))/(2 * a);
21                     x2 = (-b - sqrt(delta))/(2 * a);
22               printf("\nx1 = %8.2f",x1);
23               printf("\nx2 = %8.2f",x2);
24             }
25      }
```

Từ dòng 6 đến dòng 8 nhập a, b, c: Chuyển thành hàm nhaph().

Từ dòng 10 đến dòng 23 tính nghiệm và in kết quả: Xây dựng thành các hàm tính nghiệm.

✍Chương trình dùng hàm

```
1  #include <stdio.h>
2  #include <math.h>
3  void Nhaphs(float *x)
4  {
5                 printf("\n Nhap he so: ");scanf("%f", x);
6  }
```

```
void Delta_am(float a, float b, float c, float delta)
{
		printf("ptvn\n");
}
void Delta_duong(float a, float b,float c, float delta, float x1,float x2)
{
    x1 = (-b + sqrt(delta))/(2*a);
    x2 = (-b - sqrt(delta))/(2*a);
		printf("\nPhuong trinh co 2 nghiem phan biet:\nx1 = %f\nx2 = %f", x1, x2);
}
void Delta_khong(float a, float b,float c, float delta, float x1,float x2)
{
x1 = x2 = -b/(2*a);
printf("\nPhuong trinh co nghiem kep:\nx1 = x2 = %f",x1);
}
void Inketqua(float a, float b, float c, float delta, float x1, float x2)
{
		if (delta < 0)
		Delta_am(a, b, c, delta);
		else if (delta > 0)
		Delta_duong(a, b, c, delta, x1, x2);
		else
		Delta_khong(a, b, c, delta, x1, x2);
}
int main()
{
  float delta, a, b, c, x1, x2;
  do
  {
  Nhaphs(&a);
  }while (a == 0);
```

```
38    Nhaphs(&b);
39    Nhaphs(&c);
40    delta = b * b - 4 * a * c;
41    Inketqua(a, b, c, delta, x1, x2);
42  }
```

✍Kết quả chương trình

```
Nhap he so: 1
Nhap he so: 3
Nhap he so: 2
Phuong trinh co 2 nghiem phan biet:
x1= -1.000000
x2= -2.000000
```

🕮 Ví dụ 4.10

Dùng hàm viết chương trình thực hiện các yêu cầu sau:
Nhập vào 3 số thực a, b, c từ bàn phím.
a. Kiểm tra a, b, c có lập thành ba cạnh của tam giác không?
b. Nếu có hãy kiểm tra đây là tam giác cân, đều, vuông hay tam giác thường.
c. Tính chu vi và diện tích của tam giác. In kết quả ra màn hình.

✍Phác họa lời giải

Đây là ví dụ 3.8 chúng ta đã viết chương trình trong một hàm main, dựa vào chương trình ở dưới chúng ta chuyển thành hàm:

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <math.h>
4   main()
5   {
6   float a, b,c;
7           printf("\nNhap canh a = "); scanf("%f", &a);
8           printf("\nNhap canh b = "); scanf("%f", &b);
9           printf("\nNhap canh c = "); scanf("%f", &c);
10  if (a + b > c && b + c > a && c + a > b && a > 0 && b >0 &&
```

```
    c >0)
11  {
12          if (a == b && b == c) printf("\nTam giac deu");
13          else if (a == b ||b == c|| c == a) printf(" \nTam giac can");
14          else if (a * a + b * b == c * c ||a * a + c * c == b * b||c * c
    + b * b == a * a) printf("\nTam giac vuong");
15          else printf("\nTam giac thuong");
16          float p = (a + b + c)/2;
17          float s = sqrt(p * (p - a) * ( p - b) * (p - c));
18          printf("\nChu vi tam giac: %.2f",2 * p);
19          printf("\nDien tich tam giac %.2f", s);
20  }
21  else printf("\n khong la tam giac");
22  }
```

Dựa vào chương trình trên ta chia thành hàm như sau:

- Từ dòng 7 đến dòng 9:

```
printf("\nNhap canh a= "); scanf("%f", &a);
printf("\nNhap canh b= "); scanf("%f", &b);
printf("\nNhap canh c= "); scanf("%f", &c);
```

Để tránh lặp đi lặp lại nhiều lệnh, chúng ta sẽ tách thành 1 hàm nhập.

- Dòng 10: tách thành hàm kiểm tra tam giác.
- Từ dòng 12 đến dòng 15 tách thành hàm phân loại tam giác.

```
if (a == b && b == c) printf("\nTam giac deu");
else if (a == b ||b == c|| c == a) printf(" \nTam giac can");
else if (a * a + b * b == c * c ||a * a + c * c == b * b||c * c + b * b ==
a * a) printf("\nTam giac vuong");
else printf("\nTam giac thuong");
```

- Từ dòng 16 đến dòng 21 tách thành hàm tính diện tích, chu vi tam giác:

```
float p = (a + b + c)/2;
float s = sqrt(p * (p - a) * (p - b) * (p - c));
```

```
printf("\nChu vi tam giac: %.2f",2 * p);
printf("\nDien tich tam giac %.2f", s);
}
else printf("\n khong la tam giac");
```

Chương trình dùng hàm:

Cách 1:

```
1  #include <stdio.h>
2  #include <conio.h>
3  #include <math.h>
4  void nhap(float *n)
5  {
6          do
7          {
8                  printf("Nhap canh: "); scanf("%f", n);
9          }while (n <= 0);
10 }
11 bool kttamgiac(float a, float b, float c)
12 {
13         if (a + b > c && b + c > a && c + a > b) return 1;
14         else return 0;
15 }
16 void loaitamgiac(float a, float b, float c)
17 {
18         if (a == b && b == c) printf("\nTam giac deu");
19         else if (a == b ||b == c|| c == a) printf("\nTam giac can");
20         else if (a * a + b * b == c * c ||a * a + c * c == b * b||c * c
   + b * b == a * a) printf("\nTam giac vuong");
21         else printf("\n thuong");
22 }
23 void tinhdientich(float a, float b, float c)
24 {
25         float p = (a + b + c)/2;
26         float s = sqrt(p * (p - a) * (p - b) *( p - c));
```

```
27          printf("\nChu vi tam giac: %.2f",2 * p);
28          printf("\nDien tich tam giac %.2f", s);
29  }
30  main()
31  {
32  float a, b, c;
33  nhap(&a);
34  nhap(&b);
35  nhap(&c);
36  int x = kttamgiac(a, b, c);
37  if (x == 1)
38  {loaitamgiac(a, b, c);
39  tinhdientich(a, b, c);
40  }
41  else printf("\n khong la tam giac");
42  }
```

✍Kết quả chương trình:

```
Nhap canh: 3
Nhap canh: 4
Nhap canh: 5
Tam giac vuong
Chu vi tam giac: 12.00
Dien tich tam giac 6.00
```

✍Nhận xét

Xem xét lại ví dụ 4.10, nếu chúng ta khai báo a, b, c là biến toàn cục thì ở các hàm chúng ta không cần khai bao tham biến hình thức nữa, vì tất cả sẽ dùng biến toàn cục và chúng ta cải tiến dùng menu lựa chọn cho từng hàm ở trong chương trình chính như sau:

Cách 2:

✍Chương trình

```
1  #include <stdio.h>
2  #include <math.h>
```

```
3  float a, b, c;
4  int Kiemtra()
5  {
6    if(a + b > c && a + c > b && b + c > a && a>0 && b>0 &&
   c>0)
7       return 1;
8    else
9      return 0;
10 }
11 void Loaitamgiac()
12 {
13   if (Kiemtra()==1)
14   {
15      if(a == b && b == c)
16         printf("Tam giac deu\n");
17      else if(a == b || a == c || b == c)
18         printf("Tam giac can\n");
19      else if(a * a + b * b == c * c || a * a + c * c == b * b || c * c + b
   * b == a * a)
20         printf("Tam giac vuong\n");
21      else
22         printf("Tam giac thuong\n");
23   }
24   else printf("Khong phai la tam giac\n");
25 }
26 void DienTichChuVi()
27 {
28  if (Kiemtra()==1) {
29   float p, cv, s;
30   cv = a + b + c;
31   p = cv/2;
32   s = sqrt(p * (p - a) * (p - b) * (p - c));
33   printf("\nChu vi: %2.f", cv);
```

```
34    printf("\nDien tich: %2.f",s);}
35   else printf("\nKhong co chu vi, dien tich");
36  }
37  int main()
38  {
39    int luachon;
40    printf("a = "); scanf("%f", &a);
41    printf("b = "); scanf("%f", &b);
42    printf("c = "); scanf("%f", &c);
43    do
44    {
45                    printf("\n-----------MENU-----------------\n");
46       printf("\n 1.Kiem tra a, b, c co lap thanh tam giac duoc
     khong");
47       printf("\n 2.Kiem tra tam giac la tam giac gi");
48       printf("\n 3.Dien tich chu vi tam giac");
49       printf("\n Moi ban chon: "); scanf("%d",&luachon);
50       switch(luachon)
51       {
52       case 1:
53          if (Kiemtra()==1) printf("\n3 canh a, b, c lap thanh tam
     giac");
54          else printf("\n 3 canh a, b, c khong lap thanh tam giac");
55          break;
56       case 2:
57          Loaitamgiac();
58          break;
59       case 3:
60          DienTichChuVi();
61          break;
62       }
63    }while(luachon!= 3);
64  }
```

✍**Kết quả chương trình**

```
a = 3
b = 3
c = 4
1. Kiem tra a,b,c co lap thanh tam giac duoc khong
2. Kiem tra tam giac la tam giac gi
3. Dien tich chu vi tam giac
Moi ban chon: 1
Lap thanh 3 canh tam giac
1. Kiem tra a,b,c co lap thanh tam giac duoc khong
2. Kiem tra tam giac la tam giac gi
3. Dien tich chu vi tam giac
Moi ban chon: 2
Tam giac can
1. Kiem tra a,b,c co lap thanh tam giac duoc khong
2. Kiem tra tam giac la tam giac gi
3. Dien tich chu vi tam giac
Moi ban chon: 3
Chu vi: 10
Dien tich:  4
```

✍So sánh cách 1 và cách 2 của ví dụ 4.10

Hai cách trên đều cho kết quả giống nhau. Nhưng việc dùng biến toàn cục chưa tối ưu vì việc chúng ta khai báo biến toán cục máy tính sẽ cấp phát ô nhớ cho ba biến a, b, c trong suốt toàn bộ chương trình, dẫn đến tình trạng chiếm một phần bộ nhớ của máy tính, tốn bộ nhớ.

🕮 Ví dụ 4.11

Dùng hàm viết chương trình nhập vào một ngày/tháng/năm (ràng buộc cho ngày chỉ nhập từ một đến ngày theo tháng, tháng nhập từ tháng một đến mười hai, năm chỉ nhập năm có bốn chữ số). Tìm ngày kế tiếp (ngày/tháng/năm) của ngày vừa nhập.

✍Phác họa lời giải

Xem lại đoạn chương trình đã viết ở ví dụ 3.30.

```
#include <stdio.h>
#include <conio.h>
int main()
{
        int ng, th, n, d;
        Do
        {
        printf("\n Nhap nam: "); scanf("%d", &n);
        }while(n < 1000 || n > 9999);
        do
        {
        printf("\n Nhap thang: "); scanf("%d", &th);
        }while(th < 1|| th > 12);
        switch(th)
        {
        case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                d=31;
                break;
        case 9: case 4: case 6: d = 30; break;
        case 2:
                {
                        if(n % 4 == 0 && n % 100 !=0||n % 400 ==
0)
                                d = 29;
                        else d = 28;
                }; break;
        }
        do
        {
        printf("\n Nhap ngay: "); scanf("%d", &ng);
        }while(ng < 1|| ng > d);
```

```
30          printf("\n Ngay vua nhap %d-%d-%d\n", ng, th, n);
31          if (ng < d)
32          {
33                  ng++;
34          }
35          else if (th < 12)
36          {
37                  ng = 1; th++;
38          }
39          else
40          {
41                  ng = th = 1; n++;
42          }
43          printf("\nNgay ke tiep: %d- %d - %d", ng, th, n);
44  getch();
45  return 0;
46  }
```

Từ chương trình trên ta xây dựng thành các hàm như sau:

- Từ dòng 14 đến dòng 25 ta xây dựng hàm trả về ngày của tháng, ta sử dụng hàm có kiểu trả về và 2 tham số hình thức là tháng và năm.
- Từ dòng 31 đến dòng 43 ta xây dựng hàm tính ngày tiếp theo, hàm trả về ngày tiếp theo nên ta dùng hàm không trả về kết quả, có 3 tham số hình thức ngày, tháng, năm.

✍Chương trình dùng hàm:

```
1   #include <stdio.h>
2   #include <conio.h>
3   int Ngaycuathang(int th, int n)
4   {
5               int d;
6               switch(th)
7               {
8               case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                d = 31;
```

```
 9              break;
10              case 9: case 4: case 6: d = 30; break;
11              case 2:
12                      {
13                      if(n % 4 == 0 && n % 100! = 0||n % 400 == 0)
14                                      d = 29;
15                              else d = 28;
16                      };break;
17              }
18              return d;
19  }
20  void Timngayketiep(int ng, int th, int n)
21  {
22              int d = Ngaycuathang(th,n);
23              if (ng < d)
24              {
25                      ng++;
26              }
27              else if (th < 12)
28              {
29                      ng = 1; th++;
30              }
31              else
32              {
33                      ng = th = 1; n++;
34              }
35              printf("\nNgay ke tiep: %d- %d - %d", ng, th, n);
36  }
37  int main()
38  {
39              int ng, th, n, d;
40              do
41              {
```

```
42              printf("\n Nhap nam: "); scanf("%d", &n);
43              }while(n < 1000 || n > 9999);
44              do
45              {
46              printf("\n Nhap thang: "); scanf("%d", &th);
47              }while(th < 1||th > 12);
48              d = Ngaycuathang(th, n);
49              do
50              {
51              printf("\n Nhap ngay: "); scanf("%d", &ng);
52              }while(ng < 1||ng > d);
53              printf("\n Ngay vua nhap %d-%d-%d\n", ng,th,n);
54              Timngayketiep(ng,th,n);
55  getch();
56  return 0;
57  }
```

✍Kết quả chương trình

Nhap nam: 2019

Nhap thang: 2

Nhap ngay: 23

Ngay vua nhap 23 - 2 - 2019

Ngay ke tiep: 24 - 2 - 2019

🕮 Ví dụ 4.12

Viết chương trình thực hiện các yêu cầu sau:

Xây dựng hàm tìm ước số chung lớn nhất (USCLN) của 2 số nguyên dương bất kỳ.

Áp dụng, nhập vào 2 số nguyên tương ứng là tử số và mẫu số (mẫu số khác 0) của 1 phân số, in ra màn hình dạng tối giản của phân số đó.

✍Phác họa lời giải

Thuật toán Euclid tính USCLN chỉ đúng với hai số nguyên dương, vì vậy cần lấy giá trị tuyệt đối của hai số trước khi tiến hành thuật toán.

Để phân số có dạng in thuận tiện, ta biện luận như sau:

Nếu tử số và mẫu số khác dấu, xét mẫu số; nếu mẫu số < 0, ta đổi dấu cả tử số và mẫu số để dấu - chuyển sang tử số.

Nếu tử số và mẫu số cùng dấu, xét mẫu số; nếu mẫu số < 0 (nghĩa là tử số cũng < 0), ta đổi dấu cả tử số và mẫu số để loại dấu -.

Mô tả điều kiện trên bằng ngôn ngữ lập trình:

```
if ( ts * ms < 0 ) {
  if ( ms < 0 ) { ts = -ts; ms = -ms; }
} else {
  if ( ms < 0 ) { ts = -ts; ms = -ms; }
}
```

Ta rút gọn thành:

```
if ( ms < 0 ) { ts = -ts; ms = -ms; }
```

Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <math.h>
4   //Ham tim uscln
5   int uscln(int a, int b)
6   {
7           //dung lap kiem tra chung nao a = b thì dung
8           while (a != b)
9           {
10                  if (a > b) a - = b;
11                  else b - = a;
12          }
13          if (a == b) return a;
14  }
15  main()
16  {
17          int ts, ms, dts, dms;
18          printf("Nhap ts= "); scanf("%d", &ts);
19          //rang buoc cho ms
20          do
21          {
```

```
22                  printf("Nhap ms= "); scanf("%d",&ms);
23          }while (ms == 0);
24          //in ra phan so ban dau
25          printf("\nPhan so ban dau la %d/%d\n",ts,ms);
26          //lay so nguyen duong cua ts, ms
27          dts = abs(ts);
28          dms = abs(ms);
29          int a = uscln(dts,dms);
30          //Rut gon phan so
31          ts = ts/a;
32          ms = ms/a;
33          if (ms < 0)
34          {
35                  ts = -ts; ms = -ms;
36          }
37          if (ms == 1) printf("%d\n",ts);
38          else printf("Phan so rut gon: %d/%d\n", ts, ms);
39          getch();
40  }
```

✍Kết quả chương trình

```
Nhap ts = 4
Nhap ms = -6
Phan so ban dau la 4/-6
Phan so rut gon: -2/3
```

🕮 Ví dụ 4.13

Tính S(n) = x + x^2 + x^3 + … + x^n

✍Chương trình

```
1   #include <stdio.h>
2   #include <conio.h>
3   float luythua(float x, int n)
4   {
5           int i;
6           float t = 1;
```

```
7               for( i = 1; i <= n; i++)
8               {
9                       t = t * x;
10              }
11              return t;
12      }
13      float tong(float x, int n)
14      {
15              int i;
16              float s = 0;
17              for(i = 1; i <= n; i++)
18              {
19                      s = s + luythua(x, i);
20              }
21              return s;
22      }
23      int main()
24      {
25              int n;
26              float x;
27              printf("\nNhap n: ");
28              scanf("%d", &n);
29              printf("\nNhap x: ");
30              scanf("%f", &x);
31              float ketqua = tong(x, n);
32              printf("\nTong: %2.1f",ketqua);
33              getch();
34              return 0;
35      }
```

➢Kết quả chương trình

Nhap n: 3

Nhap x: 2

Tong: 14.0

🕮 Ví dụ 4.14

Viết chương trình nhập họ tên, điểm lập trình C, điểm Java của một sinh viên. Tính điểm trung bình và xuất ra kết quả.

✍Chương trình

```
1  #include <stdio.h>
2  #include <conio.h>
3  #include <string.h>
4  char hoten [30];
5  float java;
6  float ltc;
7  float DiemTrungBinh;
8  void Nhap()
9  {
10         fflush(stdin);
11         printf("\nNhap ho ten: ");
12         gets(hoten);
13         printf("\nNhap java: ");
14         scanf("%f", &java);
15         printf("\nNhap ltc: ");
16         scanf("%f", &ltc);
17 }
18 void XuLy()
19 {
20         DiemTrungBinh = (java + ltc) / 2.0;
21         printf("\n%8.2f", DiemTrungBinh);
22 }
23 void Xuat()
24 {
25         printf("\n%-20s|%-8s|%-8s|%-8s\n" , "hoten", "java", "ltc",
   "dtb");
26         printf("%-20s|%-8.2f|%-8.2f|%-8.2f\n" , hoten , java, ltc,
   DiemTrungBinh);
27 }
```

```
int main()
{
        int chon;
        Nhap();
        do
        {
        printf("\n---------MENU---------");
        printf("\n1. Chon Xu ly:");
        printf("\n2. Xuat danh sach: ");
        printf("\n Moi ban chon: "); scanf("%d",&chon);
        switch(chon)
        {
                case 1:
                        printf("\n Chon xu ly: ");
                        XuLy();
                        break;
                case 2:
                        printf("\n Xuat danh sach: ");
                        Xuat();
                        break;
        }
        }while (chon!=3);
        getch();
        return 0;
}
```

✍Kết quả chương trình

```
Nhap ho ten: pham thi thuan
Nhap java: 6
Nhap ltc: 7
---------MENU---------
1. Chon Xu ly:
2. Xuat danh sach:
 Moi ban chon: 1
```

```
 Chon xu ly:
   6.50
---------MENU---------
1. Chon Xu ly:
2. Xuat danh sach:
 Moi ban chon: 2
 Xuat danh sach:
hoten            |java    |ltc     |dtb
pham thi thuan      |6.00    |7.00    |6.50
---------MENU---------
1. Chon Xu ly:
2. Xuat danh sach:
 Moi ban chon: 3
--------------------------------
```

🕮 Ví dụ 4.15

Dùng hàm thực hiện các chương trình sau theo Menu lựa chọn:

a. Nhập vào 1 số nguyên dương n (n > 0). Viết hàm kiểm tra n có phải là nguyên tố không? (số nguyên tố là số tự nhiên khác 0 chỉ có hai ước số dương phân biệt là 1 và chính nó, số 1 không là số nguyên tố).

b. In ra các số nguyên tố < n.

c. Nhập vào 1 số nguyên dương n (n > 0). Viết hàm kiểm tra n có là số hoàn hảo không? (số hoàn hảo là số có tổng các ước = chính nó).

d. In ra các số hoàn hảo < n.

e. Nhập vào 2 số nguyên dương a, b. Viết hàm tìm USCLN của hai số nguyên a, b.

f. Nhập vào 2 số nguyên tương ứng là tử số và mẫu số của phần số, in ra màn hình phân số tối giản.

✍Phác họa lời giải

Mỗi một mục a, b, c , d, e, f chúng ta xây dựng một hàm. Ở cách một ta viết menu khi gọi đến hàm nào thì nhập đối số cho hàm đó.

✍Chương trình

Cách 1:

```
1    #include <stdio.h>
2    #include <math.h>
```

```
int kiemTraSNT(int n)
{
  int i;
  if(n < 2)
    return 0;
  for(i = 2; i < n;i++)
  {
    if(n % i == 0)
      return 0;
  }
  return 1;
}
void inSNT(int n)
{
  int i;
  printf("\nCac so nguyen to la: ");
  for(i = 2; i < n; i++)
  {
    if(kiemTraSNT(i) == 1)
    {
      printf("%d ", i);
    }
  }
}
int kiemTraSHH(int n)
{
  int i, S;
  S = 0;
  for(i = 1; i < n; i++)
  {
    if(n % i == 0)
    {
      S = S + i;
    }
  }
```

```
38      if(S == n)
39        return 1;
40      else
41        return 0;
42    }
43    void inSHH(int n)
44    {
45      int i;
46      printf("\nCac so hoan hao la: ");
47      for(i = 1; i < n; i++)
48      {
49        if(kiemTraSHH(i) == 1)
50        {
51          printf("%d ",i);
52        }
53      }
54    }
55    int UCLN(int a,int b)
56    {
57              int ad = abs(a);
58              int bd = abs(b);
59      while(ad != bd)
60      {
61        if(ad > bd)
62          ad = ad - bd;
63        Else
64          bd = bd - ad;
65      }
66      return ad;
67    }
68    void phanSo(int a, int b)
69    {
70      int c = UCLN(a, b);
71      int am = a/c;
72      int bm = b/c;
```

```
73      if (bm < 0)
74      {am = -am; bm = -bm;}
75      if (b == 1) printf("\n Phan so %d",a);
76      else printf("\nPhan so toi gian %d/%d = %d/%d\n", a,   b, am, bm);
77    }
78    int main()
79    {
80      int n, i, luachon;
81      int a, b;
82      Do
83      {
84        printf("\n-----------MENU-----------\n");
85        printf("\n1. Kiem tra so nguyen to");
86        printf("\n2. In cac so nguyen to");
87        printf("\n3. Kiem tra so hoan hao");
88        printf("\n4. In cac so hoan hao");
89        printf("\n5. Phan so toi gian");
90        printf("\nMoi ban chon: ");scanf("%d",&luachon);
91        switch(luachon)
92        {
93        case 1:
94              do {
95           printf("nhap n = ");
96           scanf("%d",&n);
97           if (n < 1)
98             printf("\nn duong, nhap lai\n");
99           } while (n < 1);
100          if(kiemTraSNT(n) == 1)
101            printf("\n%d la so nguyen to\n",n);
102          else
103            printf("\nkhong phai so nguyen to\n");
104          break;
105       case 2:
106             do {
107           printf("nhap n = ");
```

```
108             scanf("%d",&n);
109             if (n < 1)
110              printf("\nn duong, nhap lai\n");
111             } while (n < 1);
112           inSNT(n);
113           break;
114         case 3:
115               do {
116             printf("nhap n = ");
117             scanf("%d", &n);
118             if (n < 1)
119              printf("\nn duong, nhap lai\n");
120           } while (n < 1);
121           if(kiemTraSHH(n) == 1)
122             printf("\n%d la so hoan hao\n",n);
123           else
124          printf("\n%d khong phai la so hoan hao\n",n);
125           break;
126         case 4:
127               do {
128             printf("nhap n = ");
129             scanf("%d",&n);
130             if (n < 1)
131              printf("\nn duong, nhap lai\n");
132             } while (n < 1);
133           inSHH(n);
134           break;
135         case 5:
136               do {
137             printf("Nhap tu so = ");
138             scanf("%d", &a);
139             printf("\nNhap mau so = ");
140             scanf("%d", &b);
141            if (a == 0||b == 0)
142             printf("\nNhap so khac 0, nhap lai: \n");
```

```
143             } while (a == 0||b == 0);
144             phanSo(a, b);
145             break;
146         }
147     }while(luachon!= 5);
148 }
```

➢Kết quả chương trình

```
----------MENU----------
1. Kiem tra so nguyen to
2. In cac so nguyen to
3. Kiem tra so hoan hao
4. In cac so hoan hao
5. Phan so toi gian
Moi ban chon: 1
nhap n = 23
23 la so nguyen to
----------MENU----------
1. Kiem tra so nguyen to
2. In cac so nguyen to
3. Kiem tra so hoan hao
4. In cac so hoan hao
5. Phan so toi gian
Moi ban chon: 2
nhap n = 12
Cac so nguyen to la: 2 3 5 7 11
----------MENU----------
1. Kiem tra so nguyen to
2. In cac so nguyen to
3. Kiem tra so hoan hao
4. In cac so hoan hao
5. Phan so toi gian
Moi ban chon: 3
nhap n = 6
6 la so hoan hao
----------MENU----------
```

```
1. Kiem tra so nguyen to
2. In cac so nguyen to
3. Kiem tra so hoan hao
4. In cac so hoan hao
5. Phan so toi gian
Moi ban chon: 4
nhap n = 18
Cac so hoan hao la: 6
----------MENU----------
1. Kiem tra so nguyen to
2. In cac so nguyen to
3. Kiem tra so hoan hao
4. In cac so hoan hao
5. Phan so toi gian
Moi ban chon: 5
Nhap tu so = -6
Nhap mau so = -9
Phan so toi gian -6/-9 = 2/3
```

✍Nhận xét

Trong chương trình chúng ta thấy từ case 1 đến case 4 đều có lặp các lệnh:

```
do {
printf("nhap n = ");
scanf("%d",&n);
if (n < 1)
 printf("\nn duong, nhap lai\n");
} while ( n < 1);
```

Mục đích của chương trình là muốn gọi đến hàm nào sẽ nhập n tương ứng cho hàm đó. Nếu chúng ta chỉ nhập một lần n ở đầu của menu thì ta có thể sửa chương trình gọn hơn như sau:

✍Chương trình

✍ Cách 2:

```
1  #include <stdio.h>
2  #include <math.h>
3  int kiemTraSNT(int n)
```

```
 4  {
 5    int i;
 6    if(n < 2)
 7      return 0;
 8    for(i = 2; I < n; i++)
 9    {
10      if(n % i == 0)
11        return 0;
12    }
13    return 1;
14  }
15  void inSNT(int n)
16  {
17    int i;
18    printf("\nCac so nguyen to la: ");
19    for(i = 2; i < n; i++)
20    {
21      if(kiemTraSNT(i) == 1)
22      {
23        printf("%d ", i);
24      }
25    }
26  }
27  int kiemTraSHH(int n)
28  {
29    int i, S;
30    S = 0;
31    for(i = 1; i < n; i++)
32    {
33      if(n % i == 0)
34      {
35        S = S + i;
36      }
37    }
38    if(S == n)
39      return 1;
```

```
40      Else
41        return 0;
42    }
43    void inSHH(int n)
44    {
45      int i;
46      printf("\nCac so hoan hao la: ");
47      for(i = 1; i < n; i++)
48      {
49        if(kiemTraSHH(i) == 1)
50        {
51          printf("%d ", i);
52        }
53      }
54    }
55    int UCLN(int a,int b)
56    {
57              int ad = abs(a);
58              int bd = abs(b);
59      while(ad!= bd)
60      {
61        if(ad > bd)
62          ad = ad - bd;
63        else
64          bd = bd - ad;
65      }
66      return ad;
67    }
68    void phanSo(int a,int b)
69    {
70      int c = UCLN(a, b);
71      int am = a/c;
72      int bm = b/c;
73      if (bm < 0)
74      {am = -am; bm = -bm;}
75      if (b == 1) printf("\n Phan so %d",a);
```

```
76      else printf("\nPhan so toi gian %d/%d = %d/%d\n",a,b,am,bm);
77  }
78  int main()
79  {
80    int n,i,luachon;
81    int a,b;
82     do {
83          printf("Nhap n = ");
84          scanf("%d",&n);
85          if (n < 1)
86            printf("\nn duong, nhap lai\n");
87         } while ( n < 1);
88    Do
89    {
90      printf("\n-----------MENU-----------\n");
91      printf("\n1. Kiem tra so nguyen to");
92      printf("\n2. In cac so nguyen to");
93      printf("\n3. Kiem tra so hoan hao");
94      printf("\n4 .In cac so hoan hao");
95      printf("\n5. Phan so toi gian");
96      printf("\nMoi ban chon: ");scanf("%d",&luachon);
97      switch(luachon)
98      {
99      case 1:
100           if(kiemTraSNT(n) == 1)
101           printf("\n%d la so nguyen to\n",n);
102         else
103           printf("\nkhong phai so nguyen to\n");
104         break;
105     case 2:
106          inSNT(n);
107        break;
108     case 3:
109          if(kiemTraSHH(n) == 1)
110          printf("\n%d la so hoan hao\n",n);
111        else
```

```
112             printf("\n%d khong phai la so hoan hao\n",n);
113           break;
114         case 4:
115              inSHH(n);
116           break;
117         case 5:
118              do {
119              printf("Nhap tu so = ");
120              scanf("%d", &a);
121              printf("\nNhap mau so = ");
122              scanf("%d", &b);
123             if (a == 0||b == 0)
124              printf("\nNhap so khac 0, nhap lai: \n");
125            } while (a == 0||b == 0);
126           phanSo(a, b);
127           break;
128         }
129      }while(luachon!= 5);
130   }
```

✍Kết quả chương trình

```
Nhap n = -3
n duong, nhap lai
Nhap n = 46
-----------MENU-----------
1. Kiem tra so nguyen to
2. In cac so nguyen to
3. Kiem tra so hoan hao
4. In cac so hoan hao
5. Phan so toi gian
Moi ban chon: 1
khong phai so nguyen to
-----------MENU-----------
1. Kiem tra so nguyen to
2. In cac so nguyen to
3. Kiem tra so hoan hao
4. In cac so hoan hao
```

```
5. Phan so toi gian
Moi ban chon: 2
Cac so nguyen to la: 2 3 5 7 11 13 17 19 23 29 31 37 41 43
----------MENU----------
1. Kiem tra so nguyen to
2. In cac so nguyen to
3. Kiem tra so hoan hao
4. In cac so hoan hao
5. Phan so toi gian
Moi ban chon: 3
46 khong phai la so hoan hao
----------MENU----------
1. Kiem tra so nguyen to
2. In cac so nguyen to
3. Kiem tra so hoan hao
4. In cac so hoan hao
5. Phan so toi gian
Moi ban chon: 4
Cac so hoan hao la: 6 28
----------MENU----------
1. Kiem tra so nguyen to
2. In cac so nguyen to
3. Kiem tra so hoan hao
4. In cac so hoan hao
5. Phan so toi gian
Moi ban chon: 6
----------MENU----------
1. Kiem tra so nguyen to
2. In cac so nguyen to
3. Kiem tra so hoan hao
4. In cac so hoan hao
5. Phan so toi gian
Moi ban chon: 5
Nhap tu so = 2
Nhap mau so = -6
Phan so toi gian 2/-6 = -1/3
```

BÀI TẬP CHƯƠNG 4

A. BÀI TẬP CÓ LỜI GIẢI

🕮 Bài tập 4.1

Viết chương trình in các số nguyên tố trong phạm vi [a, b] với a, b là hai số nguyên.

✍Phác thảo lời giải

- Xây dựng hàm ktnguyento(int n) để kiểm tra số n có là số nguyên tố hay không? Sử dụng biến flag để ghi nhận trạng thái của một số. Khi khai báo biến flag nhận giá trị khởi tạo 1. Trong quá trình kiểm tra tính chia hết của số n cho i, với i có thể nhận giá trị từ 2 đến n/2, nếu n chia hết hết cho i, lập tức trạng thái của biến flag chuyển về 0, đồng thời lệnh break được thực hiện để thoát khỏi vòng lặp.
- Trong chương trình chính, hàm main(), dựa vào trạng thái của biến flag của các số trong phạm vi [a, b] để in ra các số nguyên tố.

✍Chương trình

```
1   #include <stdio.h>
2   int ktnguyento(int n);
3   int main()
4   {
5       int a, b, i, flag;
6       printf("Nhap vao hai so nguyen:\n");
7       do
8       {
9       printf("Nhap a = ");
10      scanf("%d", &a);
11      printf("Nhap b = ");
12      scanf("%d", &b);
13      if (a > b) printf("Nhap lai! Nhap vao 2 so thoa man a < b:\n");
14      } while (a >= b);
15              printf("Cac so nguyen to trong pham vi tu %d den %d la:
    ", a, b);
16      for(i = a + 1; i < b; ++i)
17      {
18          // i  la so nguyen to, co flag se nhan gia tri 1
```

```
        flag = ktnguyento(i);
        if(flag == 1)
            printf("%d ",i);
    }
    return 0;
}
// Ham do nguoi dung dinh nghia de kiem tra so nguyen to
int ktnguyento(int n)
{
    int j, flag = 1;
    for(j=2; j <= n/2; ++j)
    {
        if (n%j == 0)
        {
            flag =0;
            break;
        }
    }
    return flag;
}
```

✍Kết quả chương trình:

```
Nhap vao hai so nguyen:
Nhap a = 12
Nhap b = 8
Nhap lai! Nhap vao 2 so thoa man a < b:
Nhap a = 6
Nhap b = 27
Cac so nguyen to trong pham vi tu 6 den 27 la: 7 11 13 17 19 23
```

🕮Bài tập 4.2

Viết chương trình sử dụng hàm, chuyển đổi một số từ hệ thập phân sang hệ nhị phân. In kết quả ra màn hình.

✍Phác thảo lời giải

- Xây dựng hàm toBin(int n) chuyển số thập phân sang nhị phân, theo nguyên tắc ghi nhận số dư của phép chia n cho 2. Kết quả thu được của phép chia tiếp tục chia cho 2, đến khi nào kết quả của phép chia thu được bằng 0.

✍Chương trình

```
1   #include<stdio.h>
2   long toBin(int);
3   int main()
4   {
5      long so_np;
6      int so_tp;
7            printf("\nSu dung ham: Chuyen tu thap phan sang nhi
    phan:\n");
8            printf("----------------------------------------\n");
9      printf("Nhap vao so thap phan: ");
10     scanf("%d",&so_tp);
11     //goi ham chuyen doi thap phan sang nhi phan
12           so_np = toBin(so_tp);
13     printf("\nSo nhi phan tuong ung : %ld\n",so_np);
14     return 0;
15  }
16  long toBin(int n)
17  {
18     long np = 0, du , f = 1;
19     while(n != 0)
20     {
21        du = n % 2;
22        np = np + du * f;
23        f = f * 10;
24        n = n / 2;
25     }
26     return np;
27  }
```

✍Kết quả chương trình

Su dung ham: chuyen tu thap phan sang nhi phan:

……

Nhap vao so thap phan: 12

So nhi phan tuong ung : 1100

🕮Bài tập 4.3

Viết chương trình kiểm tra một số có là số Armstrong hay không?

✍Phác thảo lời giải

Số Armstrong là số thỏa mãn tổng lập phương các chữ số bằng chính số đó.

Dữ liệu test: 153

Kết quả mong đợi: 153 là số Armstrong, vì $153 = 1^3 + 5^3 + 3^3$

✍Chương trình

```
1   #include <stdio.h>
2   // nguyen mau ham
3   int checkArmstrong(int n1);
4   int main()
5   {
6      int n1;
7            printf("\nKiem tra mot so co la so Armstrong:\n");
8            printf("-------------------------------------\n");
9      printf("Nhap vao mot so nguyen: ");
10     scanf("%d", &n1);
11     //Goi ham checkArrmstrong()
12     if(checkArmstrong(n1))
13     {
14        printf("So %d la so Armstrong.\n", n1);
15     }
16     else
17     {
18        printf("So %d khong la so Armstrong.\n", n1);
```

```
19          }
20      }
21      //Ham kiem tra mot so co la so Armstrong
22      int checkArmstrong(int n1)
23      {
24          int ld, sum, num;
25          sum = 0;
26          num = n1;
27          while(num!=0)
28          {
29              ld = num % 10;  // tim chu so cuoi cung cua so
30              sum+ = ld * ld * ld;  // tinh lap phuong cua so cuoi cung
        va them vao tong
31              num = num/10;
32          }
33          return (n1 == sum);
34      }
```

✍Kết quả chương trình

```
Kiem tra mot so co la so Armstrong:
-----------------------------------
Nhap vao mot so nguyen: 153
So 153 la so Armstrong.
Kiem tra mot so co la so Armstrong:
-----------------------------------
Nhap vao mot so nguyen: 120
So 120 khong la so Armstrong.
```

🕮 Bài tập 4.4

Viết chương trình in ra tất cả các số Armstrong trong phạm vi [a, b], với a và b là các số nguyên dương.

✍Chương trình

```
1  #include <stdio.h>
2  /* Nguyen mau ham */
3  int isArmstrong(int num);
```

```
4  void printArmstrong(int start, int end);
5  int main()
6  {
7     int start, end;
8     /* Nhap vao can duoi va can tren*/
9     printf("Nhap vao can duoi: ");
10    scanf("%d", &start);
11    printf("Nhap vao can tren: ");
12    scanf("%d", &end);
13    printf("Tat ca cac so Armstrong trong pham vi tu so %d den so
      %d la: \n", start, end);
14    printArmstrong(start, end);
15    return 0;
16 }
17 //Ham kiem tra so Armstrong
18 int isArmstrong(int num)
19 {
20    int temp, lastDigit, sum;
21    temp = num;
22    sum = 0;
23    /* Tinh tong lap phuong cua cac chu so */
24    while(temp != 0)
25    {
26       lastDigit = temp % 10;
27       sum += lastDigit * lastDigit * lastDigit;
28       temp /= 10;
29    }
30    // kiem tra tong lap phuong cac chu so co bang so ban dau
31    if(num == sum)
32       return 1;
33    else
34       return 0;
35 }
36 //Ham in tat ca cac so Armstrong
37 void printArmstrong(int start, int end)
```

```
{
   // Lap tu dau den cuoi va in cac so hien tai, neu no la so Armstrong
   while(start <= end)
   {
      if(isArmstrong(start))
      {
         printf("%3d\t", start);
      }
      start++;
   }
}
```

✍Kết quả chương trình

```
Nhap vao can duoi: 1
Nhap vao can tren: 10000
Tat ca cac so Armstrong trong pham vi tu so 1 den so 10000 la:
1    153    370    371    407
```

🕮 Bài tập 4.5

Viết chương trình kiểm tra một số có là số đối xứng?

Dữ liệu test: 12321.

Kết quả mong muốn: 12321 là số đối xứng.

Dữ liệu test: 123123.

Kết quả mong muốn: 123123 không là số đối xứng.

✍Phác thảo lời giải

Xác định các hàm cần xây dựng để kiểm tra số nhập vào có là số đối xứng hay không?

- Sử dụng hàm isPalindrom () để kết luận một số có là số đối xứng hay không? Trong hàm isPalindrom() sử dụng 1 hàm khác có tên reverse() để lấy nghịch đảo của một số.

- Hàm isPalindrom() trả về một trong hai giá trị 0 hoặc 1. Nếu trả về giá trị 0, kết luận số đã nhập không phải số đối xứng, nếu trả về giá trị 1, kết luận số nhập vào là số đối xứng.

- Hàm reverse() được xây dựng để quy để đưa ra nghịch đảo của một số.

✍Chương trình

```
#include <stdio.h>
#include <math.h>
```

```
/*Nguyen mau ham */
int reverse(int num);
int isPalindrome(int num);
int main()
{
   int num;

   /* Nhap du lieu tu ban phim */
   printf("Nhap vao mot so: ");
   scanf("%d", &num);
   // Dua ra thong bao la so doi xung hay khong
   if(isPalindrome(num) == 1)
   {
      printf("%d la so doi xung.\n", num);
   }
   else
   {
      printf("%d khong la so doi xung.\n", num);
   }
   return 0;
}
// ham kiem tra co la so doi xung
int isPalindrome(int num)
{
   /*
    * Kiem tra so voi so nghich dao cua no
    */
   if(num == reverse(num))
   {
      return 1;
   }
   return 0;
}
// Ham lay nghich dao cua mot so
int reverse(int num)
```

```
38    {
39        /* Tim so chu so trong so da cho */
40        int digit = (int)log10(num);
41        /* Dieu kien co so cua ham de quy */
42        if(num == 0)
43            return 0;
44        // goi de quy
45        return ((num%10 * pow(10, digit)) + reverse(num/10));
46    }
```

✍Kết quả chương trình

```
Nhap vao mot so: 12121
12121 la so doi xung.
Nhap vao mot so: 123123
123123 khong la so doi xung.
```

🕮 Bài tập 4.6

Viết chương trình in ra màn hình số Fibonacci thứ n, với n được nhập từ bàn phím.

✍Phác thảo lời giải

Tìm ra số Fibonacci thứ n, sử dụng công thức đệ quy sau đây:

$$f(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ f(n-1) + f(n-2) & n > 1 \end{cases}$$

Hàm đệ quy để tìm ra số Fibonacci thứ n dựa vào 3 điều kiện sau đây:

Nếu n = 0, trả về kết quả = 0. Do đó số Fibonacci thứ 0 bằng 0.

Nếu n = 1, trả về kết quả = 1. Do đó số Fibonacci thứ 1 bằng 1.

Nếu n > 1, trả về kết quả f(n -1) + f(n -2).

Như vậy, trong chương trình:

- Xây dựng hàm fibo(int n), trả về giá trị của số Fibonacci thứ n trong dãy các số Fibonacci.

✍Chương trình

```
1  #include <stdio.h>
2  /* Nguyen mau ham */
3  unsigned long long fibo(int num);
```

```
int main()
{
    int n;
    unsigned long long fibonacci;
    /*Nhap gia tri */
    printf("Nhap so: ");
    scanf("%d", &n);
 // bien fibonacci de ghi nhan gia tri cua so fibonacci thu n
          fibonacci = fibo(n);
    printf("So fibonacci thu %d co gia tri: %llu", n, fibonacci);
    return 0;
}
// xay dung ham de quy de tim ra so fibonacci thu n
unsigned long long fibo(int num)
{
    if(num == 0)      //neo, dieu kien co so
       return 0;
    else if(num == 1) //dieu kien co so
       return 1;
    else
       return fibo(num-1) + fibo(num-2);
}
```

✍Kết quả chương trình

Nhap so: 20

So fibonacci thu 20 co gia tri: 6765

🕮Bài tập 4.7

Viết chương trình bằng ngôn ngữ C để hiển thị menu máy tính. Chương trình sẽ nhắc người dùng chọn lựa chọn thao tác (từ 1 đến 5).

Sau đó, yêu cầu người dùng nhập hai giá trị nguyên để tính toán.

CHƯƠNG TRÌNH CHÍNH

1. Thêm.
2. Trừ.
3. Nhân lên.

4. Chia.

5. Chia lấy dư.

Người dùng nhập lựa chọn, chẳng hạn chọn 1.

Nhập vào hai số: 12 15.

Kết quả = 27.

Chương trình hiển thị yêu cầu tiếp tục chương trình: “Tiếp tục chương trình? Nhấn phím Y hoặc phím y”.

Nếu người dùng nhấn phím Y hoặc phím y, chương trình sẽ được tiếp tục, người dùng lựa chọn tiếp các lựa chọn (1 đến 5).

Ngược lại chương trình sẽ thoát.

✍**Chương trình**

```
1   #include <stdio.h>
2   #include <stdlib.h>
3      void hienthi()
4      {
5
            printf("================================\n");
6                printf("     CHUONG TRINH CHINH       \n");
7           printf("================================\n");
8                printf("    1.Cong hai so\n");
9                printf("    2.Tru hai so \n");
10               printf("    3.Nhan hai so\n");
11               printf("    4.Chia hai so\n");
12               printf("    5.Chia lay du\n");
13          printf("================================\n");
14     }
15     int Add(int a,int b)
16     {
17     return(a + b);
18     }
19     int Substract(int a, int b)
20     {
21     return(a - b);
22     }
```

```
23  int Multiply(int a, int b)
24  {
25  return(a * b);
26  }
27  float Divide(int a,int b)
28  {
29  return(a/b);
30  }
31  int Modulus(int a, int b)
32  {
33  return(a % b);
34  }
35  int main()
36  {
37  //hien thi menu
38  hienthi();
39  int luachon;
40  int a;
41  int b;
42  char ok;
43  do
44  {
45          printf("Nhap vao lua chon(1-5): ");
46          scanf("%d:",&luachon);
47          printf("Nhap vao hai so nguyen: ");
48          scanf("%d %d", &a, &b);
49          switch(luachon)
50          {
51          case 1:printf("Ket qua = %d", Add(a,b));break;
52          case 2:printf("Ket qua = %d", Substract(a,b));break;
53          case 3:printf("Ket qua = %d", Multiply(a,b));break;
54          case 4:printf("Ket qua = %.2f", Divide(a,b));break;
55          case 5:printf("Ket qua = %d", Modulus(a,b));break;
56          default:printf("Khong hop le!");
57          }
```

```
58          printf("\nNhan phim y hoac Y de tiep tuc: ");
59          scanf("%s",&ok);
60          }while(ok == 'y'||ok == 'Y');
61          return 0;
62  }
```

↘Kết quả chương trình

```
================================
    CHUONG TRINH CHINH
================================
   1. Cong hai so
   2. Tru hai so
   3. Nhan hai so
   4. Chia hai so
   5. Chia lay du
================================
Nhap vao lua chon(1-5): 1
Nhap vao hai so nguyen: 12 3
Ket qua = 15
Nhan phim y hoac Y de tiep tuc: y
Nhap vao lua chon(1-5): 2
Nhap vao hai so nguyen: 12 5
Ket qua = 7
Nhan phim y hoac Y de tiep tuc: y
Nhap vao lua chon(1-5): 3
Nhap vao hai so nguyen: 12 4
Ket qua = 48
Nhan phim y hoac Y de tiep tuc: y
Nhap vao lua chon(1-5): 4
Nhap vao hai so nguyen: 12 4
Ket qua = 3.00
Nhan phim y hoac Y de tiep tuc: y
Nhap vao lua chon(1-5): 5
Nhap vao hai so nguyen: 12 5
Ket qua = 2
Nhan phim y hoac Y de tiep tuc: N
```

B. BÀI TẬP TỰ GIẢI

🕮 Bài tập 4.8

Sử dụng hàm, viết chương trình tính diện tích và chu vi của hình chữ nhật, biết rằng chiều dài và chiều rộng được nhập từ bàn phím.

Dữ liệu test:

Chiều dài = 4.

Chiều rộng = 3.

Kết quả mong muốn:

Chu vi = 14 cm.

Diện tích = 12 cm^2.

🕮 Bài tập 4.9

Sử dụng hàm, viết chương trình tính diện tích và chu vi hình tròn với bán kính được nhập từ bàn phím.

🕮 Bài tập 4.10

Sử dụng hàm, viết chương trình nhập số nguyên dương n (n > 0). Liệt kê n số chính phương đầu tiên.

🕮 Bài tập 4.11

Sử dụng hàm, viết chương trình nhập số nguyên dương n ($0 \leq n \leq 1000$) và in ra cách đọc của n.

Dữ liệu test:

Nhập *n = 105*.

Kết quả mong muốn:

In ra màn hình: *Mot tram le nam*.

🕮 Bài tập 4.12

Sử dụng hàm, viết chương trình tính tiền thuê máy dịch vụ Internet và in ra màn hình kết quả. Với dữ liệu nhập vào là giờ bắt đầu thuê (GBD), giờ kết thúc thuê (GKT), số máy thuê (SoMay).

- Điều kiện cho dữ liệu nhập: $6 \leq GBD < GKT \leq 21$. Giờ là số nguyên.

- Đơn giá: 2500 đ cho mỗi giờ máy trước 17:30 và 3000 đ cho mỗi giờ máy sau 17:30.

🕮 Bài tập 4.13

Sử dụng hàm, viết chương trình tính tiền lương ngày cho công nhân, cho biết trước giờ vào ca, giờ ra ca của mỗi người.

Giả sử rằng:

- Tiền trả cho mỗi giờ trước 12 giờ là 6000 đ và sau 12 giờ là 7500 đ.

- Giờ vào ca sớm nhất: 6 giờ sáng và giờ ra ca trễ nhất: 18 giờ *(giờ nhập vào là số nguyên).*

🕮 Bài tập 4.14

Viết chương trình nhập vào 6 số thực a, b, c, d, e, f . Giải hệ phương trình sau:

$$\begin{cases} \boldsymbol{ax + by = c} \\ \boldsymbol{dx + ey = f} \end{cases}$$

🕮 Bài tập 4.15

Sử dụng hàm, viết chương trình nhập 2 số nguyên dương a, b. Tìm USCLN và BSCNN của a, b.

🕮 Bài tập 4.16

Sử dụng hàm, viết chương trình nhập số nguyên dương n, tìm ước số lẻ lớn nhất của n (trừ ước số n).

Dữ liệu test: *27*

Kết quả mong muốn: *9.*

🕮 Bài tập 4.17

Sử dụng hàm, viết lại chương trình từ bài 3.11 đến 3.30 của Chương 3.

Chương 5
MẢNG VÀ CON TRỎ

Ngoài các kiểu dữ liệu cơ bản (số nguyên, số thực, ký tự), trong ngôn ngữ lập trình C cho phép người lập trình viên có thể tạo ra các kiểu dữ liệu mới. Trong Chương 5, các vấn đề sau đây sẽ được đề cập đến:

- Mảng một chiều và mảng hai chiều: Khai báo, cách thức truy cập các phần tử trong mảng, một số dạng bài tập với mảng một chiều và mảng hai chiều.

- Con trỏ: Định nghĩa, khai báo con trỏ, các phép toán với biến con trỏ, mối quan hệ giữa con trỏ và mảng, sử dụng con trỏ cấp phát bộ nhớ động.

- Chuỗi ký tự: Khai báo, các phép toán trên chuỗi, một số dạng bài tập vận dụng xử lý đối với chuỗi.

5.1. MẢNG

5.1.1. Mảng một chiều

Mảng một chiều là tập hợp các thành phần có cấu trúc (thường được gọi là các phần tử mảng) có thể được truy cập riêng lẻ bằng cách chỉ định vị trí của một thành phần với giá trị chỉ số duy nhất. Mảng phải được khai báo trước khi chúng có thể được sử dụng trong chương trình.

5.1.1.1. Khai báo mảng

Cú pháp:

```
data_Type arrayName [ arraySize ];
```

Trong đó:

data_Type: là kiểu giá trị của các phần tử mảng, có thể là kiểu dữ liệu cơ bản như kiểu số nguyên, số thực, ký tự, chuỗi, v.v. hoặc cũng có thể là kiểu dữ liệu có cấu trúc như kiểu bản ghi, v.v.

arrayName: là tên được đặt cho mảng, cách thức đặt tên biến mảng giống như định danh.

arraySize: kích thước của mảng, luôn được đặt trong cặp dấu [].

🕮 Ví dụ 5.1

Cho khai báo sau đây:

```
int a[5];
```

Khai báo một mảng có tên là a, số lượng phần tử của mảng là 5, kiểu dữ liệu của mỗi phần tử trong mảng a là số nguyên. Chỉ số các phần tử của mảng được đánh từ 0 đến 4. Hình ảnh minh họa của mảng một chiều:

a[0]	a[1]	a[2]	a[3]	a[4]

Lưu ý:

Có thể khai báo và khởi tạo giá trị mảng ban đầu. Khởi tạo mảng rất đơn giản trong lập trình C. Các giá trị khởi tạo được đặt trong dấu ngoặc nhọn trong khai báo và được đặt sau dấu bằng sau tên mảng. Dưới đây là một ví dụ khai báo và khởi tạo một mảng gồm năm phần tử kiểu int. Mảng cũng có thể được khởi tạo sau khi khai báo.

🕮 Ví dụ 5.2

int a[5] = {13, -6, 9, 45, 7};

Không cần thiết phải xác định kích thước của mảng trong quá trình khởi tạo, như ví dụ 5.3 sau đây:

🕮 Ví dụ 5.3

int a[] = {13, -6, 9, 45, 7};

Trong trường hợp này, trình biên dịch xác định kích thước của mảng bằng cách tính số lượng phần tử của mảng:

a[0]	a[1]	a[2]	a[3]	a[4]
13	-6	9	45	7

Điều này rất hữu ích vì kích thước của mảng có thể được kiểm soát bằng cách thêm hoặc xóa các phần tử khởi tạo khỏi định nghĩa mà không cần điều chỉnh kích thước.

Nếu kích thước mảng được xác định trước, tức được cấp phát bộ nhớ theo số phần tử đã khai báo, nhưng chỉ một số phần tử trong mảng được khởi tạo, khi đó các phần tử còn lại sẽ chứa giá trị 0. Với cách thức khai báo như vậy rất thích hợp trong trường hợp phải xử lý với mảng lớn.

🕮 Ví dụ 5.4

Cho khai báo: int a[2000] = {13, 9};

Với khai báo trên, a[0] = 13, a[1] = 9, a[2] = 0,…, a[1999] = 0.

5.1.1.2. Truy cập các phần tử của mảng

Trong C, mảng được truy cập và xử lý giống như các biến.

Cú pháp:

arrayName(index);

Trong đó:

arrayName: Tên biến mảng

index: Chỉ số phần tử truy cập;

🕮 Ví dụ 5.5

Câu lệnh chèn giá trị vào phần tử thứ 3 trong mảng a:

scanf("%d", &a[2]);

Câu lệnh in giá trị của phần tử tứ 3 trong mảng a:

printf("%d", a[2]);

Trong C, mảng có truy cập và cập nhật bằng chỉ số mảng. Nếu mảng có n phần tử, thì chỉ số của mảng được gán từ 0 đến n - 1. Không giống như ngôn ngữ Java, khi chỉ số mảng nằm ngoài phạm vi từ 0,…, n - 1, được xử lý ngoại lệ trong cặp lệnh *try…catch*, mảng trong C có thể không hiển thị bất kỳ một cảnh báo nào nếu chỉ số mảng ngoài giới hạn được truy cập. Thay vào đó, trình biên dịch có thể truy cập các phần tử ngoài phạm vi, do đó dẫn đến các lỗi nghiêm trọng về thời gian thực hiện chương trình.

Để giảm bớt các "rủi ro" các vấn đề liên quan đến chỉ số của mảng, biểu thức sizeof() thường được sử dụng để xử lý vòng lặp trong mảng. Chúng ta có thể sử dụng một marco sử dụng biểu thức sizeof() để tìm số lượng phần tử của mảng, marco này có các tên khác nhau như "lengthof()", "MY_ARRAY_SIZE()", hoặc "NUM_ELEM()", "SIZEOF_STATIC_ARRAY()" , v.v.

🕮 Ví dụ 5.6

Cho đoạn chương trình sau đây:

```
1  #include <stdio.h>
2  int main()
3  {
4  int i;
5  int a[] = {13, -6, 9, 45, 7};
6  for (i = 0; i < sizeof(a)/sizeof(int); i++)
7        {
8          printf("%d\t", a[i]);
```

```
9         }
10  return 0;
11  }
```

✍ Kết quả chương trình:

```
13    -6    9    45    7
```

Lưu ý trong ví dụ trên, kích thước của mảng không được chỉ định rõ ràng. Trình biên dịch biết kích thước của nó là 5 vì 5 giá trị trong danh sách khởi tạo. Thêm một giá trị bổ sung vào danh sách sẽ khiến nó có kích thước thành 6 và do biểu thức sizeof trong vòng lặp for, mã sẽ tự động điều chỉnh theo thay đổi này. Thực hành lập trình tốt là khai báo một kích thước thay đổi và lưu trữ số lượng phần tử trong mảng trong đó.

size = sizeof (tên_mảng) / sizeof (kiểu_dữ_liệu_mảng)

5.1.1.3. Các dạng bài tập mảng một chiều

Dạng 1: Nhập và hiển thị mảng một chiều

a. Nhập mảng

Cách 1:

Nhập từng phần tử của mảng.

Để nhập các phần tử của mảng, ta thực hiện như sau:

- Nhập từng phần tử của mảng sử dụng một vòng lặp
- Xây dựng một hàm nhap() có kiểu void, không trả về giá trị, truyền vào hai tham số là tên biến mảng và số lượng phần tử của mảng.

Cách 2: Sinh ngẫu nhiên các phần tử của mảng, sử dụng hàm rand() để sinh ra các giá trị ngẫu nhiên.

Cú pháp hàm rand():

```
int rand(void)
```

Hàm rand() không nhận tham số vào, trả về một giá trị ngẫu nhiên trong phạm vi từ 0 đến RAND_MAX. Trong đó, RAND_MAX là một hằng số có giá trị mặc định (được định nghĩa trong thư viện *"stdlib.h",* thông thường RAND_MAX = 32767)

b. Hiển thị mảng

Để hiển thị các phần tử của mảng, ta thực hiện như sau:

- Sử dụng một vòng lặp để hiển thị các phần tử của mảng.
- Xây dựng một hàm hienthi() có kiểu void, truyền vào hai tham số là tên biến mảng và số lượng phần tử của mảng.

🕮 Ví dụ 5.7

Viết chương trình nhập vào một mảng gồm n số nguyên (n là số phần tử của mảng, được nhập từ bàn phím) và hiển thị các phần tử ra màn hình, mỗi phần tử cách nhau độ dài phím tab.

✍Phác thảo lời giải

Xây dựng 2 hàm:
- Hàm nhap(): Sử dụng một vòng lặp for nhập từng phần tử của mảng.
- Hàm hienthi(): Sử dụng một vòng lặp for, hiển thị các phần tử của mảng.

✍Chương trình:

```
1   #include <stdio.h>
2   void nhap(int a[], int n) {
3       for (int i = 0; i < n; i++) {
4               printf("Gia tri phan tu a[%d]=", i);
5               scanf("%d", &a[i]);    }
6   }
7   void hienthi(int a[], int n)
8   {
9               printf("Hien thi cac phan tu cua mang :\n");
10              for(int i=0; i < n; i++)
11                      printf("%4d", a[i]);
12  }
13  int main(){
14      int a[1000];
15      int n;
16              printf("Nhap so phan tu cua mang =");
17              scanf("%d", &n);
18              nhap(a, n);
19              hienthi(a,n);
20              return 0;
21  }
```

✍Kết quả chương trình:

```
Nhap so phan tu cua mang = 3
Gia tri phan tu a[0] = 54
```

```
Gia tri phan tu a[1] = 3
Gia tri phan tu a[2] = 23
Hien thi cac phan tu cua mang:
54  3  23
```

🕮 Ví dụ 5.8

Sinh ngẫu nhiên các phần tử của mảng gồm n số nguyên, có giá trị trong phạm vi từ 0 đến 100. Hiển thị mảng ra màn hình.

✍Phác thảo lời giải

Xây dựng hai hàm:
- Hàm Sinh_ngau_nhien(): Sinh ngẫu nhiên các phần tử của mảng.
- Hàm Hien_thi(): Sử dụng một vòng lặp for, hiển thị các phần tử của mảng.

✍Chương trình:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   void Sinh_ngau_nhien(int a[], int n) {
4       int r;
5       for (int i = 0; i < n; i++)
6           {
7           r = rand() % 100;
8           a[i] = r;
9           }
10  }
11  void Hien_thi(int a[], int n)
12  {
13              printf("Hien thi cac phan tu cua mang:\n");
14              for(int i = 0; i < n; i++)
15                      printf("%4d", a[i]);
16  }
17
18  int main(){
19      int a[1000];
20      int n;
```

```
        printf("Nhap so phan tu cua mang = ");
        scanf("%d", &n);
        Sinh_ngau_nhien(a, n);
        Hien_thi(a, n);
    return 0;
}
```

Kết quả chương trình:

```
Nhap so phan tu cua mang =10
Hien thi cac phan tu cua mang:
41  67  34  0  69  24  78  58  62  64
```

Lưu ý:

Hàm rand() này sẽ không hề sinh ngẫu nhiên ra các số mới khi chạy chương trình ở các lần sau. Nghĩa là, kết quả của chương trình trên ở mọi lần chạy sẽ đều sinh ra 10 số giống nhau. Vậy muốn sinh ra các số ngẫu nhiên ở các lần chạy, chúng ta sử dụng thêm hàm srand(time).

Để tạo ra các số ngẫu nhiên khác nhau tại mọi thời điểm chạy chương trình, chúng ta sẽ thêm hàm srand() và truyền vào một tham số seed kiểu int theo thời gian hiện tại. Tham số này thay đổi thì hàm srand() sẽ sinh ra các số khác nhau, bằng cách sử dụng hàm time() trong thư viện time.h. Hàm time() trả về kiểu time_t nhưng chúng ta có thể ép kiểu về int.

Chương trình được viết lại như sau:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void Sinh_ngau_nhien(int a[], int n)
{
    int r;
    srand((int)time(0));
        for (int i = 0; i < n; i++)
      {
       r = rand() % 100;
       a[i]= r;
```

```
12          }
13  }
14  void Hien_thi(int a[], int n)
15  {
16              printf("Hien thi cac phan tu cua mang:\n");
17              for(int I = 0; i < n; i++)
18                      printf("%4d", a[i]);
19  }
20  int main(){
21      int a[1000];
22      int n;
23              printf("Nhap so phan tu cua mang = ");
24              scanf("%d", &n);
25              Sinh_ngau_nhien(a, n);
26              Hien_thi(a, n);
27      return 0;
28  }
```

Để ý về kết quả khi biên dịch, khác so với kết quả của các lần chạy trước:

```
Nhap so phan tu cua mang = 10
Hien thi cac phan tu cua mang:
42  11  19  80  22   7  69   9  54  30
```

Lưu ý:

- Muốn sinh các số nguyên ngẫu nhiên trong phạm vi [a, b], ta sử dụng câu lệnh r = a + rand() % (max +1 - b). Chẳng hạn sinh các số ngẫu nhiên có phạm vi từ 10 đến 50, ta sử dụng lệnh r = 10 + rand() % (50 + 1 - 10).

- Muốn sinh các số thực ngẫu nhiên có giá trị trong phạm vi [a, b], ta sử dụng câu lệnh r = a + rand()/(float) RAND_MAX *(b - a). Chẳng hạn sinh các số thực ngẫu nhiên trong phạm vi từ 5 đến 100, ta sử dụng lệnh r = 5 + rand()/(float) RAND_MAX *(100 - 5).

Dạng 2: Xử lý với các phần tử của mảng

Các bài tập ở dạng này thường được giải quyết thông qua việc duyệt các phần tử của mảng.

🕮 Ví dụ 5.9

Viết chương trình nhập vào điểm của n sinh viên trong lớp (n được nhập từ bàn phím). Tính và in tổng điểm của các sinh viên trong lớp.

✍ Phác thảo lời giải

- Sử dụng mảng marks để lưu điểm của các sinh viên trong lớp.
- Khai báo biến sum để ghi nhận tổng điểm của các sinh viên, ban đầu khởi tạo biến sum = 0.
- Duyệt qua từng phần tử của mảng marks, cộng điểm của từng sinh viên vào biến sum.
- Hiển thị kết quả biến sum khi kết thúc vòng lặp.

✍ Chương trình:

```
1  #include <stdio.h>
2  int main(){
3  int i, n; // khai bao bien n la so sinh vien cua lop
4  int marks[100]; // khai bao mang diem cua cac sinh vien
5  int sum = 0; // bien tong, ghi nhan tong diem cua cac sinh vien
6  printf("Nhap vao so sinh vien trong lop: ");
7  scanf("%d", &n);
8  for(i=0; i<n; i++){
9          printf("Nhap vao diem cua sinh thu %d: ", i+1);
10         scanf("%d", &marks[i]); //Nhap diem cua tung sinh vien
   trong lop
11         sum + = marks[i];
12 }
13 printf("Tong diem cua cac sinh vien trong lop = %d", sum);
14 return 0;
15 }
```

✍Kết quả chương trình:

```
Nhap vao so sinh vien trong lop: 5
Nhap vao diem cua sinh thu 1: 7
Nhap vao diem cua sinh thu 2: 6
Nhap vao diem cua sinh thu 3: 7
Nhap vao diem cua sinh thu 4: 3
Nhap vao diem cua sinh thu 5: 7
Tong diem cua cac sinh vien trong lop = 30
```

Dạng 3: Tìm kiếm và sắp xếp trên mảng

a. Tìm kiếm

Phát biểu bài toán:

Cho dãy số gồm n số nguyên và một số nguyên x. Cho biết có hay không có số x trong dãy số đã cho?

Có hai thuật toán được sử dụng để tìm kiếm một số x trong dãy số nguyên đã cho: Thuật toán tìm kiếm tuyến tính và thuật toán tìm kiếm nhị phân.

a1. Cài đặt bằng thuật toán tìm kiếm tuyến tính

Tư tưởng thuật toán:

Xét dãy số gồm n số nguyên a[0], a[1], a[2],…, a[n -1] và một số nguyên x. Lần lượt so sánh các phần tử của dãy, bắt đầu từ phần tử đầu tiên a[0] cho đến khi gặp phần tử đầu tiên a[i] trong dãy số, $\mathbf{0 \le i \le n-1}$, sao cho a[i] = x, thông báo tìm thấy phần tử trong dãy số có giá trị bằng x, hoặc đi hết đến phần tử cuối cùng a[n - 1] mà không có phần tử nào có giá trị bằng x, thông báo không tìm thấy phần tử có giá trị bằng x trong dãy số.

✍Phác thảo lời giải

- Xây dựng hàm nhập mảng gồm n số nguyên với hai tham số: tên biến mảng và số phần tử của mảng.
- Xây dựng hàm tìm kiếm với ba tham số: tên biến mảng, số phần tử của mảng và giá trị cần tìm x.
- Sử dụng vòng lặp while để tìm kiếm.

🕮 Ví dụ 5.10

Viết chương trình nhập vào mảng gồm n phần tử là các số nguyên và nhập vào một số nguyên x. Thông báo có hay không có phần tử có giá trị bằng x trong mảng?

✍Phác thảo lời giải

- Xây dựng hàm Nhap với hai tham số: tên biến mảng, số phần tử của mảng;
- Xây dựng hàm tìm kiếm tuyến tính, với ba tham số: tên biến mảng, số phần tử của mảng, số cần tìm kiếm, kết quả của hàm trả về là 1 (nếu tìm thấy) và 0 (nếu không tìm thấy).

✍ Chương trình:

```
1  #include<stdio.h>
2  #include<conio.h>
```

```
3  void Nhap(int a[], int n)
4  {
5          for (int i = 0; i<n; i++)
6          {
7                  printf("Gia tri phan tu a[%d]=", i);
8                  scanf("%d", &a[i]);
9          }
10 }
11 int TKTT(int a[], int n, int x)
12 {
13         int i = 0;
14         while ((i< n ) && (a[i]!= x)) i++;
15         if (i == n )return 0;
16         else return 1;
17
18 /*       for (int i = 0; i < n; i++)
19                 if (a[i] == x)
20                         return 1;
21         return 0;
22 */
23 }
24
25 int main()
26 {
27         int a[100];
28         int n, b; int x;
29         printf("Nhap so phan tu cua mang: ");
30         scanf("%d", &n);
31         Nhap(a, n);
32         printf("Nhap phan tu can tim");
33         scanf("%d", &x);
34         b = TKTT(a, n, x);
35         if (b == 1)
36                 printf("%d co trong mang",x);
37         Else
```

```
38          {
39                  printf("%d khong co trong mang", x);
40          }
41          return 0;
42  }
```

✍ Kết quả chương trình:

Tình huống thứ nhất: Có phần tử trong mảng có giá trị bằng x

Nhap so phan tu cua mang: 5
Gia tri phan tu a[0] = 5
Gia tri phan tu a[1] = 65
Gia tri phan tu a[2] = 34
Gia tri phan tu a[3] = 87
Gia tri phan tu a[4] = 23
Nhap phan tu can tim: 65
65 co trong mang

Tình huống thứ hai: không có phần tử nào trong mảng có giá trị bằng x

Nhap so phan tu cua mang: 5
Gia tri phan tu a[0] = 5
Gia tri phan tu a[1] = 65
Gia tri phan tu a[2] = 34
Gia tri phan tu a[3] = 87
Gia tri phan tu a[4] = 23
Nhap phan tu can tim: 88
88 khong co trong mang

a2. Tìm kiếm nhị phân

Điều kiện: Dãy số nguyên đã được sắp xếp.

Phát biểu bài toán: Cho dãy số gồm n số nguyên đã được sắp xếp tăng dần và một số nguyên x? Cho biết có số nguyên x trong dãy số đã cho hay không? Nếu có cho biết vị trí của số x?

✍Ý tưởng thuật toán:

Về cơ bản, thuật toán thu gọn phạm vi tìm kiếm sau một phép so sánh.

So sánh x với phần tử ở giữa dãy số, có chỉ số mid. Nếu x bằng phần tử này, trả về mid.

Nếu x lớn hơn phần tử có chỉ số mid, thì x nếu tìm thấy chỉ có thể nằm ở nửa sau bên phải của dãy số.

Ngược lại, phần tử cần tìm được tiếp tục tiếp kiếm ở nửa trước bên trái của dãy số.

✍Phác thảo lời giải

- Nhập hoặc khởi tạo các giá trị của mảng (điều kiện là mảng đã được sắp xếp tăng dần).
- Xây dựng hàm tìm kiếm nhị phân.
- Hiển thị chỉ số của phần tử nếu tìm thấy.

🕮 Ví dụ 5.11

Viết chương trình nhập vào mảng gồm n phần tử là các số nguyên và nhập vào một số nguyên x. Thông báo có hay không có phần tử có giá trị bằng x trong mảng? Nếu có, hãy in ra chỉ số của phần tử đó.

✍ Chương trình:

```
1  //Chuong trinh viet bang ngon ngu C thuc hien tim kiem nhi phan de quy
2  #include <stdio.h>
3  //Ham tim kiem nhi phan de quy. Tra ve vi tri cua x trong mang a[l..r], nguoc lai tra ve gia tri -1
4  int binarySearch(int a[], int l, int r, int x)
5  {
6  if (r >= l) {
7          //Xac dinh vi tri mid cua day a[l..r]
8          int mid = l + (r - l) / 2;
9          //Neu phan tu can tim o chinh giua cua day
10         //tra ve vi tri cua no
11         if (a[mid] == x)
12                 return mid;
13         //Neu phan tu can tim nho hon a[mid] thi
14         //no chi co the tim thay o nua con ben trai
15         if (a[mid] > x)
16                 return binarySearch(a, l, mid - 1, x);
17         //Nguoc lai phan tu can tim chi co the tim thay o nua con ben phai cua day
18                 return binarySearch(a, mid + 1, r, x);
```

```
19  }
20  //Tra ve gia tri - 1 neu khong co phan tu can tim trong day
21          return -1;
22  }
23  int main(void)
24  {
25  //khoi tao mang ban dau
26  int a[] = { 2, 3, 4, 10, 40 };
27  //xac dinh kich thuoc cua mang, su dung ham sizeof
28  int n = sizeof(a) / sizeof(int);
29  //khoi tao gai tri can tim kiem
30  int x = 10;
31  int result = binarySearch(a, 0, n - 1, x);
32  //Dua ra thong bao co hay khong co phan tu x, neu co chi ra vi
    tri cua phan tu
33  (result == -1) ? printf("Phan tu khong co trong mang") :
    printf("Phan tu duoc tim thay o vi tri %d ", result);
34  return 0;
35  }
```

Mảng khởi tạo giá trị ban đầu (đã sắp xếp tăng dần):

a[] = {2, 3, 4, 10, 40}

Giá trị cần tìm x = 10

✍ **Kết quả chương trình:**

Phan tu duoc tim thay o vi tri 3

✍**Yêu cầu với sinh viên:** Viết lại chương trình với ba hàm sau đây:

- Hàm Nhap(): nhập vào các giá trị của mảng, với hai tham số cơ bản là tên biến mảng, số phần tử của mảng;
- Hàm Hien_thi(): hiển thị các phần tử của mảng, với hai tham số là tên biến mảng, số phần tử của mảng;
- Hàm Sap_xep(): sắp xếp mảng theo chiều tăng dần, hoặc giảm dần, với hai tham số là tên biến mảng và số phần tử của mảng;
- Hàm Tim_kiem_NP(): tìm kiếm phần tử có giá trị bằng x, dựa trên mảng đã sắp xếp, với ba tham số: tên biến mảng, số phần tử của mảng, phần tử cần tìm kiếm x.

b. Sắp xếp mảng

✍Phát biểu bài toán:

Cho dãy số có n phần tử a[0], a[1], a[2],…, a[n -1]. Hãy sắp xếp dãy số đã cho để được dãy số theo thứ tự tăng dần của các phần tử.

✍Phác thảo lời giải

- Xây dựng hàm Nhap_mang() với hai tham số: tên biến mảng và số phần tử của mảng, để nhập giá trị cho các phần tử của mảng.
- Xây dựng hàm Hien_thi() với hai tham số: tên biến mảng và số phần tử của mảng, hiển thị các phần tử của mảng ra màn hình.
- Xây dựng hàm Sapxep() với hai tham số: tên biến mảng, số phần tử của mảng, sử dụng thuật toán đổi chỗ trực tiếp để sắp xếp mảng tăng dần theo giá trị, sử dụng biến tg khi cần tráo đổi giá trị của hai phần tử trong mảng.

🕮 Ví dụ 5.12

Viết chương trình nhập vào mảng gồm n phần tử là các số nguyên. Sắp xếp các phần tử của mảng tăng dần theo giá trị.

✍ Chương trình:

```
1  #include<stdio.h>
2  #include<conio.h>
3  #include<math.h>
4  // khai bao hang MAX = 100 la so phan tu toi da cua mang
5  #define MAX 100
6  void Nhap_mang(int a[], int n)
7  {
8          // Su dung vong lap for de nhap mang, co the su dung
   phuong phap sinh ngau nhien mang
9          for(int i = 0; i < n; i++)
10         {
11                 printf("\nPhan tu a[%d]: ", i);
12                 scanf("%d", &a[i]);
13         }
14 }
15 void Hien_thi(int a[], int n)
16 {
17
18         for(int i = 0; i < n; i++)
19         {
```

```
20                      printf("%4d", a[i]);
21              }
22              printf("\n");
23      }
24      void Sapxep(int a[], int n)
25      {
26              // ngam dinh sap xep tang dan
27              int i,j;
28              for(i=0;i<n-1;i++)
29                for(j=i+1;j<n;j++)
30                      {
31                              if(a[i]>a[j])
32                              {
33              //Su dung bien tg de trao doi gia tri cua 2 bien a[i] va a[j]
34                              int tg = a[i];
35                              a[i] = a[j];
36                              a[j]=  tg;
37                              }
38                                      }
39      }
40      int main()
41      {
42              int n;
43              int a[MAX];
44              //Kiem tra tinh hop le cua so n nhap vao, dam bao 0 <= n <=
        MAX
45              do
46              {
47                      printf("\nNhap so phan tu: ");
48                      scanf("%d", &n);
49                      if(n <= 0 || n > MAX)
50                      {
51                      printf("\Khong hop le. Nhap so trong pham vi tu 0
                        den MAX !");
52                      }
```

```
53          }while(n <= 0 || n > MAX);
54          Nhap_mang(a, n);
55          printf("In cac phan tu cua mang ban dau: \n");
56          Hien_thi(a, n);
57          Sapxep(a, n);
58          printf("In cac phan tu cua mang sap xep tang dan: \n");
59          // Goi ham Hien_thi() de in cac phan tu cua mang ra man
            hinh
60          Hien_thi(a, n);
61          //Dung man hinh
62          getch();
63          return 0;
64  }
```

✍ Kết quả chương trình:

```
MAX = 100:
Nhap so phan tu: -1
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap so phan tu: 102
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap so phan tu: 6
Phan tu a[0]: 12
Phan tu a[1]: 54
Phan tu a[2]: 6
Phan tu a[3]: 72
Phan tu a[4]: 9
Phan tu a[5]: 34
In cac phan tu cua mang ban dau:
12  54   6  72   9  34
In cac phan tu cua mang sap xep tang dan:
6   9  12  34  54  72
```

Dạng 4: Thêm, bớt phần tử trong mảng

a. Chèn một phần tử vào vị trí k của mảng

✍Phát biểu bài toán:

Cho dãy số a[0], a[1], a[2],…., a[n - 1] và một số nguyên dương k ($\mathbf{0} \leq \boldsymbol{k} \leq \boldsymbol{n}$). Chèn một phần tử vào dãy số đã cho ở vị trí k.

✍Phác thảo lời giải

- Nhập giá trị các phần tử của mảng.
- Nhập giá trị phần tử cần chèn value.
- Duyệt mảng từ phải qua trái (cuối dãy), di chuyển các phần tử từ vị trí n - 1 đến k sang phải.
- Gán giá trị phần tử a[k] bằng value.

🕮 Ví dụ 5.13

Viết chương trình nhập vào mảng gồm n phần tử là các số nguyên và nhập vào một số nguyên dương k, một số nguyên x. Hãy chèn vào mảng phần tử có giá trị x tại vị trí k trong mảng?

✍ Chương trình:

```
1   #include <stdio.h>
2   int main()
3   {
4     int a[100], k, i, n, value;
5           printf("Nhap vao so phan tu cua mang\n");
6           scanf("%d", &n);
7           printf("Nhap %d phan tu cua mang\n", n);
8           for (i = 0; i < n; i++)
9           scanf("%d", &a[i]);
10          printf("Nhap vi tri can chen phan tu:\n");
11          scanf("%d", &k);
12          printf("Nhap gia tri phan tu can chen\n");
13          scanf("%d", &value);
14          for (i = n - 1; i >= k ; i--)
15          a[i+1] = a[i];
16                  a[k] = value;
17          printf("Mang sau khi them la\n");
18          for (i = 0; i <= n; i++)
19       printf("%4d", a[i]);
20      return 0;
21  }
```

✍ Kết quả chương trình:

```
Nhap vao so phan tu cua mang
6
```

```
Nhap 6 phan tu cua mang
12
54
23
65
3
77
Nhap vi tri can chen phan tu:
3
Nhap gia tri phan tu can chen
24
Mang sau khi them la
12  54  23  24  65   3  77
```

✍ **Yêu cầu đối với sinh viên:** Xây dựng lại chương trình bằng các hàm sau đây:

- Hàm Nhap_mang()
- Hàm Hien_thi()
- Hàm Them_pt()

b. Xóa phần tử khỏi mảng

✍ **Phát biểu bài toán:**

Cho dãy số a[0], a[1], a[2],…,a[n - 1] và một số nguyên dương k ($\mathbf{0 \le k \le n}$). Xóa phần tử vào dãy số đã cho ở vị trí k.

✍ **Phác thảo lời giải**

- Nhập giá trị các phần tử của mảng.
- Nhập vị trí cần xóa.
- Duyệt mảng từ trái qua phải, lần lượt di chuyển các phần tử bắt đầu từ vị trí k + 1 đến n - 1 sang trái.

🕮 **Ví dụ 5.14**

Viết chương trình nhập vào mảng gồm n phần tử là các số nguyên và nhập vào một số nguyên dương k. Hãy xóa phần tử ở vị trí k của mảng?

✍ **Chương trình:**

```
1  #include <stdio.h>
2  int main()
```

```
{
        int a[100], vt, i, n;
        printf("Nhap vao so phan tu cua mang\n");
        scanf("%d", &n);
        printf("Nhap %d phan tu cua mang:\n", n);
        for (i = 0; i < n; i++)
    scanf("%d", &a[i]);
        printf("Nhap vi tri phan tu can xoa trong mang\n");
        scanf("%d", &vt);
                if (vt >= n+1)
                        printf("Vi tri can xoa khong ton tai.\n");
                Else
                {
                        for (i = vt; i < n - 1; i++)
                                a[i] = a[i+1];
        printf("Mang sau khi xoa:\n");
                for (i = 0; i < n - 1; i++)
                        printf("%4d", a[i]);
                }
   return 0;
}
```

Kết quả chương trình:

```
Nhap vao so phan tu cua mang
8
Nhap 8 phan tu cua mang:
54
34
12
6
33
23
7
83
```

```
Nhap vi tri phan tu can xoa trong mang
3
Mang sau khi xoa:
54  34  12  33  23   7  83
```

✍ **Yêu cầu đối với sinh viên:** Xây dựng lại chương trình bằng các hàm sau đây:

- Hàm Nhap_mang().
- Hàm Hien_thi().
- Hàm Xoa_pt().

c. Xóa bỏ các phần tử trùng lặp trong mảng

✍ **Phát biểu bài toán:**

Cho dãy số a[0], a[1], a[2],…, a[n - 1]. Nếu trong mảng có những phần tử a[i], a[j], a[k],… giống nhau thì chỉ giữ lại phần tử a[i] và xóa bỏ các phần tử khác. Hiển thị mảng sau khi loại bỏ các phần tử trùng lặp trong mảng.

✍ **Phác thảo lời giải:**

Xét phần tử a[i] ($\mathbf{0 \le i \le n - 1}$). Xét các phần tử từ vị trí i + 1 đến vị trí n - 1, so sánh các phần tử này với a[i]. Nếu phát hiện trùng, di chuyển các phần tử này sang trái một vị trí. Quá trình này được lặp đi lặp lại cho đến khi không còn phần tử nào trong phạm vi [i + 1, n - 1] trùng lặp với a[i].

🕮 **Ví dụ 5.15**

Viết chương trình nhập vào một mảng bao gồm n phần tử là các số nguyên, xác định các phần tử trùng lặp và xóa khỏi mảng. In mảng sau khi xóa ra màn hình?

✍ **Chương trình:**

```
1  #include<stdio.h>
2  #include<conio.h>
3  int main()
4  {
5     int a[20], i, j, k, n;
6     printf("Nhap so phan tu cua mang: \n");
7     scanf("%d", &n);
8     printf("Nhap %d phan tu cua mang:\n", n);
9     for(i = 0; i < n; i++)
```

```
    {
        scanf("%d", &a[i]);
    }
    printf("\nMang ban dau la: ");
    for(i = 0; i < n; i++)
    {
        printf("%4d", a[i]);
    }
    printf("\nMang sau khi loai bo phan tu trung lap: ");
    for(i = 0; i < n; i++)
    {
        for(j = i + 1 ; j < n; )
        {
            if(a[j] == a[i])
            {
                for(k = j; k < n; k++)
                {
                    a[k] = a[k+1];
                }
                n--;
            }
            else
            {
                j++;
            }
        }
    }

    for(i = 0; i < n; i++)
    {
        printf("%4d", a[i]);
    }
    getch();
    return 0;
}
```

✍ **Kết quả chương trình:**

```
Nhap so phan tu cua mang:
   8
Nhap 8 phan tu cua mang:
   45
   12
   12
   5
   45
   78
   3
   12
Mang ban dau la:   45  12  12   5  45  78   3  12
Mang sau khi loai bo phan tu trung lap: 45  12   5  78   3
```

✍**Yêu cầu đối với sinh viên:** Xây dựng lại chương trình bằng các hàm sau đây:

- Hàm Nhap_mang().
- Hàm Hien_thi().
- Hàm Xoa_Trung_Lap_pt().

5.1.2. Mảng hai chiều

5.1.2.1. Khai báo mảng hai chiều

Cú pháp:

data_Type array_Name[Row_Size][Column_Size];

data_Type: kiểu dữ liệu của các phần tử của mảng, có thể là kiểu dữ liệu cơ bản như: số nguyên, số thực, ký tự, logic,… cũng có thể là kiểu dữ liệu có cấu trúc như kiểu bản ghi.

array_Name: tên biến mảng, được đặt tên giống tên biến, tên định danh.

Row_Size, Column_Size: kích thước các chiều của mảng hai chiều; thông thường là số hàng và số cột của mảng hai chiều.

🕮 **Ví dụ 5.16**

Khai báo mảng hai chiều, kích thước 4 × 5, các phần tử của mảng có kiểu nguyên.

```
int arr[4][5];
```

Hình ảnh của mảng hai chiều kích thước 4 × 5:

arr[0][0]	arr[0][1]	arr[0][2]	arr[0][3]	arr[0][4]
arr[1][0]	arr[1][1]	arr[1][2]	arr[1][3]	arr[1][4]
arr[2][0]	arr[2][1]	arr[2][2]	arr[2][3]	arr[2][4]
arr[3][0]	arr[3][1]	arr[3][2]	arr[3][3]	arr[3][4]

Tương tự mảng một chiều, có thể vừa khai báo vừa khởi tạo giá trị ban đầu cho mảng.

🕮 Ví dụ 5.17

Khai báo và khởi tạo mảng hai chiều, kích thước 4 x 5 như sau:

```
int arr[4][5] =
{
{12, 0, 9, 5, 7},
{1, 9, 27, 15, 8},
{12, 5, 19, 25, 37},
{0, 0, 23, 63, 41},
};
```

Nếu kích thước mảng được chỉ định (số hàng và số cột), nhưng không phải tất cả các phần tử trong mảng được khởi tạo, khi đó những vị trí chưa khởi tạo trên các dòng sẽ chứa giá trị 0.

🕮 Ví dụ 5.18

Cho khai báo

```
int arr[4][5] =
{
{12, 0, 9},
{1, 9, 27, 15, 8},
{12, 5},
{0, 0, 23, 63},
};
```

Khai báo trên tương đương với:

```
int arr[4][5] =
{
{12, 0, 9, 0, 0},
{1, 9, 27, 15, 8},
```

{12, 5, 0, 0, 0},

{0, 0, 23, 63, 0},

};

Đôi khi, trong quá trình khai báo mảng hai chiều, có thể bỏ trống khai báo số lượng dòng, nhưng tuyệt đối không thể bỏ trống số lượng cột. Trong quá trình biên dịch, chương trình tự xác định số lượng dòng cần cấp phát:

int arr[][5] =

{

{12, 0, 9, 12, 65},

{1, 9, 27, 15, 8},

{12, 5, 7, 34},

{0, 0, 23, 63, 0 },

};

5.1.2.2. Truy cập đến phần tử của mảng hai chiều

Truy cập đến một phần tử trong mảng hai chiều thông qua chỉ số hàng và chỉ số cột của phần tử đó.

Cú pháp:

array_Name[row_index][col_index];

Trong đó:

array_Name: Tên mảng hai chiều;

row_index, col_index: Chỉ số hàng, chỉ số cột của phần tử cần truy cập.

🕮 Ví dụ 5.19

Cho mảng int arr[4][5] =

{

{12, 0, 9, 5, 7},

{1, 9, 27, 15, 8},

{12, 5, 19, 25, 37},

{0, 0, 23, 63, 41},

};

Khi đó phần tử arr[2][3] là phần tử ở hàng 2, cột thứ 3, có giá trị arr[2][3] = 25.

5.1.2.3. Các dạng bài tập mảng một chiều

Dạng 1: Nhập và hiển thị mảng hai chiều theo định dạng [dòng][cột]

a. Nhập từ bàn phím và hiển thị màn hình theo định dạng

🕮 Ví dụ 5.20

Viết chương trình nhập vào mảng hai chiều kích thước m × n bao gồm các phần tử là các số nguyên, với m là số hàng, n là số cột được nhập từ bàn phím. Hiển thị mảng ra màn hình gồm m hàng và n cột, mỗi phần tử cách nhau độ dài phím tab.

✍ Phác thảo lời giải

Để duyệt mảng hai chiều, sử dụng hai vòng lặp for lồng nhau, vòng lặp thứ nhất duyệt theo từng dòng của mảng, vòng lặp thứ hai tương ứng với mỗi dòng duyệt các phần tử tương ứng với từng cột.

✍ Chương trình:

```
#include<stdio.h>
#include <conio.h>
#define N 100
void Nhap_mang(int a[][N], int m, int n)
{
   for (int i=0; i<m; i++)
      for (int j=0; j<n; j++)
      {
         printf("\nNhap a[%d][%d]: ",i,j);
         scanf("%d", &a[i][j]);
      }
}
 void Hien_thi(int a[][N], int m, int n)
{
   printf("Mang sau khi nhap: \n");
         for(int i=0; i<m; i++)
   {
      for(int j=0; j<n; j++)
      {
         printf("%d\t",a[i][j]);
      }
```

```
        printf("\n");
    }
}
 int main()
{
    int m, n;
    int a[100][100];
    printf("\nNhap vao so hang cua mang hai chieu: ");
    scanf("%d",&m);
    printf("\nNhap vao so cot cua mang hai chieu: ");
    scanf("%d",&n);
    Nhap_mang(a,m,n);
    Hien_thi(a,m,n);
    getch();
    return 0;
}
```

✍ Kết quả chương trình:

```
Nhap vao so hang cua mang hai chieu: 3
Nhap vao so cot cua mang hai chieu: 2
Nhap a[0][0]: 3
Nhap a[0][1]: 43
Nhap a[1][0]: 5
Nhap a[1][1]: 12
Nhap a[2][0]: 4
Nhap a[2][1]: 5
Mang sau khi nhap:
3    43
5    12
4    5
```

b. Sinh ngẫu nhiên và hiển thị mảng hai chiều.

🕮 Ví dụ 5.21

Viết chương trình sinh ngẫu nhiên mảng hai chiều kích thước m × n bao gồm các phần tử là các số nguyên dương có giá trị trong phạm vi [0, 100] với m là số hàng, n là số cột của ma trận. Hiển thị mảng ra màn hình.

✍ Phác thảo lời giải:

- Xây dựng hàm Sinh_mang_nn để sinh mảng: sử dụng hàm rand()%(max - min + 1) để sinh ngẫu nhiên các số tự nhiên trong phạm vi từ min đến max.

- Sử dụng hàm srand(time(0)) để sinh ngẫu nhiên các số tại các thời điểm khác nhau, đồng thời khai báo thư viện *"time.h"*.

- Xây dựng hàm Hien_thi(): để hiển thị mảng hai chiều theo định dạng gồm m dòng và n cột, sử dụng câu lệnh printf("\n") để xuống dòng khi hiển thị hết dữ liệu của 1 hàng.

✍ Chương trình:

```
#include<stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#define N 100
void Sinh_mang_nn(int a[][N], int m, int n)
{
   srand(time(0));
         for (int i=0; i<m; i++)
      for (int j=0; j<n; j++)
      {
         a[i][j]= rand()%100;
      }
}
 void Hien_thi(int a[][N], int m, int n)
{
   printf("Mang sau khi nhap: \n");
         for(int i=0; i<m; i++)
   {
      for(int j=0; j<n; j++)
      {
         printf("%d\t",a[i][j]);
      }
      printf("\n");
   }
```

```
}
 int main()
{
    int m, n;
    int a[100][100];
    printf("\nNhap vao so hang cua mang hai chieu: ");
    scanf("%d",&m);
    printf("\nNhap vao so cot cua mang hai chieu: ");
    scanf("%d",&n);
    Sinh_mang_nn(a,m,n);
    Hien_thi(a,m,n);
    getch();
    return 0;
}
```

✍ Kết quả chương trình:

```
Nhap vao so hang cua mang hai chieu: 3
Nhap vao so cot cua mang hai chieu: 4
Mang sau khi nhap:
12      16      9       18
34      78      90      43
90      52      7       35
```

Dạng 2:

Duyệt các phần tử của mảng: Duyệt qua các phần tử của mảng, sử dụng chỉ số hàng và chỉ số cột, để thực hiện một số thao tác với mảng: tính tổng, tìm max, min, liệt kê các phần tử thỏa mãn một số tính chất nào đó.

🕮 Ví dụ 5.22

Nhập mảng hai chiều có kích thước m × n, trong đó m là số hàng, n là số cột. In phần tử lớn nhất, nhỏ nhất trong mảng.

✍ Phác thảo lời giải

- Xây dựng hàm nhập mảng hai chiều, sử dụng ba tham số là tên biến mảng hai chiều (trong đó chỉ số cột phải được chỉ định trước) và số hàng, số cột của mảng hai chiều.

- Xây dựng hàm hiển thị mảng hai chiều, các tham số tương tự hàm nhập mảng.
- Xây dựng hàm tìm giá trị lớn nhất, các tham số tương tự hàm nhập mảng.
- Xây dựng hàm tìm giá trị nhỏ nhất, các tham số tương tự hàm nhập mảng.

✍ **Chương trình:**

```
1  #include <stdio.h>
2  // khai bao hang MAX la gioi han cua so hang va so cot
3  #define MAX 100
4  // ham nhap vao mang hai chieu kich thuoc m x n
5  void Nhap_Mang(int a[][MAX], int m, int n)
6  {
7    for(int i = 0; i < m; i++)
8      for(int j = 0; j < n; j++)
9      {
10       printf("A[%d][%d] = ", i, j);
11       scanf("%d", &a[i][j]);
12     }
13 }
14 //hàm in mang hai chieu
15 void In_Mang(int a[][MAX], int m, int n)
16 {
17   for(int i = 0; i < m; i++)
18   {
19     for(int j = 0; j < n; j++)
20       printf("%d\t", a[i][j]);
21     printf("\n");
22   }
23 }
24 //ham tim gia tri lon nhat cua mang hai chieu
25 //thuc chat giong mang mot chieu
26 int In_Max(int a[][MAX], int m, int n)
27 {
28   int max = a[0][0];
29   for(int i = 0; i < m; i++)
```

```
30        for(int j = 0; j < n; j++)
31          if(a[i][j]>max)
32            max = a[i][j];
33    return max;
34  }
35  //ham tim gia tri nho nhat cua mang hai chieu
36  int In_Min(int a[][MAX], int m, int n)
37  {
38    int min = a[0][0];
39    for(int i = 0; i < m; i++)
40       for(int j = 0; j < n; j++)
41          if(a[i][j]<min)
42            min = a[i][j];
43    return min;
44  }
45  int main(){
46     int a[MAX][MAX];
47     int m,n;
48     //doan chuong trinh kiem tra so hang nhap dam bao lon hon 0 va
    be hon MAX
49            do
50            {
51                    printf("Nhap so hang cua mang: ");
52                    scanf("%d", &m);
53                    if(m <= 0 || m > MAX)
54                    {
55                            printf("Khong hop le. Nhap so trong pham
    vi tu 0 den MAX !\n");
56                    }
57            }while(m <= 0 || m > MAX);
58
59            //doan chuong trinh kiem tra so cot nhap vao
60                    Do
61            {
62                    printf("Nhap so cot cua mang: ");
```

```
63                  scanf("%d", &n);
64                  if(n <= 0 || n > MAX)
65                  {
66                          printf("Khong hop le. Nhap so trong pham vi tu 0 den MAX !\n");
67                  }
68          }while(n <= 0 || n > MAX);
69          //goi ham nhap mang
70          Nhap_Mang(a, m , n);
71      printf("Mang duoc nhap vao la:\n");
72          In_Mang(a, m, n);
73      printf("\nMax = %d", In_Max(a, m, n));
74      printf("\nMin = %d", In_Min(a, m, n));
75  }
```

Kết quả chương trình:

```
Nhap so hang cua mang: -1
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap so hang cua mang: 102
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap so hang cua mang: 2
Nhap so cot cua mang: -2
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap so cot cua mang: 103
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap so cot cua mang: 3
A[0][0] = 12
A[0][1] = 43
A[0][2] = 5
A[1][0] = 23
A[1][1] = 45
A[1][2] = 3
Mang duoc nhap vao la:
12      43      5
23      45      3
```

Max = 45

Min = 3

🕮 Ví dụ 5.23

Nhập vào ma trận vuông cấp n (n được nhập từ bàn phím).

a. In ma trận.

b. Tính căn bậc hai của tổng bình phương các phần tử của ma trận.

c. Tính tổng các phần tử trên đường chéo chính của ma trận.

✍Phác thảo lời giải

Ma trận vuông: là mảng hai chiều trong đó số hàng bằng số cột.

Phần tử trên đường chéo chính: là phần tử có chỉ số hàng bằng chỉ số cột.

Xây dựng hàm nhập giá trị các phần tử của ma trận vuông cấp n, gồm hai tham số: tên biến mảng hai chiều, chỉ số cột.

Xây dựng hàm hiển thị các phần tử của ma trận vuông cấp n.

Xây dựng hàm tính căn bậc hai của tổng bình phương các phần tử của ma trận.

Xây dựng hàm tính tổng các phần tử trên đường chéo chính của ma trận.

✍Chương trình:

```
1   #include <stdio.h>
2   #include <math.h>
3   // khai bao hang MAX la gioi han cua so hang va so cot
4   #define MAX 100
5   // ham nhap vao mang hai chieu kich thuoc n x n
6   void Nhap_Mang(int a[][MAX], int n)
7   {
8     for(int i = 0; i < n; i++)
9       for(int j = 0; j < n; j++)
10      {
11        printf("A[%d][%d] = ", i, j);
12        scanf("%d", &a[i][j]);
13      }
14  }
15  //hàm in ma tran vuong cap n
16  void In_Mang(int a[][MAX], int n)
17  {
```

```
18    for(int i = 0; i < n; i++)
19    {
20       for(int j = 0; j < n; j++)
21          printf("%d\t", a[i][j]);
22       printf("\n");
23    }
24 }
25 //ham tinh tong cac phan tu tren duong cheo chinh
26 int Tong_CC(int a[][MAX], int n)
27 {
28    int t_1= 0;
29    for(int i = 0; i < n; i++)
30       t_1 +=  a[i][i];
31    return t_1;
32 }
33 //ham tinh can bac hai tong binh phuong các phan tu
34 int TB(int a[][MAX], int n)
35 {
36     int bp ;
37     int t_2 = 0;
38     double normal;
39    for(int i = 0; i < n; i++)
40     {
41        for(int j = 0; j < n; j++)
42        {
43           bp = a[i][j]*a[i][j];
44           t_2 += bp;
45        }
46     }
47     normal = sqrt((double)t_2);
48    return normal;
49 }
50 int main(){
51     int a[MAX][MAX];
52     int n;
53     //doan chuong trinh kiem tra so hang nhap dam bao lon hon 0 va
```

```
be hon MAX
54          do
55          {
56                  printf("Nhap cap cua ma tran: ");
57                  scanf("%d", &n);
58                  if(n <= 0 || n > MAX)
59                  {
60                  printf("Khong hop le. Nhap so trong pham vi tu 0 den MAX !\n");
61                  }
62          }while(n <= 0 || n > MAX);
63          //goi ham nhap mang
64          Nhap_Mang(a, n );
65    printf("Mang duoc nhap vao la:\n");
66          In_Mang(a, n);
67    printf("\nTong cac phan tu tren duong cheo chinh = %d", Tong_CC(a, n));
68    printf("\nCan bac hai tong binh phuong cac phan tu cua ma tran = %d", TB(a, n));
69    return 0;
70  }
```

Kết quả chương trình:

```
Nhap cap cua ma tran: 102
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap cap cua ma tran: -1
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap cap cua ma tran: 3
A[0][0] = 12
A[0][1] = 3
A[0][2] = 43
A[1][0] = 15
A[1][1] = 4
A[1][2] = 6
A[2][0] = 33
A[2][1] = 23
```

A[2][2] = 12

Mang duoc nhap vao la:

12 3 43

15 4 6

33 23 12

Tong cac phan tu tren duong cheo chinh = 28

Can bac hai tong binh phuong cac phan tu cua ma tran = 63

🕮 Ví dụ 5.24

Viết chương trình nhập vào hai ma trận cùng cấp n × m (m, n nhập từ bàn phím). Thực hiện các yêu cầu sau đây:

a. Cộng hai ma trận và hiển thị ma trận tổng ra màn hình.

b. Trừ hai ma trận và hiển thị ma trận hiện ra màn hình.

✍Phác thảo lời giải:

Sử dụng công thức cộng, trừ hai ma trận cùng cấp nhau sau:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ b_{31} & b_{32} & \dots & b_{3n} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ a_{31} + b_{31} & a_{32} + b_{32} & \dots & a_{3n} + b_{3n} \\ \dots & \dots & \dots & \dots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_{mn} + b_{mn} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} - \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ b_{31} & b_{32} & \dots & b_{3n} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} - b_{11} & a_{12} - b_{12} & \dots & a_{1n} - b_{1n} \\ a_{21} - b_{21} & a_{22} - b_{22} & \dots & a_{2n} - b_{2n} \\ a_{31} - b_{31} & a_{32} - b_{32} & \dots & a_{3n} - b_{3n} \\ \dots & \dots & \dots & \dots \\ a_{m1} - b_{m1} & a_{m2} - b_{m2} & \dots & a_{mn} - b_{mn} \end{bmatrix}$$

Chương trình:

```
1   #include<stdio.h>
2   #define MAX 10
3   int main()
4   {
5   int n, m, i, j;
6    int a[MAX][MAX], b[MAX][MAX], Tong[MAX][MAX], Hieu[MAX][MAX];
7           do
8           {
9                   printf("Nhap so hang cua mang: ");
10                  scanf("%d", &m);
11                  if(m <= 0 || m > MAX)
12                  {
13                          printf("Khong hop le. Nhap so trong pham vi tu 0 den MAX !\n");
14                  }
15          }while(m <= 0 || m > MAX);
16          //doan chuong trinh kiem tra so cot nhap vao
17                  do
18          {
19                  printf("Nhap so cot cua mang: ");
20                  scanf("%d", &n);
21                  if(n <= 0 || n > MAX)
22                  {
23                  printf("Khong hop le. Nhap so trong pham vi tu 0 den MAX !\n");
24                  }
25          }while(n <= 0 || n > MAX);
26     printf("Nhap  %d phan tu cua ma tran thu nhat: \n", m*n);
27     for(i = 0; i < m; i++)
28        for(j = 0; j < n; j++)
29           scanf("%d", &a[i][j]);
30     printf("Nhap %d phan tu cua ma tran thu hai \n", m*n);
31     for(i = 0; i < m; i++)
32        for(j = 0; j < n; j++)
```

```
33              scanf("%d", &b[i][j]);
34      /*
35          In ma tran thu nhat
36      */
37      printf("Ma tran thu nhat la: \n");
38      for(i = 0; i < m; i++)
39      {
40          for(j = 0; j < n; j++)
41          {
42              printf("%d\t", a[i][j]);
43          }
44      printf("\n");
45      }
46      /*
47          In ma tran thu hai
48      */
49      printf("Ma tran thu hai la: \n");
50      for(i = 0; i < m; i++)
51      {
52          for(j = 0; j < n; j++)
53          {
54              printf("%d\t", b[i][j]);
55          }
56      printf("\n");
57      }
58      /*
59        Tinh tong cua hai ma tran, thu duoc ma tran dong cap voi 2
    ma tran ban dau a, b
60      */
61      for(i = 0; i < m; i++)
62          for(j = 0; j < n; j++)
63              Tong[i][j] = a[i][j] + b[i][j];
64      // In ma tran tong
65      printf("Tong cua hai ma tran la: \n");
66      for(i = 0; i < m; i++)
```

```
    {
        for(j = 0; j < n; j++)
        {
            printf("%d\t", Tong[i][j]);
        }
        printf("\n");
    }
    /*
        Tinh hieu cua hai ma tran
    */
    for(i = 0; i < m; i++)
        for(j = 0; j < n; j++)
            Hieu[i][j] = a[i][j] - b[i][j];
    // In hieu cua hai ma tran
    printf("Hieu cua hai ma tran la: \n");
    for(i = 0; i < m; i++)
    {
        for(j = 0; j < n; j++)
        {
            printf("%d\t", Hieu[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Kết quả chương trình:

```
Nhap so hang cua mang: 11
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap so hang cua mang: -1
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap so hang cua mang: 2
Nhap so cot cua mang: 12
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap so cot cua mang: -2
```

Khong hop le. Nhap so trong pham vi tu 0 den MAX !

Nhap so cot cua mang: 3

Nhap 6 phan tu cua ma tran thu nhat:

4 5 12 6 7 42

Nhap 6 phan tu cua ma tran thu hai

3 5 65 2 54 21

Ma tran thu nhat la:

4 5 12

6 7 42

Ma tran thu hai la:

3 5 65

2 54 21

Tong cua hai ma tran la:

7 10 77

8 61 63

Hieu cua hai ma tran la:

1 0 -53

4 -47 21

✍**Yêu cầu với sinh viên:**

- Xây dựng hàm nhập ma trận m × n.
- Xây dựng hàm hiển thị ma trận m × n.
- Xây dựng hàm tính tổng hai ma trận cấp m × n.
- Xây dựng hàm tính hiệu hai ma trận cấp m × n.

5.2. CON TRỎ

Con trỏ trong ngôn ngữ C là một biến chứa địa chỉ của một biến khác có cùng kiểu dữ liệu. Con trỏ được sử dụng để truy cập bộ nhớ và thao tác địa chỉ. Con trỏ là một trong những tính năng khác biệt và thú vị nhất của ngôn ngữ C. Nó cung cấp sức mạnh và tính linh hoạt cho ngôn ngữ. Mặc dù ban đầu con trỏ có thể hơi khó hiểu và phức tạp, một khi chúng ta hiểu khái niệm này, chúng ta sẽ có thể làm được nhiều hơn thế với ngôn ngữ C.

Trước khi chúng ta bắt đầu hiểu con trỏ là gì và chúng có thể làm gì, hãy bắt đầu bằng cách hiểu "Địa chỉ của bộ nhớ vị trí" nghĩa là gì?

5.2.1. Địa chỉ trong C

Khi một biến được khai báo trong C, một địa chỉ nhớ được gán cho nó, trong đó giá trị của biến được lưu trữ. Chúng ta dễ dàng kiểm tra địa chỉ bộ nhớ này bằng ký hiệu &. Nếu ta có một biến k trong chương trình của mình, &k sẽ cung cấp địa chỉ của k trong bộ nhớ, với toán tử & thường được gọi là toán tử tham chiếu.

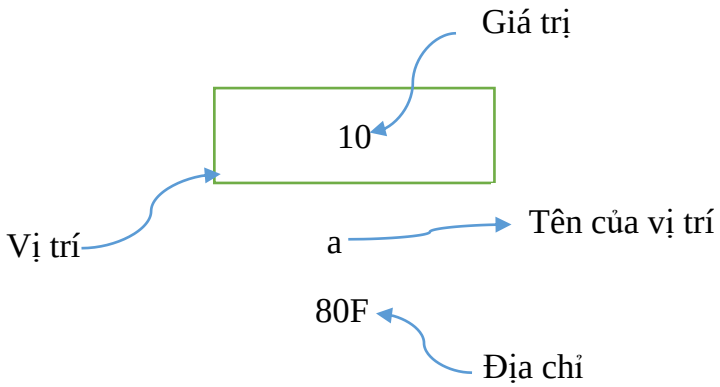
Chúng ta thường hay thấy toán tử & xuất hiện khi sử dụng hàm scanf() để nhập giá trị cho một biến. Toán tử & được sử dụng trong hàm scanf() để lưu trữ giá trị nhập vào của người dùng trong địa chỉ của biến k.

scanf("%d", &k);

Khi một biến được khai báo trong chương trình, hệ thống sẽ phân bổ một vị trí tức là một địa chỉ cho biến trong bộ nhớ, để giữ giá trị được gán. Vị trí này có một địa chỉ riêng.

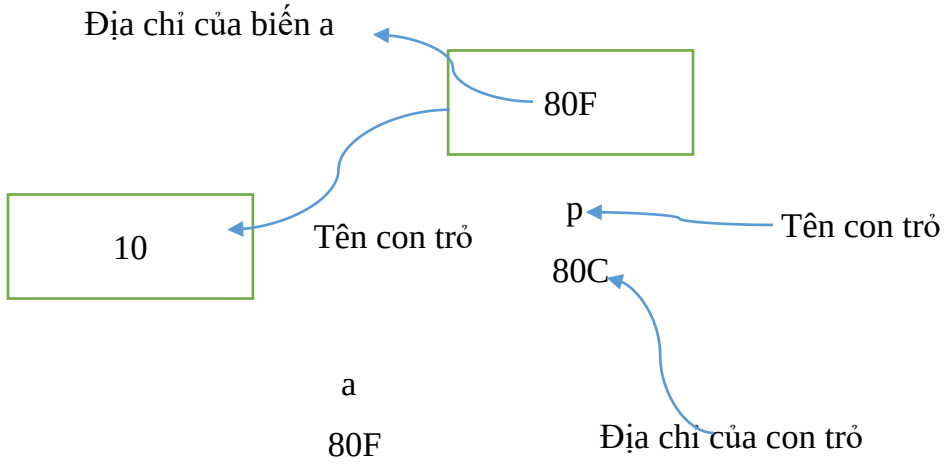
Chẳng hạn khai báo

int a =10;



Hình 5.1. Minh họa vị trí, địa chỉ và giá trị của một biến

Giả sử hệ thống đã phân bổ địa chỉ nhớ 80F cho biến a. Chúng ta có thể truy cập giá trị 10 bằng cách sử dụng tên biến a hoặc bằng cách sử dụng địa chỉ 80F. Vấn đề đặt ra làm thế nào chúng ta có thể truy cập một biến bằng địa chỉ của nó? Vì các địa chỉ bộ nhớ cũng chỉ là các số, nên chúng cũng có thể gán cho một biến khác. Các biến được sử dụng để giữ địa chỉ bộ nhớ được gọi là biến con trỏ. Do đó, một biến con trỏ không có gì khác ngoài nghĩa là một biến chứa địa chỉ của một biến khác. Và giá trị của một biến con trỏ được lưu trữ ở một địa chỉ nhớ khác.



Hình 5.2. Con trỏ, địa chỉ và giá trị

🕮 Ví dụ 5.25

✍ Cho đoạn chương trình sau đây:

```
1   #include <stdio.h>
2   int main()
3   {
4     int k = 5;
5     printf("Gia tri: %d\n", k);
6     printf("Dia chi: %u", &k);  //ky hieu & truoc bien k
7     return 0;
8   }
```

✍ **Kết quả chương trình:**

```
Gia tri: 5
Dia chi: 6487580
```

✍**Giải thích:** Trong đoạn chương trình trên, giá trị 5 được lưu ở địa chỉ 6487580, k chỉ là tên được đặt cho vị trí đó.

Lưu ý: Lợi ích của việc sử dụng con trỏ:

+ Con trỏ hiệu quả hơn trong việc xử lý mảng và cấu trúc;

+ Con trỏ cho phép tham chiếu đến 1 hàm và do đó cho phép truyền hàm dưới dạng đối số cho các hàm khác.

+ Giảm thời lượng của chương trình và thời gian thực hiện.

+ Hỗ trợ quản lý bộ nhớ động.

5.2.2. Khai báo con trỏ

Trong C, ta có thể khai báo một biến đặc biệt để lưu trữ địa chỉ (thay vì giá trị). Biến này được gọi là biến con trỏ, hay đơn giản gọi là con trỏ.

Cú pháp khai báo con trỏ:

data_type* pointer_variable_name;

Trong đó:

data_type: là kiểu dữ liệu của con trỏ, nó có thể là một trong các kiểu dữ liệu cơ bản như số nguyên, số thực, ký tự, v.v. hoặc cũng có thể là một kiểu dữ liệu có cấu trúc.

pointer_variable_name: tên biến con trỏ, được đặt giống định danh.

🕮 Ví dụ 5.26

int *k; // con trỏ k trỏ tới một số nguyên.

double *t; //con trỏ t trỏ tới một số thực kiểu double.

float *x; // con trỏ x trỏ tới một số thực kiểu float.

char *y; // con trỏ y trỏ tới một ký tự.

Các phép toán thường sử dụng với con trỏ.

Toán tử &: lấy địa chỉ của biến và gán địa chỉ đó cho biến con trỏ.

Toán tử *: trả về giá trị trong vùng nhớ mà con trỏ đang trỏ tới.

Lưu ý: Phân biệt dấu * được dùng khi khai báo một biến con trỏ với toán tử * để lấy giá trị từ địa chỉ.

Khởi tạo con trỏ.

Khởi tạo con trỏ là quá trình gán địa chỉ của một biến cho một biến con trỏ. Biến con trỏ chỉ có thể chứa địa chỉ của một biến có cùng kiểu dữ liệu. Trong ngôn ngữ lập trình C, toán tử địa chỉ & được sử dụng để xác định địa chỉ của một biến.

Trong trường hợp không chắc chắn về việc gán địa chỉ của biến nào cho biến con trỏ trong khai báo, chúng ta nên gán giá trị NULL cho biến con trỏ. Một con trỏ được gán giá trị NULL được gọi là con trỏ NULL.

🕮 Ví dụ 5.27

✍Xem xét đoạn chương trình sau đây để hiểu con trỏ hoạt động như thế nào?

```
1   #include <stdio.h>
```

```
2   int main()
3   {
4     int* pc, c;
5     c = 22;
6     printf("Dia chi cua bien c: %u\n", &c);
7     printf("Gia tri cua bien c: %d\n\n", c);
8     pc = &c;
9     printf("Dia chi cua con tro pc: %u\n", pc);
10    printf("Noi dung cua con tro pc: %d\n\n", *pc);
11    c = 11;
12    printf("Dia chi cua con tro pc: %u\n", pc);
13    printf("Noi dung cua con tro pc: %d\n\n", *pc);
14    *pc = 2;
15    printf("Dia chi cua bien  c: %u\n", &c);
16    printf("Gia tri cua bien c: %d\n\n", c);
17    return 0;
18  }
```

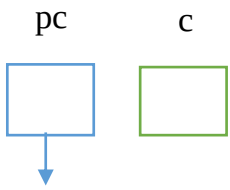
✍ **Kết quả chương trình:**

```
Dia chi cua bien c: 6487572
Gia tri cua bien c: 22
Dia chi cua con tro pc: 6487572
Noi dung cua con tro pc: 22
Dia chi cua con tro pc: 6487572
Noi dung cua con tro pc: 11
Dia chi cua bien  c: 6487572
Gia tri cua bien c: 2
```

✍Giải thích về đoạn chương trình trên:

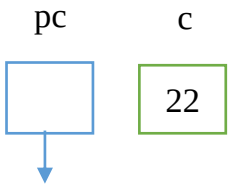
Dòng lệnh 4: int* pc, c;

Đầu tiên biến con trỏ pc và một biến bình thường c, cùng có kiểu số nguyên, được tạo ra. Do lúc đầu pc và c không được khởi tạo, nên con trỏ pc không lưu địa chỉ hoặc lưu trữ một địa chỉ ngẫu nhiên và biến c có địa chỉ nhưng chứa một giá trị ngẫu nhiên bất kỳ.



Dòng lệnh 5: c = 22;

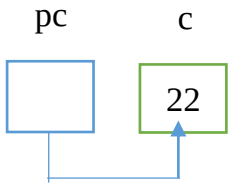
Gán giá trị 22 cho biến c, tức là 22 được lưu trữ tại vị trí bộ nhớ của biến c.



Dòng lệnh 6: printf("Dia chi cua bien c: %u\n", &c);

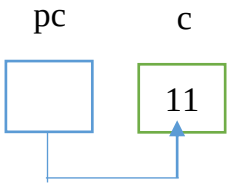
Trong câu lệnh này khi in &c (địa chỉ của biến c), chúng ta dùng %u thay vì %d vì địa chỉ thường được biểu diễn dưới dạng số nguyên không dấu (luôn dương).

Dòng lệnh 8: pc = &c;



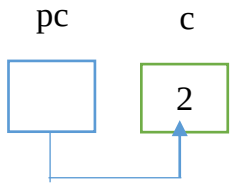
Câu lệnh này gán địa chỉ của biến c cho con trỏ pc. Như vậy, ở dòng lệnh 9, 10, giá trị của pc giống địa chỉ của biến c và nội dung của con trỏ pc là 22 (bằng giá trị của biến c).

Dòng lệnh 11: c = 11;



Gán giá trị 11 cho biến c. Vì con trỏ pc trỏ tới cùng địa chỉ của c, nên giá trị (nội dung) được trỏ bởi con trỏ pc là 11 (Dòng lệnh 12, 13).

Dòng lệnh 14: *pc = 2;



Câu lệnh trên thay đổi giá trị tại vị trí bộ nhớ được trỏ bởi con trỏ pc thành 2. Vì địa chỉ của con trỏ pc giống với địa chỉ của c, nên giá trị của c cũng được thay đổi thành 2 (Dòng lệnh 15, 16).

🕮 Ví dụ 5.28

✍Xem xét đoạn chương trình sau đây:

```
1  #include <stdio.h>
2  main() {
3  /* khai bao bien va con tro */
4  // bien q, r
5  // con tro p
6  double q , r , *p = &q ;
7  /* gan gia tri cho cac bien */
8  q = 1.34;
9  r = 6.78;
10 /* in bien, con tro va noi dung cua con tro */
11 printf ("Gia tri cua q = %f\n" , q);
12 printf ("Dia chi cua q = %p\n" , &q);
13 printf ("Gia tri cua r = %f\n" , r);
14 printf ("Dia chi cua r = %p\n" , &r);
15 printf ("Gia tri cua p = %p\n" , p);
16 printf ("Dia chi cu p = %p\n" , &p);
17 printf ("Gia tri cua *p = %f\n" , *p);
18 /* thay doi gia tri cua bien q */
19 q = -7.89;
20 /* in bien, con tro va noi dung cua con tro sau thay doi */
```

```
21 printf ("\nGia tri cua q = %f\n" , q);
22 printf ("Dia chi cua q = %p\n" , &q);
23 printf ("Gia tri cua r = %f\n" , r);
24 printf ("Dia chi cua r = %p\n" , &r);
25 printf ("Gia tri cua p = %p\n" , p);
26 printf ("Dia chi cua p = %p\n" , &p);
27 printf ("Gia tri cua *p = %f\n" , *p);
28 /* thay doi gia tri cua bien con tro p dang tro toi */
29 *p = 12.86;
30 /* in bien, con tro va noi dung cua con tro */
31 printf ("\nGia tri cua q = %f\n" , q);
32 printf ("Dia chi cua q = %p\n" , &q);
33 printf ("Gia tri cua r = %f\n" , r);
34 printf ("Dia chi cua r = %p\n" , &r);
35 printf ("Gia tri cua p = %p\n" , p);
36 printf ("Dia chi cua p = %p\n" , &p);
37 printf ("Gia tri cua *p = %f\n" , *p);
38 /* thay doi gia tri cua bien va con tro */
39 p = &r;
40 r = 5.69;
41 /* in bien, con tro va noi dung cua con tror */
42 printf ("\nGia tri cua q = %f\n" , q);
43 printf ("Dia chi cua q = %p\n" , &q);
44 printf ("Gia tri cua r = %f\n" , r);
45 printf("Dia chi cua r = %p\n" , &r);
46 printf ("Gia tri cua p = %p\n" , p);
47 printf ("Dia chi cua p = %p\n" , &p);
48 printf ("Gia tri cua *p = %f\n" , *p);
49 }
```

Kết quả chương trình:

```
Gia tri cua q = 1.340000
Dia chi cua q = 000000000062FDF8
Gia tri cua r = 6.780000
```

Dia chi cua r = 000000000062FE00
Gia tri cua p = 000000000062FDF8
Dia chi cua p = 000000000062FE08
Gia tri cua *p = 1.340000

Gia tri cua q = -7.890000
Dia chi cua q = 000000000062FDF8
Gia tri cua r = 6.780000
Dia chi cua r = 000000000062FE00
Gia tri cua p = 000000000062FDF8
Dia chi cua p = 000000000062FE08
Gia tri cua *p = -7.890000

Gia tri cua q = 12.860000
Dia chi cua q = 000000000062FDF8
Gia tri cua r = 6.780000
Dia chi cua r = 000000000062FE00
Gia tri cua p = 000000000062FDF8
Dia chi cua p = 000000000062FE08
Gia tri cua *p = 12.860000

Gia tri cua q = 12.860000
Dia chi cua q = 000000000062FDF8
Gia tri cua r = 5.690000
Dia chi cua r = 000000000062FE00
Gia tri cua p = 000000000062FE00
Dia chi cua p = 000000000062FE08
Gia tri cua *p = 5.690000

✍**Yêu cầu đối với sinh viên:**

- Giải thích Kết quả chương trình dựa vào các dòng lệnh.
- Giải thích tại sao kết quả của chương trình trên các máy khác nhau là khác nhau?

🕮 Ví dụ 5.29

Cho khai báo sau đây:

int c, *pc;

Câu lệnh gán pc = c; sai vì pc là địa chỉ trong khi c không phải địa chỉ;

Câu lệnh gán *pc = &c; sai vì *pc là nội dung (giá trị) của biến được trỏ bởi con trỏ pc, còn &c là một địa chỉ.

Câu lệnh gán pc = &c; đúng vì pc là địa chỉ và &c cũng là địa chỉ.

Câu lệnh gán *pc = c; đúng vì *pc là giá trị (nội dung) của địa chỉ mà con trỏ pc đang lưu giữ và c cũng là một giá trị (không phải địa chỉ).

5.2.3. Mối quan hệ giữa con trỏ và mảng

Trước hết chúng ta xem xét ví dụ sau đây:

🕮 Ví dụ 5.30

In địa chỉ của tất cả các phần tử trong mảng:

```
1   #include <stdio.h>
2   int main()
3   {
4     int x[4];
5     int i;
6     for(i = 0; i < 4; ++i)
7     {
8       printf("&x[%d] = %u\n", i, &x[i]);
9     }
10    printf("Dia chi cua mang  x: %u", x);
11    return 0;
12  }
```

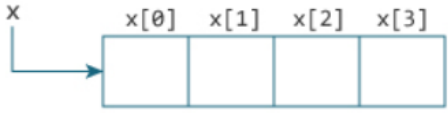
✍Kết quả chương trình

```
&x[0] = 6487552
&x[1] = 6487556
&x[2] = 6487560
&x[3] = 6487564
Dia chi cua mang  x: 6487552
```

✍Nhận xét:

Có sự khác biệt 4 bytes giữa 2 phần tử liên tiếp trong mảng x, bởi kích thước của int là 4 byte. Nhưng x và &x[0] cho cùng một kết quả.

Mối quan hệ giữa mảng và con trỏ được thể hiện qua hình ảnh sau đây:



Tương ứng với khai báo: int x[4];

Ở ví dụ 5.34, ta đã nhận xét rằng:

x và &x[0] cùng chứa 1 địa chỉ, hay &x[0] tương đương x;

Do đó, x[0] tương đương *x.

Tương tự:

&x[1] tương đương x + 1, x[1] tương đương &(x + 1);

&x[2] tương đương x + 2, x[2] tương đương &(x + 2);

…

&x[i] tương đương x + i, x[i] tương đương &(x + i).

🕮 Ví dụ 5.31

Viết chương trình sử dụng con trỏ để nhập vào n phần tử của một mảng. Tính tổng và in tổng ra màn hình.

✍Chương trình

```
1  #include <stdio.h>
2  #define MAX 50
3  int main()
4  {
5    int i,n, a[MAX], sum = 0;
6    printf("Nhap vao so phan tu cua mang:");
7    scanf("%d",&n);
8    printf("Su dung con tro, nhap vao gia tri cac phan tu:\n");
9    for(i = 0; i < n; ++i)
10   {
11      scanf("%d", a+i);
```

```
12      sum += *(a+i);
13   }
14   printf("Tong = %d", sum);
15   return 0;
16 }
```

✍Kết quả chương trình:

```
Nhap vao so phan tu cua mang: 7
Su dung con tro, nhap vao gia tri cac phan tu:
6
7
5
12
62
6
3
Tong = 101
```

Trong ví dụ 5.31, tên mảng được chuyển đổi thành con trỏ. Đó là lý do tại sao, chúng ta có thể sử dụng con trỏ có cùng tên với mảng để thao tác trên các phần tử của mảng. Tuy nhiên, con trỏ và mảng không giống nhau. Trong một vài trường hợp không chuyển đổi tên mảng thành con trỏ.

🕮Ví dụ 5.32

✍Cho đoạn chương trình sau đây:

```
1   #include <stdio.h>
2   int main()
3   {
4     int a[7] = {12, 5, 7, 4, 22, 18, 33};
5     int* p;
6     p = &a[2];
7     printf("*p = %d \n", *p);
8     printf("*p + 1 = %d \n", *p+1);
9     printf("*p - 1 = %d", *p-1);
10    return 0;
11  }
```

Kết quả chương trình:

```
*p      = 7
*p + 1 = 8
*p - 1 = 6
```

Trong ví dụ 5.32, &a[2] (địa chỉ của phần tử thứ 3 trong mảng a) được gán cho con trỏ p. Do đó giá trị a[2] = 7 được hiển thị khi thực hiện dòng lệnh 7:

printf("*p = %d \n", *p);

Dòng lệnh 8, 9 sẽ in ra các giá trị *p + 1 và *p - 1, do đó các giá trị được hiển thị trên màn hình là 8 và 6.

Trong trường hợp của ví dụ này, con trỏ p chỉ lưu địa chỉ phần tử thứ 3 của mảng a, không trỏ tới mảng a.

5.2.4. Con trỏ và hàm

Ngoài việc truyền các tham số trong hàm, chúng ta còn có thể truyền địa chỉ cho hàm, bởi vì địa chỉ cũng là một loại giá trị.

Con trỏ làm tham số hàm được sử dụng để giữ địa chỉ của các đối số được truyền trong lúc gọi hàm. Kỹ thuật này được gọi là tham chiếu, khi một hàm được gọi bằng tham chiếu, bất kỳ thay đổi nào được thực hiện cho các biến tham chiếu sẽ ảnh hưởng đến các biến ban đầu.

Xem xét ví dụ sau đây:

🕮 Ví dụ 5.33

Viết chương trình hoán đổi giá trị của hai biến.

```
1  #include <stdio.h>
2  void swap(int *n1, int *n2);
3  int main()
4  {
5     int a = 5, b = 10;
6     // Trao doi dia chi cua 2 bien a va b
7     swap( &a, &b);
8     printf("a = %d\n", a);
9     printf("b = %d", b);
10    return 0;
11 }
```

```
// Con tro  n1 va n2 chua dia chi cua a va b tuong ung
void swap(int* n1, int* n2)
{
    int temp;
    temp = *n1;
    *n1 = *n2;
    *n2 = temp;
}
```

✍ Kết quả chương trình:

```
a = 10
b = 5
```

✍ Giải thích chương trình:

Địa chỉ của hai biến a, b được truyền cho hàm swap() thông qua lời gọi trong hàm chính swap(&a, &b); con trỏ n1 và n2 chứa địa chỉ của a và b.

Khi giá trị tại địa chỉ n1 và n2 bị thay đổi, a, b cũng được thay đổi tương ứng. Nghĩa là nếu *n1 và *n2 được thay đổi bên trong hàm swap() thì a, b cũng được thay đổi bên trong hàm main().

Bên trong hàm swap(), *n1 và *n2 được tráo đổi giá trị. Do đó, giá trị trong a và b cũng được hoán đổi. Lưu ý, hàm swap() không trả về giá trị, kiểu trả về của nó là void.

Hàm trả về biến con trỏ: một hàm cũng có thể trả về một con trỏ tới hàm gọi. Trong trường hợp này, chúng ta cần thận trọng, bởi vì các biến cục bộ của hàm không nằm ngoài hàm, có phạm vi bên trong hàm. Do đó, nếu ta trả về một con trỏ được kết nối với một biến cục bộ, con trỏ đó sẽ không trỏ đến bất cứ cái gì khi hàm kết thúc.

🕮 Ví dụ 5.34

Viết chương trình tìm số lớn nhất trong hai số nguyên a, b được nhập từ bàn phím.

Chương trình

```
#include <stdio.h>
int* find_max(int*, int*);
int main()
{
```

```
5       int a,b;
6       int *p;
7       printf("Nhap vao so thu nhat: ");
8       scanf("%d",&a);
9       printf("Nhap vao so thu hai: ");
10      scanf("%d",&b);
11      p = find_max(&a, &b);
12      printf("%d la so lon hon",*p);
13  }
14  int* find_max(int *x, int *y)
15  {
16      if(*x > *y)
17          return x;
18      else
19          return y;
20  }
```

✍ **Kết quả khi biên dịch chương trình**

```
Nhap vao so thu nhat: 12
Nhap vao so thu hai: 56
56 la so lon hon
```

✍ **Giải thích chương trình:** Lời gọi trong chương trình p = find_max(&a, &b); tham chiếu đến hàm int* find_max(int *x, int *y), với x, y là hai biến con trỏ, kết quả trả về là một con trỏ p trỏ vào số lớn hơn. Do đó khi gọi lệnh printf("%d la so lon hon",*p); sẽ trả về nội dung của con trỏ p, tức là giá trị lớn nhất trong hai số a và b.

5.2.5. Quản lý bộ nhớ động

Nội dung của Mục 5.2.4, chúng ta sẽ tìm hiểu cách phân bổ bộ nhớ động và thu hồi bộ nhớ với các hàm malloc(), calloc(), free().

Mảng là một tập hợp các phần tử có cùng kiểu dữ liệu. Trước khi sử dụng, cần khai báo kích thước của mảng. Đôi khi kích thước mảng khai báo không đủ khi biên dịch chương trình, hoặc nhiều khi khai báo kích thước

mảng quá lớn, nhưng khi biên dịch chỉ sử dụng một phần nhỏ trong đó, điều đó gây lãng phí bộ nhớ. Để giải quyết vấn đề này, chúng ta có thể phân bổ bộ nhớ một cách linh động trong thời gian chạy chương trình. Có ba hàm thư viện được định nghĩa trong thư viện "stdlib.h" bao gồm malloc(), calloc(), free(), giúp chương trình cấp phát và thu hồi bộ nhớ động.

5.2.5.1. Hàm malloc()

Từ *"malloc"* viết tắt của cụm từ "*memory allocation" ("phân bổ bộ nhớ")*. Hàm malloc() dự trữ một khối bộ nhớ của số byte được chỉ định. Hàm malloc() trả về một con trỏ kiểu void có thể chuyển hóa thành con trỏ ở bất kỳ dạng nào.

Cú pháp của hàm malloc()

```
ptr = (cast-type*) malloc(byte-size)
```

🕮 Ví dụ 5.35

```
p = (int*)malloc(100* sizeof(int));
```

Kích thước của int là 4 byte, do đó câu lệnh trên phân bổ 400 byte bộ nhớ. Con trỏ p giữ địa chỉ của byte đầu tiên trong bộ nhớ được phân bổ. Tuy nhiên, nếu không đủ dung lượng, quá trình phân bổ thất bại và trả về một con trỏ NULL.

5.2.5.2. Hàm calloc()

Từ *"calloc"* viết tắt của cụm từ *"contiguous allocation"* (*"phân bổ liền kề"*). Hàm calloc() phân bổ nhiều khối bộ nhớ và khởi tạo chúng về 0.

Cú pháp của hàm calloc()

```
ptr = (cast-type*)calloc(n, element-size);
```

🕮 Ví dụ 5.36

```
p = (float*) calloc(25, sizeof(float));
```

Câu lệnh này phân bổ không gian liền kề trong bộ nhớ cho 25 phần tử, mỗi phần tử có có kích thước kiểu float.

5.2.5.3. Phân biệt giữa malloc() và calloc()

Giống nhau:

Cả hai hàm malloc() và calloc() đều được sử dụng để cấp phát bộ nhớ động cho chương trình. Nếu cấp phát thành công, hàm trả về con trỏ trỏ tới vùng nhớ được cấp phát, trả về NULL nếu không đủ bộ nhớ.

Hàm malloc() và calloc() trả về NULL trong các trường hợp sau đây:

- Kích thước vùng nhớ cần cấp phát vượt quá kích thước vật lý của hệ thống.
- Tại thời điểm gọi hàm malloc() và calloc(), tạm thời không đủ bộ nhớ để cấp phát.

Khác nhau:

Hàm malloc() nhận tham số truyền vào là số byte của vùng nhớ cần cấp phát. Hàm malloc() trả về con trỏ trỏ tới vùng nhớ nếu cấp phát thành công, trả về NULL nếu cấp phát thất bại.

Hàm calloc() nhận hai tham số truyền vào là số block và kích thước của mỗi block. Hàm calloc() trả về con trỏ trỏ tới vùng nhớ được cấp phát và vùng nhớ được khởi tạo giá trị bằng 0, trả về NULL nếu cấp phát thất bại.

Hàm malloc() nhanh hơn hàm calloc(). Hàm calloc() tốn thời gian khởi tạo vùng nhớ.

5.2.5.4. Hàm free()

Bộ nhớ được phân bổ động bằng hàm malloc() và calloc() không tự giải phóng bộ nhớ. Muốn giải phóng được bổ nhớ phải dùng hàm free().

Cú pháp hàm free()

```
free(ptr);
```

Câu lệnh này giải phóng không gian được phân bổ trong bộ nhớ được trỏ bởi ptr.

🕮 Ví dụ 5.37

Sử dụng hàm malloc() để cấp phát bộ nhớ và hàm free() để thu hồi bộ nhớ.

Viết chương trình tính tổng của n số được nhập từ bàn phím.

Để thực hiện chương trình này, bộ nhớ được cấp phát động sử dụng hàm malloc() và được giải phóng bộ nhớ bằng hàm free().

✍ Chương trình

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main()
4  {
5      int n, i, *p, tong = 0;
6      printf("Nhap vao so phan tu cua mang: ");
7      scanf("%d", &n);
```

```
    // cap phat bo nho dong bang ham malloc
        p = (int*) malloc(n * sizeof(int));
    // neu khong du bo nho
        if(p == NULL)
    {
       printf("Loi! bo nho khong duoc phan bo.");
       exit(0);
    }
    // nhap mang thong qua con tro
        printf("Nhap cac phan tu cua mang:\n");
    for(i = 0; i < n; ++i)
    {
        printf("Nhap gia tri phan tu thu %d: ", i+1);
       scanf("%d", p + i);
       tong += *(p + i);
    }
    printf("Tong cac phan tu = %d", tong);
    free(p);
    return 0;
}
```

✍ Kết quả chương trình:

```
Nhap vao so phan tu cua mang: 5
Nhap cac phan tu cua mang:
Nhap gia tri phan tu thu 1: 12
Nhap gia tri phan tu thu 2: 4
Nhap gia tri phan tu thu 3: 65
Nhap gia tri phan tu thu 4: 4
Nhap gia tri phan tu thu 5: 23
Tong cac phan tu = 108
```

🕮 Ví dụ 5.38

Làm lại ví dụ 5.37 với hàm calloc() và hàm free()

✍ Chương trình

```
#include <stdio.h>
#include <stdlib.h>
```

```
3  int main()
4  {
5     int n, i, *p, tong = 0;
6     printf("Nhap vao so phan tu cua mang: ");
7     scanf("%d", &n);
8     // cap phat bo nho dong bang ham calloc()
9     p = (int*) calloc(n, sizeof(int));
10    // neu khong du bo nho
11          if(p == NULL)
12    {
13       printf("Loi! bo nho khong duoc phan bo.");
14       exit(0);
15    }
16    // nhap mang thong qua con tro
17          printf("Nhap cac phan tu cua mang:\n");
18    for(i = 0; i < n; ++i)
19    {
20          printf("Nhap gia tri phan tu thu %d: ", i+1);
21       scanf("%d", p + i);
22       tong += *(p + i);
23    }
24    printf("Tong cac phan tu = %d", tong);
25    free(p);
26    return 0;
27 }
```

5.2.6. Một số bài tập sử dụng con trỏ

🕮 Ví dụ 5.39

Viết chương trình hoán đổi giá trị của ba số a, b, c bằng cách sử dụng hàm tham chiếu.

Nhập từ bàn phím ba giá trị, lưu trữ trong ba biến a, b, c.

Ba biến này được truyền vào hàm cyclicswap(), thay vì truyền các biến thực tế, địa chỉ của các biến được truyền vào.

Khi các biến này được hoán đổi theo thứ tự tuần hoàn trong hàm cyclicswap(), các biến a, b, c trong hàm main() cũng tự động hoán đổi.

✍ Chương trình

```
1  #include<stdio.h>
2  void cyclicSwap(int *x,int *y,int *z);
3  int main()
4  {
5     int a, b, c;
6     printf("Nhap vao ba so a, b, c tuong ung: ");
7     scanf("%d %d %d",&a,&b,&c);
8     printf("Gia tri cac bien a, b, c truoc khi hoan doi:\n");
9     printf("a = %d \nb = %d \nc = %d\n",a,b,c);
10    cyclicSwap(&a, &b, &c);
11    printf("Gia tri cua cac bien a, b, c sau khi hoan doi:\n");
12    printf("a = %d \nb = %d \nc = %d",a, b, c);
13    return 0;
14 }
15 void cyclicSwap(int *x,int *y,int *z)
16 {
17    int tg;
18    // trao doi theo thu tu tuan hoan
19    tg = *y;
20    *y = *x;
21    *x = *z;
22    *z = tg;
23 }
```

✍ Kết quả chương trình

```
Nhap vao ba so a, b, c tuong ung: 12 3 54
Gia tri cac bien a, b, c truoc khi hoan doi:
a = 12
b = 3
c = 54
Gia tri cua cac bien a, b, c sau khi hoan doi:
a = 54
b = 12
c = 3
```

🕮 Ví dụ 5.40

Viết chương trình nhập vào năm số nguyên từ bàn phím, sử dụng con trỏ.

Năm giá trị sẽ được lưu trữ vào trong mảng sử dụng con trỏ. Sau đó in các phần tử của mảng ra màn hình.

✍ Chương trình

```
1   #include<stdio.h>
2   #include<conio.h>
3     int main()
4      {
5      int arr[5],i;
6      int *p=arr;
7      printf("Nhap vao 5 gia tri cach nhau dau cach:");
8      scanf("%d%d%d%d%d",p,p+1,p+2,p+3,p+4);
9      printf("Cac gia tri cua mang:\n");
10     for(i=0;i<5;i++)
11        printf("%d\n",arr[i]);
12     getch();
13     return 0;
14     }
```

✍ Kết quả chương trình

```
Nhap vao 5 gia tri cach nhau dau cach:12 4 56 43 6
Cac gia tri cua mang:
12
4
56
43
6
```

5.3. CHUỖI KÝ TỰ

5.3.1. Định nghĩa

Một chuỗi ký tự trong C là một dãy (mảng) các ký tự, được kết thúc bằng ký tự '\0' (còn được gọi là ký tự NULL trong bảng mã ASCII).

Độ dài của một chuỗi ký tự được xác định từ ký tự đầu tiên của chuỗi đến khi gặp ký tự kết thúc null '\0'. Chẳng hạn, một chuỗi có nội dung "abcdef" gồm bảy ký tự 'a', 'b', 'c', 'd', 'e', 'f' và ký tự kết thúc '\0' (có giá trị bằng 0).

🕮 Ví dụ 5.41

Viết chương trình in chuỗi S = "DHCNGTVT" ra màn hình, cho biết độ dài của chuỗi S. In các ký tự của S từ ký tự đầu tiên đến ký tự cuối cùng và in các ký tự của S theo chiều ngược, từ ký tự cuối đến ký tự đầu tiên ra màn hình.

✍Chương trình

```
1   #include<stdio.h>
2   #include<string.h>
3   #define MAX_STRING_LEN 80
4   int main() {
5      /*  Xau la  mang cac ky tu
6      * cham dut bang ky tu NULL'\0' */
7      char S[MAX_STRING_LEN];
8     int l, i;
9     S[0] = 'D';
10    S[1] = 'H';
11    S[2] = 'C';
12    S[3] = 'N';
13    S[4] = 'G';
14    S[5] = 'T';
15    S[6] = 'V';
16    S[7] = 'T';
17    S[8] = 0;
18    l = strlen(S);
19    printf("In xau S:\t%s\n",S);
20    printf("Do dai xau S:\t%d\n",l);
21     /* In cac ky tu trong xau S */
22    printf("Tu ky tu dau tien den ky tu cuoi cung:\n");
23    for (i = 0; i < l; ++i)
24      printf("A[%d] = %c\n",i,S[i]);
```

```
25      /* In cac ky tu cua xau nghich dao cua S */
26      printf("\nCac ky tu duoc in theo chieu nguoc:\n");
27    for (i = l-1; i >= 0; --i)
28      printf("A[%d] = %c\n",i,S[i]);
29  }
```

Kết quả chương trình:

```
In xau S:      DHCNGTVT
Do dai xau S:  8
Tu ky tu dau tien den ky tu cuoi cung:
A[0] = D
A[1] = H
A[2] = C
A[3] = N
A[4] = G
A[5] = T
A[6] = V
A[7] = T
Cac ky tu duoc in theo chieu nguoc:
A[7] = T
A[6] = V
A[5] = T
A[4] = G
A[3] = N
A[2] = C
A[1] = H
A[0] = D
```

Hình ảnh của chuỗi S:

D	H	C	N	G	T	V	T	\0

Trong C, hằng chuỗi được đặt trong dấu nháy kép, chẳng hạn chuỗi *"DHCNGTVT",* được biên dịch thành mảng các ký tự và được bổ sung ký tự '\0' đánh dấu kết thúc chuỗi.

5.3.2. Khai báo

5.3.2.1. Khai báo bằng mảng

char str_name[size];

Trong đó:

str_name: tên biến chuỗi;

size: độ dài của chuỗi;

char: là kiểu dữ liệu ký tự;

Độ dài tối đa của biến chuỗi là một hằng nguyên có giá trị từ 1 đến 255. Khai báo độ dài tối đa để khi biên dịch, chương trình cấp số ô nhớ theo khai báo, tránh trường hợp khai báo thừa hoặc thiếu, dẫn đến tình trạng lãng phí bộ nhớ.

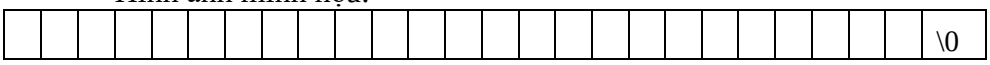
🕮 Ví dụ 5.42

Khai báo chuỗi Ho_ten có độ dài tối đa 25 ký tự:

char Ho_ten[25];

Với khai báo này, chương trình sẽ cung cấp 25 + 1 bytes để lưu trữ nội dung của biến Ho_ten, byte cuối cùng lưu trữ ký tự '\0' kết thúc chuỗi.

Hình ảnh minh họa:



5.3.2.2. Khai báo và khởi tạo giá trị

char str_name[] = < "const_str">;

Trong đó:

str_name: tên biến chuỗi;

const_str: hằng chuỗi;

🕮 Ví dụ 5.43

Cho chương trình sau đây:

```
1  #include<stdio.h>
2  int main()
3  {
4      // khai bao va khoi tao gia tri
5      char str[] = "DHCNGTVT";
6      // in chuoi ky tu
```

```
7       printf("%s",str);
8       return 0;
9   }
```

✍**Kết quả chương trình:**

```
DHCNGTVT
```

Cũng giống như mảng, trong quá trình khai báo chuỗi, ta có thể đặt kích thước chuỗi ký tự bên trong [] của khai báo.

Chẳng hạn, cho khai báo sau đây:

char str[15] = “DHCNGTVT”;

Đảm bảo rằng kích thước chuỗi đặt bên trong [] đủ lớn để chứa tất cả các ký tự trong chuỗi, cộng với ký tự ‘\0’ (NULL) để kết thúc. Trong ví dụ trên, các chỉ số từ 0 đến 8 sẽ được khởi tạo cho các ký tự của chuỗi “DHCNGTVT” và ký tự ‘\0’ (NULL). Các chỉ số còn lại từ 9 đến 14 sẽ được khởi tạo bằng \0 (giống ký tự NULL khi chuyển đổi sang char). Trong bộ nhớ, chuỗi trên được minh họa như sau:

D	H	C	N	G	T	V	T	\0	\0	\0	\0	\0	\0	\0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

5.3.2.3. Khai báo bằng con trỏ

char *<str_pointer_name>;

Chương trình sẽ dành 2 byte bộ nhớ để lưu địa chỉ của biến con trỏ đang trỏ đến, chưa cung cấp nơi để lưu trữ dữ liệu. Muốn lưu trữ dữ liệu, gọi hàm malloc() và calloc() trong thư viện “alloc.h”, sau đó mới gán dữ liệu cho biến.

🕮 **Ví dụ 5.44**

✍ Cho đoạn chương trình sau đây:

```
1   #include <stdio.h>
2   #include <string.h>
3   int main()
4   {
5           char *char_ptr = "DHCNGTVT";
6           printf(char_ptr);
```

```
7           return 0;
8   }
```

✍**Kết quả chương trình:**

```
DHCNGTVT
```

Thao tác này khởi tạo biến con trỏ char_ptr để trỏ đến ký tự đầu tiên của chuỗi "DHCNGTVT".

5.3.3. Thao tác nhập xuất chuỗi ký tự

5.3.3.1. Nhập xuất chuỗi ký tự

a. Nhập chuỗi ký tự

Hàm gets trong C được sử dụng để quét hoặc đọc một dòng văn bản từ một thiết bị đầu vào tiêu chuẩn (stdin) và lưu trữ nó trong biến chuỗi. Khi nó đọc ký tự dòng mới thì hàm gets sẽ bị chấm dứt.

Để sử dụng hàm gets, bắt buộc phải khai báo thư viện stdio.h.

Cú pháp của hàm gets:

```
gets(<str_name>);
Trong đó: str_name là tên biến chuỗi ký tự;
hoặc char *gets(char *str_pointer_name);
Trong đó: str_pointer_name: tên biến con trỏ chuỗi ký tự
```

🕮 **Ví dụ 5.45**

✍ Cho đoạn chương trình sau đây:

```
1   // Vi du ve ham gets trong C
2   #include <stdio.h>
3   int main()
4   {
5           char name[50];
6           printf("\nNhap vao ten: \n");
7           gets(name);
8           printf("=============\n");
9           printf("%s", name);
10          return 0;
11  }
```

Kết quả chương trình:

```
Nhap vao ten:
Pham Thi Thuan
=============
Pham Thi Thuan
```

Ví dụ sau đây, sẽ giúp cho sinh viên hiểu rõ hơn về hàm gets, sự khác nhau giữa hàm scanf() và hàm gets() và lý do tại sao sử dụng hàm gets() thay vì hàm scanf() khi cần làm việc với chuỗi ký tự.

Ví dụ 5.46

Xem xét đoạn chương trình sau đây:

```
1  // Vi du ve chuong trinh cho thay su khac biet giua ham gets() va ham scanf()
2  #include <stdio.h>
3  int main()
4  {
5          char name[50];
6          printf("\nNhap vao ten day du: \n");
7          scanf("%s", name);
8          printf("=============\n");
9          printf("%s", name);
10         return 0;
11 }
```

Kết quả chương trình:

```
Nhap vao ten day du:
Le Thi Chi
=============
Le
```

Thay vì sẽ sử dụng lệnh scanf(), ta sử dụng lệnh gets():

```
1  // Vi du ve chuong trinh cho thay su khac biet giua ham gets() va ham scanf()
2  #include <stdio.h>
3  int main()
4  {
```

```
5           char name[50];
6           printf("\nNhap vao ten day du: \n");
7           gets(name);
8           printf("=============\n");
9           printf("%s", name);
10          return 0;
11  }
```

Kết quả chương trình:

```
Nhap vao ten day du:
Le Thi Chi
=============
Le Thi Chi
```

Kết quả của hai đoạn chương trình hoàn toàn khác nhau, ở ví dụ sử dụng hàm gets(), chương trình in ra toàn bộ chuỗi ký tự nhập vào, trong khi ở ví dụ sử dụng hàm scanf() chỉ đọc ra từ đầu tiên "Le", lý do thật đơn giản, bởi vì hàm scanf() xem từ "Le" như một giá trị và trong C, hàm scanf() đơn giản chỉ dùng cho việc nhập chuỗi không có ký tự trắng, chẳng hạn nếu nhập "LeThiChi" thì chương trình sẽ in toàn vẹn chuỗi ký tự này.

5.3.3.2. Xuất chuỗi ký tự ra màn hình

Trong C, để in một biến chuỗi str ra màn hình chuẩn stdout, ta có thể sử dụng một trong hai câu lệnh sau đây:

```
puts(str);
printf(str);
```

Hàm puts() có thể được ưu tiên để in chuỗi vì nó thường ít tốn kém hơn (Việc thực hiện puts() đơn giản hơn hàm printf())và nếu trong chuỗi có các ký tự định dạng, chẳng hạn như "%" thì printf() sẽ đưa ra những kết quả khác không mong đợi. Ngoài ra nếu str là một chuỗi đầu vào của người dùng, việc sử dụng hàm printf() có thể gây ra sự cố về bảo mật.

Cũng lưu ý rằng, hàm puts() di chuyển con trỏ đến dòng tiếp theo. Nếu chúng ta không muốn di chuyển con trỏ, có thể sử dụng hàm fputs() là một biến thể của hàm puts() như sau:

```
fputs(str, stdout)
```

Hai ví dụ sau đây cho thấy sự khác nhau giữa hàm puts() và hàm prinft():

🕮 Ví dụ 5.47

✍ Sử dụng hàm puts():

```
1  #include<stdio.h>
2  int main()
3  {
4      puts("DHCNGTVT");
5      puts("CNGTVT");
6      getchar();
7      return 0;
8  }
```

✍ **Kết quả chương trình:**

```
DHCNGTVT
CNGTVT
```

🕮 Ví dụ 5.48

✍Sử dụng hàm puts() có kèm theo các ký tự định dạng %

```
1  #include<stdio.h>
2  int main()
3  {
4      puts("CNGTVT%sDHCNGTVT%s");
5      getchar();
6      return 0;
7  }
```

✍Kết quả chương trình:

```
CNGTVT%sDHCNGTVT%s
```

🕮 Ví dụ 5.49

✍ Sử dụng hàm printf()

```
1  #include<stdio.h>
2  int main()
```

```
{
    // % duoc dat o day de cho thay tac dung phu cua lenh printf(str)
    printf("DHCNGTVT%sCNGTVT%s");
    getchar();
    return 0;
}
```

🕮 Ví dụ 5.50

✍ Sử dụng hàm fputs()

```
#include<stdio.h>
int main()
{
    fputs("DHCNGTVT", stdout);
    fputs("CNGTVT", stdout);
    getchar();
    return 0;
}
```

✍ Kết quả khi biên dịch chương trình

```
DHCNGTVTCNGTVT
```

Tuy nhiên, tùy theo tình huống cụ thể, việc sử dụng hàm puts() hoặc printf() được áp dụng linh hoạt.

5.3.4. Các hàm xử lý chuỗi ký tự cơ bản

5.3.4.1. Hàm xác định độ dài chuỗi

Hàm strlen() tính toán độ dài của một chuỗi đã cho, không tính đến ký tự null '\0'. Hàm strlen() được định nghĩa thư viện "string.h".

Cú pháp:

int strlen(const char *str);

Chức năng: Trả về độ dài của chuỗi str. (str đại diện cho chuỗi ký tự cần xác định độ dài)

🕮 Ví dụ 5.51

✍ Cho chương trình sau đây:

```
#include<stdio.h>
```

```
#include <string.h>
int main()
{
  char ch[]={'d', 'h', 'c', 'n', 'g', 't', 'v', 't', '\0'};
  printf("Do dai cua chuoi: %d", strlen(ch));
  return 0;
}
```

✍ Kết quả chương trình:

```
Do dai cua chuoi: 8
```

🕮 Ví dụ 5.52

✍ Xem xét chương trình sau đây:

```
#include<stdio.h>s
#include<string.h>
#define MAX_STRING_LEN 80
int main() {
  char S1[MAX_STRING_LEN];
  char S2[MAX_STRING_LEN];
  int i, l;
  printf("Nhap xau ban dau S1:\t");
  scanf("%s",S1);
  //Copy tat ca cac ky tu, ket thuc voi ky tu NULL
  l = strlen(S1);
  for (i = 0; i < l+1; ++i)
    S2[i] = S1[i];
  /* thay doi xau ban dau S1 */
  S1[0] = S1[1] = S1[2] = '*';
  S1[3] = 0;
  /* In ca hai xau ra man hinh */
  printf("S1:\t%s\n",S1);
  printf("S2:\t%s\n",S2);
}
```

✍Kết quả chương trình:

```
Nhap xau ban dau S1:    dhcngtvt
S1:     ***
S2:     dhcngtvt
```

5.3.4.2. Hàm so sánh hai chuỗi ký tự

Hàm strcmp() là một hàm thư viện tích hợp trong thư viện <string.h>. Hàm này lấy 2 chuỗi làm đối số và so sánh hai chuỗi này theo từ vựng.

Cú pháp:

int strcmp(const char *leftStr, const char *rightStr);

Chức năng:

Hàm strcmp() sử dụng hai chuỗi làm tham số, trả về một giá trị nguyên dựa trên việc so sánh các chuỗi:

Strcmp() so sánh hai chuỗi theo từ vựng tiếng anh, nghĩa là bắt đầu so sánh từ ký tự đầu tiên cho đến khi các ký tự của hai chuỗi bằng nhau, hoặc gặp phải một ký tự NULL. Nếu ký tự đầu tiên trong cả hai chuỗi bằng nhau, thì hàm sẽ kiểm tra ký tự thứ hai của cả hai chuỗi, nếu hai ký tự này bằng nhau thì kiểm tra đến ký tự thứ ba, v.v. Quá trình này tiếp tục cho đến khi gặp ký tự NULL ở một trong hai chuỗi hoặc hai ký tự tương ứng ở hai chuỗi không bằng nhau.

Hàm strcmp() có thể trả về ba giá trị:

0: nếu hai chuỗi giống hệt nhau (tất cả các ký tự ở các vị trí tương ứng của hai chuỗi là giống nhau).

> 0: nếu ký tự không khớp đầu tiên trong leftStr có giá trị trong bảng mã ASCII lớn hơn ký tự tương ứng trong rightStr.

< 0: nếu ký tự không khớp đầu tiên trong leftStr có giá trị trong bảng mã ASCII nhỏ hơn ký tự tương ứng trong rightStr.

🕮 Ví dụ 5.53

Viết chương trình nhập vào hai chuỗi, đưa ra màn hình kết quả so sánh hai chuỗi.

✍Cách 1: Sử dụng hàm ctrcmp

```
1  #include <stdio.h>
2  #include <string.h>
3  int main()
4  {
5     char a[100], b[100];
6      printf("Nhap vao chuoi thu nhat:\n");
7     gets(a);
8      printf("Nhap vao chuoi thu hai:\n");
```

```
    gets(b);
    if (strcmp(a,b) == 0)
      printf("Hai chuoi giong nhau.\n");
    else
      printf("Hai chuoi khong giong nhau.\n");
    return 0;
}
```

Kết quả chương trình:

```
Trường hợp 1:
Nhap vao chuoi thu nhat:
ab
Nhap vao chuoi thu hai:
ab
Hai chuoi giong nhau.
Trường hợp 2:
Nhap vao chuoi thu nhat:
ab
Nhap vao chuoi thu hai:
abc
Hai chuoi khong giong nhau.
```

Cách 2: Xây dựng hàm compare(a, b): so sánh 2 chuỗi a và b

```
#include <stdio.h>
int compare_strings(char [], char []);
int main()
{
  char a[1000], b[1000];
  printf("Nhap vao chuoi thu nhat:\n");
  gets(a);
  printf("Nhap vao chuoi thu hai:\n");
  gets(b);
  if (compare_strings(a, b) == 0)
    printf("Hai chuoi giong nhau.\n");
  else
```

```
13        printf("Hai chuoi khong giong nhau.\n");
14      return 0;
15   }
16    int compare_strings(char a[], char b[])
17   {
18      int c = 0;
19      while (a[c] == b[c]) {
20        if (a[c] == '\0' || b[c] == '\0')
21           break;
22        c++;
23      }
24        if (a[c] == '\0' && b[c] == '\0')
25        return 0;
26      else
27        return -1;
28   }
```

✍Kết quả chương trình:

```
Nhap vao chuoi thu nhat:
Cntt
Nhap vao chuoi thu hai:
Cngtvt
Hai chuoi khong giong nhau.
```

✍Cách 3: Sử dụng con trỏ:

```
1   #include<stdio.h>
2   int compare_string(char*, char*);
3   int main()
4   {
5       char a[1000], b[1000];
6       int kq;
7       printf("Nhap vao chuoi thu nhat:\n");
8       gets(a);
9       printf("Nhap vao chuoi thu hai:\n");
10      gets(b);
11      kq = compare_string(a, b);
```

```
    if (kq == 0)
      printf("Hai chuoi giong nhau.\n");
    else
      printf("Hai chuoi khac nhau.\n");
    return 0;
}
int compare_string(char *a, char *b) {
  while (*a == *b) {
    if (*a == '\0' || *b == '\0')
      break;
    a++;
    b++;
  }

  if (*a == '\0' && *b == '\0')
    return 0;
  else
    return -1;
}
```

Kết quả chương trình:

```
Nhap vao chuoi thu nhat:
abc
Nhap vao chuoi thu hai:
abc
Hai chuoi giong nhau.
```

5.3.4.3. Hàm sao chép chuỗi

Hàm strcpy() là một hàm trong thư viện chuẩn "string.h" của C, được sử dụng để sao chép nội dung từ chuỗi này đến chuỗi khác.

Cú pháp:

```
char* strcpy(char* dest, const char* src);
```

Có hai tham số của hàm:

dest: con trỏ trỏ tới mảng đích nơi nội dung sẽ được sao chép.

src: chuỗi sẽ được sao chép.

Trả về giá trị: sau khi sao chép chuỗi nguồn vào chuỗi đích, hàm strcpy() trả về con trỏ trỏ vào chuỗi đích.

🕮 Ví dụ 5.54

Viết chương trình khai báo hai chuỗi a, b. Nhập nội dung cho chuỗi a, sau đó sao chép nội dung của chuỗi a sang chuỗi b và in nội dung của hai chuỗi ra màn hình.

✍ Cách 1: sử dụng hàm strcpy()

```
1  #include <stdio.h>
2  #include <string.h>
3  int main()
4  {
5     char a[1000], b[1000];
6     printf("Nhap chuoi\n");
7     gets(a);
8     strcpy(b, a);
9     printf("Chuoi nguon: %s\n", a);
10    printf("Chuoi sao chep: %s\n", b);
11    return 0;
12 }
```

✍Kết quả chương trình:

```
Nhap chuoi
abc
Chuoi nguon: abc
Chuoi sao chep: abc
```

✍ Cách 2: Không sử dụng hàm strcpy()

```
1  #include <stdio.h>
2   void copy_string(char [], char []);
3   int main() {
4  char s[1000], d[1000]="dhcngtvt";
5  printf("Noi dung cua chuoi luc ban dau: %s\n", d);
6    printf("Nhap vao chuoi:\n");
7    gets(s);
```

```
    copy_string(d, s);
    printf("Chuoi sao chep: %s\n", d);
     return 0;
  }
   void copy_string(char d[], char s[]) {
    int c = 0;
     while (s[c] != '\0') {
      d[c] = s[c];
      c++;
    }
    d[c] = '\0';
  }
```

Kết quả chương trình:

```
Noi dung cua chuoi luc ban dau: dhcngtvt
Nhap vao chuoi:
abc
Chuoi sao chep: abc
```

Cách 3: Xây dựng hàm sao chép chuỗi ký tự

```
#include <stdio.h>
 void copy_string(char [], char []);
 int main() {
  char s[1000], d[1000];
  printf("Nhap vao chuoi ky tu s:\n");
  gets(s);
  copy_string(d, s);
  printf("Sao chep noi dung chuoi s va chuoi d: %s\n", d);
  return 0;
}
 void copy_string(char d[], char s[]) {
  int c = 0;
   while (s[c] != '\0') {
    d[c] = s[c];
    c++;
```

```
16    }
17    d[c] = '\0';
18  }
```

Kết quả chương trình:

```
Nhap vao chuoi ky tu s:
cngtvt
Sao chep noi dung chuoi s va chuoi d: cngtvt
```

Cách 4: Sử dụng con trỏ

```
1  #include <stdio.h>
2   void copy_string(char * , char * );
3   int main() {
4    char s[1000], d[1000];
5    printf("Nhap vao chuoi s:\n");
6    gets(s);
7    copy_string(d, s);
8    printf("Sao chep noi dung chuoi s vao chuoi d: %s\n", d);
9    return 0;
10  }
11  void copy_string(char *target, char *source) {
12     while (*source) {
13        *target = *source;
14        source++;
15        target++;
16     }
17     *target = '\0';
18  }
```

Kết quả chương trình:

```
Nhap vao chuoi s:
abc
Sao chep noi dung chuoi s vao chuoi d: abc
```

5.3.4.4. Hàm ghép chuỗi

Hàm strcat() là một hàm trong thư viện chuẩn "string.h", có nhiệm vụ ghép nối nội dung của một chuỗi vào cuối của chuỗi khác.

Cú pháp:

char *strcat(char *s1, const char *s2);

Chức năng: Hàm strcat() nối thêm một bản sao của chuỗi được trỏ bởi con trỏ s2 vào cuối chuỗi được trỏ bởi con trỏ s1. Kết quả của hàm trả về một con trỏ trỏ vào tới s1, nơi chứa kết quả của phép ghép hai chuỗi.

Hai tham số của hàm strcat():

s1: Con trỏ trỏ tới chuỗi sẽ được thay đổi, nội dung của chuỗi mà con trỏ s2 đang trỏ tới được sao chép vào cuối chuỗi, nơi con trỏ s1 đang trỏ tới.

s2: Con trỏ trỏ tới chuỗi sẽ được nối vào cuối chuỗi con trỏ s1 đang trỏ tới.

Lưu ý: Sử dụng hàm strcat() một cách thận trọng, vì nó dễ dàng sao chép nhiều bytes vào biến, có thể gây ra những hành vi không thể đoán trước.

🕮 Ví dụ 5.55

Nhập vào hai chuỗi ký tự, nối và in màn hình kết quả của phép nối hai chuỗi ký tự?

✍ Cách 1: Sử dùng hàm strcat()

```
1  #include <stdio.h>
2  #include <string.h>
3   int main()
4  {
5    char a[1000], b[1000];
6     printf("Nhap vao chuoi thu nhat:\n");
7    gets(a);
8     printf("Nhap vao chuoi thu hai:\n");
9    gets(b);
10   // noi hai chuoi a va b bang ham strcat, ghep chuoi b vao chuoi a
11   strcat(a, b);
12    printf("Chuoi ky tu thu duoc khi ghep hai chuoi tren: %s\n", a);
13    return 0;
14 }
```

✍ **Kết quả chương trình:**

Nhap vao chuoi thu nhat:

```
ab
Nhap vao chuoi thu hai:
bcde
Chuoi ky tu thu duoc khi ghep hai chuoi tren: abbcde
```

Cách 2: Không sử dụng hàm strcat()

```
1  #include <stdio.h>
2  void concatenate(char [], char []);
3  int main()
4  {
5    char p[100], q[100];
6    printf("Nhap vao chuoi p:\n");
7    gets(p);
8    printf("Nhap vao chuoi q\n");
9    gets(q);
10   concatenate(p, q);
11   printf("Chuoi thu duoc khi ghep q vao p: \%s\n", p);
12    return 0;
13 }
14  void concatenate(char p[], char q[]) {
15   int c, d;
16     c = 0;
17    while (p[c] != '\0') {
18     c++;
19   }
20    d = 0;
21    while (q[d] != '\0') {
22     p[c] = q[d];
23     d++;
24     c++;
25   }
```

```
26      p[c] = '\0';
27  }
```

Kết quả chương trình:

```
Nhap vao chuoi p:
Truong Dai hoc
Nhap vao chuoi q
Cong nghe GTVT
Chuoi thu duoc khi ghep q vao p: Truong Dai hoc Cong nghe GTVT
```

Cách 3: Sử dụng con trỏ

```
1  #include <stdio.h>
2  void concatenate_string(char*, char*);
3  int main()
4  {
5     char a[100], b[100];
6     printf("Nhap vao chuoi a:\n");
7     gets(a);
8     printf("Nhap vao chuoi b:\n");
9     gets(b);
10    // ghep b vao a
11    concatenate_string(a, b);
12    //ket qua cua phep ghep
13 printf("Chuoi thu duoc sau khi ghep b vao a: \"%s\"\n", a);
14    return 0;
15 }
16 void concatenate_string(char *a, char *b)
17 {
18   while(*a)
19     a++;
20   while(*b)
21   {
22      *a = *b;
23      b++;
```

```
24        a++;
25     }
26     *a = '\0';
27  }
```

✍Kết quả chương trình:

```
Nhap vao chuoi a:
abc
Nhap vao chuoi b:
bcdef
Chuoi thu duoc sau khi ghep b vao a: "abcbcdef"
```

5.3.4.5. Hàm xác định chuỗi nghịch đảo

Hàm strrev() là hàm được dựng sẵn trong C, được định nghĩa trong thư viện "string.h". Hàm strrev() được sử dụng để đảo ngược chuỗi đã cho.

Cú pháp:

```
char *strrev(char *str);
```

Hàm trả về một chuỗi ký tự, sau khi đã đảo ngược chuỗi str đã cho

🕮 Ví dụ 5.56

Viết chương trình nhập vào 1 chuỗi ký tự. In chuỗi đảo ngược?

✍ Cách 1 : Sử dùng hàm strrev

✍ Chương trình:

```
1   #include <stdio.h>
2   #include <string.h>
3   int main()
4   {
5      char arr[100];
6       printf("Nhap vao mot xau\n");
7      gets(arr);
8       strrev(arr);
9       printf("Xau dao nguoc la \n%s\n", arr);
10      return 0;
11  }
```

✍ Kết quả chương trình:

```
Nhap vao mot xau
abcdef
Xau dao nguoc la
fedcba
```

✍ Cách 2: Không sử dùng hàm strrev

```
1  #include <stdio.h>
2  int main()
3  {
4    char s[1000], r[1000];
5    int begin, end, count = 0;
6     printf("Nhap vao chuoi:\n");
7    gets(s);
8     // tinh do dai cua chuoi
9     while (s[count] != '\0')
10   count++; // bien dem do dai chuoi
11   end = count - 1;
12   for (begin = 0; begin < count; begin++)
13   {
14     r[begin] = s[end];
15     end--;
16   }
17   r[begin] = '\0';
18   printf("Chuoi dao nguoc la:\n");
19         printf("%s\n", r);
20   return 0;
21 }
```

✍ Kết quả chương trình:

```
Nhap vao chuoi:
truong dai hoc cong nghe giao thong van tai
Chuoi dao nguoc la:
iat nav gnoht oaig ehgn gnoc coh iad gnourt
```

Cách 3: Sử dụng hàm đệ quy

```
1   #include <stdio.h>
2   #include <string.h>
3   void reverse(char*, int, int);
4    int main()
5   {
6      char a[100];
7      printf("Nhap vao chuoi:\n");
8      gets(a);
9      reverse(a, 0, strlen(a)-1);
10     printf("Chuoi dao nguoc la:\n");
11     printf("%s\n", a);
12     return 0;
13  }
14   void reverse(char *x, int begin, int end)
15  {
16     char c;
17      if (begin >= end)
18        return;
19      c        = *(x+begin);
20     *(x+begin) = *(x+end);
21     *(x+end)   = c;
22      reverse(x, ++begin, --end);
23  }
```

Kết quả khi biên dịch:

```
Nhap vao chuoi:
bo mon he thong thong tin
Chuoi dao nguoc la:
nit gnoht gnoht eh nom ob
```

Cách 4: Chương trình không đệ quy, sử dụng con trỏ

```
1   #include<stdio.h>
2    int string_length(char*);
```

```
3  void reverse(char*);
4  main()
5  {
6    char s[100];
7     printf("Nhap vao chuoi:\n");
8     gets(s);
9     reverse(s);
10    printf("Chuoi dao nguoc la:\n");
11    printf("%s\n", s);
12    return 0;
13 }
14  void reverse(char *s)
15 {
16   int length, c;
17   char *begin, *end, temp;
18    length = string_length(s);
19   begin  = s;
20   end    = s;
21    for (c = 0; c < length - 1; c++)
22      end++;
23    for (c = 0; c < length/2; c++)
24   {
25      temp   = *end;
26      *end   = *begin;
27      *begin = temp;
28       begin++;
29      end--;
30   }
31 }
32  int string_length(char *pointer)
33 {
34   int c = 0;
35    while( *(pointer + c) != '\0' )
```

```
36        c++;
37      return c;
38  }
```

✍ Kết quả chương trình:

```
Nhap vao chuoi:
khoa cong nghe thong tin
Chuoi dao nguoc la:
nit gnoht ehgn gnoc aohk
```

5.3.4.6. Chuyển đổi ký tự hoa sang ký tự thường và ngược lại

Hàm strlwr() là một hàm dựng sẵn trong C, được sử dụng để chuyển đổi một chuỗi ký tự đã cho thành các ký tự thường.

Cú pháp:

```
char *strlwr(char *str);
```

Trong đó, con trỏ str trỏ tới chuỗi đang cần chuyển đổi sang ký tự thường, kết quả trả về của hàm là một con trỏ trỏ vào chuỗi ký tự đã được chuyển đổi sang ký tự thường.

Hàm strupr() được sử dụng để chuyển đổi các ký tự của chuỗi str đã cho sang ký tự viết hoa.

Cú pháp:

```
char *strupr(char *str);
```

Trong đó, con trỏ str trỏ tới chuỗi đang cần chuyển đổi sang ký tự hoa, kết quả trả về là một con trỏ trỏ vào chuỗi đã được chuyển đổi sang ký tự viết hoa.

🕮 Ví dụ 5.57

Chuyển đổi ký tự thường thành ký tự hoa.

✍ Cách 1: Sử dụng hàm strlwr()

```
1  #include <stdio.h>
2  #include <string.h>
3  int main()
4  {
5    char a[1000];
6    printf("Nhap chuoi can chuyen doi:\n");
```

```
7     gets(a);
8     printf("Chuoi ky tu viet thuong: %s\n", strlwr(a));
9     return  0;
10  }
```

✍ Cách 2: Không sử dụng hàm strlwr()

```
1   #include <stdio.h>
2   void lower_string(char []);
3   int main()
4   {
5     char a[100];
6     printf("Chuoi ban dau chua cac ky tu can chuyen doi thanh ky
    tu thuong:\n");
7     gets(a);
8     // goi ham chuyen doi tu hoa thanh thuong
9     lower_string(a);
10    printf("Chuoi thu duoc sau chuyen doi: %s\n", a);
11    return 0;
12  }
13  void lower_string(char s[]) {
14    int c = 0;
15    // dua vao bang ma, ma cua ky tu thuong = ma cua ky tu hoa +
    32
16    while (s[c] != '\0') {
17      if (s[c] >= 'A' && s[c] <= 'Z') {
18        s[c] = s[c] + 32;
19      }
20      c++;
21    }
22  }
```

✍ Kết quả chương trình:

Nhap chuoi can chuyen doi:

TRUONG DAI HOC CNGTVT

Chuoi ky tu viet thuong: truong dai hoc cngtvt

🕮 Ví dụ 5.58

Chuyển đổi chuỗi ký tự từ viết thường sang viết hoa

✍ Cách 1: Sử dụng hàm strupr()

```
#include <stdio.h>
#include <string.h>
int main()
{
   char a[1000];
   printf("Nhap vao chuoi ky tu:\n");
   gets(a);
   printf("Chuoi sau khi chuyen doi thanh ky tu hoa: %s\n", strupr(a));
   return  0;
}
```

✍ Cách 2: Không sử dụng hàm strupr()

```
#include <stdio.h>
void upper_string(char []);
int main()
{
   char a[100];

   printf("Nhap vao chuoi can chuyen doi thanh ky tu viet hoa:\n");
   gets(a);
   // goi ham chuyen doi
   upper_string(a);
   printf("Chuoi thu duoc sau chuyen doi: %s\n", a);
   return 0;
}
// xay dung ham chuyen doi
void upper_string(char s[]) {
   int c = 0;
   // dua vao bang ma ASCII
   while (s[c] != '\0') {
```

```
19      if (s[c] >= 'a' && s[c] <= 'z') {
20         s[c] = s[c] - 32;
21      }
22      c++;
23    }
24  }
```

✍ Kết quả chương trình:

Nhap vao chuoi ky tu:

truong dai hoc cngtvt

Chuoi sau khi chuyen doi thanh ky tu hoa: TRUONG DAI HOC CNGTVT

5.3.4.7. Tóm tắt các hàm trong thư viện string.h

strlen(s): Trả về độ dài của chuỗi s.
strcpy(s1,s2): Sao chép nội dung chuỗi s2 vào chuỗi s1.
strncpy(s1, s2, n): Chép n ký tự từ chuỗi s2 sang chuỗi s1. Nếu chiều dài s2 < n thì hàm sẽ điền khoảng trắng cho đủ n ký tự vào s1.
strcat(s1, s2): Nối chuỗi s2 vào chuỗi s1.
strncat(s1, s2, n): Nối n ký tự đầu tiên của chuỗi s2 vào chuỗi s1.
strcmp(s1, s2): So sánh 2 chuỗi s1 và s2 theo nguyên tắc thứ tự từ điển (Phân biệt chữ hoa và thường). Trả về: • 0: nếu s1 bằng s2. • > 0: nếu s1 lớn hơn s2. • < 0: nếu s1 nhỏ hơn s2.
strncmp(s1,s2, n): Tương tự như strcmp, nhưng chỉ so sánh n ký tự đầu tiên của 2 chuỗi.
stricmp(s1, s2): Tương tự như strcmp, nhưng không phân biệt chữ hoa hay thường.
strnicmp(s1, s2, n): Tương tự như stricmp, nhưng chỉ so sánh n ký tự đầu của 2 chuỗi.
strchr(s, c): Tìm lần xuất hiện đầu tiên của ký tự c trong chuỗi s. Trả về: • NULL: nếu không có.

• Địa chỉ c: nếu tìm thấy.

strtok(s1, s2):

• Nếu s2 có xuất hiện trong s1: Tách chuỗi s1 thành hai chuỗi: Chuỗi đầu là những ký tự cho đến khi gặp chuỗi s2 đầu tiên, chuỗi sau là những ký tự còn lại của s1 sau khi đã bỏ đi chuỗi s2 xuất hiện trong s1.

• Nếu s2 không xuất hiện trong s1 thì kết quả chuỗi tách vẫn là s1.

5.3.5. Một số dạng bài tập về chuỗi ký tự

🕮 Ví dụ 5.59

Nhập vào một chuỗi ký tự. Đưa ra màn hình cho biết chuỗi ký tự đó có đối xứng hay không?

✍**Gợi ý:** Cho chuỗi s, s được gọi là chuỗi đối xứng nếu chuỗi đảo ngược s thu được một chuỗi giống s. Chẳng hạn, "abcba" là một chuỗi đối xứng, "abcdba" không là chuỗi đối xứng.

Cách 1:

✍**Phác thảo lời giải**

- Nhập vào một chuỗi a;
- Sử dụng hàm strcpy() để sao chép nội dung của chuỗi a vào chuỗi b;
- Sử dụng hàm strrev() để đảo ngược chuỗi b, kết quả lưu vào b;
- Sử dụng hàm strcmp(a, b) để so sánh hai chuỗi a, b có bằng nhau;
- Nếu strcmp(a, b) = 0 thì kết luận a là chuỗi đối xứng, ngược lại kết luận a không phải là chuỗi đối xứng.

✍**Chương trình**

```
1  #include <stdio.h>
2  #include <string.h>
3  int main()
4  {
5    char a[100], b[100];
6    printf("Nhap vao mot chuoi:\n");
7    gets(a);
8    strcpy(b, a);  // sao chep a vao b
9    strrev(b);  // dao nguoc xau b
10   if (strcmp(a, b) == 0)  // so sanh chuoi ban dau voi chuoi dao
     nguoc
11     printf("%s la chuoi doi xung.\n", a);
```

```
12    else
13      printf("%s khong la chuoi doi xung.\n",a);
14    return 0;
15  }
```

✍Kết quả chương trình:

```
Trường hợp 1:
Nhap vao mot chuoi:
abcba
abcba la chuoi doi xung.
Trường hợp 2:
Nhap vao mot chuoi:
abcab
abcab khong la chuoi doi xung.
```

Cách 2:

✍Phác thảo lời giải

- Nhập vào chuỗi a;
- Sử dụng các biến begin, end, middle để đánh dấu các vị trí đầu, cuối, ở giữa của chuỗi a.
- Sử dụng biến length để lưu trữ chiều dài của chuỗi a (Sử dụng hàm strlen() hoặc sử dụng một vòng lặp while đếm từ ký tự đầu tiên cho đến khi gặp ký tự '\0' thì dừng); chuỗi a chứa các ký tự có chỉ số từ 0 đến length -1;
- Gán middle = length/2 (vị trí giữa của chuỗi), end = length - 1 (đánh dấu ký tự cuối của chuỗi a).
- Lần lượt so sánh các phần tử a[begin] với a[end], a[begin +1] với a[end -1],… (Sử dụng một vòng lặp for để xử lý).
- Nếu tất cả các cặp đều giống nhau, khẳng định a là chuỗi đối xứng, ngược lại chỉ cần 1 cặp không thõa mãn tính chất, sử dụng lệnh break để thoát và thông báo a không phải là chuỗi đối xứng.

✍Chương trình

```
1   #include <stdio.h>
2    int main()
3   {
4     char a[100];
5     int begin, middle, end, length = 0;
6     printf("Nhap vao mot chuoi:\n");
```

```
7     gets(a);
8      while (a[length] != '\0')
9       length++;
10     end = length - 1;
11    middle = length/2;
12     for (begin = 0; begin < middle; begin++)
13    {
14      if (a[begin] != a[end])
15      {
16        printf("Khong phai chuoi doi xung.\n");
17        break;
18      }
19      end--;
20    }
21    if (begin == middle)
22      printf("%s la chuoi doi xung.\n", a);
23     return 0;
24  }
```

✍ Kết quả chương trình:

```
Nhap vao mot chuoi:
abcddcba
abcddcba la chuoi doi xung.
Nhap vao mot chuoi:
abcdabc
Khong phai chuoi doi xung.
```

Cách 3:

✍Phác thảo lời giải

- Xây dựng hàm doixung(str) để kiểm tra tính đối xứng của chuỗi str.
- Sử dụng hàm strlen() để xác định độ dài của chuỗi str, giả sử chuỗi str có độ dài n, tức là n = strlen(str);
- Sử dụng các biến l, r để đánh dấu các ký tự bên trái và bên phải của chuỗi str; khởi tạo các chỉ số l = 0 và r = n - 1;
- Trong khi l < r:

▪ Nếu str(l) khác với str(r) thì thông báo str không phải là chuỗi đối xứng và dừng;

▪ Nếu str(l) = str(r) thì tăng l, giảm r, tức là l++ và r --;

- Thoát khỏi vòng lặp, thông báo str là chuỗi đối xứng.

✍Chương trình:

```
#include <stdio.h>
#include <string.h>
// Xay dung ham doixung de kiem tra chuoi str co doi xung khong?
void doixung(char str[])
{
   // danh dau vi tri ben trai va ben phai chuoi str
   int l = 0;
   int r = strlen(str) - 1;
    // Tiep tuc so sanh cac ky tu khi chung con giong nhau
   while (r > l)
   {
      if (str[l++] != str[r--])
      {
         printf("%s khong phai chuoi doi xung", str);
         return;
      }
   }
   printf("%s la chuoi doi xung", str);
}

// Chuong trinh chinh
int main()
{
   char s[100];
   printf("Nhap vao chuoi s:\n");
   gets(s);
   // goi ham kiem tra doi xung
         doixung(s);
```

```
29      return 0;
30  }
```

Kết quả chương trình:

```
Nhap vao chuoi s:
ababccbaba
ababccbaba la chuoi doi xung
Nhap vao chuoi s:
abcdabc
abcdabc khong phai chuoi doi xung
```

Cách 4: Sử dụng hàm và con trỏ, không sử dụng các hàm strcpy, strcmp, strrev

Phác thảo lời giải

- Xây dựng các hàm is_palindrome(): kiểm tra tính đối xứng;
- Xây dựng hàm copy_string(): sao chép chuỗi;
- Xây dựng hàm reverse_string(): đảo ngược chuỗi ký tự;
- Xây dựng hàm string_length(): xác định độ dài chuỗi;
- Xây dựng hàm copare_string(): so sánh hai chuỗi;
- Thực chất là xây dựng các hàm để thay thế các hàm strcpy, strcmp, strrev, strlen trong thư viện "string.h"

Chương trình:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   int is_palindrome(char*);
5   void copy_string(char*, char*);
6   void reverse_string(char*);
7   int string_length(char*);
8   int compare_string(char*, char*);
9
10  int main()
11  {
12    char a[100];
13    int kq;
14     printf("Nhap vao chuoi ky tu\n");
15    gets(a);
```

```
16      kq = is_palindrome(a);
17       if (kq == 1)
18         printf("\"%s\" la chuoi doi xung.\n", a);
19      else
20         printf("\"%s\" khong la chuoi doi xung.\n", a);
21       return 0;
22  }
23
24  int is_palindrome(char *a)
25  {
26     int kt, length;
27     char *reverse;
28
29     length = string_length(a);
30     reverse = (char*)malloc(length+1);
31
32     copy_string(reverse, a);
33     reverse_string(reverse);
34      kt = compare_string(a, reverse);
35      free(reverse);
36      if (kt == 0)
37        return 1;
38     else
39        return 0;
40  }
41
42  int string_length(char *a)
43  {
44     int length = 0;
45
46     while(*a)
47     {
48        length++;
49        a++;
50     }
```

```
51
52      return length;
53  }
54
55  void copy_string(char *t, char *s)
56  {
57      while(*s)
58      {
59          *t = *s;
60          s++;
61          t++;
62      }
63      *t = '\0';
64  }
65  void reverse_string(char *a)
66  {
67      int length, c;
68      char *begin, *end, tg;
69      length = string_length(a);
70      begin = a;
71      end = a;
72      for (c = 0; c < (length - 1); c++)
73          end++;
74      for (c = 0; c < length/2; c++)
75      {
76          tg = *end;
77          *end = *begin;
78          *begin = tg;
79          begin++;
80          end--;
81      }
82  }
83  int compare_string(char *first, char *second)
84  {
85      while(*first==*second)
```

```
86      {
87        if (*first == '\0' || *second == '\0')
88          break;
89        first++;
90        second++;
91      }
92      if (*first == '\0' && *second == '\0')
93        return 0;
94      else
95        return -1;
96    }
```

✍Kết quả chương trình:

```
Nhap vao chuoi ky tu
abcdcba
"abcdcba" la chuoi doi xung.
Nhap vao chuoi ky tu
abccab
"abccab" khong la chuoi doi xung.
```

🕮 Ví dụ 5.60

Nhập vào một số nguyên dương? Kiểm tra số nguyên dương đó có đối xứng không?

✍ Phác thảo lời giải

- Nhập vào số nguyên dương n;
- Xác định số nghịch đảo r của n;
- Nếu n = r thì kết luận n là số đối xứng, ngược lại kết luận n không phải số đối xứng.

✍ Chương trình:

```
1  #include <stdio.h>
2   int main()
3  {
4     int n, r = 0, t;
5      printf("Nhap vao mot so nguyen:\n");
6     scanf("%d", &n);
7      t = n;
```

```
8     while (t != 0) // xac dinh so nghich dao r
9     {
10      r = r * 10;
11      r = r + t%10;
12      t = t/10;
13    }
14    if (n == r)
15      printf("%d la so doi xung.\n", n);
16    else
17      printf("%d khong la so doi xung.\n", n);
18    return 0;
19  }
```

Kết quả chương trình:

```
Nhap vao mot so nguyen:
1234321
1234321 la so doi xung.
Nhap vao mot so nguyen:
123654
123654 khong la so doi xung.
```

BÀI TẬP CHƯƠNG 5

A. BÀI TẬP CÓ LỜI GIẢI

🕮 Bài tập 5.1

Sinh ngẫu nhiên mảng gồm n số thực trong phạm vi từ 5 đến 100 và hiển thị các phần tử của mảng ra màn hình.

✍ Chương trình:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void Sinh_ngau_nhien(float a[], int n) {
   float r;
   srand((int)time(0));
         for (int i = 0; i < n; i++)
      {
       r = 5 + (rand()/(float) RAND_MAX*(100 - 5));
       a[i]= r;
       }
}
void Hien_thi(float a[], int n)
{
          printf("Hien thi cac phan tu cua mang:\n");
          for(int i=0; i<n; i++)
                  printf("%f \t", a[i]);
}

int main(){
   float a[1000];
   int n;
         printf("Nhap so phan tu cua mang = ");
         scanf("%d", &n);
         Sinh_ngau_nhien(a, n);
         Hien_thi(a,n);
   return 0;
}
```

✍ Kết quả chương trình:

```
Nhap so phan tu cua mang = 10
Hien thi cac phan tu cua mang:
98.031403    87.193977    97.529831    74.747459
55.821102    39.637440    28.904387    76.005737
46.360821    43.713795
```

🕮 Bài tập 5.2

Lập trình nhập vào một mảng gồm n số nguyên. Hiển thị mảng ra màn hình, đồng thời in các số lẻ ra màn hình?

✍ Phác thảo lời giải

- Xây dựng hàm Nhap_mang() để nhập các phần tử của mảng.
- Xây dựng hàm Hien_thi() để hiển thị các phần tử của mảng
- Xây dựng hàm In_so_le() để liệt kê các số lẻ của mảng (số lẻ là số không chia hết cho 2).
- Cả ba hàm trên đều sử dụng vòng lặp for.

✍ Chương trình:

```
1   #include<stdio.h>
2   #include<conio.h>
3   void Nhap_mang(int a[], int n)
4           {
5            for (int i = 0; i < n; i++)
6                    {
7                    printf("Phan tu thu %d: ", i);
8                   scanf("%d", &a[i]);
9                    }
10          }
11  void Hien_thi(int a[], int n)
12          {
13          for (int i = 0; i < n; i++)
14                   {
15                   printf("%d \t", a[i]);
16                   }
17          }
18
```

```
void In_so_le (int a[], int n)
        {
        printf("\nCac so le la: ");

 for (int i=0; i < n; i++)
       {
        if(a[i] % 2 != 0) printf("%d \t", a[i]);
       }
}
int main()
        {
        int a[50], n;
        printf("Nhap so luong phan tu: ");
        scanf("%d", &n);
        Nhap_mang(a, n);
        Hien_thi(a, n);
        In_so_le(a, n);
        return 0;
        }
```

✍Kết quả chương trình:

```
Nhap so luong phan tu: 6
Phan tu thu 0: 43
Phan tu thu 1: 5
Phan tu thu 2: 64
Phan tu thu 3: 44
Phan tu thu 4: 57
Phan tu thu 5: 4
43      5       64      44      57      4
Cac so le la: 43        5       57
```

🕮 Bài tập 5.3

Viết chương trình nhập vào mảng gồm n phần tử là các số nguyên. Sắp xếp các phần tử của mảng giảm dần theo giá trị (sử dụng thuật toán sắp xếp chọn).

Chương trình:

```
1   #include<stdio.h>
2   #include<conio.h>
3   #include<math.h>
4   // khai bao hang MAX = 100 la so phan tu toi da cua mang
5   #define MAX 100
6   void Nhap_mang(int a[], int n)
7   {
8           // Su dung vong lap for de nhap mang, co the su dung
    phuong phap sinh ngau nhien mang
9           for(int i = 0; i < n; i++)
10          {
11                  printf("\nPhan tu a[%d]: ", i);
12                  scanf("%d", &a[i]);
13          }
14  }
15  void Hien_thi(int a[], int n)
16  {
17
18          for(int i = 0; i < n; i++)
19          {
20                  printf("%4d", a[i]);
21          }
22          printf("\n");
23  }
24  void Sapxep(int a[], int n)
25  {
26          int i,imax,j,tg;
27     for (i=0; i<=n-2; i++)
28     {
29        imax = a[i]; //Tim imax
30        for (j=i+1; j<=n-1; j++)
31           if (a[j] > imax)
32           {
```

```
33                  imax = a[j];
34
35                  //Hoan doi a[i] va a[j]
36                  tg = a[i];
37                  a[i] = a[j];
38                  a[j] = tg;
39              }
40      }
41  }
42  int main()
43  {
44          int n;
45          int a[MAX];
46      //Kiem tra tinh hop le cua so n nhap vao, dam bao 0 <= n <= MAX
47          do
48          {
49                  printf("\nNhap so phan tu: ");
50                  scanf("%d", &n);
51                  if(n <= 0 || n > MAX)
52                  {
53                          printf("Khong hop le. Nhap so trong pham vi
                            tu 0 den MAX !");
54                  }
55          }while(n <= 0 || n > MAX);
56          Nhap_mang(a, n);
57          printf("In cac phan tu cua mang ban dau: \n");
58          Hien_thi(a, n);
59          Sapxep(a, n);
60          printf("In cac phan tu cua mang sap xep giam dan dan: \n");
61          // Goi ham Hien_thi() de in cac phan tu cua mang ra man hinh
62          Hien_thi(a, n);
63          //Dung man hinh
64          getch();
65          return 0;
66  }
```

✍ Kết quả chương trình:

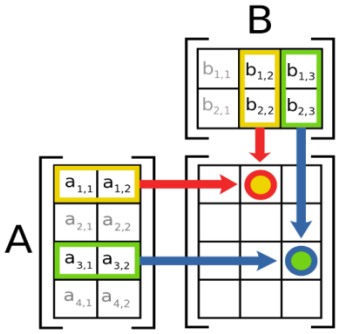
```
MAX = 100:
Nhap so phan tu: -1
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap so phan tu: 102
Khong hop le. Nhap so trong pham vi tu 0 den MAX !
Nhap so phan tu: 6
Phan tu a[0]: 43
Phan tu a[1]: 3
Phan tu a[2]: 55
Phan tu a[3]: 43
Phan tu a[4]: 23
Phan tu a[5]: 25
In cac phan tu cua mang ban dau:
 43  3  55  43  23  25
In cac phan tu cua mang sap xep giam dan:
 55  43  43  25  23  3
```

🕮 Bài tập 5.4

Lập trình tính và hiển thị tích của hai ma trận.

✍Phác thảo lời giải:

Phép nhân hai ma trận được xác định khi và chỉ khi số cột của ma trận bên trái bằng số hàng của ma trận bên phải. Nếu A là một ma trận m × n và B là một ma trận n × p, thì ma trận tích AB là ma trận m × p với các phần tử được xác định theo tích vô hướng của hàng tương ứng trong A với cột tương ứng trong B:



Hình 5.3. Minh họa phép nhân hai ma trận

$$[\mathbf{AB}]_{\mathbf{i,j}} = \mathbf{A_{i,1}B_{1,j}} + \mathbf{A_{i,2}B_{2,j}} + \cdots + \mathbf{A_{i,n}B_{n,j}} = \sum_{r=1}^{n} \boldsymbol{A_{i,r}B_{r,j}}$$

với $\mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{m}, \mathbf{1} \leq \boldsymbol{j} \leq \boldsymbol{p}$.

Phép nhân ma trận thỏa mãn quy tắc:

Tính chất kết hợp: (AB)C = A(BC).

Luật phân phối trái và phải: (A + B)C = AC + BC và C(A + B) = CA + CB khi kích thước của các ma trận tham gia vào phép nhân thỏa mãn yêu cầu của tích hai ma trận.

Tích AB có thể xác định trong khi BA không nhất thiết phải xác định, tức là nếu A và B lần lượt có số chiều m×n và n×k và m ≠ k. Thậm chí khi cả hai tích này đều tồn tại thì chúng không nhất thiết phải bằng nhau, tức là AB ≠ BA.

Chương trình:

```
1  #include<stdio.h>
2  #define MAX 10
3  int main()
4  {
5     int n, m, p, q, k, a[MAX][MAX], b[MAX][MAX],
   Tich[MAX][MAX],Tong = 0;
6     int i, j;
7     printf("Nhap so hang va so cot cua ma tran thu nhat:\n");
8                 do
9           {
10                printf("+ Nhap so hang cua ma tran thu nhat: ");
11                scanf("%d", &m);
12                if(m <= 0 || m > MAX)
13                {
14                      printf("Khong hop le. Nhap so trong pham
   vi tu 0 den MAX !\n");
15                }
16          }while(m <= 0 || m > MAX);
17          //doan chuong trinh kiem tra so cot nhap vao
18                do
19          {
20                printf("+ Nhap so cot cua ma tran thu nhat: ");
21                scanf("%d", &n);
```

```
22                  if(n <= 0 || n > MAX)
23                  {
24                  printf("Khong hop le. Nhap so trong pham vi tu 0 den MAX !\n");
25                  }
26          }while(n <= 0 || n > MAX);
27    printf("Nhap %d phan tu cua ma tran thu nhat:\n", m*n);
28    for(i = 0; i < m; i++)   // to iterate the rows
29       for(j = 0; j < n; j++)   // to iterate the columns
30          scanf("%d", &a[i][j]);
31    printf("Nhap so hang va so cot cua ma tran thu hai:\n");
32          do
33          {
34                  printf("+ Nhap so hang cua ma tran thu hai: ");
35                  scanf("%d", &p);
36                  if(p <= 0 || p > MAX)
37                  {
38                          printf("Khong hop le. Nhap so trong pham vi tu 0 den MAX !\n");
39                  }
40          }while(p <= 0 || p > MAX);
41                  do
42          {
43                  printf("+ Nhap so cot cua ma tran thu hai: ");
44                  scanf("%d", &q);
45                  if(q <= 0 || q > MAX)
46                  {
47                  printf("Khong hop le. Nhap so trong pham vi tu 0 den MAX !\n");
48                  }
49          }while(q <= 0 || q > MAX);
50  // Kiem tra so hang cua ma tran thu hai co bang so cot cua ma tran thu nhat
51    if(n != p)
52       printf("Hai ma tran nay khong the nhan voi nhau\n");
53    else
```

```
{
    printf("Nhap %d phan tu cua ma tran thu hai:\n",p*q);

    for(i = 0; i < p; i++)
        for(j = 0; j < q; j++)
            scanf("%d", &b[i][j]);
    // In ma tran thu nhat
    printf("Ma tran thu nhat la: \n");
    for(i = 0; i < m; i++)
    {
        for(j = 0; j < n; j++)
        {
            printf("%d\t", a[i][j]);
        }
        printf("\n");
    }
    // In ma tran thu hai
    printf("Ma tran thu hai la: \n");
    for(i = 0; i < p; i++)
    {
        for(j = 0; j < q; j++)
        {
            printf("%d\t", b[i][j]);
        }
        printf("\n");
    }
    // tinh ma tran tich
    for(i = 0; i < m; i++)
    {
        for(j = 0; j < q; j++)
        {
            for(k = 0; k < p; k++)
            {
                Tong = Tong + a[i][k]*b[k][j];
            }
```

```
89              Tich[i][j] = Tong;
90              Tong = 0;
91              }
92          }
93          // In cac phan tu cua ma tran tich
94          printf("Tich cua hai ma tran la:\n");
95          for(i = 0; i < m; i++)
96          {
97              for(j = 0; j < q; j++)
98              {
99                  printf("%d\t", Tich[i][j]);
100             }
101             printf("\n");
102         }
103     }
104     return 0;
105 }
```

Kết quả chương trình:

Trường hợp 1: Nhập số hàng của ma trận thứ hai bằng số cột của ma trận thứ nhất

```
Nhap so hang va so cot cua ma tran thu nhat:
+ Nhap so hang cua ma tran thu nhat: 3
+ Nhap so cot cua ma tran thu nhat: 2
Nhap 6 phan tu cua ma tran thu nhat:
4
12
54
3
22
5
Nhap so hang va so cot cua ma tran thu hai:
+ Nhap so hang cua ma tran thu hai: 2
+ Nhap so cot cua ma tran thu hai: 4
Nhap 8 phan tu cua ma tran thu hai:
65
```

3
21
4
5
61
1
13
Ma tran thu nhat la:
4 12
54 3
22 5
Ma tran thu hai la:
65 3 21 4
5 61 1 13
Tich cua hai ma tran la:
320 744 96 172
3525 345 1137 255
1455 371 467 153

Trường hợp 2: Số hàng của ma trận thứ hai khác số cột của ma trận thứ nhất

Nhap so hang va so cot cua ma tran thu nhat:
+ Nhap so hang cua ma tran thu nhat: 3
+ Nhap so cot cua ma tran thu nhat: 2
Nhap 6 phan tu cua ma tran thu nhat:
12
43
54
5
65
2
Nhap so hang va so cot cua ma tran thu hai:
+ Nhap so hang cua ma tran thu hai: 3
+ Nhap so cot cua ma tran thu hai: 3
Hai ma tran nay khong the nhan voi nhau

✍**Yêu cầu với sinh viên:** Đoạn chương trình còn một số đoạn code bị lặp lại. Viết lại chương trình bằng các hàm sau:

- Hàm nhập ma trận.
- Hàm hiển thị ma trận.
- Hàm nhân hai ma trận, kiểm tra số hàng của ma trận thứ hai bằng số cột của ma trận thứ nhất.

🕮 Bài tập 5.5

Cải tiến chương trình để in ra năm số nguyên theo thứ tự đảo ngược với quá trình nhập vào?

✍ Chương trình:

```
1  #include<stdio.h>
2  #include<conio.h>
3    int main()
4     {
5      int arr[5],i;
6      int *p=arr;
7      printf("Nhap 5 phan tu cach nhau boi dau cach:");
8      scanf("%d%d%d%d%d",p,p+1,p+2,p+3,p+4);
9      printf("In mang theo thu tu nguoc:\n");
10     for(i=4;i>=0;i--)
11        printf("%d\n",*(p+i));
12     getch();
13     return 0;
14    }
```

✍ **Kết quả chương trình:**

```
Nhap 5 phan tu cach nhau boi dau cach:12 54 3 65 4
In mang theo thu tu nguoc:
4
65
3
54
12
```

🕮 Bài tập 5.6

Cải tiến chương trình ở bài tập 5.5, viết chương trình nhập vào mảng gồm n phần tử. In mảng theo thứ tự đảo ngược.

Chương trình

```
#include <stdio.h>
int main()
{
   int n, i, arr1[15];
   int *pt;
 printf("\nIn cac phan tu cua mang theo thu tu dao nguoc :\n");
          printf("-------------------------------------\n");
   printf("Nhap vao so phan tu cua mang (toi da 15) : ");
   scanf("%d",&n);
   pt = &arr1[0];  //  con tro pt chua dia chi cua phan tu dau tien cua
mang
   printf("Nhap %d phan tu cua mang: \n",n);
   for(i=0;i<n;i++)
      {
            printf("Phan tu thu - %d : ",i+1);
            scanf("%d",pt);
            pt++;
            }
   pt = &arr1[n - 1];
   printf("\nCac phan tu cua mang theo thu tu dao nguoc :");
   for (i = n; i > 0; i--)
   {
      printf("\nPhan tu - %d : %d  ", i, *pt);
      pt--;
   }
printf("\n\n");
}
```

Kết quả chương trình

```
In cac phan tu cua mang theo thu tu dao nguoc:
-----------------------------------------------
Nhap vao so phan tu cua mang (toi da 15): 4
Nhap 4 phan tu cua mang:
Phan tu thu - 1: 12
Phan tu thu - 2: 3
```

```
Phan tu thu - 3: 54
Phan tu thu - 4: 22
Cac phan tu cua mang theo thu tu dao nguoc:
Phan tu - 4: 22
Phan tu - 3: 54
Phan tu - 2: 3
Phan tu - 1: 12
```

🕮 Bài tập 5.7

Viết chương trình tìm phần tử lớn nhất của một mảng được cấp phát bộ nhớ động.

✍ Chương trình

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main()
4  {
5     int i,n;
6     float *p;
7           printf("\nTim kiem phan tu lon nhat cua mang :\n");
8           printf("-----------------------------------\n");
9      printf(" Nhap so phan tu cua mang (1 den 100): ");
10     scanf("%d",&n);
11    // su dung ham calloc() cap phat bo nho cho n phan tu
12    p=(float*)calloc(n,sizeof(float));
13    if(p==NULL)
14    {
15       printf("Khong du bo nho cap phat.");
16       exit(0);
17    }
18    printf("\n");
19    for(i=0;i<n;++i)
20    {
21      printf("Nhap phan tu thu %d: ",i+1);
22      scanf("%f",p+i);
23    }
```

```
    for(i=1;i<n;++i)
    {
      if(*p<*(p+i))
         *p=*(p+i);
    }
    printf("Phan tu lon nhat cua mang:  %.2f \n\n",*p);
    return 0;
}
```

✍ Kết quả chương trình

```
Tim kiem phan tu lon nhat cua mang:
----------------------------------
Nhap so phan tu cua mang (1 den 100): 4
Nhap phan tu thu 1: 43
Nhap phan tu thu 2: 2
Nhap phan tu thu 3: 55
Nhap phan tu thu 4: 23
Phan tu lon nhat cua mang: 55.00
```

🕮 Bài tập 5.8

Sử dụng con trỏ cấp phát bộ nhớ động, viết chương trình sắp xếp mảng gồm n phần tử.

✍ Chương trình

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int *a,i,j,tmp,n;
        printf("Sap xep mang su dung con tro :\n");
        printf("-----------------------------\n");
  printf("Nhap so phan tu cua mang : ");
  scanf("%d",&n);
  a = (int*) malloc(n * sizeof(int));
  printf("Nhap %d phan tu cua mang : \n",n);
  // nhap mang
  for(i=0;i<n;i++)
```

```
14      {
15              printf("Phan tu thu - %d : ",i+1);
16              scanf("%d",a+i);
17              }
18    for(i=0;i<n;i++)
19    {
20     for(j=i+1;j<n;j++)
21     {
22       if( *(a+i) > *(a+j))
23       {
24       tmp = *(a+i);
25       *(a+i) = *(a+j);
26       *(a+j) = tmp;
27        }
28     }
29    }
30    printf("In mang sau khi sap xep : \n");
31    for(i=0;i<n;i++)
32      {
33              printf("Phan tu - %d : %d \n",i+1,*(a+i));
34              }
35  printf("\n");
36  free(a);
37  }
```

Kết quả chương trình:

```
Sap xep mang su dung con tro:
-----------------------------
Nhap so phan tu cua mang: 5
Nhap 5 phan tu cua mang:
Phan tu thu - 1: 12
Phan tu thu - 2: 5
Phan tu thu - 3: 65
Phan tu thu - 4: 33
Phan tu thu - 5: 4
In mang sau khi sap xep:
```

```
Phan tu - 1: 4
Phan tu - 2: 5
Phan tu - 3: 12
Phan tu - 4: 33
Phan tu - 5: 65
```

🕮 Bài tập 5.9

Cải tiến chương trình được viết ở ví dụ 5.48, hãy viết chương trình sắp xếp mảng gồm mười phần tử theo thứ tự tăng dần.

Hàm sắp xếp gồm hai tham số: một con trỏ trỏ tới mảng và kích thước của mảng. Hàm trả về một con trỏ trỏ tới mảng đã được sắp xếp.

🖉 Chương trình

```
1  #include<stdio.h>
2  #include<conio.h>
3  int *sortAsc(int *p, int size);
4    int main()
5     {
6      int arr[]={23,34,2,3,5,12,42,56,89,8};
7      int *p=sortAsc(arr,10);
8      // dau ra la mang da sap xep
9            int i;
10     printf("Mang sau khi sap xep:\n");
11           for(i=0;i<10;i++)
12       printf("%d\n",*(p+i));
13     return 0;
14    }
15    int *sortAsc(int *p, int n){
16    int i,j;
17    for(i=0;i<n;i++)
18       for(j=i+1;j<n;j++)
19          if(*(p+j)<*(p+i))
20            {
21              int temp=*(p+j);
22              *(p+j)=*(p+i);
23              *(p+i)=temp;
```

```
24                  }
25      return p;
26  }
```

✍ Kết quả chương trình:

```
Mang sau khi sap xep:
2
3
5
8
12
23
34
42
56
89
```

🕮 Bài tập 5.10

Nhập vào một chuỗi ký tự str và một ký tự ch. Cho biết số lần xuất hiện của ký tự ch trong chuỗi str?

✍ Phác thảo lời giải

Trong chương trình này, chuỗi được nhập bởi người dùng được lưu trữ trong biến str.

Sau đó, người dùng được yêu cầu nhập ký tự ch.

Sử dụng vòng lặp for, mỗi ký tự trong chuỗi được kiểm tra cho ký tự được nhập.

Nếu ký tự được tìm thấy, số lần xuất hiện được tăng lên. Nếu không, vòng lặp tiếp tục.

Kết thúc vòng lặp, in số lần xuất hiện của ký tự ch trong chuỗi str.

✍ Chương trình

```
1   #include <stdio.h>
2   int main()
3   {
4       char str[1000], ch;
5       int i, solan = 0;
```

```
6     printf("Nhap vao mot chuoi ky tu: ");
7     gets(str);
8     printf("Nhap vao ky tu can dem so lan xuat hien trong chuoi: ");
9     scanf("%c",&ch);
10    for(i = 0; str[i] != '\0'; ++i)
11    {
12       if(ch == str[i])
13          ++solan;
14    }
15    printf("So lan xuat hien cua ky tu %c trong chuoi = %d", ch,
      solan);
16    return 0;
17 }
```

✍ Kết quả chương trình:

```
Nhap vao mot chuoi ky tu: abbcccdddd
Nhap vao ky tu can dem so lan xuat hien trong chuoi: c
So lan xuat hien cua ky tu c trong chuoi = 3
```

🕮 Bài tập 5.11

Nhập vào một chuỗi bao gồm các ký tự trong bảng chữ cái tiếng Anh. Đếm số ký tự nguyên âm, phụ âm, dấu cách trong chuỗi đã nhập?

✍ Phác thảo lời giải

Đầu tiên, nhập vào một chuỗi ký tự từ bàn phím bao gồm các ký tự trong bảng chữ cái tiếng Anh và lưu trữ trong một biến.

Các biến đếm số nguyên âm, phụ âm, chữ số và dấu cách được khởi tạo bằng 0.

Sử dụng vòng lặp, khi tìm thấy nguyên âm, biến đếm số nguyên âm được tăng thêm 1. Tương tự với các biến đếm số phụ âm, chữ số và khoảng trắng được tăng lên khi tìm thấy các ký tự này.

In các biến đếm số nguyên âm, phụ âm, dấu cách ra màn hình.

✍ Chương trình:

```
1  #include <stdio.h>
2  int main()
3  {
```

```
4     char str[150];
5     // khai bao bien i dieu khien vong lap
6     // cac bien vowels: dem so nguyen am, consonants: so dem phu
  am, digits: dem so chu so, spaces: dem khoang trang
7           int i, vowels, consonants, digits, spaces;
8     // khoi tao cac bien dem = 0
9     vowels = consonants = digits = spaces = 0;
10    printf("Nhap vao mot chuoi ky tu: ");
11    scanf("%[^\n]", str);
12    // dem so nguyen am
13    // trong bang chu cai tieng anh
14    // nguyen am bao gom: a, e, i, o, u, A, E, I, O, U
15    for(i=0; str[i]!='\0'; ++i)
16    {
17       if(str[i]=='a' || str[i]=='e' || str[i]=='i' ||
18          str[i]=='o' || str[i]=='u' || str[i]=='A' ||
19          str[i]=='E' || str[i]=='I' ||str[i]=='O' ||
20          str[i]=='U')
21       {
22          ++vowels;
23       }
24       // dem so phu am, la nhung ky tu con lai trong bang chu cai
25       else if((str[i]>='a'&& str[i]<='z') || (str[i]>='A'&& str[i]<='Z'))
26       {
27          ++consonants;
28       }
29       // dem chu so trong xau
30                else if(str[i]>='0' && str[i]<='9')
31       {
32          ++digits;
33       }
34       // dem khoang trang
35                else if (str[i]==' ')
36       {
37          ++spaces;
```

```
        }
    }
    // In ket qua so nguyen am, phu am, chu so, khaong trang dem duoc
    printf("So nguyen am: %d",vowels);
    printf("\nSo phu am: %d",consonants);
    printf("\nSo chu so: %d",digits);
    printf("\nSo khoang trang: %d", spaces);
    return 0;
}
```

✍ Kết quả chương trình

```
Nhap vao mot chuoi ky tu: abbcuikthqo uuabiemmmnk
So nguyen am: 9
So phu am: 13
So chu so: 0
So khoang trang: 1
```

🕮 Bài tập 5.12

Viết chương trình nhập vào một chuỗi ký tự. Hiển thị chuỗi ký tự thu được ra màn hình sau khi xóa hết các nguyên âm trong chuỗi?

Cách 1:

✍ Chương trình:

```
#include <stdio.h>
#include <string.h>
int check_vowel(char);
int main()
{
  char s[100], t[100];
  int c, d = 0;
   printf("Nhap vao chuoi s:\n");
  gets(s);
   for(c = 0; s[c] != '\0'; c++) {
    if(check_vowel(s[c]) == 0) {    // Neu ky tu s[c] khong phai la mot nguyen am
      t[d] = s[c];
      d++;
```

```
    }
  }
  t[d] = '\0';
  strcpy(s, t);  // thay doi cau truc cua chuoi ban dau
  printf("Chuoi thu duoc sau khi xoa di cac nguyen am: %s\n", s);
  return 0;
}
// ham kiem tra 1 ky tu co phai la nguyen am hay khong
// cac nguyen am trong bang chu cai tieng Anh: a, A, e, E, i, I, o, ), u, U (viet hoa va viet thuong)
// tra ve 1 neu la mot nguyen am, tra ve 0 neu khong phai la nguyen am
int check_vowel(char ch)
{
   if (ch == 'a' || ch == 'A' || ch == 'e' || ch == 'E' || ch == 'i' || ch == 'I' || ch =='o' || ch=='O' || ch == 'u' || ch == 'U')
     return 1;
   else
     return 0;
}
```

Kết quả chương trình:

```
Nhap vao chuoi s:
abcabutikmnopq
Chuoi thu duoc sau khi xoa di cac nguyen am: bcbtkmnpq
```

Cách 2: Sử dụng con trỏ

Chương trình:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int check_vowel(char);
main()
{
   char a[100], *tg, *p, ch, *start;
    printf("Nhap vao chuoi ban dau:\n");
```

```
 9      gets(a);
10      tg = a;
11     p = (char*)malloc(100);
12
13    if( p == NULL )
14     {
15        printf("Khong the phan bo bo nho.\n");
16        exit(EXIT_FAILURE);
17     }
18      start = p;
19      while(*tg)
20     {
21        ch = *tg;
22         if ( !check_vowel(ch) )
23        {
24           *p = ch;
25           p++;
26        }
27        tg++;
28     }
29     *p = '\0';
30      p = start;
31     strcpy(a, p); /* neu muon thay doi chuoi ban dau */
32     free(p);
33      printf("Chuoi sau khi xoa di cac nguyen am (s): \"%s\"\n", a);
34      return 0;
35  }
36   // kiem tra nguyen am
37  int check_vowel(char a)
38  {
39    if (a == 'a' || a == 'e' || a == 'i' || a == 'o' || a == 'u')
40      return 1;
41    else if (a == 'A' || a == 'E' || a == 'I' || a == 'O' || a == 'U')
42      return 1;
43    else
```

```
44      return 0;
45  }
```

Kết quả chương trình:

```
Nhap vao chuoi ban dau:
abciokmueqp
Chuoi sau khi xoa di cac nguyen am (s): "bckmqp"
```

Bài tập 5.13

Viết chương trình nhập vào 10 chuỗi ký tự. Sắp xếp và hiển thị các chuỗi ký tự này theo vần anphabet?

Chương trình:

```
1   #include <string.h>
2   int main()
3   {
4   int i, j;
5   char str[10][50], temp[50];
6   printf("Nhap 10 tu:\n");
7   for(i=0; i<10; ++i)
8   scanf("%s[^\n]",str[i]);
9   for(i=0; i<9; ++i)
10  for(j=i+1; j<10 ; ++j)
11  {
12  if(strcmp(str[i], str[j])>0)
13  {
14  strcpy(temp, str[i]);
15  strcpy(str[i], str[j]);
16  strcpy(str[j], temp);
17  }
18  }
19  printf("\nCac chuoi duoc in theo thu tu tu dien: \n");
20  for(i=0; i<10; ++i)
21  {
22  puts(str[i]);
23  }
24  return 0; }
```

✍ Kết quả chương trình:

```
Nhap 10 tu:
ab
a
cdr
gg
acb
dgghs
acbd
gfrwe
yta
uqot
Cac chuoi duoc in theo thu tu tu dien:
a
ab
acb
acbd
cdr
dgghs
gfrwe
gg
uqot
yta
```

🕮 Bài tập 5.14

Viết chương trình nhập vào một chuỗi các ký tự. In ra chuỗi thu được sau khi sắp xếp lại các ký tự của chuỗi theo vần alphabet?

✍ Phác thảo lời giải

Xây dựng chương trình bằng ngôn ngữ C để sắp xếp một chuỗi theo thứ tự chữ cái: chẳng hạn, nếu người dùng nhập chuỗi "programming" thì đầu ra sẽ là "aggimmnoprr", vì vậy chuỗi đầu ra sẽ chứa các ký tự theo thứ tự bảng chữ cái. Giả sử chuỗi đầu vào chỉ chứa các chữ cái in thường. Đầu tiên, chúng tôi đếm số lần xuất hiện của ký tự 'a' đến 'z' trong chuỗi đầu vào và sau đó tạo một chuỗi khác chứa các ký tự 'a' đến 'z' với số lần xuất hiện đếm được.

✍ Cách 1:

Chương trình:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   int main()
5   {
6     char ch, in[100], out[100];
7     int no[26] = {0}, n, i, t, x;
8     printf("Nhap vao chuoi ky tu:\n");
9     scanf("%s", in);
10    n = strlen(in); // so ky tu cua chuoi a
11    /* Luu tru so lan xuat hien cua cac ky tu tu a den z
12     xuat hien trong chuoi in vao trong mot mang  */
13    for (i = 0; i < n; i++)
14    {
15      ch = in[i] - 'a';
16      no[ch]++;
17    }
18    t = 0;
19    /* Chen cac ky tu tu 'a' den 'z'  vao chuoi dau ra
20    voi so lan xuat hien trong chuoi dau vao  */
21    for (ch = 'a'; ch <= 'z'; ch++)
22    {
23      x = ch - 'a';
24      for (i = 0; i < no[x]; i++)
25      {
26        out[t] = ch;
27        t++;
28      }
29    }
30    out[t] = '\0';
31    printf("Chuoi duoc sap xep lai: %s\n", out);
32    return 0;
33  }
```

Kết quả chương trình:

```
Nhap vao chuoi ky tu:
aproach
Chuoi duoc sap xep lai: aachopr
```

Cách 2: Sử dụng con trỏ:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   void sort_string(char*);
5   int main()
6   {
7     char a[100];
8     printf("Nhap vao chuoi:\n");
9     gets(a);
10    sort_string(a);
11    printf("Chuoi sau khi sap xep:\n");
12    printf("%s\n", a);
13    return 0;
14  }
15   void sort_string(char *s)
16  {
17    int i, d = 0, length;
18    char *p, *kq, ch;
19    length = strlen(s);
20    kq = (char*)malloc(length+1);
21    p = s;
22     for ( ch = 'a' ; ch <= 'z' ; ch++ )
23    {
24       for ( i = 0 ; i < length ; i++ )
25       {
26          if ( *p == ch )
27          {
28             *(kq+d) = *p;
29             d++;
```

```
30            }
31            p++;
32          }
33          p = s;
34       }
35       *(kq+d) = '\0';
36        strcpy(s, kq);
37       free(kq);
38    }
```

✍ Kết quả chương trình:

```
Nhap vao chuoi:
sort a string
Chuoi sau khi sap xep:
aginorrsstt
```

🕮 Bài tập 5.15

Viết chương trình nhập vào một chuỗi ký tự. Hãy in ra màn hình chuỗi ký tự gồm n ký tự được lấy ra từ chuỗi đã cho từ vị trí t trong chuỗi đã cho, n và t được nhập từ bàn phím?

Xây dựng chương trình ngôn ngữ C để tìm chuỗi con của chuỗi và tất cả các chuỗi con của chuỗi. Một chuỗi con là một phần của chuỗi dài hơn. Ví dụ: các chuỗi con của chuỗi "the" là "" (chuỗi rỗng), "t", "th", "the", "h", "he" và "e".

Thư viện "string.h" không chứa hàm để tìm một chuỗi con trực tiếp.

✍ Chương trình:

```
1   #include <stdio.h>
2   int main()
3   {
4     char a[1000], sub[1000];
5     int t,n , i = 0;
6     printf("Nhap vao chuoi:\n");
7     gets(a);
8     printf("Nhap vao vi tri bat dau lay ra chuoi con:\n");
9     scanf("%d", &t);
10    printf("Nhap vao so ky tu cua chuoi con:\n");
```

```
    scanf("%d", &n);
    while (i < n) {
      sub[i] = a[t + i-1];
      i++;
    }
    sub[i] = '\0';
    printf("Chuoi con theo yeu cau: \"%s\"\n", sub);
    return 0;
}
```

Kết quả chương trình:

```
Nhap vao chuoi:
teacher
Nhap vao vi tri bat dau lay ra chuoi con:
3
Nhap vao so ky tu cua chuoi con:
2
Chuoi con theo yeu cau: "ac"
```

Cách 2: Viết chương trình sử dụng hàm

```
#include <stdio.h>
void substring(char [], char[], int, int);
int main()
{
   char a[1000], sub[1000];
   int vt, n, i = 0;
   printf("Nhap vao chuoi ky tu:\n");
   gets(a);
   printf("Nhap vao vi tri bat dau cua chuoi con:\n");
   scanf("%d", &vt);
   printf("Nhap vao do dai cua chuoi con:\n");
   scanf("%d",&n);
   substring(a, sub, vt, n);
   printf("Chuoi con duoc cat ra la: \"%s\"\n", sub);
   return 0;
}
```

```
17  //Dinh nghia ham lay ra chuoi con cua chuoi ky tu
18  void substring(char s[], char sub[], int p, int l) {
19    int i = 0;
20
21    while (i < l) {
22      sub[i] = s[p+i-1];
23      i++;
24    }
25    sub[i] = '\0';
26  }
```

Kết quả chương trình:

```
Nhap vao chuoi ky tu:
dhcngtvt
Nhap vao vi tri bat dau cua chuoi con:
5
Nhap vao do dai cua chuoi con:
4
Chuoi con duoc cat ra la: "gtvt"
```

Bài tập 5.16

Xóa khoảng cách

Viết chương trình nhập vào một chuỗi các ký tự. In ra màn hình chuỗi ký tự, sau khi xóa các khoảng trắng thừa của chuỗi ký tự?

Chương trình:

```
1   #include <stdio.h>
2   int main()
3   {
4     // khai bao 2 chuoi a: chuoi ban dau chua cac khoang trang thua
5     // b: chuoi thu duoc tu a sau khi xoa cac khoang trang thua
6     char a[1000], b[1000];
7     int i = 0, j = 0;
8     printf("Nhap vao chuoi ky tu:\n");
9     gets(a);
10    while (a[i] != '\0') {
11      if (a[i] == ' ') {
```

```
        int tg = i + 1;
        if (a[tg] != '\0') {
          while (a[tg] == ' ' && a[tg] != '\0') {
            if (a[tg] == ' ') {
              i++;
            }
            tg++;
          }
        }
      }
      b[j] = a[i];
      i++;
      j++;
    }
    b[j] = '\0';
  printf("Chuoi ky tu sau khi xoa cac khoang trang thua:\n%s\n", b);
    return 0;
}
```

✍ Kết quả chương trình:

```
Nhap vao chuoi ky tu:
truong  dai  hoc cngtvt
Chuoi ky tu sau khi xoa cac khoang trang thua:
truong dai hoc cngtvt
```

🕮 Bài tập 5.17

Hoán đổi hai chuỗi ký tự:

```
#include <stdio.h>
#include <string.h>
int main()
{
  char a[100], b[100], tg[100];
  printf("Nhap chuoi thu nhat:\n");
  gets(a);
  printf("Nhap chuoi thu hai:\n");
  gets(b);
```

```
10    printf("\nTruoc khi hoan doi\n");
11    printf("Chuoi thu nhat: %s\n", a);
12    printf("Chuoi thu hai: %s\n\n", b);
13    //hoan doi hai chuoi, su dung bien tg
14    strcpy(tg, a);
15    strcpy(a, b);
16    strcpy(b, tg);
17    printf("Sau khi hoan doi:\n");
18    printf("Chuoi thu nhat: %s\n", a);
19    printf("Chuoi thu hai: %s\n", b);
20    return 0;
21 }
```

✍ Kết quả chương trình:

```
Nhap chuoi thu nhat:
abc
Nhap chuoi thu hai:
ccd
Truoc khi hoan doi
Chuoi thu nhat: abc
Chuoi thu hai: ccd
Sau khi hoan doi:
Chuoi thu nhat: ccd
Chuoi thu hai: abc
```

🕮 Bài tập 5.18

Tần số xuất hiện của các ký tự trong chuỗi

Viết chương trình bằng ngôn ngữ lập trình C để tìm tần số của các ký tự trong một chuỗi: Chương trình đếm tần số của các ký tự trong một chuỗi, tức là, ký tự đó xuất hiện bao nhiêu lần trong chuỗi. Ví dụ: trong chuỗi "code", mỗi ký tự 'c', 'd', 'e' và 'o' đã xuất hiện ra một lần. Trong chương trình này, chỉ xét với bảng chữ cái viết thường, các ký tự khác (chữ hoa và ký tự đặc biệt) được bỏ qua. Chúng ta có thể dễ dàng sửa đổi chương trình này để xử lý chữ hoa và ký hiệu đặc biệt.

✍ Chương trình:

```
1  #include <stdio.h>
2  #include <string.h>
```

```
3   int main()
4   {
5     char st[100];
6     int i = 0, count[26] = {0}, x;
7     printf("Nhap vao chuoi ky tu:\n");
8     gets(st);
9     while (st[i] != '\0') {
10    /*chi xem xet cac ky tu tu 'a' den 'z', bo qua cac ky tu khac */
11      if (st[i] >= 'a' && st[i] <= 'z') {
12        x = st[i] - 'a';
13        count[x]++;
14      }
15      i++;
16    }
17    for (i = 0; i < 26; i++)
18      printf("%c xuat hien %d lan trong chuoi.\n", i + 'a', count[i]);
19    return 0;
20  }
```

Kết quả chương trình:

```
Nhap vao chuoi ky tu:
truong dai hoc cong nghe gtvt
a xuat hien 1 lan trong chuoi.
b xuat hien 0 lan trong chuoi.
c xuat hien 2 lan trong chuoi.
d xuat hien 1 lan trong chuoi.
e xuat hien 1 lan trong chuoi.
f xuat hien 0 lan trong chuoi.
g xuat hien 4 lan trong chuoi.
h xuat hien 2 lan trong chuoi.
i xuat hien 1 lan trong chuoi.
j xuat hien 0 lan trong chuoi.
k xuat hien 0 lan trong chuoi.
l xuat hien 0 lan trong chuoi.
m xuat hien 0 lan trong chuoi.
n xuat hien 3 lan trong chuoi.
```

```
o xuat hien 3 lan trong chuoi.
p xuat hien 0 lan trong chuoi.
q xuat hien 0 lan trong chuoi.
r xuat hien 1 lan trong chuoi.
s xuat hien 0 lan trong chuoi.
t xuat hien 3 lan trong chuoi.
u xuat hien 1 lan trong chuoi.
v xuat hien 1 lan trong chuoi.
w xuat hien 0 lan trong chuoi.
x xuat hien 0 lan trong chuoi.
y xuat hien 0 lan trong chuoi.
z xuat hien 0 lan trong chuoi.
```

✍ Cách 2: Xây dựng hàm đếm số lần xuất hiện

```
1  #include <stdio.h>
2  #include <string.h>
3  void find_frequency(char [], int []);
4  int main()
5  {
6     char st[100];
7     int c, count[26] = {0};
8     printf("Nhap vao chuoi ky tu:\n");
9     gets(st);
10    find_frequency(st, count);
11    printf("Ky tu  So lan\n");
12    for (c = 0 ; c < 26 ; c++)
13       printf("%c \t  %d\n", c + 'a', count[c]);
14    return 0;
15 }
16 // xay dung ham xac dinh so lan xuat hien cua cac ky tu tu 'a' den
   'z' trong chuoi
17 void find_frequency(char s[], int count[]) {
18    int c = 0;
19    while (s[c] != '\0') {
20       if (s[c] >= 'a' && s[c] <= 'z' )
21          count[s[c]-'a']++;
22       c++;
```

```
23     }
24  }
```

Kết quả chương trình:

```
Nhap vao chuoi ky tu:
truong dai hoc cong nghe giao thong van tai
Ky tu  So lan

a       4
b       0
c       2
d       1
e       1
f       0
g       5
h       3
i       3
j       0
k       0
l       0
m       0
n       5
o       5
p       0
q       0
r       1
s       0
t       3
u       1
v       1
w       0
x       0
y       0
z       0
```

🕮 Bài tập 5.19

Viết chương trình nhập vào một chuỗi ký tự, đếm số ký tự, số chữ số và số các ký tự đặc biệt trong chuỗi vừa nhập?

✍ Chương trình:

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #define MAX 100 //so ky tu cua chuoi
5  int main()
6  {
7     char str[MAX];
8     // khai bao cac bien: so_kt:so ky tu trong chuoi
9     //so_cs: so cac chu so 0, 1, 2,..., 9 trong chuoi
10    //so_ktdb: so cac ky tu dac biet trong chuoi nhu cach dau, dau .
11    int so_kt, so_cs, so_ktdb, i;
12    // khoi tao gia tri ban dau
13          so_kt = so_cs= so_ktdb = i = 0;
14       printf("\n\nChuong trinh in so ky tu, so chu so, so ky tu dac
      biet trong chuoi :\n");
15       printf("----------------------------------------\n");
16       printf("Nhap vao chuoi can dem : ");
17       fgets(str, sizeof str, stdin);
18     /* Kiem tra moi phan tu cua chuoi, duyet tu dau cho den khi
      gap ky tu ket thuc '\0'*/
19    while(str[i]!='\0')
20    {
21       if((str[i]>='a' && str[i]<='z') || (str[i]>='A' && str[i]<='Z')) //
      neu gap cac ky tu viet hoa va viet thuong
22       {
23          so_kt++; // tang so ky tu len 1 don vi
24       }
25       else if(str[i]>='0' && str[i]<='9') //neu gap cac chu so tu 0
      den 9
26       {
27          so_cs++; // tang so chu so len 1 don vi
```

```
        }
        else
        {
            so_ktdb++; // con lai, tang so ky tu dac biet len 1 don vi
        }
        i++;
    }
    printf("So cac ky tu trong chuoi : %d\n", so_kt);
    printf("So cac chu so trong chuoi : %d\n", so_cs);
    printf("So cac ky tu dac biet trong chuoi : %d\n\n", so_ktdb);
return 0;
}
```

✍ Kết quả chương trình:

```
Chuong trinh in so ky tu, so chu so, so ky tu dac biet trong chuoi :
--------------------------------------------------
Nhap vao chuoi can dem : SinhvienK69 CNTT DHCNGTVT
So cac ky tu trong chuoi : 21
So cac chu so trong chuoi : 2
So cac ky tu dac biet trong chuoi : 3
```

🕮 Bài tập 5.20

Nhập vào một chuỗi ký tự, in ra màn hình chuỗi sau khi loại bỏ các ký tự đặc biệt và các chữ số (chỉ giữ lại các ký tự)?

✍ Phác thảo lời giải

- Nhập vào chuỗi s;
- Kiểm tra từng ký tự của chuỗi s, nếu không phải là ký tự (ký tự trong bảng mã ASCII nằm trong giới hạn từ a → z hoặc A→Z), thì thực hiện xóa ký tự đó.
- Để xóa 1 ký tự đặc biệt, chữ số thực hiện giống như xóa 1 phần tử trong mảng: Chẳng hạn, để xóa ký tự i, dịch chuyển các ký tự từ i + 1 đến ký tự cuối cùng của chuỗi sang trái 1 vị trí.
- Thực hiện quá trình trên cho đến khi hết chuỗi.

✍ Chương trình:

```
#include<stdio.h>
int main()
{
```

```
4       char s[150];
5       int i, j;
6       printf("Nhap vao mot chuoi:\n ");
7       gets(s);
8        // lap tu ky tu dau tien den khi gap ky tu ket thuc chuoi '\0'
9              //co the su dung vong lap while
10      for(i = 0; s[i] != '\0'; ++i)
11       { // neu gap 1 ky tu khong phai la ky tu (nam ngoai pham vi tu
    a->z hoac A->Z)
12          while (!( (s[i] >= 'a' && s[i] <= 'z') || (s[i] >= 'A' && s[i] <=
    'Z') || s[i] == '\0') )
13          { // neu tai vi tri i la 1 ky tu dac biet hoac chu so
14            // lan luot di doi cac ky tu cac ky tu tu phai qua trai, bat dau
    tu i+1
15              for(j = i; s[j] != '\0'; ++j)
16              {
17                 s[j] = s[j+1];
18              }
19              s[j] = '\0';
20          }
21      }
22      printf("Chuoi sau khi loai bo cac ky tu dac biet, chu so:\n ");
23      puts(s);
24      return 0;
25  }
```

✍ **Kết quả chương trình:**

```
Nhap vao mot chuoi:
Sinh vien k69 .khoa cntt@truong dhcngtvt
Chuoi sau khi loai bo cac ky tu dac biet, chu so:
Sinhvienkkhoacntttruongdhcngtvt
```

🕮 Bài tập 5.21

Viết chương trình nhập vào một chuỗi ký tự. Cho biết ký tự nào xuất hiện nhiều nhất trong chuỗi và số lần xuất hiện của ký tự đó?

✍ Chương trình

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #define MAX 100 //hang MAX: so ky tu toi da cua chuoi
5  #define chr_no 255 //Ky tu tu doi da duoc phep cua chuoi
6  int main()
7  {
8     char str[MAX];
9           int ch_fre[chr_no];
10    int i = 0, max;
11    int ascii;
12       printf("\n\nTim ky tu xuat hien nhieu nhat trong chuoi :\n");
13       printf("---------------------------------------\n");
14       printf("Nhap vao chuoi : ");
15       fgets(str, sizeof str, stdin);
16    for(I = 0; i<chr_no; i++)  // Thiet lap tat cac ky tu co so lan
   xuat hien = 0
17    {
18       ch_fre[i] = 0;
19    }
20    /* Doc tan xuat xuat hien cua tung ky tu */
21    i=0;
22    while(str[i] != '\0') // ky hieu ket thuc cua chuoi '\0'
23    {
24       ascii = (int)str[i]; // lay ma so ascii trong bang ma
25       ch_fre[ascii] += 1; //tang 1 don vi
26       i++; // xet voi ky tu tiep theo
27    }
28    max = 0;
29    for(i = 0; i<chr_no; i++)
30    {
31       if(I != 32)  //tim ky tu co tan xuat xuat hien nhieu nhat
32       {
33       if(ch_fre[i] > ch_fre[max])
34          max = i;
```

```
        }
    }
    printf("Ky tu xuat hien nhieu nhat trong chuoi '%c' voi so lan xuat hien la : %d \n\n", max, ch_fre[max]);
    return 0;
}
```

✍ Kết quả chương trình:

```
Tim ky tu xuat hien nhieu nhat trong chuoi:
-------------------------------------------------
Nhap vao chuoi: truong dai hoc cong nghe giao thong van tai
Ky tu xuat hien nhieu nhat trong chuoi 'g' voi so lan xuat hien la: 5
```

🕮 Bài tập 5.22

Viết chương trình xây dựng một chuỗi rút gọn của một chuỗi bất kỳ nhập vào từ bàn phím.

Ví dụ: X = “aaaaabbbcccchhhaaat” thì chuỗi rút gọn sẽ là: “5a3b4c3h3a1t”.

✍ Phác thảo lời giải

Sử dụng các hàm sau:

itoa(int dem,char *str,10); chuyển đổi số nguyên int *dem* từ hệ 10 thành xâu *str*.

strncpy(char *s1,char *s2, int n); sao chép tối đa *n* ký tự đầu của chuỗi *s2* vào trong *s1*.

strcat(char *s1,char *s2); ghép xâu *s2* vào cuối sâu *s1*.

strcpy(str+i,str+i+dem); xóa *dem* ký tự bắt đầu từ ký tự *i* của xâu *str*.

✍ Chương trình:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void Nhap(char *s)
{
    printf("Nhap chuoi ky tu: ");
        fflush(stdin);
        gets(s);
}
```

```
void RutGon(char *s)
{    int i,len,dem,length;
     char kq[100], s1[100], tg[100];
     strcpy(kq,""); strcpy(tg,""); strcpy(s1,"");
     length=strlen(s);
     do {
     dem = 1; len = strlen(s);
     for (i = 0; i<len-1; i++)
      if (s[i] == s[i+1]) dem++; //dem so ky tu giong nhau dau chuoi s
     else break;
     itoa(dem,s1,10); //doi dem thanh chuoi s1
     strncpy(tg,s,1); //lay 1 ky tu dau chuoi s dua vao chuoi tg
     if ((length ==1)||(len >= 1))
      strcat(kq,s1); //ghep chuoi s1 va cuoi chuoi kq
      strcat(kq,tg);//Ghep  ky tu  dai dien vao cuoi chuoi kq
     strcpy(s+0,s+0+dem);  // Xoa dem ky tu dau tien trong chuoi s
        }  while (len>1);
     printf("\nChuoi rut gon: %s", kq);
     }
int main()
{
  char s[100];
  Nhap(s);
  RutGon(s);
  return 0;
}
```

✍ Kết quả chương trình:

```
Nhap chuoi ky tu: abbbbccddd
Chuoi rut gon: a4b2c3d
```

B. BÀI TẬP TỰ GIẢI

🕮 Bài tập 5.23

Viết chương trình thực hiện các yêu cầu sau đây:

- Nhập mảng gồm n số nguyên. Hiển thị các phần tử của mảng ra màn hình.

- Tìm phần tử lớn nhất và nhỏ nhất của mảng.
- Tính tổng các phần tử là số lẻ trong mảng, in kết quả tính được ra màn hình.
- Liệt kê các số nguyên tố trong mảng.
- Nhập một số x, đưa ra thông báo số x có trong mảng hay không?
- Sắp xếp mảng tăng dần.

Yêu cầu:

Viết thành hàm, với mỗi ý của bài được thể hiện bằng một hàm.

🕮 Bài tập 5.24

Viết chương trình nhập vào một dãy n số thực a[0], a[1],..., a[n-1], sắp xếp dãy số theo thứ tự giảm dần. In dãy số sau khi sắp xếp.

Yêu cầu:

- Sinh mảng ngẫu nhiên.

- Sử dụng các thuật toán sắp xếp, với mỗi thuật toán xây dựng một chương trình.

🕮 Bài tập 5.25

Viết chương trình sắp xếp một mảng theo thứ tự tăng dần sau khi đã loại bỏ các phần tử trùng nhau.

🕮 Bài tập 5.26

Viết chương trình thực hiện các yêu cầu sau đây:
- Nhập vào số phần tử của mảng, kiểm soát số phần tử của mảng lớn hơn 0 và bé hơn 100.
- Nhập vào từng phần tử của mảng, kiểm soát giá trị mảng nhập vào có giá trị trong phạm vi [-10, 10].
- Liệt kê các số chẵn trong mảng.
- Tính tổng bình phương của các số âm trong mảng.
- Liệt kê các số hoàn hảo trong mảng. Số hoàn hảo là số có tổng các ước số (trừ chính nó) bằng chính số đó.
- Nhập vào số x, đếm và in ra màn hình có bao nhiêu có bao nhiêu số x trong mảng đã cho.
- Sắp xếp mảng tăng dần bằng thuật toán Quick_sort.

Yêu cầu:

Thể hiện mỗi ý của bài bằng một hàm.

🕮 Bài tập 5.27

Viết chương trình thực hiện các yêu cầu sau đây:
- Viết chương trình nhập vào một mảng gồm n số tự nhiên.

- Hãy xuất ra màn hình:

+ Dòng 1: các phần tử của mảng.

+ Dòng 2: gồm các số lẻ, tổng cộng có bao nhiêu số lẻ.

+ Dòng 3: gồm các số chẵn, tổng cộng có bao nhiêu số chẵn.

+ Dòng 4: gồm các số nguyên tố, có tất cả bao nhiêu số nguyên tố trong mảng.

+ Dòng 5: gồm các số không phải là số nguyên tố.

Yêu cầu:

Xây dựng chương trình với mỗi ý được thể hiện bằng một hàm.

🕮 Bài tập 5.28

Viết chương trình thực hiện các yêu cầu sau đây:

- Viết chương trình nhập vào một dãy các số theo thứ tự tăng, nếu nhập sai quy cách thì yêu cầu nhập lại.
- In dãy số sau khi đã nhập xong.
- Nhập thêm một số mới và chèn số đó vào dãy đã có sao cho dãy vẫn đảm bảo thứ tự tăng. In lại dãy số để kiểm tra.

Yêu cầu:

Mỗi ý của bài toán được xây dựng bằng một hàm.

🕮 Bài tập 5.29

Viết chương trình thực hiện việc đảo mảng một chiều. Chẳng hạn cho mảng sau: 1 2 3 4 5 7 9 10, kết quả sau khi đảo thành 10 9 7 5 4 3 2 1.

Yêu cầu:

Không dùng mảng phụ để lưu trữ kết quả.

🕮 Bài tập 5.30

Viết chương trình thực hiện các yêu cầu sau đây:

- Nhập vào số phần tử n của mảng. Kiểm soát giá trị nhập vào có giá trị trong phạm vi từ 0 đến 100.
- Sinh ngẫu nhiên các phần tử của mảng có giá trị từ 100 đến 1000.
- Hiển thị các phần tử của mảng ra màn hình.
- Chèn một phần tử có giá trị x vào vị trí k của mảng, x và k được nhập từ bàn phím và được kiểm soát giá trị ($100 \leq x \leq 1000$, $0 \leq k \leq n$).

Yêu cầu:

Với mỗi ý, thể hiện bằng một hàm. Xây dựng hàm main() để gọi các hàm vừa xây dựng.

🕮 Bài tập 5.31

Viết chương trình thực hiện các yêu cầu sau đây:

- Sinh ngẫu nhiên các giá trị của mảng gồm n phần tử, n là số nguyên dương được nhập từ bàn phím, các phần tử của mảng có giá trị từ 0 đến 100.

- Liệt kê các số lẻ trong mảng, đồng thời tính tổng của các số lẻ, in các kết quả ra màn hình.

- Xóa phần tử thứ k trong mảng, k được nhập từ bàn phím $\mathbf{0} \leq \boldsymbol{k} \leq \boldsymbol{n}$. Hiển thị mảng sau khi xóa ra màn hình.

🕮 Bài tập 5.32

Viết chương trình thực hiện việc trộn hai dãy có thứ tự thành một dãy có thứ tự.

Yêu cầu:

Không được trộn chung rồi mới sắp thứ tự. Khi trộn phải tận dụng được tính chất đã sắp của hai dãy con.

🕮 Bài tập 5.33

Viết chương trình thực hiện các yêu cầu sau đây:

- Nhập giá trị của một ma trận (mảng hai chiều) các số nguyên, bao gồm m hàng, n cột (với m, n được nhập từ bàn phím).

- In ma trận đó lên màn hình theo định dạng gồm m hàng và mỗi hàng gồm n phần tử.

- Nhập từ bàn phím một số nguyên x, cho biết có những phần tử nào của ma trận trùng với x không? Cho biết số lượng phần tử trong ma trận có giá trị bằng x và vị trí của các phần tử này trong ma trận.

- Tìm phần tử lớn nhất ở hàng 1 của ma trậnvà phần tử nhỏ nhất nằm ở cột n. In ra màn hình các giá trị này.

- In các phần tử trên đường chéo chính ra màn hình, tính tổng các phần tử trên đường chéo chính.
- Sắp xếp mảng tăng dần, in kết quả sau sắp xếp ra màn hình.

🕮 Bài tập 5.34

Viết chương trình để chuyển đổi vị trí từ dòng thành cột của một ma trận (ma trận chuyển vị) vuông 4 hàng 4 cột.

Sau đó viết cho ma trận tổng quát cấp m * n.

🕮 Bài tập 5.35

Viết chương trình nhập vào hai ma trận A và B có cấp m, n. In hai ma trận lên màn hình. Tổng hai ma trận A và B là ma trận C được tính bởi công thức:

$c_{ij} = a_{ij} + b_{ij}$ với i = 0, 1, 2, …, m - 1 và j = 0, 1, 2,…, n - 1;

Tính ma trận tổng C và in kết quả lên màn hình.

🕮 Bài tập 5.36

Viết chương trình thực hiện các yêu cầu sau đây:

- Nhập vào ma trận A cấp m × k, ma trận B cấp k × n, với m, n, k được nhập từ bàn phím và kiểm soát giá trị trong phạm vi từ 1 đến 10.

- Tích của ma trận A với ma trận B được xác định như sau:

$c_{ij} = a_{i1} * b_{1j} + a_{i2} * b_{2j} + a_{i3} * b_{3j} + \cdots + a_{ik} * b_{kj}$ với i = 0, 1, 2, …, m - 1 và j = 0, 1, 2,…, n - 1;

- Tính ma trận tích C và in kết quả lên màn hình.

🕮 Bài tập 5.37

Viết chương trình thực hiện các yêu cầu sau đây:

- Nhập vào số hàng m và số cột n, với m, n là các số nguyên dương nhỏ hơn 10.

- Sinh ngẫu nhiên các phần tử của ma trận A cấp m × n.

- Hiển thị các phần tử của ma trận A gồm m hàng, mỗi hàng gồm n phần tử.

- Tìm phần tử lớn nhất của mảng, cho biết vị trí của phần tử này?

- Sắp xếp lại ma trận tăng dần, in các phần tử trên cùng một hàng.

🕮 Bài tập 5.38

Xét ma trận A vuông cấp n, các phần tử a[i, i] (i = 1, 2, 3,..., n) được gọi là đường chéo chính của ma trận vuông A. Ma trận vuông A được gọi là ma trận tam giác nếu tất cả các phần tử dưới đường chéo chính đều bằng 0. Định thức của ma trận tam giác bằng tích các phần tử trên đường chéo chính. Hãy biến đổi ma trận A về ma trận tam giác. In kết quả từng bước lên màn hình.

🕮Bài tập 5.39

Thực hiện các bài tập ở phần mảng từ bài 5.23 đến bài tập 5.39 bằng cách sử dụng con trỏ.

🕮 Bài tập 5.40

Viết chương trình thực hiện các yêu cầu sau đây:

- Nhập vào một chuỗi ký tự. In chuỗi ra màn hình.
- Đếm có bao nhiêu khoảng trắng trong chuỗi.
- Loại bỏ các khoảng trắng thừa trong chuỗi (Định nghĩa khoảng trắng thừa nếu có từ hai khoảng cách liên tiếp trở lên). In chuỗi sau khi loại bỏ các khoảng trắng thừa ra màn hình.

🕮 Bài tập 5.41

Viết chương trình nhập vào hai chuỗi s1 và s2, nối chuỗi s2 vào s1. Xuất chuỗi s1 ra màn hình.

Yêu cầu:

- Xây dựng hàm nhập chuỗi.

- Xây dựng hàm nối chuỗi s2 vào s1.

- Xây dựng hàm hiển thị chuỗi.

🕮 Bài tập 5.42

Viết chương trình thực hiện các yêu cầu sau đây:

- Nhập vào một chuỗi ký tự, in chuỗi ký tự ra màn hình.
- Viết chương trình đổi tất cả các kí tự có trong chuỗi thành chữ thường (không dùng hàm strlwr).
- Đổi tất cả các kí tự trong chuỗi sang chữ in hoa (không dùng hàm struppr).
- Đổi những kí tự đầu tiên của mỗi từ thành chữ in hoa.
- Chữ xen kẻ 1 chữ hoa và 1 chữ thường. Chẳng hạn, nhập ABCDEfgh đổi thành AbCdEfGh.
- Đảo ngược các kí tự trong chuỗi. Chẳng hạn, nhập ABCDE, xuất ra màn hình là: EDCBA.

Yêu cầu:

Mỗi ý trong bài được thể hiện bằng một hàm.

🕮 Bài tập 5.43

Viết chương trình thực hiện các yêu cầu sau đây:

- Nhập vào một chuỗi ký tự s, in chuỗi ký tự ra màn hình.
- Nhập một ký tự ch, tìm kiếm và trả lời ký tự ch có trong chuỗi không, nếu có xuất ra vị trí của từ chứa kí tự đó. Chẳng hạn, chuỗi s là "dai hoc cong nghe gtvt", nếu nhập ký tự 'c', vị trí được in ra màn hình là 3 (từ thứ 3 của chuỗi s).

- Nhập vào ký tự ch, đếm số lần xuất hiện của ký tự ch trong chuỗi s. Chẳng hạn, chuỗi s là “dai hoc cong nghe gtvt”, ký tự được nhập vào là o, kết quả trả về là 2.

Yêu cầu:

Thể hiện mỗi ý trong bài bằng một hàm, xây dựng hàm main() để gọi các kết quả đạt được.

🕮 Bài tập 5.44

Viết chương trình thực hiện các yêu cầu sau đây:

- Nhập vào chuỗi s1 và s2, hiển thị chuỗi s1 và s2 ra màn hình.

- Cho biết vị trí xuất hiện của chuỗi s2 trong s1.

Yêu cầu:

- Xây dựng hàm nhập chuỗi ký tự.

- Xây dựng hàm cho biết vị trí xuất hiện của chuỗi s2 trong s1.

🕮 Bài tập 5.45

Viết chương trình thực hiện các yêu cầu sau đây:

- Nhập vào một chuỗi ho_ten (chứa cả họ và tên), in chuỗi ra màn hình.

- Nhập vào một từ là tên của một người, thực hiện tìm kiếm tên trong chuỗi họ tên. Nếu có tên này trong chuỗi ho_ten thì thì xuất ra màn hình “Ten cua nguoi nay da nhap!”, ngược lại thông báo “Da nhap sai!”.

🕮 Bài tập 5.46

Viết chương trình đổi chữ xen kẻ 1 chữ hoa và 1 chữ thường.

Test: nhập ABCDEfgh đổi thành AbCdEfGh.

🕮 Bài tập 5.47

Viết chương trình đảo ngược các ký tự trong chuỗi.

Test: nhập ABCDE, xuất ra màn hình là: EDCBA.

Yêu cầu:

- Xây dựng hàm nhập chuỗi.

- Xây dựng hàm nghịch đảo các ký tự trong chuỗi.

- Xây dựng hàm hiển thị chuỗi.

🕮 Bài tập 5.48

a. Viết chương trình nhập vào một chuỗi ký tự S.

b. Nhập vào một ký tự, thông báo có hay không có ký tự đó trong chuỗi S, nếu có đưa ra vị trí của ký tự đó.

Test: Chuỗi ABCDEF, ký tự nhập vào D.

Thông báo " Có ký tự D, ở vị trí 3".

c. Đếm xem có bao nhiêu ký tự vừa nhập trong chuỗi S.

🕮 Bài tập 5.49

Viết chương trình nhập vào một chuỗi ký tự.

Đảo vị trí của từ đầu và từ cuối. In chuỗi kết quả thu được.

Test: Chuỗi " bo an co", chuỗi thu được sau khi đảo vị trí " co an bo".

Yêu cầu:

- Xây dựng hàm nhập vào chuỗi.
- Xây dựng hàm đảo vị trí của từ đầu và từ cuối.
- Xây dựng hàm hiển thị chuỗi.

🕮 Bài tập 5.50

Viết chương trình nhập vào một chuỗi ký tự.

Nhập vào từ bắt đầu tách, in màn hình chuỗi thu được cắt ra bắt đầu từ từ vừa nhập.

Test: Nhập chuỗi S1: "Khoa Công Nghệ Thông tin".

Người nhập muốn tách bắt đầu từ chữ "Công" thì sẽ đưa ra màn hình chuỗi "Công Nghệ Thông Tin".

🕮 Bài tập 5.51

Viết chương trình nhập vào một chuỗi ký tự.

Nếu trong chuỗi có ký tự viết hoa thì được chuyển sang ký tự viết thường và những ký tự viết thường được chuyển sang ký tự viết hoa.

Test: Chuỗi "nGuYen vAN a" được đổi thành: "Nguyễn Văn A".

🕮 Bài tập 5.52(*)

Viết chương trình nhập vào một chuỗi đếm xem chuỗi có bao nhiêu từ. Các từ cách nhau bằng khoảng trắng, dấu chấm câu: dấu chấm (.), dấu phẩy (,), dấu chấm phẩy (;), dấu hỏi (?) và dấu chấm than (!).

CHƯƠNG 6

KIỂU CẤU TRÚC

Nội dung của Chương 6, trình bày:

- Định nghĩa kiểu dữ liệu do người dùng định nghĩa, cho phép nhiều thành phần dữ liệu khác nhau được chứa đựng trong một cấu trúc;

- Khai báo và khởi tạo kiểu cấu trúc;

- Cách thức truy cập đến các thành phần của kiểu cấu trúc;

- Mối quan hệ giữa kiểu cấu trúc và con trỏ;

- Định nghĩa mảng cấu trúc, cách cấp phát bộ nhớ động cho mảng cấu trúc.

6.1. ĐỊNH NGHĨA

Cấu trúc là kiểu dữ liệu do người dùng định nghĩa cho phép kết hợp nhiều kiểu dữ liệu khác nhau. Kiểu cấu trúc giúp xây dựng một kiểu dữ liệu phức tạp có ý nghĩa hơn. Nó có phần giống với mảng, nhưng mảng chỉ chứa dữ liệu cùng loại, trong khi, cấu trúc có thể lưu trữ dữ liệu thuộc bất kỳ loại nào, điều này thực tế hữu ích hơn.

Chẳng hạn, nếu chúng ta phải viết chương trình lưu trữ thông tin sinh viên, trong đó thông tin sinh viên bao gồm tên sinh viên (chuỗi ký tự, có độ dài tối đa 20 ký tự), tuổi (số nguyên), giới tính (kiểu logic, hoặc số nguyên nhận hai giá trị 0, 1), địa chỉ (chuỗi ký tự), v.v. chúng ta có thể sử dụng mảng cho bài toán này, với mỗi phần tử của mảng có kiểu dữ liệu có thể lưu giữ các kiểu dữ liệu khác nhau (chuỗi ký tự, số nguyên, logic, v.v.).

Trong cấu trúc, dữ liệu được lưu trữ dưới dạng bản ghi.

6.1.1. Khai báo kiểu cấu trúc

Từ khóa struct được sử dụng để xác định một cấu trúc, struct định nghĩa một kiểu dữ liệu mới là tập hợp của nhiều kiểu dữ liệu khác nhau.

Cú pháp:

```
struct  [struct_name_tag]
{
Data_type member_1;
```

Data_type member_2;

…

Data_type member_n;

};

Trong đó:

struct_name: tên kiểu cấu trúc;

member_1, member_2,…, member_ n: các thành phần, hay còn gọi là thành viên của kiểu cấu trúc. Các thành phần của một kiểu cấu trúc có kiểu dữ liệu khác nhau.

Data_type: Kiểu dữ liệu của các thành phần, có thể là các kiểu dữ liệu cơ bản như kiểu số nguyên, số thực, ký tự, chuỗi, cũng có thể là kiểu dữ liệu có cấu trúc.

Trong cú pháp khai báo ở trên, chúng ta bắt đầu với từ khóa struct, sau đó là tên cấu trúc (nên đặt tên), tiếp theo bên trong dấu ngoặc nhọn, khai báo các biến thành viên, không có gì ngoài các biến ngôn ngữ C bình thường thuộc các loại khác nhau như int, float, mảng, v.v. Sau khi đóng ngoặc nhọn, chúng ta có thể chỉ định một hoặc nhiều biến cấu trúc.

Lưu ý:

Dấu ngoặc nhọn đóng trong khai báo kiểu cấu trúc phải được theo sau bởi dấu;

🕮 Ví dụ 6.1

Khai báo kiểu cấu trúc Nhan_vien, bao gồm các thông tin họ tên, địa chỉ và lương như sau:

```
1  struct Nhan_vien
2  {
3  char ho_ten[20];
4  char dia_chi[50];
5  float luong;
6  };
```

🕮 Ví dụ 6.2

Khai báo kiểu cấu trúc Sinh_vien để lưu trữ các thông tin của một sinh viên bao gồm bốn trường dữ liệu: Tên sinh viên, tuổi, địa chỉ và giới tính. Các trường này được gọi là các thành phần hoặc thành viên của kiểu cấu trúc Sinh_vien. Mỗi thành phần này có kiểu dữ liệu khác nhau, trong

đó tên là một chuỗi các ký tự, có độ dài tối đa 20 ký tự, tuổi thuộc kiểu nguyên int, địa chỉ là một chuỗi các ký tự có độ dài tối đa 50 ký tự và giới tính nhận một trong hai giá 0, 1, (hoặc F (female), M (male)). Sinh_vien là tên của cấu trúc và được gọi là thẻ cấu trúc.

```
1   struct Sinh_vien
2   {
3   char ten[20];
    int tuoi;
4   char dia_chi[50];
5   char gioitinh; // nhan gia tri F (female) hoặc M (male)
6   };
```

Khi một cấu trúc được xác định, nó tạo ra một kiểu do người dùng định nghĩa (lúc này nó được sử dụng giống như các kiểu dữ liệu cơ bản đã biết như số nguyên, số thực, ký tự, v.v.). Tuy nhiên, không có không gian lưu trữ hoặc bộ nhớ được phân bổ. Để phân bổ bộ nhớ của một loại cấu trúc nhất định và làm việc với nó, chúng ta cần tạo các biến.

Có thể khai báo các biến của một cấu trúc, cùng với định nghĩa cấu trúc hoặc sau khi cấu trúc được xác định. Khai báo biến cấu trúc tương tự như khai báo của bất kỳ biến thông thường nào của bất kỳ kiểu dữ liệu nào khác. Các biến cấu trúc có thể được khai báo theo hai cách sau:

🕮 Ví dụ 6.3

Khai báo các biến có kiểu Nhan_vien

```
1   struct Nhan_vien
2           {
3                   char ho_ten[20];
4                   char dia_chi[50];
5                   float luong;
6           };
7           int main()
8           {
9           // khai báo 2 biến nv1, nv2 có kiểu dữ liệu Nhan_vien
10          // khai báo mảng nv có tối đa 20 phần tử, mỗi phần tử có
    kiểu Nhan_vien
11                  struct Nhan_vien nv1, nv2, nv[20];
12                  return 0;
13          }
```

Có thể khai báo như sau:

```
1       struct Nhan_vien
2       {
3       char ho_ten[20];
4       char dia_chi[50];
5       float luong;
6       } nv1, nv2, nv[20];
```

Cú pháp khai báo biến kiểu cấu trúc:

Cách 1: Khai báo Kiểu cấu trúc, sau đó khai báo biến cấu trúc

```
//khai báo kiểu cấu trúc
struct [struct_name_tag]
{
  data_type member;
};
//Khai báo biến cấu trúc
 struct struct_name_tag variable_struct;
```

Trong đó:

variable_struct: danh sách các biến kiểu cấu trúc.

Cách 2: Khai báo danh sách biến cấu trúc cùng với khai báo kiểu cấu trúc

```
struct [struct_name_tag]
{
   data_type member;
} variable_struct;
```

Trong đó:

variable_struct: Danh sách các biến cấu trúc

6.1.2. Truy cập đến các thành phần của kiểu cấu trúc

Các thành phần của cấu trúc có thể được truy cập và gán giá trị. Để truy cập đến thành phần của cấu trúc, ta sử dụng toán tử (.)

Cú pháp:

variable_struct.member;

Trong đó:

variable_struct: Tên biến cấu trúc;

member: Thành phần cần truy cập;

Chẳng hạn nv1.ho_ten, nv2.luong, nv[15].diachi;

Khi đã truy xuất được tới các thành phần của cấu trúc thì mỗi thành phần đó là một biến bình thường và ta gán giá trị hoặc nhập xuất giá trị cho chúng như cách bình thường mà chúng ta vẫn làm.

🕮 Ví dụ 6.4

✍ Xét chương trình sau đây:

```
1   #include<stdio.h>
2   #include<string.h>
3   struct Sinh_vien
4   {
5      char ten_sv[20];
6      int tuoi_sv;
7      char dia_Chi[50];
8      //F (female) va M (male)
9      char gioi_tinh;
10  };
11  int main()
12  {
13     struct Sinh_vien s1;
14     /*
15      s1 la bien kieu cau truc Sinh_vien va
16            tuoi la thanh phan cua Sinh_vien
17     */
18     s1.tuoi_sv = 18;
19     /*
20        su dung ham strcpy de them ten sinh vien
21     */
22     strcpy(s1.ten_sv, "Kien");
23     /*
24        Hien thi gia tri
```

```
    */
    printf("Ten cua sinh vien: %s\n", s1.ten_sv);
    printf("Tuoi cua sinh vien: %d\n", s1.tuoi_sv);
    return 0;
}
```

✍ Kết quả chương trình:

```
Ten cua sinh vien: Kien
Tuoi cua sinh vien: 18
```

Chúng ta có thể dụng lệnh scanf() để nhập giá trị cho các thành phần của cấu trúc:

scanf(“%s”, s1.ten_sv);

scanf(“%d”, &s1.tuoi_sv);

🕮 Ví dụ 6.5

```
#include<stdio.h>
#include<string.h>
struct Sinh_vien
{
    char ten_sv[20];
    int tuoi_sv;
    char dia_Chi[50];
    //F (female) va M (male)
    char gioi_tinh;
};
int main()
{
    struct Sinh_vien s1;
    printf("Nhap ten sinh vien s1: \n");
        scanf("%s", s1.ten_sv);
    printf("Nhap tuoi sinh vien s1: \n");
        scanf("%d", &s1.tuoi_sv);
    printf("Ten cua sinh vien: %s\n", s1.ten_sv);
    printf("Tuoi cua sinh vien: %d\n", s1.tuoi_sv);
    return 0;
}
```

✍ **Kết quả chương trình:**

```
Nhap ten sinh vien s1:
Nam
Nhap tuoi sinh vien s1:
19
Ten cua sinh vien: Nam
Tuoi cua sinh vien: 19
```

Lưu ý: Nếu sử dụng lệnh scanf("%s", s1.ten_sv); khi nhập vào "Le Thanh Nam", khi sử dụng lệnh in ra, trên màn hình chỉ ghi nhận "Le". Để xử lý vấn đề này, chúng ta sử dụng câu lệnh:

```
fflush(stdin);
gets(s1.ten_sv);
```

Hàm fflush() trong thư viện stdio.h cũng có tác dụng cho phép lựa chọn xóa bộ nhớ đệm cho stream nào. Ở đây ta truyền vào stdin để xóa bộ đệm cho dòng nhập chuẩn, tức là bàn phím.

🕮 **Ví dụ 6.6**

Khai báo kiểu Nhan_Vien bao gồm các thông tin: Mã nhân viên, họ tên, lương. Khai báo 2 đối tượng nv1, nv2 thuộc kiểu Nhan_Vien, nhập và hiển thị thông tin của hai nhân viên nv1, nv2.

✍ Chương trình

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   // khai bao struct Nhan_Vien
4   struct Nhan_Vien
5   {
6      char manv[20]; // ma nhan vien
7      char hoten[30]; // ho ten nhan vien
8      double luong; // luong nhan vien
9   };
10  int main()
11  {
12  // khai bao 2 bien nv1, nv2
13     struct Nhan_Vien nv1, nv2
14  //Nhap thong tin cua nhan vien thu nhat
```

```
15      printf("Nhap thong tin cho nv1:\n");
16      printf("Ma nhan vien: ");
17            fflush(stdin);
18      gets(nv1.manv);
19      printf("Ho ten nhan vien: ");
20            fflush(stdin);
21      gets(nv1.hoten);
22      printf("Nhap luong cua nhan vien: ");
23            fflush(stdin);
24      scanf("%lf", &nv1.luong);
25    //Nhap thong tin cua nhan vien thu hai
26      printf("Nhap thong tin cho nv2:\n");
27      printf("Ma nhan vien: ");
28            fflush(stdin);
29      gets(nv2.manv);
30      printf("Ho ten nhan vien: ");
31            fflush(stdin);
32      gets(nv2.hoten);
33      printf("Nhap luong cua nhan vien: ");
34            fflush(stdin);
35      scanf("%lf", &nv2.luong);
36    //Hien thi thong tin cua cac nhan vien
37      printf("\n ------ Hien thi thong tin nhan vien: ----\n");
38      printf("%-20s %-30s %-7s\n", "Ma NV", "Ho ten", "Luong");
39      printf("%-20s %-30s %-7.2lf\n", nv1.manv, nv1.hoten,
      nv1.luong);
40      printf("%-20s %-30s %-7.2lf \n", nv2.manv, nv2.hoten,
      nv2.luong);
41      return 0;
42    }
```

Kết quả chương trình:

```
Nhap thong tin cho nv1:
Ma nhan vien: nv01
Ho ten nhan vien: Le Kien
Nhap luong cua nhan vien: 7000000
```

```
Nhap thong tin cho nv2:
Ma nhan vien: nv02
Ho ten nhan vien: Le Luan
Nhap luong cua nhan vien: 9000000
--------- Hien thi thong tin nhan vien: -----
Ma NV        Ho ten          Luong
nv01         Le Kien         7000000.00
nv02         Le Luan         9000000.00
```

6.1.3. Khởi tạo kiểu cấu trúc

Giống như biến của bất kỳ kiểu dữ liệu nào khác, biến cấu trúc có thể được khởi tạo tại thời điểm biên dịch.

🕮 Ví dụ 6.7

Cho khai báo kiểu cấu trúc Sinh_vien và khai báo 1 biến p1 kiểu Sinh_vien được khởi tạo giá trị sau đây:

```
1  struct Sinh_vien
2  {
3  char ten[20];
   int tuoi;
4  char dia_chi[50];
5  char gioitinh; // nhan gia tri F (female) hoặc M (male)
6  };
7  struct Sinh_vien p1 ={“Nam”,19, “Ha Noi”, ‘F’};
```

Hoặc khởi tạo như sau:

```
1  struct Sinh_vien
2  {
3  char ten[20];
   int tuoi;
4  char dia_chi[50];
5  char gioitinh; // nhan gia tri F (female) hoặc M (male)
6  };
7  struct Sinh_vien p1 ={“Nam”,19, “Ha Noi”, ‘F’};
8  p1.ten = “Nam”;
9  p1.tuoi =19;
10 p1.dia_chi = “Ha Noi”;
11 P1.gioitinh = ‘F’;
```

🕮 Ví dụ 6.8

Xem xét chương trình sau đây:

```
1   #include<stdio.h>
2   #include<string.h>
3   struct Sinh_vien
4   {
5      char ten_sv[20];
6      int tuoi_sv;
7      char dia_chi[50];
8      //F (female) va M (male)
9      char gioi_tinh;
10  };
11  int main()
12  {
13     struct Sinh_vien s1 ={"Nam",19, "Ha Noi", 'F'};
14     printf("Ten cua sinh vien: %s\n", s1.ten_sv);
15     printf("Tuoi cua sinh vien: %d\n", s1.tuoi_sv);
16     printf("Dia chi cua sinh vien: %s\n", s1.dia_chi);
17     printf("Gioi tinh cua sinh vien: %c\n", s1.gioi_tinh);
18     return 0;
19  }
```

✍ Kết quả chương trình:

```
Ten cua sinh vien: Nam
Tuoi cua sinh vien: 19
Dia chi cua sinh vien: Ha Noi
Gioi tinh cua sinh vien: F
```

6.1.4. Khai báo với từ khóa typedef

typedef là một từ khóa được sử dụng trong ngôn ngữ C để gán tên thay thế cho các kiểu dữ liệu hiện có. Nó chủ yếu được sử dụng với các kiểu dữ liệu do người dùng định nghĩa, khi tên của các kiểu dữ liệu trở nên hơi phức tạp để sử dụng trong các chương trình.

Sau đây là cú pháp chung để sử dụng typedef:

typedef <existing_name> <alias_name>

Trong đó:

existing_name: Kiểu_dữ_liệu_hiện_Có;

alias_name: Tên_thay_thế;

Có thể sử dụng từ khóa typedef với cấu trúc để định nghĩa một kiểu dữ liệu mới và sau đó sử dụng kiểu dữ liệu đó để định nghĩa các biến cấu trúc một cách trực tiếp như sau:

```
typedef struct [struct_name_tag]
   {
   data_type member_1;
   data_type member_2;
   …
   data_type member_n;
}type_name;
 type_name struct_variable_name;
```

Trong đó:

struct_varible_name: danh sách các biến cấu trúc;

🕮 Ví dụ 6.9

Xem xét ví dụ sau đây:

```
1  #include<stdio.h>
2  #include<string.h>
3  typedef struct Sinh_vien
4  {
5     char ten_sv[20];
6     int tuoi_sv;
7     char dia_Chi[50];
8     //F (female) va M (male)
9     char gioi_tinh;
10 }SV;
11 int main()
12 {
13     SV s1;
14    // Khoi tao gia tri cua bien s1, la bien cau truc thay the duoc su
   dung trong chuong trinh
15          strcpy(s1.ten_sv,"Le Thi Chi");
```

```
16          s1.tuoi_sv = 18;
17      /*
18         su dung ham strcpy de them ten sinh vien
19      */
20      strcpy(s1.dia_Chi, "Ha Noi");
21      s1.gioi_tinh='M';
22      /*
23         Hien thi gia tri
24      */
25      printf("Ten cua sinh vien: %s\n", s1.ten_sv);
26      printf("Tuoi cua sinh vien: %d\n", s1.tuoi_sv);
27      printf("Dia chi cua sinh vien: %s\n", s1.dia_Chi);
28      printf("Gioi tinh cua sinh vien: %c\n", s1.gioi_tinh);
29      return 0;
30  }
```

✍ Kết quả chương trình:

```
Ten cua sinh vien: Le Thi Chi
Tuoi cua sinh vien: 18
Dia chi cua sinh vien: Ha Noi
Gioi tinh cua sinh vien: M
```

6.1.5. Phép gán các biến cùng có kiểu cấu trúc

Việc thực hiện gán các biến cùng kiểu cấu trúc được thực hiện giống như phép gán hai biến cùng kiểu dữ liệu.

🕮 Ví dụ 6.10

```
1   struct Sinh_vien
2   {
3   char ten[20];
    int tuoi;
4   char dia_chi[50];
5   char gioitinh; // nhan gia tri F (female) hoặc M (male)
6   };
7   struct Sinh_vien s1 = {"Nam",19, "Ha Noi", 'F'};
8   struct s2;
9   s2 = s1;
```

Sau lệnh gán s2 = s1; mọi thông tin về Sinh_vien s1 được gán cho s2.

🕮 Ví dụ 6.11

✍ Xem xét chương trình sau đây:

```
1   #include<stdio.h>
2   #include<string.h>
3   struct Sinh_vien
4   {
5       char ten_sv[20];
6       int tuoi_sv;
7       char dia_chi[50];
8       //F (female) va M (male)
9       char gioi_tinh;
10  };
11  int main()
12  {
13      struct Sinh_vien s1 ={"Nam",19, "Ha Noi", 'F'};
14      struct Sinh_vien s2;
15      s2 = s1;
16      printf("Ten cua sinh vien thu 2: %s\n", s1.ten_sv);
17      printf("Tuoi cua sinh vien thu 2: %d\n", s1.tuoi_sv);
18      printf("Dia chi cua sinh vien thu 2: %s\n", s1.dia_chi);
19      printf("Gioi tinh cua sinh vien thu 2: %c\n", s1.gioi_tinh);
20      return 0;
21  }
```

✍ Kết quả chương trình:

```
Ten cua sinh vien thu 2: Nam
Tuoi cua sinh vien thu 2: 19
Dia chi cua sinh vien thu 2: Ha Noi
Gioi tinh cua sinh vien thu 2: F
```

6.1.6. Cấu trúc lồng nhau

Ngôn ngữ lập trình C cho phép một cấu trúc lồng trong một cấu trúc khác. Khi đó một cấu trúc có một cấu trúc khác là một biến thành viên.

Có hai cách để định nghĩa cấu trúc lồng nhau trong C:

6.1.6.1. Định nghĩa cấu trúc riêng biệt

Khai báo hai cấu trúc, cấu trúc phụ thuộc được sử dụng bên trong cấu trúc chính như là thành viên.

🕮 Ví dụ 6.12

✍ Xem xét đoạn khai báo sau đây:

```
1   struct NgaySinh
2   {
3     int ngay;
4     int thang;
5     int nam;
6   };
7   struct Nhan_vien
8   {
9     int id_nv;
10    char ten_nv[20];
11    struct NgaySinh ns;
12  }nv1, nv2;
```

Trong ví dụ trên, ns là biến kiểu NgaySinh, ns được sử dụng như một thành phần trong cấu trúc Nhan_vien. Theo cách khai báo trên, cấu trúc NgaySinh có thể được sử dụng trong nhiều cấu trúc khác nhau.

6.1.6.2. Định nghĩa cấu trúc nhúng

Cấu trúc nhúng cho phép khai báo cấu trúc bên trong cấu trúc. Do đó, nó đòi hỏi ít dòng mã hơn, nhưng nó không thể được sử dụng trong nhiều cấu trúc.

🕮 Ví dụ 6.13

✍ Xem xét đoạn khai báo sau đây:

```
1   struct Nhan_vien
2   {
3     int id_nv;
4     char ten_nv[20];
```

```
    struct NgaySinh
{
    int ngay;
    int thang;
    int nam;
}ns;
}nv1, nv2;
```

6.1.6.3. Truy cập đến thành phần của cấu trúc lồng

Để truy cập đến một thành phần của cấu trúc lồng nhau, ta sử dụng toán tử dấu chấm (.)

🕮 Ví dụ 6.14

```
nv1.ns.ngay;
nv1.ns.thang;
nv1.ns.nam;
```

6.1.6.4. Minh họa cấu trúc lồng

🕮 Ví dụ 6.15

✍ Xem xét đoạn chương trình sau đây:

```
#include <stdio.h>
#include <string.h>
struct Nhan_vien
{
   int id_nv;
   char ten_nv[20];
   struct Ngay_Sinh
    {
      int ngay;
      int thang;
      int nam;
    }ns;
}nv1;
int main( )
{
   //gan gia tri cho bien cau truc nv1
   nv1.id_nv=101;
```

```
18    strcpy(nv1.ten_nv, "Le Trung Kien");//su dung ham strcpy de sao chep
19    nv1.ns.ngay=30;
20    nv1.ns.thang=6;
21    nv1.ns.nam=1980;
22    //In thong tin cua nhan vien nv1
23    printf( "Ma nhan vien : %d\n", nv1.id_nv);
24    printf( "Ten nhan vien : %s\n", nv1.ten_nv);
25    printf( "Ngay sinh cua nhan vien (dd/mm/yyyy): %d/%d/%d\n", nv1.ns.ngay,nv1.ns.thang,nv1.ns.nam);
26    return 0;
27  }
```

✍ Kết quả chương trình:

Ma nhan vien : 101

Ten nhan vien : Le Trung Kien

Ngay sinh cua nhan vien (dd/mm/yyyy) : 30/6/1980

6.1.7. Cấu trúc được sử dụng như một tham số của hàm

Chúng ta có thể truyền cấu trúc như là một đối số hàm giống như truyền các biến, mảng trở thành các đối số trong một hàm.

🕮 Ví dụ 6.16

Xem xét chương trình sau đây, sử dụng cấu trúc Sinh_vien làm một tham số trong hàm:

```
1   #include<stdio.h>
2   struct Sinh_Vien
3   {
4      char ma_sv[10];
5      char ten_sv[20];
6      float diemtb;
7   };
8   void Hienthi(struct Sinh_Vien sv);
9   int main()
10  {
11     struct Sinh_Vien s;
12     printf("Nhap thong tin sinh vien:\n");
```

```
13      printf("Ma sinh vien:\n");
14      fflush(stdin);
15      gets(s.ma_sv);
16      printf("Ten sinh vien:\n");
17      fflush(stdin);
18      gets(s.ten_sv);
19      printf("Nhap diem cua sinh vien:\n");
20      scanf("%f", &s.diemtb);
21      Hienthi(s);
22      return 0;
23  }
24  void Hienthi(struct Sinh_Vien sv)
25  {
26   printf("\n%s %s %0.2f", sv.ma_sv,sv.ten_sv, sv.diemtb);
27  }
```

✍ Kết quả chương trình:

```
Nhap thong tin sinh vien:
Ma sinh vien:
sv01
Ten sinh vien:
Le Kien
Nhap diem cua sinh vien:
8.8
sv01 Le Kien 8.8
```

6.1.8. Cấu trúc và con trỏ

//Khai báo kiểu cấu trúc:

struct [struct_name_tag]

{

Data_type member;

};

Khai báo biến con trỏ kiểu cấu trúc:

Struct struct_name_tag * variable_struct_pointer;

Sử dụng toán tử → để truy cập đến các thành phần của cấu trúc thông qua biến con trỏ: variable_struct_pointer → member;

📖 Ví dụ 6.17

✍ Cho khai báo cấu trúc Sinh_vien sau đây:

```
1  struct Sinh_vien
2  {
3    char ma_sv[15];
4    char ten_sv[30];
5    float diem;
6  };
```

Khai báo con trỏ kiểu cấu trúc Sinh_vien như sau:

```
struct Sinh_vien *sv;
```

Truy cập đến thành phần ten_sv:

```
  sv->ten_sv;
```

📖 Ví dụ 6.18

✍ Xem xét chương trình sau đây:

```
1  #include <stdio.h>
2  // dinh nghia kieu cau truc
3    struct Sinh_vien {
4      char ma_sv[15];
5      char ten_sv[30];
6      float diem_tb;
7    };
8  int main()
9  {
10   // khai bao bien cau truc
11   struct Sinh_vien sv;
12   // bien con tro kieu Sinh_vien
13   struct Sinh_vien *p = NULL;
14   // con tro p tro toi sv
15   p = &sv;
16   // Nhap thong tin
17   printf("Nhap vao ma sinh vien: ");
18   scanf("%s", p->ma_sv);
19   printf("Nhap vao ten sinh vien: ");
```

```
    scanf("%s", p->ten_sv);
    printf("Nhap diem cua sinh vien: ");
    scanf("%f", &p->diem_tb);
      // Hien thi thong tin qua bien sv
    printf("\nHien thi ket qua qua bien sv:\n");
    printf("Ma_sv: %s\n", sv.ma_sv);
    printf("Ten sv: %s\n", sv.ten_sv);
    printf("Diem TB: %f\n", sv.diem_tb);
    // Hien thi thong tin qua bien con tro p
    printf("\nHien thi ket qua qua bien con tro p:\n");
    printf("Ma sv: %s\n", p->ma_sv);
    printf("Ten sv: %s\n", p->ten_sv);
    printf("Diem TB: %f\n", p->diem_tb);
    return 0;
}
```

✍ Kết quả chương trình:

```
Nhap vao ma sinh vien: sv01
Nhap vao ten sinh vien: Kien
Nhap diem cua sinh vien: 8.9
Hien thi ket qua qua bien sv:
Ma_sv: sv01
Ten sv: Kien
Diem TB: 8.900000
Hien thi ket qua qua bien con tro p:
Ma sv: sv01
Ten sv: Kien
Diem TB: 8.9
```

6.2. MẢNG CẤU TRÚC

Xem xét tình huống sau đây, chúng ta cần lưu trữ thông tin của ba sinh viên, thông tin của sinh viên bao gồm mã sinh viên, tên sinh viên, điểm của sinh viên. Với những gì đã học ở phần trước, chúng ta có thể đưa ra cấu trúc để lưu trữ như sau:

📖 Ví dụ 6.19

✍Xem xét đoạn chương trình sau đây:

```
1  #include<stdio.h>
2  #include <stdlib.h>
3  struct Sinh_vien
4  {
5     char ten_sv[20];
6     char ma_sv[10];
7     float diemtb;
8  };
9  int main()
10 {
11    struct Sinh_vien s1,s2,s3;
12    printf("Nhap vao ten cua sinh vien thu nhat:\n");
13    fflush(stdin);
14    gets(s1.ten_sv);
15    printf("Nhap vao ma cua sinh vien thu nhat:\n");
16    fflush(stdin);
17    gets(s1.ma_sv);
18    printf("Nhap vao diem cua sinh vien thu nhat:\n");
19    scanf("%f",&s1.diemtb);
20    printf("Nhap vao ten cua sinh vien thu hai:\n");
21    fflush(stdin);
22    gets(s2.ten_sv);
23    printf("Nhap vao ma cua sinh vien thu hai:\n");
24    fflush(stdin);
25    gets(s2.ma_sv);
26    printf("Nhap vao diem cua sinh vien thu hai:\n");
27    scanf("%f",&s2.diemtb);
28    printf("Nhap vao ten cua sinh vien thu ba:\n");
```

```
    fflush(stdin);
    gets(s3.ten_sv);
    printf("Nhap vao ma cua sinh vien thu ba:\n");
    fflush(stdin);
    gets(s3.ma_sv);
    printf("Nhap vao diem cua sinh vien thu ba:\n");
    scanf("%f",&s3.diemtb);
    printf("In thong tin chi tiet cua cac sinh vien....\n");
    printf("%s %s %0.2f\n",s1.ten_sv,s1.ma_sv,s1.diemtb);
    printf("%s %s %0.2f\n",s2.ten_sv,s2.ma_sv,s2.diemtb);
    printf("%s %s %0.2f\n",s3.ten_sv,s3.ma_sv,s3.diemtb);
return 0;
}
```

Kết quả chương trình:

```
Nhap vao ten cua sinh vien thu nhat:
Le Kien
Nhap vao ma cua sinh vien thu nhat:
sv01
Nhap vao diem cua sinh vien thu nhat:
8.8
Nhap vao ten cua sinh vien thu hai:
Le Chi
Nhap vao ma cua sinh vien thu hai:
sv02
Nhap vao diem cua sinh vien thu hai:
8.9
Nhap vao ten cua sinh vien thu ba:
Pham Thuan
Nhap vao ma cua sinh vien thu ba:
sv03
Nhap vao diem cua sinh vien thu ba:
9.2
```

```
In thong tin chi tiet cua cac sinh vien....
Le Kien sv01 8.8
Le Chi sv02 8.9
Pham Thuan sv03 9.2
```

Trong chương trình trên, chúng ta đã lưu trữ dữ liệu của ba sinh viên trong chương trình. Tuy nhiên, độ phức tạp của chương trình sẽ tăng lên nếu có hai mươi sinh viên. Trong trường hợp đó, chúng ta sẽ phải khai báo hai mươi biến cấu trúc khác nhau và lưu trữ từng biến một. Điều này sẽ gây khó khăn vì chúng ta sẽ phải khai báo một biến mỗi khi chúng ta thêm một sinh viên. Ghi nhớ tên của tất cả các biến cũng là một nhiệm vụ rất khó khăn. Bên cạnh đó, các câu lệnh được sử dụng để nhập cho mỗi sinh viên được lặp đi lặp lại, khiến cho số dòng lệnh của chương trình nhiều, không cần thiết.

C cho phép chúng ta khai báo một mảng các cấu trúc. Với cách làm này, chúng ta có thể tránh khai báo các biến cấu trúc khác nhau; giảm thiểu số lượng dòng mã code bị trùng lặp.

6.2.1. Định nghĩa mảng cấu trúc

Một mảng các cấu trúc trong C có thể được định nghĩa là tập hợp của nhiều biến cấu trúc trong đó mỗi biến chứa thông tin về các thực thể khác nhau. Mảng cấu trúc trong C được sử dụng để lưu trữ thông tin về nhiều thực thể thuộc các loại dữ liệu khác nhau. Mảng cấu trúc còn được gọi là tập hợp các cấu trúc.

Cú pháp:

```
//Khai báo kiểu cấu trúc:
struct [struct_name_tag]
{
data_type member;
};
Khai báo mảng cấu trúc
struct kiểu_cấu_trúc Tên_biến_mảng_cấu_trúc[số_phần_tử];
```

🕮 Ví dụ 6.20

Cho khai báo:

struct Sinh_vien SV[10];

Khai báo mảng SV gồm 10 phần tử, mỗi phần tử có kiểu cấu trúc Sinh_vien;

🕮 Ví dụ 6.21

✍ Xét chương trình sau đây:

```
1  #include<stdio.h>
2  #include <stdlib.h>
3  #define MAX 100
4  struct Sinh_vien
5  {
6     char ten_sv[20];
7     char ma_sv[10];
8     float diemtb;
9  };
10 int main()
11 {
12    struct Sinh_vien sv[MAX];
13    int n, i;
14    printf("Nhap vao so luong sinh vien:\n");
15    scanf("%d", &n);
16    //nhap thong tin cua cac sinh vien
17          printf("Nhap vao thong tin cua %d sinh vien:\n", n);
18    for(i = 0;i < n;i++)
19    {
20    printf("Nhap vao ten cua sinh vien thu %d:\n", i);
21          fflush(stdin);
22    gets(sv[i].ten_sv);
23          printf("Nhap vao ma cua sinh vien thu %d:\n",i);
24          fflush(stdin);
25          gets(sv[i].ma_sv);
26          printf("Nhap vao diem cua sinh vien thu %d:\n",i);
27          scanf("%f", &sv[i].diemtb);
28    }
29    // in thong tin cua cac sinh vien
30          printf("In thong tin chi tiet cua %d sinh vien:\n", n);
31    for(i = 0;i < n;i++)
32    {
33          printf("%s %s %f\n", sv[i].ten_sv, sv[i].ma_sv,
   sv[i].diemtb);
```

```
34      }
35  return 0;
36  }
```

Kết quả chương trình:

```
Nhap vao so luong sinh vien:
4
Nhap vao thong tin cua 4 sinh vien:
Nhap vao ten cua sinh vien thu 0:
Le Kien
Nhap vao ma cua sinh vien thu 0:
sv01
Nhap vao diem cua sinh vien thu 0:
8.8
Nhap vao ten cua sinh vien thu 1:
Pham Thuan
Nhap vao ma cua sinh vien thu 1:
sv02
Nhap vao diem cua sinh vien thu 1:
9.2
Nhap vao ten cua sinh vien thu 2:
Le Chi
Nhap vao ma cua sinh vien thu 2:
sv03
Nhap vao diem cua sinh vien thu 2:
8.7
Nhap vao ten cua sinh vien thu 3:
Le An
Nhap vao ma cua sinh vien thu 3:
sv04
Nhap vao diem cua sinh vien thu 3:
9.0
In thong tin chi tiet cua 4 sinh vien:
Le Kien sv01 8.8
Pham Thuan sv02 9.2
Le Chi sv03 8.7
Le An sv04 9.0
```

6.2.2. Cấp phát bộ nhớ động cho kiểu cấu trúc

Đôi khi, số lượng biến cấu trúc chúng ta khai báo có thể không đủ. Chúng ta có thể cần phân bổ bộ nhớ trong thời gian chạy. Đây là cách chúng ta có thể đạt được điều này trong ngôn ngữ lập trình C.

🕮 Ví dụ 6.22

✍Xem xét ví dụ sau đây:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  struct sinhvien {
4    char masv[10];
5    char ten_sv[30];
6    float diem_tb;
7  };
8  int main()
9  {
10   struct sinhvien *p;
11   int i, n;
12   printf("Nhap so sinh vien: ");
13   scanf("%d", &n);
14   // cap phat bo nho dong
15   p = (struct sinhvien*) malloc(sizeof(struct sinhvien));
16   for(i = 0; i < n; ++i)
17   {
18      printf("Nhap ma sinh vien thu %d:\n",i);
19      scanf("%s", &(p+i)->masv);
20      printf("Nhap ten sinh vien:\n");
21      scanf("%s", &(p+i)->ten_sv);
22            printf("Nhap diem sinh vien:\n");
23      scanf("%f", &(p+i)->diem_tb);
24   }
25   printf("Hien thi thong tin sinh vien:\n");
26   for(i = 0; i < n; ++i)
27      printf("MaSV: %s\tTenSV: %s\tDiemTB: %0.2f\n", (p+i)-
   >masv, (p+i)->ten_sv,(p+i)->diem_tb);
28   return 0;
29 }
```

✍ Kết quả chương trình:

```
Nhap so sinh vien: 3
Nhap ma sinh vien thu 0:
sv01
Nhap ten sinh vien:
An
Nhap diem sinh vien:
8.7
Nhap ma sinh vien thu 1:
sv02
Nhap ten sinh vien:
Binh
Nhap diem sinh vien:
7.9
Nhap ma sinh vien thu 2:
sv03
Nhap ten sinh vien:
Cuong
Nhap diem sinh vien:
9.2
Hien thi thong tin sinh vien:
MaSV: sv01      TenSV: An       DiemTB: 8.70
MaSV: sv02      TenSV: Binh     DiemTB: 7.90
MaSV: sv03      TenSV: Cuong    DiemTB: 9.20
```

6.2.3. Một số dạng bài tập với cấu trúc

🕮 Ví dụ 6.23

Khai báo kiểu struct và khởi tạo giá trị

Viết chương trình thực hiện các yêu cầu sau đây:

- Khai báo kiểu cấu trúc sinh viên, bao gồm các trường thông tin như: tên sinh viên, điểm môn toán, điểm môn tiếng anh và điểm môn vật lý.
- Khởi tạo bốn bản ghi.
- In thông tin của các bản ghi này ra màn hình.

✍Phác thảo lời giải

- Khai báo cấu trúc sinhvien.
- Khởi tạo mảng sv[] gồm bốn sinh viên, mỗi sinh viên có thông tin của tên sinh viên, điểm môn tiếng anh, điểm môn toán, điểm môn vật lý.
- Hiển thị thông tin của các sinh viên.

✍Chương trình:

```
1   #include <stdio.h>
2   #define N 4
3   struct sinhvien{
4       char ten_sv[20];
5       int d_anh;
6       int d_toan;
7       int d_ly;
8   };
9   struct sinhvien sv[]={
10      {"Cuong", 82, 72, 58},
11      {"Nam", 77, 82, 79},
12      {"Sinh", 52, 62, 39},
13      {"Thuy", 61, 82, 88}
14  };
15
16  int main()
17  {
18      int i;
19     for(i=0; i<N; i++){
20     printf("%9s: TiengAnh = %3d   Toan = %3d     VatLy =
    %3d\n", sv[i].ten_sv, sv[i].d_anh, sv[i].d_toan,
21    sv[i].d_ly);
22      }
23      return (0);
24  }
```

✍Kết quả chương trình khi thực hiện:

```
Cuong: TiengAnh =  82  Toan =  72   VatLy =  58
Nam:  TiengAnh =  77   Toan =  82   VatLy =  79
Sinh: TiengAnh =  52   Toan =  62   VatLy =  39
Thuy: TiengAnh =  61   Toan =  82   VatLy =  88
```

🕮 Ví dụ 6.24

Khai báo kiểu struct và khởi tạo giá trị

Viết chương trình thực hiện các yêu cầu sau đây:

- Khai báo kiểu cấu trúc sinh viên, bao gồm các trường thông tin như: tên sinh viên, điểm môn toán, điểm môn tiếng anh và điểm môn vật lý, trung bình môn.

- Khởi tạo 4 sinh viên, bao gồm các thông tin: tên sinh viên, điểm môn toán, điểm môn tiếng anh, điểm môn vật lý.

- Tính điểm trung bình cho các sinh viên. In thông tin của các sinh viên ra màn hình.

✍Phác thảo lời giải

- Khai báo cấu trúc sinhvien.

- Khởi tạo mảng sv[] gồm 4 sinh viên, mỗi sinh viên có thông tin của tên sinh viên, điểm môn tiếng anh, điểm môn toán, điểm môn vật lý.

- Sử dụng công thức điểm trung bình = (điểm môn tiếng anh + điểm môn toán + điểm môn vật lý)/3.

- Hiển thị thông tin của các sinh viên sau khi đã cập nhật điểm trung bình.

✍Chương trình

```
1   #include <stdio.h>
2   #define N 4
3   struct sinhvien{
4      char ten_sv[20];
5      int d_anh;
6      int d_toan;
7      int d_ly;
8      double d_tb;
9   };
10  static struct sinhvien sv[]={
11     {"Cuong", 82, 72, 58,0.0},
12     {"Nam", 77, 82, 79, 0.0},
13     {"Sinh", 52, 62, 39, 0.0},
14     {"Thuy", 61, 82, 88, 0.0}
15  };
16  int main()
17  {
18     int i;
19     //tinh diem trung binh cho moi sinh vien
20     for(i=0; i<N; i++)
21           {
```

```
22              sv[i].d_tb = (sv[i].d_anh + sv[i].d_toan +sv[i].d_ly)/3;
23      }
24      //hien thi thong tin cua tung sinh vien
25      for(i=0; i<N; i++){
26          printf("%8s: TiengAnh = %3d  Toan = %3d   VatLy = %3d
        DiemTB  =  %5.1f\n",   sv[i].ten_sv,  sv[i].d_anh,  sv[i].d_toan,
        sv[i].d_ly, sv[i].d_tb);
27      }
28      return (0);
29  }
```

✍ Kết quả chương trình:

```
Cuong: TiengAnh =  82  Toan =  72   VatLy =  58 DiemTB =  70.0
 Nam: TiengAnh =  77  Toan =  82   VatLy =  79 DiemTB =  79.0
Sinh: TiengAnh =  52  Toan =  62   VatLy =  39 DiemTB =  51.0
Thuy: TiengAnh =  61  Toan =  82   VatLy =  88 DiemTB =  77.0
```

🕮 **Ví dụ 6.25**.

Khai báo kiểu struct và khởi tạo giá trị

Viết chương trình thực hiện các yêu cầu sau đây:

- Khai báo kiểu cấu trúc sinh viên, bao gồm các trường thông tin như: tên sinh viên, điểm môn toán, điểm môn tiếng anh và điểm môn vật lý, trung bình môn, xếp loại.
- Khởi tạo 4 sinh viên, bao gồm các thông tin: tên sinh viên, điểm môn toán, điểm môn tiếng anh, điểm môn vật lý.
- Tính điểm trung bình cho các sinh viên và xếp loại của từng sinh viên. In thông tin của các sinh viên ra màn hình.

✍**Phác thảo lời giải**

- Khai báo cấu trúc sinhvien.
- Khởi tạo mảng sv[] gồm 4 sinh viên, mỗi sinh viên có thông tin của tên sinh viên, điểm môn tiếng anh, điểm môn toán, điểm môn vật lý.
- Sử dụng công thức điểm trung bình = (điểm môn tiếng anh + điểm môn toán + điểm môn vật lý)/3.
- Xếp loại cho từng sinh viên, dựa vào tiêu chí:

+ Điểm trung bình >=90: Xuất sắc.
+ 90 > Điểm trung bình >=80: Giỏi.
+ 80 > Điểm trung bình >= 65: Khá.
+ 65 > Điểm trung bình >= 50: Trung bình.

+ 50 > Điểm trung bình: Yếu.
- Hiển thị thông tin của các sinh viên sau khi đã cập nhật điểm trung bình.

Chương trình:

```
#include <stdio.h>
#include <string.h>
#define N 4
struct sinhvien{
   char ten_sv[20];
   int d_anh;
   int d_toan;
   int d_ly;
   double d_tb;
   char xeploai[15];
};
static struct sinhvien sv[]={
   {"Cuong", 90, 82, 88},
   {"Nam", 95, 90, 89},
   {"Sinh", 52, 62, 39},
   {"Thuy", 61, 82, 88}
};
int main()
{
   int i;
   //tinh diem trung binh cho moi sinh vien
   for(i=0; i<N; i++)
         {
         sv[i].d_tb = (sv[i].d_anh + sv[i].d_toan +sv[i].d_ly)/3;
   }
   //xep loai dua vao diem trung binh
    for(i = 0; i < N; i++)
         {
         if (sv[i].d_tb >= 90) strcpy(sv[i].xeploai,"Xuat sac");
         else if (sv[i].d_tb >= 80) strcpy(sv[i].xeploai,"Gioi");
                  else if (sv[i].d_tb >= 65)
strcpy(sv[i].xeploai,"Kha");
```

```
32                      else if (sv[i].d_tb >= 50) strcpy(sv[i].xeploai,"Trung binh");
33                          else strcpy(sv[i].xeploai,"Yeu");
34      }
35      //hien thi thong tin cua tung sinh vien
36      for(i = 0; I < N; i++){
37          printf("*-----------------------------------*\n");
38                  printf("Sinh vien %s:\n",sv[i].ten_sv );
39          printf("TiengAnh = %2d  Toan = %2d   VatLy = %2d \n", sv[i].d_anh, sv[i].d_toan, sv[i].d_ly);
40          printf("DiemTB = %4.1f \n", sv[i].d_tb);
41          printf("XepLoai = %2s\n", sv[i].xeploai);
42      }
43     printf("*-----------------------------------*\n");
44      return (0);
45  }
```

✍ Kết quả chương trình:

```
*-----------------------------------*
Sinh vien Cuong:
TiengAnh = 90  Toan = 82   VatLy = 88
DiemTB = 86.0
XepLoai = Gioi
*-----------------------------------*
Sinh vien Nam:
TiengAnh = 95  Toan = 90   VatLy = 89
DiemTB = 91.0
XepLoai = Xuat sac
*-----------------------------------*
Sinh vien Sinh:
TiengAnh = 52  Toan = 62   VatLy = 39
DiemTB = 51.0
XepLoai = Trung binh
```

```
*-----------------------------------*
Sinh vien Thuy:
TiengAnh = 61  Toan = 82   VatLy = 88
DiemTB = 77.0
XepLoai = Kha
*-----------------------------------*
```

✍ **Yêu cầu với sinh viên:** Viết lại chương trình từ ví dụ 6.23 đến ví dụ 6.25:

- Xây dựng hàm nhập thông tin các sinh viên;
- Xây dựng hàm hiển thị thông tin sinh viên;
- Xây dựng hàm tính điểm trung bình cho mỗi sinh viên;
- Xây dựng hàm xếp loại cho mỗi sinh viên.

🕮 **Ví dụ 6.26**

Sử dụng biến con trỏ và từ khóa typedef, viết lại chương trình ở ví dụ 6.23.

```
1   #include <stdio.h>
2   #define N 4
3   struct sinhvien{
4       char ten_sv[20];
5       int d_anh;
6       int d_toan;
7       int d_ly;
8       double d_tb;
9   };
10  static struct sinhvien sv[]={
11      {"Cuong", 82, 72, 58},
12      {"Nam", 77, 82, 79},
13      {"Sinh", 52, 62, 39},
14      {"Thuy", 61, 82, 88}
15  };
16
17  int main()
18  {
19      int i;
20      //tinh diem trung binh cho moi sinh vien
```

```
21      for(i=0; i<N; i++)
22            {
23            sv[i].d_tb = (sv[i].d_anh + sv[i].d_toan +sv[i].d_ly)/3;
24      }
25      //hien thi thong tin cua tung sinh vien
26      for(i = 0; i < N; i++){
27         printf("%8s: TiengAnh = %3d  Toan = %3d   VatLy = %3d
    DiemTB = %5.1f\n",
28         sv[i].ten_sv,    sv[i].d_anh,    sv[i].d_toan,    sv[i].d_ly,
    sv[i].d_tb);
29      }
30      return (0);
31   }
```

✍ Kết quả chương trình:

```
Cuong:   TiengAnh =  82  Toan =  72   VatLy =    58
Nam:     TiengAnh =  77  Toan =  82   VatLy =    79
Sinh:    TiengAnh =  52  Toan =  62   VatLy =    39
Thuy:    TiengAnh =  61  Toan =  82   VatLy =    88
```

🕮 Ví dụ 6.27

Bài toán tuyển sinh của trường đại học.

Trường Đại học Công nghệ GTVT có n thí sinh khối A nộp hồ sơ xét tuyển (n được nhập từ bàn phím). Thông tin của một thí sinh bao gồm họ tên, số báo danh, điểm môn Toán, điểm môn Lý, điểm môn Hóa và tổng điểm của ba môn. Điểm chuẩn để trúng tuyển vào trường là 18.

- Lập trình nhập vào thông tin của n thí sinh khối A đã nộp hồ sơ.
- In ra màn hình danh sách thí sinh khối A đã nộp hồ sơ xét tuyển.
- In danh sách thí sinh trúng tuyển, thí sínhh trúng tuyển có tổng điểm >= 18.

✍Phác thảo lời giải

- Khai báo 1 cấu trúc thisinh bao gồm các thông tin họ tên, số báo danh, điểm môn toán, điểm môn lý, điểm môn hóa, điểm tổng.
- Sử dụng mảng cấu trúc để giải quyết bài toán;
- Xây dựng hàm nhapttts(), gồm hai tham số là tên biến mảng kiểu thí sinh, số thí sinh;
- Xây dựng hàm hienthi() để hiển thị thông tin thí sinh đăng ký xét tuyển;
- Xây dựng hàm thisinhTT(): hiển thị danh sách các thí sinh trúng tuyển (là thí sinh có tổng điểm ≥ điểm chuẩn).

Chương trình:

```
#include<stdio.h>
#define max 50
#define diemchuan 18
struct thisinh
{
char ht[30];
char sbd[10];
float d_toan,d_ly,d_hoa,tong_diem;
} ;
void nhapttts(thisinh a[], int n)
{
int i;
printf("Nhap thong tin cua cac thi sinh:\n");
for (i = 0;i < n;i++)
{
printf("Nhap thong tin thu %d:\n", i+1);
printf("Nhap ho ten thi sinh:\n");
fflush(stdin);
gets(a[i].ht);
printf("Nhap so bao danh thi sinh:\n");
fflush(stdin);
gets(a[i].sbd);
printf("Nhap diem mon Toan: \n");
scanf("%f", &a[i].d_toan);
printf("Nhap diem mon Ly: \n");
scanf("%f", &a[i].d_ly);
printf("Nhap diem mon Hoa: \n");
scanf("%f", &a[i].d_hoa);
a[i].tong_diem=a[i].d_toan+a[i].d_ly+a[i].d_hoa;
}
}
void hienthi(thisinh a[], int n)
{
int i;
```

```
35      printf("Danh sach thi sinh du thi: \n");
36      printf("%-4s%-20s%-10s%-10s%-10s%-10s%-10s\n",
        "Stt","Ho_ten","SBD","D_toan","D_ly","D_hoa","Tong_die
        m");
37      for (i = 0;i < n;i++)
38      printf("%-4d%-20s%-10s%-10.1f%-10.1f%-10.1f%-10.1f\n",
        i+1,a[i].ht,a[i].sbd,a[i].d_toan,a[i].d_ly,a[i].d_hoa,a[i].tong_di
        em);
39      }
40      void thisinhTT(thisinh a[], int n)
41      {
42      int i, dem ;
43      printf("\nDanh sach thi sinh trung tuyen :\n");
44      printf("%-4s%-20s%-10s%-10s%-10s%-10s%-10s\n",
        "Stt","Ho_ten","SBD","D_toan","D_ly","D_hoa","Tong_die
        m");
45      for (i = 0,dem = 0;i<n;i++)
46      if (a[i].tong_diem>=diemchuan)
47      {dem++;
48      printf("%-4d%-20s%-10s%-10.1f%-10.1f%-10.1f%-10.1f\n",
        dem,a[i].ht,a[i].sbd,a[i].d_toan,a[i].d_ly,a[i].d_hoa,a[i].tong_d
        iem);
49      }
50      }
51      int main()
52      {
53      int n;
54      struct thisinh a[max];
55      printf("Nhap vao so luong thi sinh khoi A:\n");
56      scanf("%d",&n);
57      nhapttts(a,n);
58      hienthi(a,n);
59      thisinhTT(a,n);
60      printf("\nNhan Enter ket thuc ...!");
61      return 0;
62      }
```

✍Kết quả chương trình:

```
Nhap vao so luong thi sinh khoi A:
5
Nhap thong tin cua cac thi sinh:
Nhap thong tin thu 1:
Nhap ho ten thi sinh:
An
Nhap so bao danh thi sinh:
a1901
Nhap diem mon Toan:
8
Nhap diem mon Ly:
8
Nhap diem mon Hoa:
7
Nhap thong tin thu 2:
Nhap ho ten thi sinh:
Binh
Nhap so bao danh thi sinh:
a1902
Nhap diem mon Toan:
5
Nhap diem mon Ly:
4
Nhap diem mon Hoa:
4
Nhap thong tin thu 3:
Nhap ho ten thi sinh:
Cuong
Nhap so bao danh thi sinh:
a1903
Nhap diem mon Toan:
8
Nhap diem mon Ly:
9
Nhap diem mon Hoa:
```

9
Nhap thong tin thu 4:
Nhap ho ten thi sinh:
Thuy
Nhap so bao danh thi sinh:
a1904
Nhap diem mon Toan:
6
Nhap diem mon Ly:
8
Nhap diem mon Hoa:
9
Nhap thong tin thu 5:
Nhap ho ten thi sinh:
Yen
Nhap so bao danh thi sinh:
a1905
Nhap diem mon Toan:
4
Nhap diem mon Ly:
4
Nhap diem mon Hoa:
4
Danh sach thi sinh du thi:

Stt	Ho_ten	SBD	D_toan	D_ly	D_hoa	Tong_diem
1	An	a1901	8.0	8.0	7.0	23.0
2	Binh	a1902	5.0	4.0	4.0	13.0
3	Cuong	a1903	8.0	9.0	9.0	26.0
4	Thuy	a1904	6.0	8.0	9.0	23.0
5	Yen	a1905	4.0	4.0	4.0	12.0

Danh sach thi sinh trung tuyen :

Stt	Ho_ten	SBD	D_toan	D_ly	D_hoa	Tong_diem
1	An	a1901	8.0	8.0	7.0	23.0
2	Cuong	a1903	8.0	9.0	9.0	26.0
3	Thuy	a1904	6.0	8.0	9.0	23.0

🕮 **Ví dụ 6.28.**

Bài toán tính tiền điện cho các hộ gia đình trong khu phố:

Viết chương trình tính tiền điện cho các hộ gia đình của một khu phố có n hộ (n được nhập từ bàn phím).

Thông tin về một hộ gia đình gồm có tên chủ hộ, số điện tiêu thụ tháng trước, số điện tiêu thụ tháng này, số tiền phải trả. Biết rằng:

- Số tiền phải trả được tính dựa vào số điện tiêu thụ.
- Điện tiêu thụ = Điện tiêu thụ tháng này - Điện tiêu thụ tháng trước.
- Đơn giá/1 kWh được tính theo hình thức lũy tiến như sau:

Bậc tính	Giá
Bậc 1: Cho kWh từ 0 - 50	1.678
Bậc 2: Cho kWh từ 51 - 100	1.734
Bậc 3: Cho kWh từ 101 - 200	2.014
Bậc 4: Cho kWh từ 201 - 300	2.536
Bậc 5: Cho kWh từ 301 - 400	2.834
Bậc 6: Cho kWh từ 401 trở lên	2.927

Viết chương trình thực hiện các yêu cầu sau đây:

- Nhập thông tin của các hộ gia đình, sử dụng mảng các cấu trúc;
- Tính tiền điện cho các hộ gia đình và hiển thị ra màn hình.
- Thống kê hộ sử dụng theo từng loại:

c1. $\leq$ 100 kWh

c2. $\geq$ 101 kWh và $\leq$ 300 kWh

c3. $\geq$ 301 kWh

✍ **Chương trình:**

```
1  #include<stdio.h>
2
3  #define max 100
4  struct hogd
5  {
6  char ht[30];
7  float csd, css;
8  double tiendien;
9  };
```

```
10  void nhaptt(hogd hgd[], int n)
11  {
12  int i;
13  float dtt;
14  double td;
15  printf("Nhap thong tin cua cac ho gia dinh:\n");
16    for(i=0;i<n;i++)
17      {
18       dtt =0;
19       td =0;
20       printf("Nhap thong tin cua ho thu %d:\n", i+1);
21       printf("Nhap ten chu ho:");
22       fflush(stdin);
23       gets(hgd[i].ht);
24       printf("Nhap vao chi so truoc:");
25       scanf("%f", &hgd[i].csd);
26       printf("Nhap vao chi so sau:");
27       scanf("%f", &hgd[i].css);
28       dtt = hgd[i].css - hgd[i].csd;
29       if (dtt<=50) td =1678*dtt;
30            else if (dtt<=100) td= 1678*50 +1734*(dtt-50);
31                 else if (dtt<=200) td = 1678*50 + 50* 1734 +
     2014*(dtt-100);
32                 else if (dtt<=300) td = 1678*50 + 50* 1734 +
     2014*100 + 2536*(dtt -200);
33                      else if (dtt<=400) td = 1678*50 + 50*
     1734 + 2014*100 + 2536*100 + 2834*(dtt-300);
34                      else td = 1678*50 + 50* 1734 +
     2014*100 + 2536*100 + 2834*100 + 2927*(dtt-400);
35          hgd[i].tiendien = td;
36           }
37  }
38  void hienthi(hogd hgd[], int n)
39  {
40  int i,j;
41  printf("%-4s%-20s%-15s%-15s%-15s\n", "Stt","Chu ho","Chi
```

```
so dau","Chi so cuoi","Tien dien");
for (i=0,j=0;i<n;i++)
{j++;
printf("%-4d%-20s%-15.0f%-15.0f%-15.0f\n",                j,
hgd[i].ht,hgd[i].csd,hgd[i].css, hgd[i].tiendien);
}
}
int thongke(hogd hgd[],int n,float d,float c)
// thong ke dien tieu thu trong khoang tu d den c
{
int i, dem=0;float dtt;
for (i=0;i<n;i++)
{dtt = hgd[i].css-hgd[i].csd;
if ((dtt>=d) && (dtt<=c)) dem++;
}
return dem;
}
int main()
{
hogd h[max];
float d,c;
int i,j,n,k;
printf("Nhap so ho gia dinh:\n");
scanf("%d", &n);
nhaptt(h,n);
printf("Bang tong hop tien dien cua cac ho gia dinh:\n");
hienthi(h,n);
k=thongke(h,n,0,100);
printf("So ho gia dinh su dung <=100 so la %d, chiem %2.1f
%\n",k,(float)k/n*100);
k=thongke(h,n,101,200);
printf("So ho gia dinh su dung  tu 101 den 100 so la %d,
chiem %2.1f %\n",k,(float)k/n*100);
k=thongke(h,n,201,300);
printf("So ho gia dinh su dung tu 201 den 300 so la %d,
chiem %2.1f %\n",k,(float)k/n*100);
```

```
73  k=thongke(h,n,301,400);
74  printf("So ho gia dinh su dung tu 301 den 400 so la %d,
    chiem %2.1f %\n",k,(float)k/n*100);
75  k=thongke(h,n,401,500);
76  printf("So ho gia dinh su dung tu 401 den 500 so la %d,
    chiem %2.1f %\n",k,(float)k/n*100);
77  return 0;
78  }
```

Kết quả chương trình

```
Nhap so ho gia dinh:
3
Nhap thong tin cua cac ho gia dinh:
Nhap thong tin cua ho thu 1:
Nhap ten chu ho: Anh
Nhap vao chi so truoc: 45
Nhap vao chi so sau: 501
Nhap thong tin cua ho thu 2:
Nhap ten chu ho: Dan
Nhap vao chi so truoc: 101
Nhap vao chi so sau: 302
Nhap thong tin cua ho thu 3:
Nhap ten chu ho: Thang
Nhap vao chi so truoc: 136
Nhap vao chi so sau: 432
Bang tong hop tien dien cua cac ho gia dinh:
Stt     Chu ho   Chi so dau   Chi so cuoi   Tien dien
1       Anh      45           501           1072912
2       Dan      101          302           374536
3       Thang    136          432           615456
So ho gia dinh su dung <= 100 so la 0, chiem 0.0
So ho gia dinh su dung  tu 101 den 100 so la 0, chiem 0.0%
So ho gia dinh su dung tu 201 den 300 so la 2, chiem 66.7%
So ho gia dinh su dung tu 301 den 400 so la 0, chiem 0.0%
So ho gia dinh su dung tu 401 den 500 so la 1, chiem 33.3%
```

6.3. CASE STYDY

6.3.1. Phát biểu bài toán

Viết chương trình bằng ngôn ngữ lập trình C để lưu trữ và thực hiện phân tích, thống kê cho một lớp có 20 sinh viên. Thông tin của mỗi sinh viên bao gồm: mã sinh viên, tên sinh viên, giới tính, hai điểm thành phần, điểm chuyên cần, điểm giữa kỳ, điểm thi cuối kỳ và điểm tổng kết.

Chương trình nhắc người dùng chọn thao tác với các bản ghi từ Menu sau đây:

Chương trình trình

---

1. Thêm hồ sơ sinh viên.
2. Xóa hồ sơ sinh viên.
3. Cập nhật hồ sơ sinh viên.
4. Hiển thị toàn bộ thông tin sinh viên.
5. Tính điểm tổng kết của sinh viên được chọn.
6. Hiển thị thông tin sinh viên có điểm cao nhất.
7. Hiển thị thông tin sinh viên có điểm thấp nhất.
8. Tìm kiếm sinh viên dựa vào mã sinh viên.
9. Sắp xếp danh sách sinh viên dựa vào điểm tổng kết.

6.3.2. Hướng giải quyết vấn đề

6.3.2.1. Lựa chọn cấu trúc dữ liệu để giải quyết bài toán

Lưu ý: Thông tin của các sinh viên được lưu trữ trong một mảng cấu trúc.

Để giải quyết bài toàn này một cách đơn giản và dễ thực hiện nhất, ta chia giải pháp thành các bước sau đây:

🕮 **Bước 1:** Định nghĩa một cấu trúc có tên sinhvien để lưu trữ các bản ghi. Cấu trúc sinhvien chứa 9 thành phần sau đây:

- masv (chuỗi ký tự, có độ dài tối đa 10 ký tự) để lưu mã sinh viên, phân biệt sinh viên này với sinh viên khác;
- tensv (chuỗi ký tự, hoặc mảng các ký tự, độ dài tối đa 25 ký tự) để lưu tên của sinh viên;
- gioi_tinh (ký tự) lưu giới tính của sinh viên, nhận một trong hai giá trị F (female - nữ) hoặc M (male - nam);

- d_tp1 (số thực - float) để lưu điểm thành phần thứ nhất;
- d_tp2 (số thực - float) để lưu điểm thành phần thứ hai;
- d_cc (số thực - float) để lưu điểm chuyên cần của sinh viên;
- d_gk (số thực - float) để lưu điểm giữa kỳ của sinh viên;
- d_kt (số thực - float) để lưu điểm thi cuối kỳ của sinh viên;
- diem_tong (số thực - float) để lưu điểm tổng kết của sinh viên.

```
typedef struct
{
char masv[10];
char tensv[10];
char gt;
float d_tp1;
float d_tp2;
float d_cc;
float d_gk;
float d_kt;
float tong_diem;
} sinhvien;
```

🕮 **Bước 2:** Khai báo các hàm được sử dụng trong chương trình và xây dựng hàm để hiển thị các chức năng của chương trình. Chương trình cho phép lựa chọn các công việc (chức năng) từ 1 đến 9 để thao tác với các bản ghi.

```
// khai báo các hàm được sử dụng trong chương trình
void Themsv(sinhvien sv[], int *spt);
int timkiem(sinhvien sv[], char id[], int spt);
void xoasv(sinhvien sv[], int *spt);
void khoitao(sinhvien sv[], int index);
void capnhat(sinhvien sv[], int spt);
void Sx_theo_tong(sinhvien sv[], int n);
void hienthi(sinhvien sv[], int spt);
void tieude();
```

```
// Hàm cho phép hiển thị hệ thống lựa chọn các chức năng
//Ham de hien thi cac menu duoc chon
void hienthi_menu(){
printf("----------------------------------------\n");
printf("            CHUONG TRINH CHINH \n");
```

```
5   printf("--------------------------------------\n");
6   printf(" 1. Them ho so sinh vien\n");
7   printf(" 2. Xoa ho so sinh vien\n");
8   printf(" 3. Cap nhat ho so sinh vien\n");
9   printf(" 4. Hien thi toan bo thong tin sinh vien\n");
10  printf(" 5. Tinh diem tong ket cua sinh vien\n");
11  printf(" 6. Hien thi sinh vien co diem tong ket cao nhat\n");
12  printf(" 7. Hien thi sinh vien co diem tong ket thap nhat\n");
13  printf(" 8. Tim kiem sinh vien dua vao ma sinh vien\n");
14  printf(" 9. Sap xep danh sach sinh vien dua vao diem tong ket\n");
15  }
```

📖 **Bước 3:** Xây dựng hàm Themsv(sinhvien sv[], int *spt) để thêm một bản ghi mới vào mảng các đối tượng sinh viên. Hàm này có hai đối số, đối số đầu là mảng các đối tượng sinh viên (sv), đối số thứ hai là số phần tử trong mảng. Đầu tiên, hàm này sẽ kiểm tra bản ghi mới (sử dụng hàm tìm kiếm, được xây dựng ở bước sau) trước khi nó được thêm vào mảng để tránh các bản ghi trùng lặp. Khi bản ghi mới được thêm vào, số phần tử của mảng n được tăng thêm 1, đồng nghĩa với số lượng bản ghi trong danh sách được tăng lên.

```
// Hàm để thêm một bản ghi mới
1   void Themsv(sinhvien sv[], int *spt )
2   {
3    kt:
4           printf("Nhap vao ma sinh vien:");
5           fflush(stdin);
6           scanf("%s",&sv[*spt].masv);
7           if (timkiem(sv,sv[*spt].masv,*spt)!=-1)
8            {
9                   printf("Ma sinh vien da ton tai\n"); goto kt;
10           }
11  printf("Nhap ten sinh vien:");
12  fflush(stdin);
13  gets(sv[*spt].tensv);
14  printf("Nhap gioi tinh sinh vien:(F hoac M):");
15  fflush(stdin);
16  scanf("%s", &sv[*spt].gt);
```

```
17    printf("Nhap diem thanh phan thu nhat:");
18    scanf("%f", &sv[*spt].d_tp1);
19    printf("Nhap diem thanh phan thu hai:");
20    scanf("%f", &sv[*spt].d_tp2);
21    printf("Nhap diem chuyen can:");
22    scanf("%f", &sv[*spt].d_cc);
23    printf("Nhap diem giua ky:");
24    scanf("%f", &sv[*spt].d_gk);
25    printf("Nhap diem ket thuc mon:");
26    scanf("%f", &sv[*spt].d_kt);
27    sv[*spt].tong_diem=sv[*spt].d_tp1+sv[*spt].d_tp2+sv[*spt].d_
      cc+sv[*spt].d_gk+sv[*spt].d_kt;
28    (*spt)++;
29    }
```

Hiển thị các bản ghi trong danh sách

🕮 **Bước 4:** Xây dựng hàm tìm kiếm search(sinhvien sv[], char id[], int spt): tìm kiếm theo mã sinh viên. Hàm này rất hữu ích vì chúng ta cần sử dụng nó để tìm vị trí của bản ghi trong mảng các đối tượng sinh viên. Hàm tìm kiếm còn được sử dụng để bảo đảm bản ghi này tồn tại trước khi thực hiện xóa hoặc cập nhật bản ghi. Nếu phần tử được tìm thấy, hàm sẽ trả về chỉ số của phần tử này, trả về -1, nếu không tìm thấy phần tử cần tìm (theo mã số sinh viên) trong mảng.

```
//Hàm tìm kiếm vị trí của bản ghi
1   int timkiem(sinhvien sv[], char id[], int spt)
2   {
3           int timthay =-1, i;
4           for(i=0; i< spt && timthay==-1; i++ )
5           {
6                   if(strcmp(sv[i].masv, id)==0) timthay = i;
7                   else timthay=-1;
8           }
9           return timthay;
10  }
```

🕮 **Bước 5:** Xây dựng hàm hiển thị tất cả các bản ghi hienthi(sinhvien sv[], int spt) để hiển thị tất cả các thông tin của các sinh viên trong lớp. Để hiển thị tất cả các bản ghi, chúng ta cần sử dụng một vòng lặp để duyệt qua tất cả các đối tượng trong mảng các sinh viên.

```
//Hàm để hiển thị tất cả các bản ghi

// Ham hien thi thong tin cua sinh vien
1   void hienthi(sinhvien sv[], int spt)
2   {
3           int i=0;
4           tieude();
5           while (i<spt)
6           {
7                   if(sv[i].masv!="")
8                   {
9                           printf("%-7s",sv[i].masv);
10                          printf("%-10s",sv[i].tensv);
11                          printf("%-6c",sv[i].gt);
12                          printf("%-7.1f",sv[i].d_tp1);
13                          printf("%-7.1f",sv[i].d_tp2);
14                          printf("%-7.1f",sv[i].d_cc);
15                          printf("%-7.1f",sv[i].d_gk);
16                          printf("%-7.1f",sv[i].d_kt);
17                          printf("%-7.1f",sv[i].tong_diem);
18                          printf("\n");
19                  }
20                  i=i+1;
21          }
22          if (spt==0) printf("Khong co ban ghi nao!\n");
23  }
24  void tieude()
25  {
26  printf("MASV  TENSV  GT   D_TP1  D_TP2  D_CC  D_GK
    D_KT  TONG  \n ");
27  printf("=========================================
    ==========\n");
28  }
```

Xóa bản ghi

🕮 **Bước 6:** Xây dựng hàm xoasv(sinhvien sv[], int *spt) để xóa một bản ghi từ mảng các đối tượng sinh viên. Đầu tiên người dùng sẽ nhập mã sinh viên cần xóa. Bước tiếp theo, mã sinh viên này sẽ được kiểm tra để đảm bảo nó tồn tại trong danh sách. Nếu bản ghi hoặc phần tử đó thực sự tồn tại, quá trình xóa có thể được thực hiện. Hàm xoasv() được thực hiện bằng việc kiểm tra bản ghi cần xóa là bản ghi đầu tiên, bắt đầu hay ở giữa danh sách. Nếu bản ghi cần xóa là bản ghi cuối cùng trong danh sách, chúng ta chỉ cần xóa bản ghi bằng cách cung cấp cho nó 1 phương thức khoitao(sinhvien sv[], int index) (khởi tạo tất cả giá trị về null hoặc 0). Bản ghi cuối cùng là bản ghi có chỉ số bằng số phần tử trong danh sách - 1. Nếu bản ghi cần xóa ở đầu hoặc ở giữa danh sách, chúng ta cần sử dụng một vòng lặp cho phép phần tử trước tiếp quản phần tử kế tiếp (Dịch chuyển phần tử sang trái, bắt đầu từ vị trí của phần từ cần xóa đến hết danh sách, ở mỗi bước phương thức khoitao() được gọi để xóa thông tin của phần tử cuối cùng trong danh sách, thực chất đây là thao tác ghi đè dữ liệu lên phần tử cần xóa). Sau khi phần tử được xóa, số phần tử của danh sách bị giảm đi 1 đơn vị.

```
1   void xoasv(sinhvien sv[], int *spt)
2   {
3           char id[10];
4           int index, i;
5           if (*spt>0)
6            {
7                   printf("Nhap ma sv can xoa:"); scanf("%s", &id);
8                   index=timkiem(sv, id, *spt);
9
10                  if (index!=-1 && *spt!=0)
11                   {
12                  if (index==*spt-1) // xoa phan tu cuoi cung
13                          {
14                          khoitao(sv, index);
15                          *spt--;
16                          printf("Ban ghi cua sinh vien da bi xoa.\n");
17                          }
18                          else
19                                  {
20                                  for(i=index;i<*spt;i++) //Xoa phan
    tu dau tien hoac phan tu o giua
```

```
21                                          {
22                                          sv[i]=sv[i+1];
23                                          khoitao(sv, *spt);
24                                          (*spt)--;
25                                          }
26                                  }
27                          }
28                          else printf("Ban ghi khong ton tai. Kiem tra Masv
    va thu lai.\n");
29                  }
30                  else printf("Khong co ban gi de xoa\n");
31  }
```

//Hàm khoitao

```
1   void khoitao(sinhvien sv[], int index)
2   {
3   strcpy(sv[index].masv,"");
4   strcpy(sv[index].tensv,"");
5   sv[index].gt =NULL;
6   sv[index].d_tp1 = 0;
7   sv[index].d_tp2 = 0;
8   sv[index].d_cc = 0;
9   sv[index].d_gk = 0;
10  sv[index].d_kt = 0;
11  sv[index].tong_diem = 0;
12  }
```

Cập nhật bản ghi

🕮 **Bước 7:** Xây dựng hàm capnhat(sinhvien sv[], int spt) để cập nhật một bản ghi được chỉ định. Quá trình cập nhật bắt đầu bằng cách yêu cầu người dùng nhập mã sinh viên của sinh viên cần cập nhật thông tin. Giá trị mã sinh viên sẽ được kiểm tra để đảm bảo sinh viên được cập nhật thực sự tồn tại. Nếu giá trị mã sinh viên đó tồn tại, việc thay đổi với bản ghi được thực hiện sau khi yêu cầu người dùng nhập giá trị mới của trường cần thay đổi.

```
//Hàm cập nhật bản ghi
void capnhat(sinhvien sv[], int spt)
{
char id[10];
int cot_capnhat;
printf("Nhap ma sinh vien:");
scanf("%s",&id);
printf("Truong ban muon cap nhat(1-7)?:");
scanf("%d",&cot_capnhat);

int index = timkiem(sv, id, spt);

if (index != -1)
{
        if (cot_capnhat == 1)
        {
        printf("Nhap ten sinh vien:");
        fflush(stdin);
        scanf("%s",&sv[index].tensv);
        }

        else if (cot_capnhat == 2)
        {
        printf("Nhap gioi tinh sinh vien (F hoac M):");
        scanf("%c",&sv[index].gt);
        }
        else if (cot_capnhat == 3)
        {
        printf("Nhap diem thanh phan 1:");
        scanf("%f",&sv[index].d_tp1);
        }
        else if (cot_capnhat == 4)
        {
        printf("Nhap diem thanh phan 2:");
        scanf("%f",&sv[index].d_tp2);
```

```
35          }
36          else if (cot_capnhat == 5)
37          {
38          printf("Nhap diem chuyen can:");
39          scanf("%f",&sv[index].d_cc);
40          }
41          else if (cot_capnhat == 6)
42          {
43          printf("Nhap diem giua ky:");
44          scanf("%f",&sv[index].d_gk);
45          }
46          else if (cot_capnhat == 7)
47          {
48          printf("Nhap diem cuoi ky:");
49          scanf("%f",&sv[index].d_kt);
50          }
51          else printf("Khong phai cot can cap nhat");
52  sv[index].tong_diem = sv[index].d_tp1 + sv[index].d_tp2 +
    sv[index].d_cc + sv[index].d_gk + sv[index].d_kt;
53  }
54  else printf("Ban ghi khong ton tai. Kiem tra lai Masv va thu lai.");
55  }
```

Tính điểm trung bình của sinh viên

🕮 **Bước 8:** Xây dựng hàm dtb(sinhvien[] sv, int n) để tính điểm trung bình của sinh viên được lựa chọn. Hàm bắt đầu bằng việc yêu cầu người dùng nhập mã sinh viên muốn tính điểm trung bình. Mã sinh viên được yêu cầu đảm bảo tồn tại. Tính điểm trung bình của sinh viên theo công thức dtb = (điểm thành phần 1 + điểm thành phần 2 + điểm chuyên cần + điểm giữa kỳ + điểm thi)/5.

```
//Ham tinh diem trung binh cua sinh vien
1   void tb_sv(sinhvien sv[], int spt)
2   {
3   char id[10];
4   float dtb=0;
5   printf("Nhap ma sinh vien:");
```

```
6   scanf("%s",&id);
7   int index = timkiem(sv, id,spt);
8   if (index != -1 && spt>0)
9   {
10  sv[index].tong_diem = sv[index].d_tp1 + sv[index].d_tp2 +
    sv[index].d_cc + sv[index].d_gk + sv[index].d_kt;
11  dtb = sv[index].tong_diem/5;
12  printf("Diem trung binh cua sinh vien la: %f",dtb);
13  printf("\n");
14  }
15  else printf("Ban ghi khong ton tai. Kiem tra lai Masv va thu lai.");
16  }
```

Tính tổng điểm cao nhất và thấp nhất.

🕮 **Bước 9:** Định nghĩa hàm caonhat(sinhvien sv[], int spt) và thapnhat(sinhvien sv[], int spt) để hiển thị thông tin của sinh viên nhận điểm cao nhất và điểm thấp nhất. Để tìm điểm cao nhất hoặc điểm thấp nhất, chúng ta cần so sánh điểm tổng của từng sinh viên.

```
//Ham hien thi diem cao nhat
1   void caonhat(sinhvien sv[], int spt)
2   {
3   float max = sv[0].tong_diem;
4   int index=0,j;
5   if (spt >= 2)
6   {
7
8     for (j = 0; j < spt-1; ++j)
9           if (max < sv[j+1].tong_diem)
10          {
11          max = sv[j+1].tong_diem;
12          index = j+1;
13          }
14  printf("Sinh vien co ma sinh vien ID %s  co tong diem cao nhat
    la: %f",sv[index].masv,max);
15  printf("\n");
16  }
17  else if (spt == 1)
```

```
18  {
19  index = 0;
20  max = sv[0].tong_diem;
21  printf("Sinh vien co ma sinh vien ID %s  co tong diem cao nhat
    la: %f",sv[index].masv,max);
22  printf("\n");
23  }
24  else printf("Khong co ban ghi nao!\n");
25  }
```

//Ham hien thi diem thap nhat

```
1   void thapnhat(sinhvien sv[], int spt)
2   {
3   float min = sv[0].tong_diem;
4   int index=0,j;
5   if (spt >= 2)
6   {
7     for (j = 0; j < spt-1; ++j)
8           if (min > sv[j+1].tong_diem)
9           {
10          min = sv[j+1].tong_diem;
11          index = j+1;
12          }
13  printf("Sinh vien co ma sinh vien ID %s  co tong diem thap nhat
    la: %f: %f",sv[index].masv,min);
14  printf("\n");
15  }
16  else if (spt == 1)
17  {
18  index = 0;
19  min = sv[0].tong_diem;
20  printf("Sinh vien co ma sinh vien ID %s  co tong diem thap nhat
    la: %f: %f",sv[index].masv,min);
21  printf("\n");
22  }
23  else printf("Khong co ban ghi nao!\n");
24  }
```

Tìm kiếm bản ghi

🕮 **Bước 10:** Xây dựng hàm tìm kiếm find(sinhvien sv[], int spt) để tìm kiếm một bản ghi trong danh sách. Hàm này yêu cầu người dùng nhập mã sinh viên. Mã sinh viên được kiểm tra để đảm bảo nó thực sự tồn tại. Nếu bản ghi được tìm thấy, tất cả các thông tin của sinh viên tìm được sẽ hiển thị. Ngược lại, nếu không tìm thấy bản ghi, một thông báo "Bản ghi không tồn tại" được hiển thị.

```
//Ham tim kiem thong tin sinh vien
1   void find(sinhvien sv[], int spt)
2   {
3   char id[10];
4   printf("Nhap ma sinh vien can tim:");
5   scanf("%s",&id);
6   int index=timkiem(sv,id, spt);
7   if (index != -1)
8   { //hien thi thong tin ban ghi tim thay
9   tieude();
10  printf("%-7s",sv[index].masv);
11  printf("%-10s",sv[index].tensv);
12  printf("%-6c",sv[index].gt);
13  printf("%-7.1f",sv[index].d_tp1);
14  printf("%-7.1f",sv[index].d_tp2);
15  printf("%-7.1f",sv[index].d_cc);
16  printf("%-7.1f",sv[index].d_gk);
17  printf("%-7.1f",sv[index].d_kt);
18  printf("%-7.1f",sv[index].tong_diem);
19  printf("\n");
20  }
21  else printf("Ban ghi khong ton tai.\n");
22  }
```

Sắp xếp danh sách

🕮 **Bước 11:** Định nghĩa hàm bubblesort(sinhvien sv[], int n) để sắp xếp các bản ghi tăng dần theo điểm tổng. Hàm này được dựa trên thuật toán bubble sort.

```
//Ham sap xep theo tong diem
void Sx_theo_tong(sinhvien sv[], int n)
{
int i, j;
sinhvien tg;
for (i = 0; i < n; i++)
        for (j = n - 1; j > i; j--)
                if (sv[j].tong_diem < sv[j - 1].tong_diem )
                {
                tg = sv[j];
                sv[j] = sv[j - 1];
                sv[j - 1] = tg;
                }
}
```

🕮 **Bước 12:** Xây dựng hàm main() thực hiện các công việc

```
int main()
{
        sinhvien sv[MAX];
        int spt = 0;
        hienthi_menu();
        int luachon;
        char ok;
        do
        {
        printf("Lua chon cong viec(1-9):"); scanf("%d", &luachon);
                switch(luachon)
                {
                        case 1: Themsv(sv, &spt); break;
                        case 2: xoasv(sv, &spt); break;
                        case 3: capnhat(sv, spt); break;
                        case 4: hienthi(sv, spt); break;
                        case 5: tb_sv(sv, spt);break;
```

```
                        case 6: caonhat(sv, spt);break;
                        case 7: thapnhat(sv, spt);break;
                        case 8:find(sv, spt);break;
                        case 9: {
                                Sx_theo_tong(sv, spt);
                                hienthi(sv, spt);
                                break;
                                }
                default: printf("Khong co lua chon nay\n"); break;
                }
                printf("Nhan phim y de tiep tuc:");
                scanf("%s", &ok);
        } while (ok == 'y');

        if (ok != 'y') printf("CAM ON BAN DA THUC HIEN CHUONG TRINH!\n");
        return 0;
}
```

6.3.2.2. Giải quyết vấn đề

✍**Chương trình**:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#define MAX 100
//dinh nghia kieu cau truc sinh vien

typedef struct
{
char masv[10];
char tensv[10];
char gt;
float d_tp1;
float d_tp2;
float d_cc;
float d_gk;
```

```
float d_kt;
float tong_diem;
} sinhvien;
void Themsv(sinhvien sv[], int *spt);
int timkiem(sinhvien sv[], char id[], int spt);
void xoasv(sinhvien sv[], int *spt);
void khoitao(sinhvien sv[], int index);
void capnhat(sinhvien sv[], int spt);
void Sx_theo_tong(sinhvien sv[], int n);
void hienthi(sinhvien sv[], int spt);
void tieude();
//Ham de hien thi cac menu duoc chon
void hienthi_menu(){
printf("-----------------------------------------\n");
printf("            CHUONG TRINH CHINH \n");
printf("-----------------------------------------\n");
printf(" 1. Them ho so sinh vien\n");
printf(" 2. Xoa ho so sinh vien\n");
printf(" 3. Cap nhat ho so sinh vien\n");
printf(" 4. Hien thi toan bo thong tin sinh vien\n");
printf(" 5. Tinh diem tong ket cua sinh vien\n");
printf(" 6. Hien thi sinh vien co diem tong ket cao nhat\n");
printf(" 7. Hien thi sinh vien co diem tong ket thap nhat\n");
printf(" 8. Tim kiem sinh vien dua vao ma sinh vien\n");
printf(" 9. Sap xep danh sach sinh vien dua vao diem tong ket\n");
}
void Themsv(sinhvien sv[], int *spt )
{
 kt:
 printf("Nhap vao ma sinh vien:");
 fflush(stdin);
 scanf("%s",&sv[*spt].masv);
 if (timkiem(sv,sv[*spt].masv,*spt) != -1)
  {
            printf("Ma sinh vien da ton tai\n"); goto kt;
```

```
  }
 printf("Nhap ten sinh vien:");
 fflush(stdin);
 gets(sv[*spt].tensv);
 printf("Nhap gioi tinh sinh vien:(F hoac M):");
 fflush(stdin);
 scanf("%s", &sv[*spt].gt);
 printf("Nhap diem thanh phan thu nhat:");
 scanf("%f", &sv[*spt].d_tp1);
 printf("Nhap diem thanh phan thu hai:");
 scanf("%f", &sv[*spt].d_tp2);
 printf("Nhap diem chuyen can:");
 scanf("%f", &sv[*spt].d_cc);
 printf("Nhap diem giua ky:");
 scanf("%f", &sv[*spt].d_gk);
 printf("Nhap diem ket thuc mon:");
 scanf("%f", &sv[*spt].d_kt);
 (*spt)++;
}
int timkiem(sinhvien sv[], char id[], int spt)
{
 int timthay = -1, i;
 for(i = 0; i < spt && timthay == -1; i++)
 {
        if(strcmp(sv[i].masv, id) == 0) timthay = i;
        else timthay = -1;
 }
 return timthay;
}
void xoasv(sinhvien sv[], int *spt)
{
 char id[10];
 int index, i;
 if (*spt > 0)
  {
```

```
        printf("Nhap ma sv can xoa:"); scanf("%s", &id);
        index = timkiem(sv, id, *spt);

        if (index != -1 && *spt != 0)
         {
                if (index == *spt-1) // xoa phan tu cuoi cung
                {
                khoitao(sv, index);
                *spt--;
                printf("Ban ghi cua sinh vien da bi xoa.\n");
                }
                else
                        {
                        for(i = index;i<*spt;i++) //Xoa phan tu dau tien
 hoac phan tu o giua
                                {
                                sv[i] = sv[i+1];
                                khoitao(sv, *spt);
                                (*spt)--;
                                }
                  }
         }
        else printf("Ban ghi khong ton tai. Kiem tra Masv va thu
 lai.\n");
  }
 else printf("Khong co ban gi de xoa\n");
}
void khoitao(sinhvien sv[], int index)
{
strcpy(sv[index].masv,"");
strcpy(sv[index].tensv,"");
sv[index].gt = NULL;
sv[index].d_tp1 = 0;
sv[index].d_tp2 = 0;
sv[index].d_cc = 0;
```

```
sv[index].d_gk = 0;
sv[index].d_kt = 0;
sv[index].tong_diem = 0;
}
void capnhat(sinhvien sv[], int spt)
{
char id[10];
int cot_capnhat;
printf("Nhap ma sinh vien:");
scanf("%s",&id);
printf("Truong ban muon cap nhat(1-7)?:");
scanf("%d",&cot_capnhat);
int index = timkiem(sv, id, spt);
if (index != -1)
{
 if (cot_capnhat == 1)
 {
 printf("Nhap ten sinh vien:");
 fflush(stdin);
 scanf("%s",&sv[index].tensv);
 }
 else if (cot_capnhat == 2)
 {
 printf("Nhap gioi tinh sinh vien (F hoac M):");
 scanf("%c",&sv[index].gt);
 }
 else if (cot_capnhat == 3)
 {
 printf("Nhap diem thanh phan 1:");
 scanf("%f",&sv[index].d_tp1);
 }
 else if (cot_capnhat == 4)
 {
 printf("Nhap diem thanh phan 2:");
 scanf("%f",&sv[index].d_tp2);
```

```
 }
 else if (cot_capnhat == 5)
 {
 printf("Nhap diem chuyen can:");
 scanf("%f",&sv[index].d_cc);
 }
 else if (cot_capnhat == 6)
 {
 printf("Nhap diem giua ky:");
 scanf("%f",&sv[index].d_gk);
 }
 else if (cot_capnhat == 7)
 {
 printf("Nhap diem cuoi ky:");
 scanf("%f",&sv[index].d_kt);
 }
 else printf("Khong phai cot can cap nhat");
sv[index].tong_diem = sv[index].d_tp1 + sv[index].d_tp2 +
 sv[index].d_cc + sv[index].d_gk + sv[index].d_kt;
}
else printf("Ban ghi khong ton tai. Kiem tra lai Masv va thu lai.");
}
// Ham hien thi thong tin cua sinh vien
void hienthi(sinhvien sv[], int spt)
{
 int i = 0;
 tieude();
 while (i < spt)
 {
        if(sv[i].masv!="")
        {
                printf("%-7s",sv[i].masv);
                printf("%-10s",sv[i].tensv);
                printf("%-6c",sv[i].gt);
                printf("%-7.1f",sv[i].d_tp1);
```

```
                printf("%-7.1f",sv[i].d_tp2);
                printf("%-7.1f",sv[i].d_cc);
                printf("%-7.1f",sv[i].d_gk);
                printf("%-7.1f",sv[i].d_kt);
                printf("%-7.1f",sv[i].tong_diem);
                printf("\n");
        }
        i = i+1;
 }
 if (spt == 0) printf("Khong co ban ghi nao!\n");
}
//Ham tinh diem trung binh cua sinh vien
void tb_sv(sinhvien sv[], int spt)
{
char id[10];
float dtb = 0;
printf("Nhap ma sinh vien:");
scanf("%s",&id);
int index = timkiem(sv, id,spt);
if (index != -1 && spt>0)
{
sv[index].tong_diem = sv[index].d_tp1 + sv[index].d_tp2 +
 sv[index].d_cc + sv[index].d_gk + sv[index].d_kt;
dtb = sv[index].tong_diem/5;
printf("Diem trung binh cua sinh vien la: %f",dtb);
printf("\n");
}
else printf("Ban ghi khong ton tai. Kiem tra lai Masv va thu lai.");
}
//Ham hien thi diem cao nhat
void caonhat(sinhvien sv[], int spt)
{
float max = sv[0].tong_diem;
int index = 0,j;
if (spt >= 2)
```

```
{
  for (j = 0; j < spt-1; ++j)
 if (max < sv[j+1].tong_diem)
 {
 max = sv[j+1].tong_diem;
 index = j+1;
 }
printf("Sinh vien co ma sinh vien ID %s  co tong diem cao nhat la:
 %f",sv[index].masv,max);
printf("\n");
}
else if (spt == 1)
{
index = 0;
max = sv[0].tong_diem;
printf("Sinh vien co ma sinh vien ID %s  co tong diem cao nhat la:
 %f",sv[index].masv,max);
printf("\n");
}
else printf("Khong co ban ghi nao!\n");
}
//Ham hien thi diem thap nhat
void thapnhat(sinhvien sv[], int spt)
{
float min = sv[0].tong_diem;
int index = 0,j;
if (spt >= 2)
{
  for (j = 0; j < spt-1; ++j)
 if (min > sv[j+1].tong_diem)
 {
 min = sv[j+1].tong_diem;
 index = j+1;
 }
printf("Sinh vien co ma sinh vien ID %s  co tong diem thap nhat la: %f:
```

```
 %f",sv[index].masv,min);
printf("\n");
}
else if (spt == 1)
{
index = 0;
min = sv[0].tong_diem;
printf("Sinh vien co ma sinh vien ID %s  co tong diem thap nhat la: %f:
 %f",sv[index].masv,min);
printf("\n");
}
else printf("Khong co ban ghi nao!\n");
}
//Ham tim kiem thong tin sinh vien
void find(sinhvien sv[], int spt)
{
char id[10];
printf("Nhap ma sinh vien can tim:");
scanf("%s",&id);
int index=timkiem(sv,id, spt);
if (index != -1)
{ //hien thi thong tin ban ghi tim thay
tieude();
printf("%-7s",sv[index].masv);
printf("%-10s",sv[index].tensv);
printf("%-6c",sv[index].gt);
printf("%-7.1f",sv[index].d_tp1);
printf("%-7.1f",sv[index].d_tp2);
printf("%-7.1f",sv[index].d_cc);
printf("%-7.1f",sv[index].d_gk);
printf("%-7.1f",sv[index].d_kt);
printf("%-7.1f",sv[index].tong_diem);
printf("\n");
}
else printf("Ban ghi khong ton tai.\n");
```

```
}
//Ham sap xep theo tong diem
void Sx_theo_tong(sinhvien sv[], int n)
{
int i, j;
sinhvien tg;
for (i = 0; i < n; i++)
 for (j = n - 1; j > i; j--)
            if (sv[j].tong_diem < sv[j - 1].tong_diem )
            {
            tg = sv[j];
            sv[j] = sv[j - 1];
            sv[j - 1] = tg;
            }
}
void tieude()
{
printf("MASV TENSV GT D_TP1 D_TP2 D_CC D_GK D_KT TONG
 \n   ");
printf("==========================================
 ==\n");
int main()
{
 sinhvien sv[MAX];
 int spt=0;
 hienthi_menu();
 int luachon;
 char ok;
 do
 {
            printf("Lua chon cong viec(1-9):"); scanf("%d", &luachon);
            switch(luachon)
            {
                        case 1: Themsv(sv, &spt); break;
                        case 2: xoasv(sv, &spt); break;
```

```
                case 3: capnhat(sv, spt); break;
                case 4: hienthi(sv, spt); break;
                case 5: tb_sv(sv, spt);break;
                case 6: caonhat(sv, spt);break;
                case 7: thapnhat(sv, spt);break;
                case 8:find(sv, spt);break;
                case 9: {
                            Sx_theo_tong(sv, spt);
                                hienthi(sv, spt);
                                break;
                           }
                default: printf("Khong co lua chon nay\n"); break;
        }
        printf("Nhan phim y de tiep tuc:");
        scanf("%s", &ok);
  } while (ok == 'y');
  if (ok!='y') printf("CAM ON BAN DA THUC HIEN CHUONG TRINH!\n");
  return 0;
 }
```

✍Kết quả chương trình

```
----------------------------------------
          CHUONG TRINH CHINH
----------------------------------------
 1. Them ho so sinh vien
 2. Xoa ho so sinh vien
 3. Cap nhat ho so sinh vien
 4. Hien thi toan bo thong tin sinh vien
 5. Tinh diem tong ket cua sinh vien
 6. Hien thi sinh vien co diem tong ket cao nhat
 7. Hien thi sinh vien co diem tong ket thap nhat
 8. Tim kiem sinh vien dua vao ma sinh vien
 9. Sap xep danh sach sinh vien dua vao diem tong ket
Lua chon cong viec(1-9):1
Nhap vao ma sinh vien:sv01
```

Nhap ten sinh vien:An Huy
Nhap gioi tinh sinh vien:(F hoac M):M
Nhap diem thanh phan thu nhat:7
Nhap diem thanh phan thu hai:8
Nhap diem chuyen can:7
Nhap diem giua ky:9
Nhap diem ket thuc mon:8
Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):4

MASV	TENSV	GT	D_TP1	D_TP2	D_CC	D_GK	D_KT	TONG
sv01	An Huy	M	7.0	8.0	7.0	9.0	8.0	39.0

Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):1
Nhap vao ma sinh vien:sv01
Ma sinh vien da ton tai
Nhap vao ma sinh vien:sv02
Nhap ten sinh vien:Kien
Nhap gioi tinh sinh vien:(F hoac M):M
Nhap diem thanh phan thu nhat:8
Nhap diem thanh phan thu hai:9
Nhap diem chuyen can:8
Nhap diem giua ky:9
Nhap diem ket thuc mon:8
Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):4

MASV	TENSV	GT	D_TP1	D_TP2	D_CC	D_GK	D_KT	TONG
sv01	An Huy	M	7.0	8.0	7.0	9.0	8.0	39.0
sv02	Kien	M	8.0	9.0	8.0	9.0	8.0	42.0

Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):3
Nhap ma sinh vien:sv02
Truong ban muon cap nhat(1-7)?:1
Nhap ten sinh vien:Luan
Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):3

Nhap ma sinh vien:sv02
Truong ban muon cap nhat(1-7)?:7
Nhap diem cuoi ky:9
Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):4

MASV	TENSV	GT	D_TP1	D_TP2	D_CC	D_GK	D_KT	TONG
sv01	An Huy	M	7.0	8.0	7.0	9.0	8.0	39.0
sv02	Luan	M	8.0	9.0	8.0	9.0	9.0	43.0

Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):5
Nhap ma sinh vien:sv01
Diem trung binh cua sinh vien la: 7.800000
Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):6
Sinh vien co ma sinh vien ID sv02 co tong diem cao nhat la: 43.0
Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):7
Sinh vien co ma sinh vien ID sv01 co tong diem thap nhat la: 39.0
Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):9

MASV	TENSV	GT	D_TP1	D_TP2	D_CC	D_GK	D_KT	TONG
sv01	An Huy	M	7.0	8.0	7.0	9.0	8.0	39.0
sv02	Luan	M	8.0	9.0	8.0	9.0	9.0	43.0

Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):2
Nhap ma sv can xoa:sv01
Nhan phim y de tiep tuc:y
Lua chon cong viec(1-9):4

MASV	TENSV	GT	D_TP1	D_TP2	D_CC	D_GK	D_KT	TONG
sv02	Luan	M	8.0	9.0	8.0	9.0	9.0	43.0

Nhan phim y de tiep tuc:K
CAM ON BAN DA THUC HIEN CHUONG TRINH!

BÀI TẬP CHƯƠNG 6

A. BÀI TẬP CÓ LỜI GIẢI

🕮 Bài tập 6.1

Viết chương trình quản lý các nhân viên của một công ty. Biết rằng thông tin của một nhân viên bao gồm: mã nhân viên, tên nhân viên và tuổi của nhân viên.

Hiển thị menu sau đây cho phép người dùng lựa chọn:

CHUONG TRINH CHINH

---

1. Nhap thong tin nhan vien
2. In danh sach nhan vien da sap xep theo ten
3. In danh sach nhan vien sap xep theo tuoi

Nếu người dùng lựa chọn 1, cho phép nhập thông tin của một nhân viên. Nếu người dùng chọn 2, cho phép hiển thị danh sách nhân viên theo tên, nếu người dùng chọn 3 cho phép hiển thị danh sách nhân viên theo tuổi. Quá trình thực hiện chương trình kết thúc nếu người dùng nhập vào 1 phím khác phím y, nhập y để tiếp tục công việc.

✍Chương trình:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  struct Nhanvien
5  {
6          char tennv[30];
7          char manv[30];
8          int tuoi;
9  };
10
11 void tieude();
12 void hienthi();
13 int timkiem(Nhanvien ds_nv[], char id[], int n);
14 void tieude()
```

```
{
	printf("--------------------------------\n");
	printf("|  TEN NV  |   MANV   | TUOi  | \n");
	printf("--------------------------------\n");
}
void hienthi(Nhanvien ds_nv[], int n)
{
	int i = 0;
	tieude();
	if (n == 0) printf("Danh sach nhan vien rong!\n ");
	while(i < n)
	{
		if(ds_nv[i].manv!="")
			{
				printf("%-27s", ds_nv[i].tennv);
				printf("%-15s", ds_nv[i].manv);
				printf("%-5d", ds_nv[i].tuoi);
				printf("\n");
			}
		i++;
	}
}
void Themnv(Nhanvien ds_nv[], int *n)
{
kt:
	printf("Nhap ma nhan vien:");
	scanf("%s", &ds_nv[*n].manv);
	fflush(stdin);
	if (timkiem(ds_nv, ds_nv[*n].manv, *n)!=-1)
		{
			printf("Ma nhan vien da ton tai\n");
			goto kt;
		}
	printf("Nhap ten nhan vien:");
	fflush(stdin);
```

```
50          gets(ds_nv[*n].tennv);
51          printf("Nhap tuoi nhan vien:");
52          scanf("%d", &ds_nv[*n].tuoi);
53  (*n)++;
54  }
55  int timkiem(Nhanvien ds_nv[], char id[], int n)
56  {
57          int timthay = -1;
58          for (int i = 0; i<n && timthay == -1; i++)
59          {
60                  if (strcmp(ds_nv[i].manv, id) == 0) timthay = i;
61                  else timthay = -1;
62          }
63  return timthay;
64  }
65  void In_ds_theo_ten(Nhanvien *ds_nv, int n)
66  {
67          int i, j;
68          Nhanvien tg;
69          for(i = 0; i < n; i++)
70                  for (j = i+1; j < n; j++)
71                  {
72                          if(strcmp((ds_nv+i)->tennv,      (ds_nv+j)-
    >tennv)>0)
73                          {
74                          tg = *(ds_nv+j);
75                          *(ds_nv+j) = *(ds_nv+i);
76                          *(ds_nv+i) = tg;
77                          }
78                  }
79          hienthi(ds_nv, n);
80  }
81  void In_ds_theo_tuoi(Nhanvien *ds_nv, int n)
82  {
83          int i, j;
```

```
84          Nhanvien tg;
85          for(i = 0; I < n; i++)
86                  for (j = i+1; j < n; j++)
87                  {
88                          if((ds_nv+i)->tuoi < (ds_nv+j)->tuoi)
89                          {
90                          tg = *(ds_nv+j);
91                          *(ds_nv+j) = *(ds_nv+i);
92                          *(ds_nv+i) = tg;
93                          }
94                  }
95  hienthi(ds_nv, n);
96  }
97  int main()
98  {
99          char ok;
100         int luachon;
101         Nhanvien ds_nv[20];
102         int n = 0;
103         printf("-----------------------------------\n");
104         printf("     CHUONG TRINH CHINH          \n");
105         printf("-----------------------------------\n");
106         printf("1. Nhap thong tin nhan vien\n");
107         printf("2. In danh sach nhan vien da sap xep theo ten\n");
108         printf("3. In danh sach nhan vien sap xep theo tuoi\n\n");
109 do
110 {
111         printf("Nhap vao lua chon (1-3): ");
112         scanf("%d", &luachon);
113         switch (luachon)
114         {
115         case 1: Themnv(ds_nv, &n); break;
116         case 2: In_ds_theo_ten(ds_nv, n); break;
117         case 3: In_ds_theo_tuoi(ds_nv, n); break;
118         default: printf("Khong ton tai lua chon nay!\n");
```

```
119             }
120  printf("Nhan phim y de tiep tuc:"); scanf("%s", &ok);
121  }while(ok == 'y');
122  return 0;
123  }
```

↘Kết quả chương trình

```
---------------------------------------------------
                CHUONG TRINH CHINH
---------------------------------------------------
1. Nhap thong tin nhan vien
2. In danh sach nhan vien da sap xep theo ten
3. In danh sach nhan vien sap xep theo tuoi
Nhap vao lua chon (1-3): 1
Nhap ma nhan vien:nv01
Nhap ten nhan vien:Luan
Nhap tuoi nhan vien:40
Nhan phim y de tiep tuc:y
Nhap vao lua chon (1-3): 1
Nhap ma nhan vien:nv02
Nhap ten nhan vien:Kien
Nhap tuoi nhan vien:40
Nhan phim y de tiep tuc:y
Nhap vao lua chon (1-3): 1
Nhap ma nhan vien:nv03
Nhap ten nhan vien:Thuan
Nhap tuoi nhan vien:38
Nhan phim y de tiep tuc:y
Nhap vao lua chon (1-3): 1
Nhap ma nhan vien:nv04
Nhap ten nhan vien:Chi
Nhap tuoi nhan vien:36
Nhan phim y de tiep tuc:y
Nhap vao lua chon (1-3): 2
```

```
-------------------------------------------------
|     TEN NV        |    MANV     |   TUOI   |
-------------------------------------------------
          Chi                 nv04         36
          Kien                nv02         40
          Luan                nv01         40
          Thuan               nv03         38
Nhan phim y de tiep tuc:y
Nhap vao lua chon (1-3): 3
-------------------------------------------------
|     TEN NV        |    MANV     |   TUOI   |
-------------------------------------------------
          Kien                nv02         40
          Luan                nv01         40
          Thuan               nv03         38
          Chi                 nv04         36
```

B. BÀI TẬP TỰ GIẢI

🕮 Bài tập 6.2

Cho biết kết quả chương trình và giải thích:

```
1    #include <stdio.h>
2    #include <string.h>
3    #include <stdlib.h>>
4    struct personal
5    {
6    char name[20];
7    int day;
8    char month[10];
9    int year;
10   float salary;
11   };
12   main()
13   {
14   struct personal person;
15   printf("Input Values\n");
```

```
16    scanf("%s %d %s %d %f", person.name, &person.day, person.month,
17    &person.year, &person.salary);
18    printf("%s %d %s %d %f\n", person.name, person.day, person.month,
19    person.year, person.salary);
20    }
```

🕮 Bài tập 6.3

a. Viết chương trình sử dụng con trỏ cấu trúc để hiển thị giờ, phút, giây ra màn hình và tính khoảng cách giữa 2 mốc thời gian.

b. Viết chương trình sử dụng con trỏ cấu trúc thể hiện ngày, tháng, năm ra màn hình và tính khoảng cách giữa 2 ngày.

c. Viết chương trình khai báo kiểu dữ liệu thể hiện một số phức. Sử dụng kiểu này để viết hàm tính tổng, hiệu, tích của hai số phức.

d. Viết chương trình khai báo kiểu dữ liệu để biểu diễn một phân số. Hãy viết hàm thực hiện những công việc sau:

- Tính tổng, hiệu, tích, thương hai phân số.

- Rút gọn phân số.

- Quy đồng hai phân số.

- So sánh hai phân số.

🕮 Bài tập 6.4.

Viết chương trình quản lý thông tin của các sinh viên trong một lớp học, biết rằng thông tin của một sinh viên bao gồm: mã sinh viên, tên sinh viên, điểm các môn CSDL, Toán rời rạc, Lập trình C.

+ Nhập một danh sách n(1<n<50) sinh viên.
+ Đối với mỗi sinh viên, hãy tính điểm trung bình và phân loại như sau:
  - DTB = (Điểm CSDL + Điểm Toán rời rạc + Điểm Lập trình C)/3
  - Và phân loại:
    - Xuất sắc, nếu DTB > 9.0
    - Giỏi, nếu 9.0 > DTB > 8.0
    - Khá, nếu 8 > DTB > 7.0
    - Trung bình, nếu 7.0 > DTB > 5.0
    - Kém, nếu DTB < 5.0

+ Nhập vào một mã sinh viên, hiển thị thông tin của sinh viên có mã sinh viên vừa nhập;

+ Nhập vào một mã sinh viên, xóa sinh viên trùng với mã sinh viên trên;

+ Sắp xếp danh sách trên theo thứ tự giảm dần của DTB, in danh sách trên lên màn hình, mỗi người trên một dòng gồm các mục: Stt, Mã sinh viên, Họ tên, điểm CSDL, điểm Toán rời rạc, điểm lập trình C, điểm DTB, xếp loại của sinh viên.

Yêu cầu:

- Thể hiện mỗi ý của bài tập bằng một hàm.

🕮 Bài tập 6.5.

Viết chương trình quản lý thông tin của các sách trong một cửa hàng. Thông tin của một cuốn sách bao gồm: Mã sách, tên sách, số lượng, tác giả, năm xuất bản và đơn giá.

+ Nhập một danh sách n đầu sách (1 < b < 50).

+ Với mỗi đầu sách tính giá tiền(GT) như sau:

o GT= SL * DG nếu SL < 10

= SL * DG - 5% SL * DG nếu 10 <= SL <= 30

= SL * DG - 10% SL * DG nếu 31 <= SL <= 50

= SL * DG - 5% SL * DG nếu SL >= 51

+ Tính tổng số tiền chi cho việc mua sách kể cả 10 % thuế giá trị gia tăng đánh vào tổng giá tiền.

+ Tìm xem trong danh sách có sách “Ngon ngu lap trinh C” không? Nếu có thì cho biết có bao nhiêu cuốn và đơn giá của nó.

Yêu cầu:

- Thể hiện mỗi ý của bài tập bằng một hàm.

🕮 Bài tập 6.6

Thông tin về một mặt hàng trong siêu thị bao gồm: Mã mặt hàng (ký tự, 10), Tên mặt hàng (ký tự, 20), màu sắc (ký tự, 10), năm sản xuất (số nguyên), giá bán (số nguyên).

Hãy thực hiện các yêu cầu sau đây bằng ngôn ngữ lập trình C

a. Nhập vào thông tin của n sản phẩm trong siêu thị.

b. Viết hàm liệt kê mã và tên của các mặt hàng có năm sản xuất là 2019, có bao nhiêu mặt hàng như vậy, cho biết số lượng đếm được.

c. Viết hàm đếm các mặt hàng có màu đỏ và có giá bán lớn hơn 200000. Nếu không có thì thông báo không có, nếu có cho biết số lượng đếm được.

d. Viết hàm liệt kê ra các mặt hàng có giá bán thấp nhất trong siêu thị.

e. Viết hàm nhập vào một mã số x (x- ký tự), kiểm tra và trả lời có mặt hàng nào có mã số x trong siêu thị hay không?

f. Viết hàm sắp xếp lại danh sách giảm dần theo năm sản xuất.

Yêu cầu:

- Thể hiện mỗi ý của bài tập bằng một hàm.

🕮 Bài tập 6.7

Thông tin về cầu thủ của một đội bóng bao gồm: Mã cầu thủ (chuỗi ký tự, độ dài tối đa 10), Tên cầu thủ (chuỗi ký tự, 20), quê quán (chuỗi ký tự, 30), tuổi (số nguyên), vị trí (chuỗi ký tự, 15), hợp đồng (số nguyên).

Hãy thực hiện các yêu cầu sau đây bằng ngôn ngữ lập trình C

a. Nhập thông tin của n cầu thủ trong một câu lạc bộ.

b. Viết hàm liệt kê mã và tên của các cầu thủ đá ở vị trí tiền đạo, cho biết có bao nhiêu cầu thủ đá ở vị trí này.

c. Viết hàm cho biết có bao nhiêu cầu thủ sắp giải nghệ (có tuổi lớn hơn 32). Nếu không có thì thông báo không có, nếu có cho biết số lượng đếm được.

d. Viết hàm in ra các cầu thủ có số năm hợp đồng bé hơn 2 (để câu lạc bộ có thể biết để ký tiếp hợp đồng khác)

e. Viết hàm nhập vào tên của một cầu thủ, cho biết có cầu thủ này trong đội bóng hay không?

f. Viết hàm xoá một cầu thủ có mã số là x, x nhập vào từ bàn phím.

🕮 Bài tập 6.8

Viết chương trình thực hiện các công việc sau đây:

a. Khai báo cấu trúc nhân viên, biết rằng thông tin của một nhân viên: Mã số nhân viên (kiểu ký tự), họ tên nhân viên (30 ký tự), ngày sinh (kiểu ngày), lương (số thực), giới tính (0: nữ, 1: nam).

b. Viết hàm nhập thông tin cho n nhân viên, n được nhập từ bàn phím.

c. Liệt kê tên các nhân viên sinh năm 1972.

d. Đếm số lượng nhân viên có lương lớn hơn 1 triệu.

e. Xoá một nhân viên có mã số bằng x, x nhập từ bàn phím, nếu không có thì thông báo là không có.

f. Sắp xếp các nhân viên giảm dần theo năm sinh.

🕮 Bài tập 6.9

Tổ chức dữ liệu để quản lí sinh viên bằng cấu trúc mẫu tin trong một mảng N phần tử, mỗi phần tử có cấu trúc như sau:

- *Mã sinh viên.*

- *Tên.*

- *Năm sinh.*

- *Điểm toán, lý, hoá, điểm trung bình.*

Viết chương trình thực hiện những công việc sau:

a. Nhập danh sách các sinh viên cho một lớp học.

b. Xuất danh sách sinh viên ra màn hình.

c. Tìm sinh viên có điểm trung bình cao nhất.

d. Sắp xếp danh sách lớp theo thứ tự tăng dần của điểm trung bình.

e. Sắp xếp danh sách lớp theo thứ tự giảm dần của điểm toán.

f. Tìm kiếm và in ra các sinh viên có điểm trung bình > 5 và không có môn nào dưới 3.

g. Tìm sinh viên có tuổi lớn nhất.

h. Nhập vào tên của sinh viên. Tìm và in ra các thông tin liên quan đến sinh viên đó (nếu có).

🕮 Bài tập 6.10

Tổ chức dữ liệu quản lí danh mục các bộ phim VIDEO, các thông tin liên quan đến bộ phim này như sau:

- *Tên phim (tựa phim).*

- *Thể loại (3 loại: hình sự, tình cảm, hài).*

- *Tên đạo diễn.*

- *Tên điễn viên nam chính.*

- *Tên diễn viên nữ chính.*

- *Năm sản xuất.*

- *Hãng sản xuất*

Viết chương trình thực hiện những công việc sau:

a. Nhập vào bộ phim mới cùng với các thông tin liên quan đến bộ phim này.

b. Nhập một thể loại: In ra danh sách các bộ phim thuộc thể loại này.

c. Nhập một tên nam diễn viên. In ra các bộ phim có diễn viên này đóng.

d. Nhập tên đạo diễn. In ra danh sách các bộ phim do đạo diễn này dàn dựng.

🕮 Bài tập 6.11

Một thư viện cần quản lý thông tin về các đầu sách. Mỗi đầu sách bao gồm các thông tin sau: MaSSach (mã số sách), TenSach (tên sách), TacGia (tác giả), SL (số lượng các cuốn sách của đầu sách).

Viết chương trình thực hiện các chức năng sau:

a. Nhập vào một danh sách các đầu sách (tối đa là 100 đầu sách).

b. Nhập vào tên của quyển sách. In ra thông tin đầy đủ về các sách có tên đó, nếu không có thì tên của quyển sách đó thì báo là: Không Tìm Thấy.

c. Tính tổng số sách có trong thư viện.

🕮 Bài tập 6.12

Viết chương trình quản lý vé tàu, thông tin một vé tàu như sau:

- *Ngày giờ khởi hành, ngày giờ đến.*

- *Ga đi, ga đến.*

- *Loại tàu, loại chỗ ngồi (ngồi, nằm, cứng, mềm).*

- *Số toa, số ghế.*

a. Viết hàm nhập vào danh sách các vé tàu.

b. In danh sách các vé tàu có ga đến là Huế.

c. In danh sách các vé tàu có ga đến là Hà Nội và đi ngày 08/8/2019.

d. Đếm xem có bao nhiêu khách đi tàu loại chỗ ngồi là *nằm cứng*.

CHƯƠNG 7
TỆP

Nếu máy tính của bạn chỉ có thể xử lý dữ liệu được lưu trữ trong bộ nhớ chính của máy (Ram) thì phạm vi và sự đa dạng của các ứng dụng của bạn có thể xử lý sẽ bị hạn chế nghiêm trọng. Hầu hết tất cả ứng dụng trong kinh doanh đều đòi hỏi về phạm vi lưu trữ dữ liệu bên ngoài bộ nhớ chính và sự linh động phụ thuộc vào khả năng xử lý dữ liệu được lưu trữ trên một thiết bị mở rộng như ổ đĩa cứng. Ngôn ngữ lập trình C cung cấp một số hàm trong thư viện stdio.h phục vụ cho quá trình đọc, ghi dữ liệu từ các tệp được lưu trữ trên các thiết bị mở rộng. Các thư viện được sử dụng để làm việc với các tệp độc lập với các thiết bị, được áp dụng cho hầu hết tất cả các thiết bị lưu trữ mở rộng. Tuy nhiên, trong phạm vi của chương này, trong tất cả các ví dụ của Chương 7, chúng ta đều xử lý và truy xuất các tệp được lưu trữ trên các ổ cứng của máy.

Nội dung của Chương 7 đề cập đến các vấn đề sau đây:

- Tập tin là gì?
- Cách xử lý tệp tin.
- Cách đọc và ghi dữ liệu vào tệp tin theo định dạng.
- Cách đọc và ghi dữ liệu với tệp nhị phân.
- Cách đọc và ghi dữ liệu với tệp ngẫu nhiên.

7.1. KHÁI NIỆM VỀ TỆP TIN

7.1.1. Khái niệm tệp

Trong hầu hết các ví dụ và bài tập từ chương một đến chương sáu, mọi dữ liệu mà người dùng nhập sẽ mất khi chương trình kết thúc. Nếu người dùng muốn chạy chương trình có dữ liệu, thao tác đều tiên yêu cầu người dùng nhập lại mỗi lần biên dịch chương trình. Với những chương trình nhỏ, dữ liệu không lớn, việc nhập dữ liệu có thể dễ dàng, chưa đưa đến cảm giác khó chịu cho người dùng, nhưng với những chương trình mà bộ dữ liệu tương đối lớn, chẳng hạn chương trình quản lý nhân sự, với 100 nhân viên, mỗi nhân viên đều có những thông tin cá nhân như họ tên, mã nhân viên, địa chỉ, bậc lương, v.v. với mỗi lần biên dịch để xử lý một nghiệp vụ nào đó, người dùng bắt buộc phải nhập lại dữ liệu của tất cả các nhân viên, lúc này vấn đề không chỉ là sự bất tiện, mà đôi khi nó còn khiến cho nhiệm vụ lập trình trở nên bất khả thi.

Đáp án cho những khó khăn trên là lưu trữ dữ liệu trên bộ nhớ ngoài (vĩnh viễn), giúp cho dữ liệu được duy trì và tái sử dụng sau khi

máy tính bị tắt. Dữ liệu được lưu dưới dạng các tệp, sẽ bảo vệ dữ liệu của chương trình ngay cả khi chương trình kết thúc. Ngoài ra, nếu chúng ta có một tệp chứa tất cả dữ liệu, chúng ta có thể dễ dàng truy cập nội dung của tệp bằng vài lệnh trong C. Chúng ta có thể dễ dàng di chuyển dữ liệu của mình từ máy tính này sang máy tính khác mà không có bất kỳ thay đổi nào.

Theo định nghĩa của Wikipedia về tệp (file): Một tệp là một tài nguyên dùng để lưu trữ thông tin lâu dài, sử dụng cho các chương trình máy tính.

Cũng giống như việc lưu trữ dữ liệu tạm thời trên RAM, tệp cũng lưu trữ dữ liệu dưới dạng nhị phân (0 hoặc 1), tuy nhiên tùy vào định dạng của tệp và cách chuyển đổi của mỗi phần mềm đọc tệp mà chúng ta có những kiểu thông tin khác nhau. Ví dụ tệp .png thì được chuyển về dạng hình ảnh, phần mềm Microsoft Word chuyển dãy bit nhị phân về dạng text, v.v.

Trong ngôn ngữ lập trình C: Tệp là kiểu đối tượng, nó xác định một stream và chứa các thông tin cần thiết để điều khiển, bao gồm một con trỏ trỏ đến buffer của nó, các chỉ mục và trạng thái của nó.

7.1.2. Phân loại tệp

Trong ngôn ngữ lập trình C, có hai loại tệp được đề cập:

Tập tin văn bản.

Tệp nhị phân.

7.1.2.1. Tệp văn bản

Các tệp văn bản là các tệp .txt, có thể dễ dàng tạo bằng Notepad hoặc bất kỳ trình soạn thảo văn bản đơn giản nào. Khi chúng ta mở các tệp đó, sẽ thấy tất cả nội dung trong tệp dưới dạng văn bản thuần túy, có thể dễ dàng chỉnh sửa hoặc xóa nội dung.

7.1.2.2. Tệp nhị phân

Các tệp nhị phân chủ yếu là các tệp .bin trong máy tính. Thay vì lưu trữ dữ liệu trong văn bản thuần túy, dữ liệu được lưu trữ ở dạng nhị phân (0 và 1). Chúng có thể chứa lượng dữ liệu cao hơn, không thể đọc được dễ dàng và cung cấp bảo mật tốt hơn các tệp văn bản.

7.2. THAO TÁC XỬ LÝ VỚI TỆP

7.2.1. Các thao tác với tệp tin

Trong ngôn ngữ lập trình C, có thể thực hiện bốn thao tác chính trên tệp, dưới dạng tệp văn bản hoặc tệp nhị phân:

- Tạo một tệp mới.
- Mở tệp tin hiện có.
- Đóng tệp tin.
- Đọc và ghi nội dung trong tệp tin.

7.2.2. Khai báo tệp tin

Khi làm việc với các tệp, bạn cần khai báo một con trỏ của loại tệp. Khai báo này là cần thiết để liên lạc giữa các tập tin và chương trình.

Cú pháp:

```
FILE * fptr;
```

Trong đó fptr là biến con trỏ, đại diện cho tệp tin.

🕮 Ví dụ 7.1

```
FILE * f1, f2;
```

Trong đó f1, f2 là các biến con trỏ, đại diện cho tệp tin.

7.2.3. Mở tệp tin

Việc mở tệp được thực hiện thông hàm fopen() trong thư viện “stdio.h”.

Cú pháp mở một tệp tin tiêu chuẩn I/O:

```
ptr = fopen("fileopen","mode")
```

🕮 Ví dụ 7.2

```
fopen("E:\\cprogram\\data1.txt","w");
```

Giả sử tệp data1.txt không tồn tại ở thư mục E:\cprogram (cprogram là một thư mục trên máy của người dùng, lưu các chương trình viết bằng ngôn ngữ C). Khi đó việc đầu tiên hàm fopen() sẽ tạo ra một tệp trong thưc mục E:\cprogram có tên data1.txt và mở tệp tin để ghi theo chế độ “w”. Chế độ ghi cho phép bạn tạo và chỉnh sửa (ghi đè) nội dung của tệp tin.

```
fopen("E:\\cprogram\\data2.bin","rb");
```

Giả sử tệp nhị phân thứ hai data2.bin tồn tại ở vị trí E:\ cprogram. Hàm fopen() mở tệp hiện có để đọc ở chế độ nhị phân 'rb'. Chế độ đọc chỉ cho phép bạn đọc tệp, bạn không thể ghi vào tệp.

Các chế độ mở tệp trong chế độ chuẩn I/O:

Bảng 7.1. Mở tệp tin trong chế độ chuẩn I/O

Chế độ	Ý nghĩa
r	Mở để đọc Nếu tệp không tồn tại, hàm fopen() trả về giá trị NULL
rb	Mở để đọc ở chế độ nhị phân Nếu tệp không tồn tại, hàm fopen() trả về giá trị NULL
w	Mở để ghi Nếu tập tin tồn tại, nội dung của nó được ghi đè. Nếu tập tin không tồn tại, nó sẽ được tạo mới với tên tệp đã được ấn định trong khai báo của hàm fopen()
wb	Mở để ghi ở chế độ nhị phân Nếu tập tin tồn tại, nội dung của nó được ghi đè. Nếu tập tin không tồn tại, nó sẽ được tạo mới với tên tệp đã được ấn định trong khai báo của hàm fopen()
a	Mở để ghi tiếp, dữ liệu được ghi tiếp ở cuối tệp Nếu tệp tin không tồn tại, nó sẽ được tạo mới
ab	Mở để ghi tiếp ở chế độ nhị phân, dữ liệu được ghi tiếp ở cuối tệp. Nếu tệp tin không tồn tại, nó sẽ được tạo mới
r+	Mở tệp cho phép đọc và ghi Nếu tệp không tồn tại, hàm fopen() trả về giá trị NULL
rb+	Mở tệp cho phép đọc và ghi ở chế độ nhị phân Nếu tệp không tồn tại, hàm fopen() trả về giá trị NULL
w+	Mở tệp cho phép đọc và ghi Nếu tập tin tồn tại, nội dung của nó được ghi đè. Nếu tập tin không tồn tại, nó sẽ được tạo mới với tên tệp đã được ấn định trong khai báo của hàm fopen()
wb+	Mở tệp cho phép đọc và ghi ở chế độ nhị phân Nếu tập tin tồn tại, nội dung của nó được ghi đè. Nếu tập tin không tồn tại, nó sẽ được tạo mới với tên tệp đã được ấn định trong khai báo của hàm fopen()
a+	Mở tệp cho phép đọc và ghi tiếp ở cuối tệp Nếu tệp tin không tồn tại, nó sẽ được tạo mới
ab+	Mở tệp cho phép đọc và ghi tiếp ở cuối tệp ở chế độ nhị phân Nếu tệp tin không tồn tại, nó sẽ được tạo mới

Lưu ý:

- Hàm fopen() trả về một con trỏ tệp tin. Chương trình sẽ không thay đổi giá trị của con trỏ này. Nếu có lỗi xuất hiện khi mở tệp tin thì hàm trả về giá trị NULL.
- Muốn mở tệp tin chỉ cho phép đọc, thì tệp tin đó phải tồn tại, nếu không một lỗi sẽ xuất hiện.
- Phân biệt chế độ mở tệp để ghi đè (w, wb, w+, wb+) với mở tệp để ghi tiếp (a, ab, a+, ab+).

7.2.4. Đóng tệp

Tệp tin (cả tệp văn bản và tệp nhị phân) sẽ được đóng sau thao tác đọc/ghi tệp. Đóng tệp được thực hiện bằng hàm fclose() trong thư viện "stdio.h"

Cú pháp:

```
fclose(fptr);
```

Trong đó: fptr là biến con trỏ được liên kết với tệp tin sẽ được đóng.

7.2.5. Đọc và ghi với tệp văn bản

Để đọc và ghi tệp văn bản, chúng ta sử dụng các câu lệnh fprintf() và fscanf(). Trong đó fprintf() và fscanf() là các phiên bản của lệnh printf() và scanf() được áp dụng với tệp. Sự khác biệt duy nhất, lệnh fprintf() và fscanf() mong đợi một con trỏ trỏ tới cấu trúc FILE.

🕮 Ví dụ 7.3

Ghi vào tệp văn bản với lệnh fprintf()

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main()
4  {
5    int num;
6    FILE *fptr;
7    fptr = fopen("data.txt","w");
8    if(fptr == NULL)
9    {
10     printf("Loi!");
11     exit(1);
12   }
```

```
13      printf("Nhap vao so: ");
14      scanf("%d",&num);
15      fprintf(fptr,"%d",num);
16      fclose(fptr);
17      return 0;
18   }
```

✍ Kết quả chương trình:

Nhap vao so: 15

Mở tệp data.txt trong thư mục chứa tệp chương trình:

15

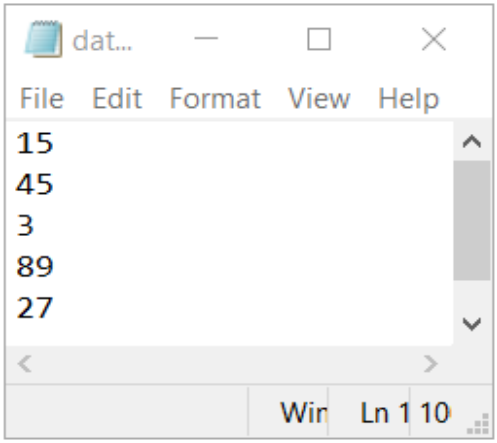
Số 15 đã được ghi vào tệp

🕮 Ví dụ 7.4

✍ Đọc tệp văn bản với lệnh fscanf()

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int main()
4   {
5      int num;
6      FILE *fptr;
7      if ((fptr = fopen("data1.txt","r")) == NULL){
8         printf("Loi! khi mo tep");
9         //Chuong trinh se thoat neu con tro tep tra ve NULL
10        exit(1);
11     }
12     fscanf(fptr,"%d", &num);
13     printf("Gia tri cua n = %d", num);
14     fclose(fptr);
15     return 0;
16  }
```

Giả sử tệp data1.txt đang nằm cùng thư mục với tệp chương trình, chứa dữ liệu như sau:



Kết quả chương trình:

Gia tri cua n = 15

Chương trình trên sẽ đọc một số nguyên có trong tệp và in ra màn hình. Nếu chúng ta đã tạo thành công tệp từ ví dụ 7.3, khi chạy chương trình này sẽ in ra số nguyên đã nhập và ghi vào tệp data.txt trong ví dụ 7.3.

Tại sao khi in, chương trình chỉ in giá trị đầu tiên trong tệp? Lý do trong chương trình, chúng ta chỉ sử dụng câu lệnh fscanf() để đọc dòng đầu tiên của tệp data1.txt.

Nếu muốn in tất cả các giá trị được ghi trong tệp data1.txt, ta cần thêm các dòng lệnh để đọc từng dòng của tệp, sử dụng vòng lặp while để đọc hết tệp:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int main()
4   {
5     int num;
6     FILE *fptr;
7     if ((fptr = fopen("data1.txt","r")) == NULL){
8        printf("Loi! khi mo tep");
9      //Chuong trinh se thoat neu con tro tep tra ve NULL
10       exit(1);
11    }
12    while (!feof(fptr))
13    {
```

```
    fscanf(fptr,"%d ", &num);
    printf("%d\n", num);
    }
    fclose(fptr);
    return 0;
}
```

Lưu ý:

Các hàm khác như fgetchar(), fputc(), v.v. có thể được sử dụng theo cách tương tự.

7.2.6. Đọc và ghi tệp nhị phân

Các hàm fread() và fwrite() được sử dụng để đọc và ghi tệp nhị phân.

7.2.6.1. Ghi vào tệp nhị phân

Để ghi vào tệp nhị phân, chúng ta sử dụng câu lệnh fwrite/

Cú pháp:

fwrite(address_data, size_data, numbers_data, pointer_to_file);

Trong đó:

address_data: Địa chỉ của dữ liệu được ghi vào thiết bị nhớ ngoài;

size_data: Kích thước của dữ liệu;

numbers_data: Số kiểu dữ liệu;

pointer_to_file: Con trỏ trỏ tới tệp

🕮 Ví dụ 7.5

Xem xét đoạn chương trình sau đây:

```
#include <stdio.h>
#include <stdlib.h>
struct threeNum
{
   int n1, n2, n3;
};
int main()
{
   int n;
   struct threeNum num;
```

```
11    FILE *fptr;
12    if ((fptr = fopen("databinary.bin","wb")) == NULL){
13        printf("Loi!");
14        // chuong trinh thoat neu con tro tep tra ve gia tri NULL
15        exit(1);
16    }
17    for(n = 1; n < 5; ++n)
18    {
19      num.n1 = n;
20      num.n2 = 5*n;
21      num.n3 = 5*n + 1;
22      fwrite(&num, sizeof(struct threeNum), 1, fptr);
23    }
24    fclose(fptr);
25    return 0;
26  }
```

Trong ví dụ trên, đầu tiên người lập trình tạo ra tệp nhị phân databinary.bin trong cùng vị trí với tệp chạy chương trình.

Khai báo cấu trúc threeNum với thành phần là 3 số n1, n2, n3 và định nghĩa 1 biến cấu trúc trong hàm chính là num;

Khai báo biến con trỏ fptr.

Ánh xạ biến con trỏ fptr vào tệp databinary. Sử dụng hàm fopen với tham số wb cho phép ghi tệp tên nhị phân;

Bên trong vòng lặp for, chúng ta sử dụng hàm fwrite() để lưu trữ giá trị vào tệp.

Tham số đầu tiên của hàm fwrite() là địa chỉ của biến num;

Tham số thứ hai lấy kích thước của cấu trúc threeNum;

Tham số thứ ba là 1, vì chúng ta chỉ đưa vào 1 kiểu thể hiện của num;

Tham số thứ 4 là biến con trỏ fptr trỏ tới tệp lưu trữ dữ liệu.

Đóng tệp tin nhị phân bằng hàm fclose().

7.2.6.2. Đọc nội dung từ tệp nhị phân

Sử dụng hàm fread() để đọc nội dung được ghi trong tệp nhị phân.

Cú pháp:

fread(address_data, size_data, numbers_data, pointer_to_file);

Các tham số tương tự hàm fwrite().

🕮 Ví dụ 7.6

Xem xét đoạn chương trình sau đây:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  struct threeNum
4  {
5     int n1, n2, n3;
6  };
7  int main()
8  {
9     int n;
10    struct threeNum num;
11    FILE *fptr;
12    if ((fptr = fopen("databinary.bin","rb")) == NULL){
13        printf("Loi");
14        exit(1);
15    }
16    for(n = 1; n < 5; ++n)
17    {
18       fread(&num, sizeof(struct threeNum), 1, fptr);
19       printf("n1: %d\tn2: %d\tn3: %d \n", num.n1, num.n2, num.n3);
20    }
21    fclose(fptr);
22
23    return 0;
24 }
```

Kết quả chương trình:

```
n1: 1   n2: 5   n3: 6
n1: 2   n2: 10  n3: 11
n1: 3   n2: 15  n3: 16
n1: 4   n2: 20  n3: 21
```

7.2.6.3. Lấy dữ liệu với hàm fseek()

Nếu bạn có nhiều bản ghi trong một tệp và cần truy cập vào một bản ghi tại một vị trí cụ thể, bạn cần phải lặp qua tất cả các bản ghi trước bản

ghi đó. Điều này sẽ lãng phí rất nhiều bộ nhớ và thời gian hoạt động. Một cách dễ dàng hơn để có được dữ liệu cần thiết có thể đạt được bằng cách sử dụng fseek(). Hàm tìm kiếm fseek() đưa con trỏ tới bản ghi đã cho trong tệp.

Hàm fseek() trong C được sử dụng để đặt con trỏ tập tin vào vị trí offset được chỉ định. Nó được sử dụng để ghi dữ liệu vào file tại vị trí mong muốn. Nó sẽ giữ lại dữ liệu của file từ vị trí bắt đầu đến vị trí chỉ định, dữ liệu còn lại bị thay thế bởi dữ liệu mới.

Cú pháp hàm fseek()

fseek(FILE * stream, long int offset, int whence)

Trong đó:

Tham số đầu tiên stream là con trỏ trỏ tới tệp;

Tham số thứ hai là offset là số byte sẽ di chuyển con trỏ file bắt đầu từ vị trí whence;

Tham số thứ ba là vị trí ban đầu của con trỏ file. Có 3 vị trí được định nghĩa sẵn trong thư viện "stdio.h";

SEEK_CUR: vị trí hiện tại của con trỏ file;

SEEK_END: vị trí cuối file;

SEEK_SET: vị trí đầu file.

Lưu ý:

Hàm fseek() di chuyển con trỏ tới vị trí mong muốn:

Hàm fseek() trả về 0 nếu di chuyển thành công;

Hàm fseek() trả về giá trị khác 0 nếu xảy ra lỗi.

🕮 Ví dụ 7.7

Xem xét ví dụ sau đây:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  struct threeNum
4  {
5     int n1, n2, n3;
6  };
7  int main()
```

```
{
   int n;
   struct threeNum num;
   FILE *fptr;
   if ((fptr = fopen("databinary.bin","rb")) == NULL){
       printf("Loi");
       // Chuong trinh thoat neu con tro tra ve gia tri NULL
       exit(1);
   }
   // di chuyen con tro den vi tri cuoi cua file
   fseek(fptr, -sizeof(struct threeNum), SEEK_END);
   for(n = 1; n < 5; ++n)
   {
      fread(&num, sizeof(struct threeNum), 1, fptr);
      printf("n1: %d\tn2: %d\tn3: %d\n", num.n1, num.n2, num.n3);
      fseek(fptr, -2*sizeof(struct threeNum), SEEK_CUR);
   }
   fclose(fptr);
   return 0;
}
```

Kết quả chương trình

```
n1: 4   n2: 20  n3: 21
n1: 3   n2: 15  n3: 16
n1: 2   n2: 10  n3: 11
n1: 1   n2: 5   n3: 6
```

Yêu cầu với sinh viên:

Nhận xét kết quả thu được giữa hai ví dụ 7.6 và ví dụ 7.7.

Chương trình sẽ đọc các bản ghi từ tệp databinary.bin theo thứ tự ngược lại từ bản ghi cuối và in kết quả ra màn hình.

🕮 Ví dụ 7.8

Cho đoạn chương trình sau đây:

```
1   #include <stdio.h>
2   int main () {
3     FILE *fp;
4     fp = fopen("dulieu.bin","w+");
5     fputs("Truong Dai hoc", fp);
6     fseek( fp, 14, SEEK_SET );
7     fputs("DH Cong nghe GTVT", fp);
8     fclose(fp);
9     return(0);
10  }
```

✍ Giải thích chương trình:

- Đầu tiên khai báo biến con trỏ fp, sau đó con trỏ fp ánh xạ đến tệp dulieu.bin được tạo và lưu cùng thư mục của chương trình. Sử dụng hàm fputs() để thêm nội dung vào tệp dulieu.bin.
- Sử dụng hàm fseek() đặt con trỏ vào vị trí 7.
- Ghi đè nội dung bắt đầu từ vị trí này với hàm fputs().
- Khi biên dịch chương trình, sẽ tạo ra tệp dulieu.bin. Đầu tiên nội dung của tệp dulieu.bin được ghi là " Truong Dai hoc", sau đó hàm fseek() sẽ đặt con trỏ bắt đầu từ vị trí thứ 7 và câu lệnh fputs() sẽ ghi đè nội dung bắt đầu từ vị trí này. Kết quả nội dung mới của tệp dulieu.bin sẽ là "Truong DH Cong nghe GTVT". Chúng ta sẽ đọc nội dung của tệp dulieu.bin bằng đoạn chương trình sau đây:

🕮 Ví dụ 7.9

```
1   #include <stdio.h>
2   int main () {
3     FILE *fp;
4     int c;
5     fp = fopen("dulieu.bin","r");
6     while(1) {
7       c = fgetc(fp);
8       if( feof(fp) ) {
9         break;
```

```
10          }
11          printf("%c", c);
12       }
13       fclose(fp);
14       return(0);
15    }
```

✍ **Kết quả chương trình:**

Truong DH Cong nghe GTVT

✍ **Yêu cầu đối với sinh viên:** Sử dụng ví dụ 7.8 và ví dụ 7.9 viết thành một chương trình gồm hai hàm:

Hàm ghi dữ liệu vào tệp;

Hàm đọc và hiển thị dữ liệu ra màn hình.

7.2.7. Một số hàm khác được sử dụng để xử lý tệp

Trong các ví dụ ở trên, chúng ta đã sử dụng các hàm fputs(), hàm fgets(), nhưng chưa rõ chức năng của các hàm này. Mục 7.2.7 sẽ giải thích kỹ hơn về một số hàm được sử dụng để xử lý với tệp.

7.2.7.1. Hàm fputs()

Cú pháp:

int fputs(const char * s, FILE * stream)

Chức năng: Ghi một chuỗi ký tự vào tệp.

🕮 **Ví dụ 7.10**

✍ Xét đoạn chương trình sau đây:

```
1    #include <stdio.h>
2    main() {
3       FILE *fp;
4       // Gan tep
5       fp = fopen("data_7_10.txt", "w");
6       // Ghi chuoi vao tep
7       fputs("Hoc lap trinh C!", fp);
8       // dong tep
9       fclose(fp);
10   }
```

Kiểm tra tệp data_7_10.txt cùng thư mục với tệp chương trình, nội dung được ghi vào tệp là "Hoc lap trinh C!"

7.2.7.2. Hàm fgets()

Cú pháp:

char* fgets(char *s, int n, FILE *stream)

Chức năng: Đọc một dòng ký tự từ tệp.

🕮 Ví dụ 7.11

✍ Xét đoạn chương trình sau đây:

```
1   #include<stdio.h>
2   #include<conio.h>
3   int main(){
4       FILE *fp;
5       char text[300];
6       fp=fopen("data_7_10.txt", "r");
7       printf("%s",fgets(text,200,fp));
8       fclose(fp);
9       getch();
10  }
```

✍ Kết quả chương trình

Hoc lap trinh C!

7.2.7.3. Hàm fputc()

Cú pháp:

int fputc(int c, FILE *stream)

Chức năng: Ghi một ký tự vào tệp

🕮 Ví dụ 7.12

Xem xét đoạn chương trình sau đây:

```
1  #include <stdio.h>
2  main() {
3     FILE *fp;
4     // Mo tep, con tro fp anh xa vao tep data_7_12.txt, mo de ghi
5     fp = fopen("data_7_12.txt", "w");
```

```
    // Ghi tung ky tu vao tep
    fputc('C', fp);
    // dong tep
    fclose(fp);
}
```

Kiểm tra tệp data_7_12.txt cùng thư mục với tệp chương trình, nội dung được ghi vào tệp là "C".

✍Yêu cầu với sinh viên:

Sử dụng hàm fputc() ghi 1 chuỗi ký tự vào tệp.

7.2.7.4. Hàm fgetc()

Cú pháp:

int fgetc(FILE *stream)

Chức năng: Sử dụng để đọc từng ký tự một từ một file đã cho. Nó trả về EOF khi kết thúc file.

🕮 Ví dụ 7.13

Cho nội dung của tệp data_7_13.txt có nội dung như sau:

"Truong Dai hoc Cong nghe GTVT"

Sử dụng hàm fgetc() đọc tệp data_7_13.txt

```
#include<stdio.h>
#include<conio.h>
int main() {
    FILE *fp;
    char c;
    fp=fopen("data_7_13.txt", "r");
    while ((c = fgetc(fp)) != EOF) {
        printf("%c", c);
    }
    fclose(fp);
   return 0;
}
```

✍ Kết quả chương trình:

Truong Dai hoc Cong nghe GTVT

7.3. MỘT SỐ VÍ DỤ VỀ TỆP

🕮 Ví dụ 7.14

Viết chương trình nhập tên và điểm của sinh viên. Lưu kết quả vào tệp.

✍Phác thảo lời giải

- Khai báo một biến tệp fptr;

- Dùng phương thức fopen() mở một tệp Sinhvien.txt được lưu cùng thư mục của chương trình.

- Nhập thông tin bao gồm tên và điểm của n sinh viên, dùng phương thức fpritnf() để ghi vào tệp.

- Đóng tệp.

✍Chương trình

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int main()
4   {
5     char tensv[50];
6     int diem, i, n;
7     printf("Nhap vao so sinh vien: ");
8     scanf("%d", &n);
9     FILE *fptr;
10    fptr = (fopen("Sinhvien.txt", "w"));
11    if(fptr == NULL)
12    {
13       printf("Loi!");
14       exit(1);
15    }
16    for(i = 0; i < n; ++i)
17    {
18      printf("Nhap ten sinh vien thu %d : ", i+1);
19      scanf("%s", tensv);
20      printf("Nhap diem: ");
21      scanf("%d", &diem);
22      fprintf(fptr,"\nTen SV: %s \nDiem =%d \n", tensv, diem);
23    }
```

```
24      fclose(fptr);
25      return 0;
26  }
```

Kết quả chương trình (Ở màn hình Console)

```
Nhap vao so sinh vien: 4
Nhap ten sinh vien thu 1 : An
Nhap diem: 8
Nhap ten sinh vien thu 2 : Cuong
Nhap diem: 9
Nhap ten sinh vien thu 3 : Thuy
Nhap diem: 6
Nhap ten sinh vien thu 4 : Vinh
Nhap diem: 8
```

Nội dung của tệp Sinhvient.txt:

```
Ten SV: An
Diem = 8
Ten SV: Cuong
Diem = 9
Ten SV: Thuy
Diem = 6
Ten SV: Vinh
Diem = 8
```

Ví dụ 7.15

Viết chương trình nhập vào tên và điểm của n sinh viên. Nếu tệp đã tồn tại trước đó, hãy thêm thông tin của n sinh viên này vào tệp.

Chương trình

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int main()
4   {
5       char tensv[50];
6       int diem, i, n;
7       printf("Nhap vao so sinh vien: ");
8       scanf("%d", &n);
```

```
    FILE *fptr;
    //mo tep ghi tiep
    fptr = (fopen("Sinhvien.txt", "a"));
    if(fptr == NULL)
    {
        printf("Loi!");
        exit(1);
    }
    for(i = 0; i < n; ++i)
    {
      printf("Nhap ten sinh vien thu %d : ", i+1);
      scanf("%s", tensv);
      printf("Nhap diem: ");
      scanf("%d", &diem);
      fprintf(fptr,"Ten SV: %s \nDiem =%d \n", tensv, diem);
    }
    fclose(fptr);
    return 0;
}
```

✍Kết quả chương trình

✍ Nhập dữ liệu ở màn hình Console

```
Nhap vao so sinh vien: 3
Nhap ten sinh vien thu 1: Binh
Nhap diem: 8
Nhap ten sinh vien thu 2: Nam
Nhap diem: 6
Nhap ten sinh vien thu 3: Yen
Nhap diem: 8
```

✍ Dữ liệu trong tệp Sinhvien.txt, được ghi tiếp từ tệp Sinhvien.txt trong ví dụ 7.14

```
Ten SV: An
Diem = 8
Ten SV: Cuong
Diem = 9
```

```
Ten SV: Thuy
Diem = 6
Ten SV: Vinh
Diem = 8
Ten SV: Binh
Diem = 8
Ten SV: Nam
Diem = 6
Ten SV: Yen
Diem = 8
```

🕮 Ví dụ 7.16

Viết chương trình ghi tất cả các thành phần của một mảng các cấu trúc vào một tệp sử dụng hàm fwrite(). Đọc mảng từ tệp và hiển thị ra màn hình.

✍Chương trình

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #define MAX 100
4   struct Sinhvien
5   {
6     char tensv[50];
7     float diem;
8   };
9   int main(){
10     struct Sinhvien sv1[MAX], sv2[MAX];
11           int n,i ;
12     FILE *fptr;
13     printf("Nhap vao so sinh vien: ");
14     scanf("%d", &n);
15     fptr = fopen("sv.txt","wb");
16     for(i = 0; i < n; ++i)
17     {
18        fflush(stdin);
19        printf("Nhap thong tin cua sinh vien thu %d :\n", i +1);
20        printf("Nhap ten sinh vien:\n");
```

```
21          gets(sv1[i].tensv);
22          printf("Nhap diem sinh vien:\n ");
23          scanf("%f", &sv1[i].diem);
24      }
25      fwrite(sv1, sizeof(sv1), 1, fptr);
26      fclose(fptr);
27      fptr = fopen("sv.txt", "rb");
28      fread(sv2, sizeof(sv2), 1, fptr);
29      for(i = 0; i < n; ++i)
30      {
31          printf("Ten Sinh vien: %s    Diem sV: %0.2f \n",
    sv2[i].tensv, sv2[i].diem);
32      }
33      fclose(fptr);
34  }
```

✍Kết quả chương trình

✍ ở Màn hình Console

```
Nhap vao so sinh vien: 4
Nhap thong tin cua sinh vien thu 1:
Nhap ten sinh vien:
An
Nhap diem sinh vien:
 9.1
Nhap thong tin cua sinh vien thu 2:
Nhap ten sinh vien:
Cuong
Nhap diem sinh vien:
 4.5
Nhap thong tin cua sinh vien thu 3:
Nhap ten sinh vien:
Thuy
Nhap diem sinh vien:
 6.8
Nhap thong tin cua sinh vien thu 4:
Nhap ten sinh vien:
Nam
```

```
Nhap diem sinh vien:
 8.8
Ten Sinh vien: An  Diem sV: 9.10
Ten Sinh vien: Cuong  Diem sV: 4.50
Ten Sinh vien: Thuy  Diem sV: 6.80
Ten Sinh vien: Nam  Diem sV: 8.80
```

🕮 Ví dụ 7.17

Viết chương trình để ghép hai tệp và ghi vào một tệp mới.

Giả sử tệp thứ nhất test1.txt ghi nội dung: "Bo mon he thong thong tin, Khoa CNTT". Tệp thứ hai test1.txt ghi nội dung ", Truong Dai hoc Cong nghe GTVT".

Tệp kết quả khi ghép tệp 1 và tệp 2 có tên test3.txt có nội dung.

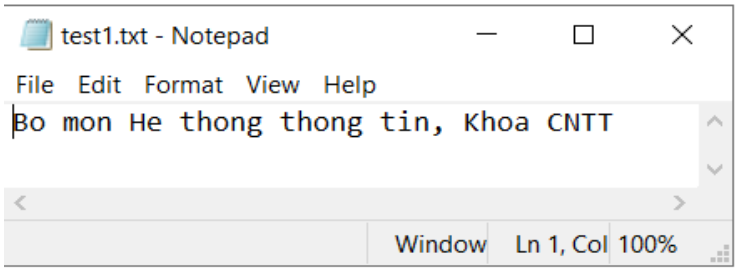
"Bo mon he thong thong tin, Khoa CNTT, Truong Dai hoc Cong nghe GTVT"

✍ Chương trình:

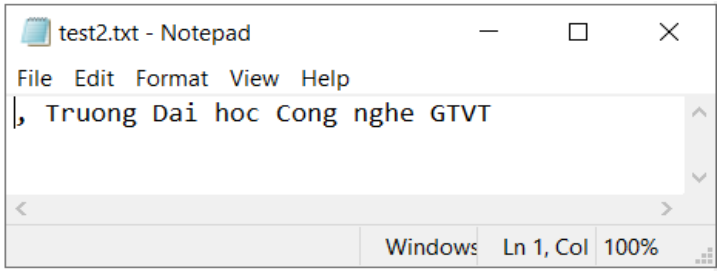
```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int main()
4   {
5           FILE *f1, *f2, *fnew;
6           char ch, fname1[20], fname2[20], fname3[30];
7
8           printf("\n\n Chuong trinh ghep hai tep va ghi vao mot tep
    moi :\n");
9           printf("-------------------------------------\n");
10          // nhap dung ten tep 1 da tao tren cung thu muc cua
    chuong trinh, vd test1.txt
11          // ghi san noi dung cua test1.txt
12          printf("Nhap ten tep 1 : ");
13          scanf("%s",fname1);
14          // nhap dung ten tep 2 da tao tren cung thu muc cua
    chuong trinh, vd test2.txt
15          // ghi san noi dung cua test2.txt
16          printf("Nhap ten tep 2 : ");
17          scanf("%s",fname2);
18    // nhap dung ten tep 3 da tao tren cung thu muc cua chuong
    trinh, vd test3.txt
```

```
19          // test3.txt de trang noi dung, duoc cap nhat ket qua khi bien dich chuong trinh
20          printf("Nhap ten tep moi khi hop nhat hai tep: ");
21          scanf("%s",fname3);
22          f1=fopen(fname1, "r");
23          f2=fopen(fname2, "r");
24          if(f1==NULL || f2==NULL)
25          {
26  //              Thong bao loi
27                  printf(" Tep khong ton tai hoac loi khi mo ra...!!\n");
28                  exit(EXIT_FAILURE);
29          }
30          fnew=fopen(fname3, "w");
31          if(fnew==NULL)
32          {
33  //              Thong bao loi
34                  printf(" Tep khong ton tai hoac loi khi mo ra...!!\n");
35                  exit(EXIT_FAILURE);
36          }
37          while((ch=fgetc(f1))!=EOF)
38          {
39                  fputc(ch, fnew);
40          }
41          while((ch=fgetc(f2))!=EOF)
42          {
43                  fputc(ch, fnew);
44          }
45          printf(" Ghep noi 2 tep vao tep %s thanh cong..!!\n\n", fname3);
46          fclose(f1);
47          fclose(f2);
48          fclose(fnew);
49  }
```

✍ Nội dung của tệp thứ nhất:



✍ Nội dung của tệp thứ hai:



✍ Kết quả chương trình

Chuong trinh ghep hai tep va ghi vao mot tep moi :

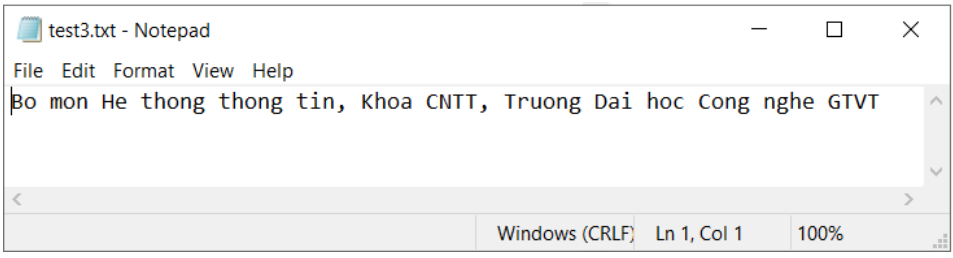
Nhap ten tep 1 : test1.txt

Nhap ten tep 2 : test2.txt

Nhap ten tep moi khi hop nhat hai tep: test3.txt

Ghep noi 2 tep vao tep test3.txt thanh cong..!!

✍ Nội dung của tệp test3.txt



🕮 Ví dụ 7.18

Chương trình quản lý sinh viên

- Khai báo kiểu dữ liệu SinhVien có các trường họ tên, giới tính, tuổi, điểm Toán - Lý - Hóa và điểm trung bình.

- Nhập vào danh sách N sinh viên;
- Xuất danh sách N sinh viên;
- Tính điểm trung bình cho N sinh viên;
- Sắp xếp N sinh viên theo thứ tự tăng dần của điểm trung bình;
- Xếp loại N sinh viên;
- Xuất danh sách N sinh viên ra file;
- Viết chương trình dạng menu cho phép sử dụng các tính năng trên.

Chương trình

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
 struct SinhVien{
    char ten[30];
    char gt[5];
    float dT, dL, dH;
    float dtb = 0;
};
 typedef SinhVien SV;
 void nhap(SV &sv);
void nhapN(SV a[], int n);
void xuat(SV sv);
void xuatN(SV a[], int n);
void tinhDTB(SV &sv);
void sapxep(SV a[], int n);
void xeploai(SV a);
void xeploaiN(SV a[], int n);
void xuatFile(SV a[], int n, char fileName[]);
 int main(){
    int key;
    char fileName[] = "DSSV.txt";
    int n;
    bool daNhap = false;
    do{
        printf("\nNhap so luong SV: "); scanf("%d", &n);
    }while(n <= 0);
    SV a[n];
```

```
29      while(true){
30          system("cls");
31
printf("*****************************************\n");
32        printf("**    CHUONG TRINH QUAN LY SINH VIEN
**\n");
33        printf("**      1. Nhap du lieu            **\n");
34        printf("**      2. In danh sach sinh vien    **\n");
35        printf("**      3. Sap xep sinh vien theo DTB**\n");
36        printf("**      4. Xep loai sinh vien        **\n");
37        printf("**      5. Xuat DS sinh vien         **\n");
38        printf("**      0. Thoat                   **\n");
39        printf("*****************************************\n");
40          printf("** Nhap lua chon cua ban**\n");
41          scanf("%d",&key);
42          switch(key){
43              case 1:
44              printf("\nBan da chon nhap DS sinh vien!");
45                  nhapN(a, n);
46                  printf("\nBan da nhap thanh cong!");
47                  daNhap = true;
48               printf("\nBam phim bat ky de tiep tuc!");
49                  getch();
50                  break;
51              case 2:
52                  if(daNhap){
53              printf("\nBan da chon xuat DS sinh vien!");
54                      xuatN(a, n);
55                  }else{
56                      printf("\nNhap DS SV truoc!!!!");
57                  }
58               printf("\nBam phim bat ky de tiep tuc!");
59                  getch();
60                  break;
61              case 3:
```

```
62              if(daNhap){
63          printf("\nBan da chon sap xep SV theo STB!");
64                  sapxep(a, n);
65                  xuatN(a, n);
66              }else{
67                  printf("\nNhap DS SV truoc!!!!");
68              }
69           printf("\nBam phim bat ky de tiep tuc!");
70              getch();
71              break;
72           case 4:
73              if(daNhap){
74           printf("\nBan da chon thoat xep loai SV!");
75                  xeploaiN(a, n);
76              }else{
77                  printf("\nNhap DS SV truoc!!!!");
78              }
79           printf("\nBam phim bat ky de tiep tuc!");
80              getch();
81              break;
82           case 5:
83              if(daNhap){
84                printf("\nBan da chon xuat DS SV!");
85                  xuatFile(a, n, fileName);
86              }else{
87                printf("\nNhap DS SV truoc!!!!");
88              }
89           printf("\nXuat DSSV thanh cong vao file %s!",
fileName);
90          printf("\nBam phim bat ky de tiep tuc!");
91              getch();
92              break;
93           case 0:
94          printf("\nBan da chon thoat chuong trinh!");
95              getch();
```

```
96                 return 0;
97              default:
98               printf("\nKhong co chuc nang nay!");
99               printf("\nBam phim bat ky de tiep tuc!");
100                getch();
101                break;
102          }
103      }
104  }
105   void tinhDTB(SV &sv){
106      sv.dtb = (sv.dH + sv.dL + sv.dT)/3;
107  }
108  void nhap(SV &sv){
109      printf("\nNhap ten: ");
110            fflush(stdin);
111            gets(sv.ten);
112      printf("Nhap gioi tinh: F (nu) M (nam) ");
113            gets(sv.gt);
114      printf("Nhap diem mon Toan - Ly - Hoa: ");
   scanf("%f%f%f", &sv.dT, &sv.dL, &sv.dH);
115      tinhDTB(sv);
116  }
117   void nhapN(SV a[], int n){
118      printf("\n_____\n");
119      for(int i = 0; i< n; ++i){
120         printf("\nNhap SV thu %d:", i+1);
121         nhap(a[i]);
122      }
123      printf("\n_____\n");
124  }
125   void xuat(SV sv){
126      printf("\nHo ten SV: %s", sv.ten);
127      printf("\nGioi tinh: %s", sv.gt);
128      printf("\nDiem Toan - Ly - Hoa: %.2f - %.2f - %.2f", sv.dT,
   sv.dL, sv.dH);
```

```
129     printf("\nDiem TB: %.2f", sv.dtb);
130 }
131  void xuatN(SV a[], int n){
132     printf("\n_____\n");
133     for(int i = 0;i < n;++i){
134         printf("\nThong tin SV thu %d:", i+1);
135         xuat(a[i]);
136     }
137     printf("\n_____\n");
138 }
139  void sapxep(SV a[], int n){
140     //Sap xep theo DTB tang dan
141     SV tmp;
142     for(int i = 0;i < n;++i){
143         for(int j = i+1; j < n;++j){
144             if(a[i].dtb > a[j].dtb){
145                 tmp = a[i];
146                 a[i] = a[j];
147                 a[j] = tmp;
148             }
149         }
150     }
151 }
152  void xeploai(SV sv){
153     if(sv.dtb >= 8) printf("Gioi");
154     else if(sv.dtb >= 6.5) printf("Kha");
155     else if(sv.dtb >= 4) printf("Trung binh");
156     else printf("Yeu");
157 }
158 void xeploaiN(SV a[], int n){
159     printf("\n_____\n");
160     for(int i = 0;i < n;++i){
161         printf("\nXep loai cua SV thu %d la: ", i+1);
162         xeploai(a[i]);
163     }
```

```
164     printf("\n_____\n");
165  }
166   void xuatFile(SV a[], int n, char fileName[]){
167     FILE * fp;
168     fp = fopen (fileName,"w");
169     fprintf(fp,        "%20s%5s%10s%10s%10s%10s\n",        "Ho
    Ten","GT", "DiemToan", "DiemLy", "DiemHoa", "DiemTB");
170     for(int i = 0;i < n;i++){
171        fprintf(fp,                  "%20s%5s%10f%10f%10f%10f\n",
    a[i].ten,a[i].gt, a[i].dT, a[i].dL, a[i].dH, a[i].dtb);
172     }
173     fclose (fp);
174  }
```

Kết quả chương trình:

```
Nhap so luong SV: 4
****************************************
**    CHUONG TRINH QUAN LY SINH VIEN   **
**      1. Nhap du lieu              **
**      2. In danh sach sinh vien      **
**      3. Sap xep sinh vien theo DTB  **
**      4. Xep loai sinh vien          **
**      5. Xuat DS sinh vien           **
**      0. Thoat                     **
****************************************
**       Nhap lua chon cua ban        **
1
Ban da chon nhap DS sinh vien!
_____

Nhap SV thu 1:
Nhap ten: An
Nhap gioi tinh: F (nu) M (nam) M
Nhap diem mon Toan - Ly - Hoa: 8 8 8

Nhap SV thu 2:
```

Nhap ten: Cuong
Nhap gioi tinh: F (nu) M (nam) M
Nhap diem mon Toan - Ly - Hoa: 6 7 8

Nhap SV thu 3:
Nhap ten: Thuy
Nhap gioi tinh: F (nu) M (nam) F
Nhap diem mon Toan - Ly - Hoa: 9 8 8

Nhap SV thu 4:
Nhap ten: Mai
Nhap gioi tinh: F (nu) M (nam) F
Nhap diem mon Toan - Ly - Hoa: 9 6 8

Ban da nhap thanh cong!
Bam phim bat ky de tiep tuc!
** Nhap lua chon cua ban **
2
Ban da chon xuat DS sinh vien!

Thong tin SV thu 1:
Ho ten SV: An
Gioi tinh: M
Diem Toan - Ly - Hoa: 8.00 - 8.00 - 8.00
Diem TB: 8.00
Thong tin SV thu 2:
Ho ten SV: Cuong
Gioi tinh: M
Diem Toan - Ly - Hoa: 6.00 - 7.00 - 8.00
Diem TB: 7.00
Thong tin SV thu 3:
Ho ten SV: Thuy
Gioi tinh: F
Diem Toan - Ly - Hoa: 9.00 - 8.00 - 8.00

Diem TB: 8.33
Thong tin SV thu 4:
Ho ten SV: Mai
Gioi tinh: F
Diem Toan - Ly - Hoa: 9.00 - 6.00 - 8.00
Diem TB: 7.67

Bam phim bat ky de tiep tuc!
** Nhap lua chon cua ban **
3
Ban da chon sap xep SV theo DTB!

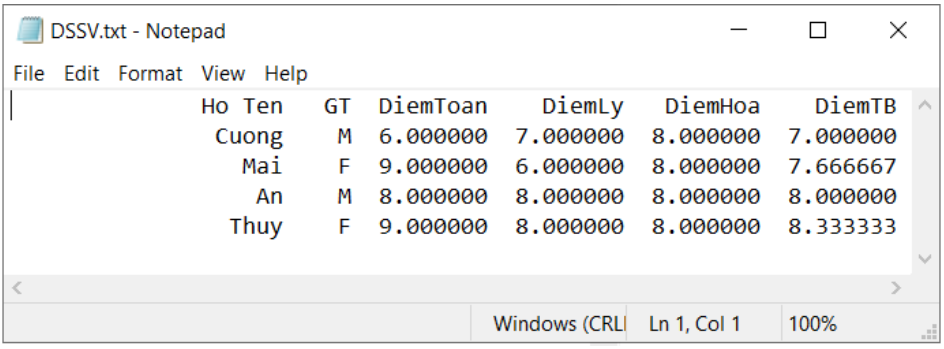
Thong tin SV thu 1:
Ho ten SV: Cuong
Gioi tinh: M
Diem Toan - Ly - Hoa: 6.00 - 7.00 - 8.00
Diem TB: 7.00
Thong tin SV thu 2:
Ho ten SV: Mai
Gioi tinh: F
Diem Toan - Ly - Hoa: 9.00 - 6.00 - 8.00
Diem TB: 7.67
Thong tin SV thu 3:
Ho ten SV: An
Gioi tinh: M
Diem Toan - Ly - Hoa: 8.00 - 8.00 - 8.00
Diem TB: 8.00
Thong tin SV thu 4:
Ho ten SV: Thuy
Gioi tinh: F
Diem Toan - Ly - Hoa: 9.00 - 8.00 - 8.00
Diem TB: 8.33

Bam phim bat ky de tiep tuc!
** Nhap lua chon cua ban **

4
Ban da chon thoat xep loai SV!

Xep loai cua SV thu 1 la: Kha
Xep loai cua SV thu 2 la: Kha
Xep loai cua SV thu 3 la: Gioi
Xep loai cua SV thu 4 la: Gioi

Bam phim bat ky de tiep tuc!
** Nhap lua chon cua ban **
5
Ban da chon xuat DS SV!
Xuat DSSV thanh cong vao file DSSV.txt!
Bam phim bat ky de tiep tuc!



BÀI TẬP CHƯƠNG 7

A. BÀI TẬP CÓ LỜI GIẢI

🕮 Bài tập 7.1

Viết chương trình tạo và lưu thông tin trong tệp văn bản

✍ Phác thảo lời giải

- Sử dụng hàm fgets() để nhập 1 dòng văn bản.
- Sử dụng hàm fprintf() để in.

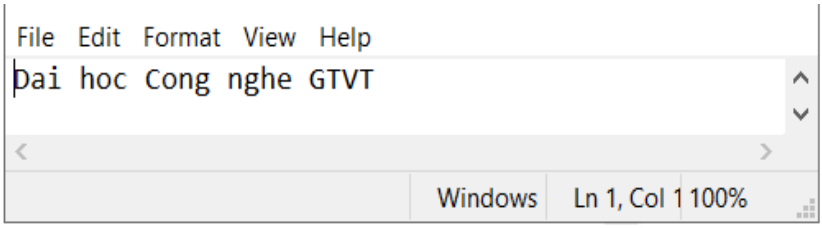
✍Chương trình:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int main()
4   {
5     char str[1000];
6     FILE *fptr;
7     char fname[20]="test_7_1.txt";
8     printf("\nTao tep (test_7_1.txt) va nhap dong van ban:\n");
9           printf("-----------------------------------\n");
10    fptr=fopen(fname,"w");
11    if(fptr==NULL)
12    {
13      printf(" Loi khi mo tep!");
14      exit(1);
15    }
16   printf(" Nhap dong van ban luu vao tep test_6_1.txt : ");
17    fgets(str, sizeof str, stdin);
18    fprintf(fptr,"%s",str);
19    fclose(fptr);
20   printf("\n Tep %s duoc tao thanh cong...!!\n\n",fname);
21    return 0;
22  }
```

✍Kết quả chương trình:

```
Tao tep (test_7_1.txt) va nhap dong van ban:
-----------------------------------------
Nhap dong van ban luu vao tep test_7_1.txt : Dai hoc Cong nghe GTVT
Tep test_7_1.txt duoc tao thanh cong...!!
```

✍ Nội dung được ghi vào tệp test_7_1.txt



🕮 Bài tập 7.2

Viết chương trình đọc và hiển thị nội dung của tệp văn bản đang tồn tại.

✍ Phác thảo lời giải:

- Nhập vào tên tệp, kiểm tra sự tồn tại của tệp.
- Sử dụng hàm fgetc() đọc từng ký tự, kết hợp vòng lặp while để đọc tất cả các ký tự của tệp.

✍ Chương trình

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int main()
4   {
5           FILE *fptr;
6           char fname[20];
7           char str;
8    printf("\nChuong trinh in noi dung cua tep van ban :\n");
9           printf("------------------------------\n");
10          printf(" Nhap vao ten tep tin can doc noi dung : ");
11          // Vi du doc lai tep test_7_1.txt cua bai tap 7.1
12          scanf("%s",fname);
13          fptr = fopen (fname, "r");
14          if (fptr == NULL)
15          {
16                  printf("Tep khong ton tai hoac khong the mo.\n");
17                  exit(0);
18          }
19          printf("\nNoi dung cua tep %s duoc hien thi  :\n",fname);
20          str = fgetc(fptr);
21          while (str != EOF)
```

```
22                  {
23                          printf ("%c", str);
24                          str = fgetc(fptr);
25                  }
26          fclose(fptr);
27      printf("\n\n");
28  }
```

✍ Kết quả chương trình:

```
Chuong trinh in noi dung cua tep van ban:
------------------------------
Nhap vao ten tep tin can doc noi dung: test_7_1.txt
Noi dung cua tep test_7_1.txt duoc hien thi:
Dai hoc Cong nghe GTVT
Trường hợp tệp không tồn tại
Chuong trinh doc noi dung cua tep van ban:
------------------------------
Nhap vao ten cua tep tin can doc noi dung: test_6.txt
Tep khong ton tai hoac khong the mo.
```

🕮 Bài tập 7.3

Viết chương trình để viết nhiều dòng trong một tệp văn bản

✍ Chương trình

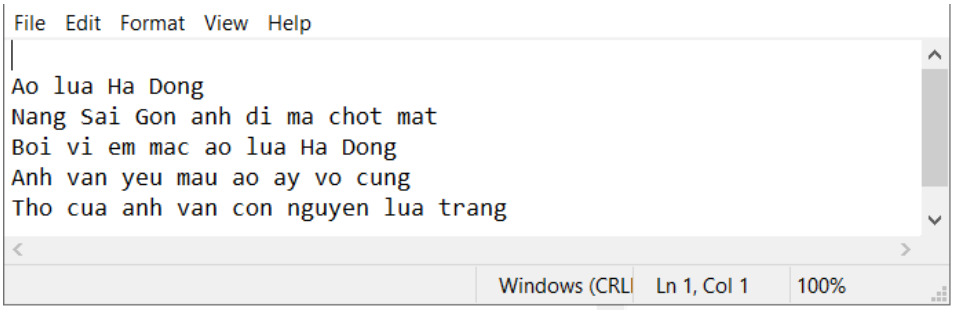
```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int main ()
4   {
5     FILE * fptr;
6     int i,n;
7     char str[100];
8     char fname[20]="test_7_3.txt";
9     char str1;
10      printf("\n\nChuong trinh ghi nhieu dong vao 1 tep van ban va
    doc noi dung tep :\n");
11  printf("-----------------------------------\n");
12  printf("Nhap vao so dong can ghi: ");
```

```
13    scanf("%d", &n);
14    printf("\nNoi dung cac dong:\n");
15    fptr = fopen (fname,"w");
16    for(i = 0; i < n+1;i++)
17              {
18              fgets(str, sizeof str, stdin);
19              fputs(str, fptr);
20              }
21     fclose (fptr);
22    /*------------ Doc noi dung tep va hien thi -----------*/
23    fptr = fopen (fname, "r");
24    printf("\n Noi dung cua tep %s la :\n",fname);
25    str1 = fgetc(fptr);
26    while (str1 != EOF)
27              {
28                        printf ("%c", str1);
29                        str1 = fgetc(fptr);
30              }
31       printf("\n\n");
32       fclose (fptr);
33       return 0;
34    }
```

✍ Kết quả chương trình

```
Chuong trinh ghi nhieu dong vao 1 tep van ban va doc noi dung tep:
-------------------------------------------------------
Nhap vao so dong can ghi: 5
Noi dung cac dong:
Ao lua Ha Dong
Nang Sai Gon anh di ma chot mat
Boi vi em mac ao lua Ha Dong
Anh van yeu mau ao ay vo cung
Tho cua anh van con nguyen lua trang
Noi dung cua tep test_7_3.txt la:
Ao lua Ha Dong
Nang Sai Gon anh di ma chot mat
```

Boi vi em mac ao lua Ha Dong
Anh van yeu mau ao ay vo cung
Tho cua anh van con nguyen lua trang



Nội dung được ghi trong tệp test_7_3.txt

🕮 Bài tập 7.4

Viết chương trình đọc tệp và lưu các dòng vào mảng.

✍Chương trình:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #define dong 128
5   #define cot 10
6   int main(void)
7   {
8      char line[dong][cot];
9   char fname[20];
10     FILE *fptr = NULL;
11     int i = 0;
12     int tot = 0;
13    printf("\n\nChuong trinh doc tu tep va luu vao mang :\n");
14  printf("---------------------------------------\n");
15  printf(" Nhap ten tep can mo : ");
16  scanf("%s",fname);
17     fptr = fopen(fname, "r");
18     while(fgets(line[i], dong, fptr))
19  {
20        line[i][strlen(line[i]) - 1] = '\0';
```

```
        i++;
    }
    tot = i;
  printf("\n Noi dung cua tep %s  la : \n",fname);
    for(i = 0; i < tot; ++i)
    {
        printf(" %s\n", line[i]);
    }
    printf("\n");
    return 0;
}
```

✍ Kết quả chương trình:

```
Chuong trinh doc tu tep va luu vao mang:
-------------------------------------------------------
 Nhap ten tep can mo : test_7_3.txt
 Noi dung cua tep test_7_3.txt  la:
 Ao lua Ha Dong
 Nang Sai Gon anh di ma chot mat
 Boi vi em mac ao lua Ha Dong
 Anh van yeu mau ao ay vo cung
 Tho cua anh van con nguyen lua trang
```

🕮 Bài tập 7.5

Viết chương trình để đếm số dòng trong một tệp văn bản.

✍Phác thảo lời giải:

- Nhập vào tên của tệp, kiểm tra sự tồn tại của tệp.
- Trích xuất từng ký tự của tệp vào biến ký tự c.
- Sử dụng vòng lặp để duyệt các ký tự, nếu gặp ký tự xuống dòng "\n" thì tăng biến đếm lên một đơn vị.

✍ Chương trình

```
#include <stdio.h>
#define FSIZE 100
int main()
{
    FILE *fptr;
```

```
    int ctr = 0;
    char fname[FSIZE];
    char c;
    printf("\n\nDoc tep va dem so dong cua tep :\n");
        printf("----------------------------------------\n");
        printf("Nhap ten tep can mo (tep da duoc luu trong may):
");
        scanf("%s",fname);
    fptr = fopen(fname, "r");
    if (fptr == NULL)
    {
       printf("Khong the mo tep %s", fname);
       return 0;
    }
    // Trich xuat cac ky tu tu tep va luu tru trong bien c
    for (c = getc(fptr); c != EOF; c = getc(fptr))
       if (c == '\n') // Tang bien diem neu gap ky tu xuong dong
          ctr = ctr + 1;
    fclose(fptr);
    printf("So dong cua tep %s la : %d \n", fname, ctr-1);
    return 0;
}
```

✍Kết quả chương trình:

```
Doc tep va dem so dong cua tep:
-----------------------------------------------
Nhap ten tep can mo (tep da duoc luu trong may) : test_7_3.txt
So dong cua tep test_7_3.txt la: 5
```

🕮 Bài tập 7.6

Viết chương trình đếm số từ và đếm số ký tự trong một tệp văn bản

✍ Chương trình:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
```

```
5     FILE *fptr;
6     char ch;
7     // wrd: bien dem so tu, charctr: bien dem so ky tu
8           int wrd=1,charctr=1;
9     char fname[20];
10    printf("\nDem so tu va so ky tu trong mot tep :\n");
11          printf("-----------------------------------\n");
12          printf("Nhap ten tep can mo : ");
13          scanf("%s",fname);
14    fptr=fopen(fname,"r");
15    if(fptr==NULL)
16     {
17        printf("Tep khong ton tai hoac khong the mo!.");
18      }
19    else
20       {
21        ch=fgetc(fptr);
22        printf("Noi dung cua tep %s la : ",fname);
23        while(ch!=EOF)
24          {
25             printf("%c",ch);
26       // gap ky tu la dau cach ' ' hoac dau xuong dong \n
27                              if(ch==' '||ch=='\n')
28               {
29                    wrd++;
30               }
31                 else
32               {
33                    charctr++;
34               }
35             ch=fgetc(fptr);
36          }
37       printf("\nSo tu trong tep %s la : %d\n",fname,wrd-2);
38        printf("So ky tu trong tep %s la : %d\n\n",fname,charctr-1);
39        }
40    fclose(fptr);
41 }
```

🕮 Kết quả chương trình:

```
Dem so tu va so ky tu trong mot tep:
-----------------------------------------------------
Nhap ten tep can mo : test_7_3.txt
Noi dung cua tep test_7_3.txt la:
Ao lua Ha Dong
Nang Sai Gon anh di ma chot mat
Boi vi em mac ao lua Ha Dong
Anh van yeu mau ao ay vo cung
Tho cua anh van con nguyen lua trang

So tu trong tep test_7_3.txt la: 36
So ky tu trong tep test_7_3.txt la: 107
```

🕮 Bài tập 7.7

Viết chương trình xóa một dòng được chỉ định trong một tệp văn bản.

Chương trình:

```
1   #include <stdio.h>
2   #include <string.h>
3   #define MAX 256
4    int main()
5    {
6        int lno, ctr = 0;
7        char ch;
8        FILE *fptr1, *fptr2;
9           char fname[MAX];
10       char str[MAX], temp[] = "temp.txt"; // tep ghi tam
11          printf("\nXoa mot dong trong tep van ban :\n");
12          printf("--------------------------------\n");
13          printf("Nhap ten tep can mo ra: ");
14          scanf("%s",fname);
15       fptr1 = fopen(fname, "r");
16       if (!fptr1)
17          {
18       printf("Tep khong ton tai hoac khong the mo!!\n");
```

```
19              return 0;
20          }
21          fptr2 = fopen(temp, "w"); // Mo tep "ghi tam" o che do ghi
22          if (!fptr2)
23              {
24          printf("Khong the mo tep tam thoi de ghi!!\n");
25              fclose(fptr1);
26              return 0;
27          }
28       printf("Nhap vao chi so dong trong tep can xoa : ");
29          scanf("%d", &lno);
30             lno++;
31          // Sao chep tat ca noi dung vao tep tam thoi tru dong can
    xoa noi dung
32          while (!feof(fptr1))
33          {
34             strcpy(str, "\0");
35             fgets(str, MAX, fptr1);
36             if (!feof(fptr1))
37             {
38                ctr++;
39                /* bo qua dong da chon de xoa */
40                if (ctr != lno)
41                {
42                   fprintf(fptr2, "%s", str);
43                }
44             }
45          }
46          fclose(fptr1);
47          fclose(fptr2);
48          remove(fname);              // Xoa toan bo tep ban dau
49          rename(temp, fname);    // Doi ten tep tam thoi thanh tep
    ban dau
50    /*------ doc noi dung tep ----------------*/
51      fptr1=fopen(fname,"r");
```

```
52              ch=fgetc(fptr1);
53            printf("Noi dung tep sau khi xoa %s la : \n",fname);
54            while(ch!=EOF)
55              {
56                 printf("%c",ch);
57                  ch=fgetc(fptr1);
58              }
59          fclose(fptr1);
60    /*------- Ket thuc doc ----------------*/
61          return 0;
62
63     }
```

✍ Kết quả chương trình

```
Xoa mot dong trong tep van ban:
----------------------------------------
Nhap ten tep can mo ra: test_7_3.txt
Nhap vao chi so dong trong tep can xoa: 3
Noi dung tep sau khi xoa test_7_3.txt la:

Ao lua Ha Dong
Nang Sai Gon anh di ma chot mat
Anh van yeu mau ao ay vo cung
Tho cua anh van con nguyen lua trang
```

🕮 Bài tập 7.8

Viết chương trình đếm số lần xuất hiện của tất cả các từ trong một tệp văn bản.

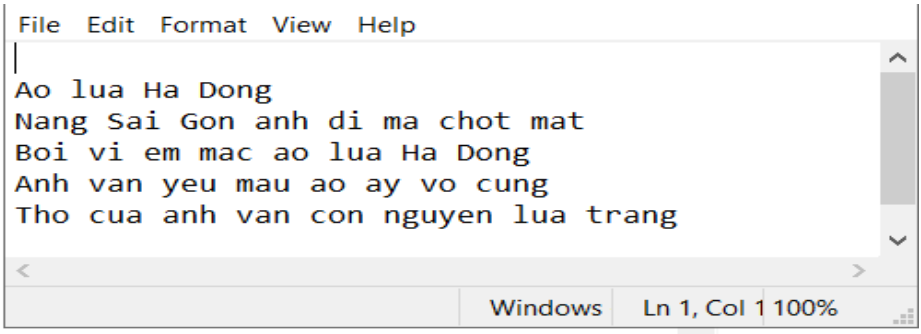
✍Chương trình:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #include <ctype.h>
5   #define MAX_WORDS   1000
6   int main()
7   {
```

```
8       FILE *fptr;
9       char path[100];
10      int i, len, index, isUnique;
11      // Danh sach cua cac tu rieng biet
12      char words[MAX_WORDS][50];
13      char word[50];
14      // Dem cac tu rieng biet
15      int  count[MAX_WORDS];
16      /* Nhap ten tep van ban can dem cac tu*/
17      printf("Nhap bao ten tep: ");
18      scanf("%s", path);
19      /* Mo tep o che do doc */
20      fptr = fopen(path, "r");
21      /* Thoat neu mo tep khong thanh cong */
22      if (fptr == NULL)
23      {
24          printf("Khong the mo tep.\n");
25          printf("Kiem tra lai tep!\n");
26          exit(EXIT_FAILURE);
27      }
28      // Khoi tao mang bien dem cac tu trong tep ve 0
29      for (i=0; i<MAX_WORDS; i++)
30          count[i] = 0;
31      index = 0;
32      while (fscanf(fptr, "%s", word) != EOF)
33      {
34          // Chuyen doi sang ky tu thuong
35          strlwr(word);
36          // Xoa ky tu dau cham cau cuoi cung
37          // ham ispunct() kiem tra ky tu co phai la ky tu dac biet
    hay khong
38          // bao gom cac ky tu  ! " # $ % & ' ( ) * + , - . / : ; < = > ?
    @ [ \ ] ^ _ ` { | } ~
39          len = strlen(word);
40          if (ispunct(word[len - 1]))
```

```
41              word[len - 1] = '\0';
42
43          // Kiem tra tu da ton tai trong danh sach cac tu phan biet
    cua tep
44          isUnique = 1;
45          for (i=0; i<index && isUnique; i++)
46          { // hàm kiem tra 2 xau ký tu strcmp
47              if (strcmp(words[i], word) == 0)
48                  isUnique = 0;
49          }
50          // Neu tu  la duy nhat thi them tu vao danh sach cac tu
    rieng biet
51          // va tang chi so. Nguoc lai tang so lan xuat hien
52          // dem tu hien tai
53          if (isUnique)
54          {
55              strcpy(words[index], word);
56              count[index]++;
57              index++;
58          }
59          else
60          {
61              count[i - 1]++;
62          }
63      }
64      // dong tep
65      fclose(fptr);
66      /*
67       * In so lan xuat hien cua tat ca tu trong tep
68       */
69      printf("\nSo lan xuat hien cua tat ca cac tu trong tep: \n");
70      for (i=0; i<index; i++)
71      {
72          /*
73           * %-15s in chuoi co do rong 15 ky tu.
```

```
74          * - duoc su dung de in chuoi can le ben trai
75          * 15 la do rong cua ky tu
76          */
77         printf("%-15s => %d\n", words[i], count[i]);
78      }
79      return 0;
80  }
```



Kết quả chương trình:

Nhap vao ten tep: test_7_3.txt
So lan xuat hien cua tat ca cac tu trong tep:
ao => 3
lua => 3
ha => 2
dong => 2
nang => 1
sai => 1
gon => 1
anh => 3
di => 1
ma => 1
chot => 1
mat => 1
boi => 1
vi => 1
em => 1
mac => 1
van => 2

```
yeu          => 1
mau          => 1
ay           => 1
vo           => 1
cung         => 1
tho          => 1
cua          => 1
con          => 1
nguyen       => 1
trang        => 1
```

Bài tập 7.9 (*) Bài đọc thêm

Viết chương trình để tìm và thay thế một từ trong tệp văn bản.

Chương trình:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #define BUFFER_SIZE 1000
5   /* Dinh nghia ham thay the */
6   void replaceAll(char *str, const char *oldWord, const char
    *newWord);
7   int main()
8   {
9      /* Con tro tep anh xa den tep van ban */
10     FILE * fPtr;
11     FILE * fTemp;
12     char path[100];
13     char buffer[BUFFER_SIZE];
14     char oldWord[100], newWord[100];
15     printf("Nhap tep van ban ban dau: ");
16     scanf("%s", path);
17     printf("Nhap tu can thay the: ");
18     scanf("%s", oldWord);
19     printf("Duoc thay the bang tu : ");
20     scanf("%s", newWord);
21     /* Mo tat ca cac tep tin can thiet */
22     fPtr  = fopen(path, "r");
```

```
23      fTemp = fopen("replace.tmp", "w");
24      /* Ham fopen() tra ve NULL neu khong mo duoc tep. */
25      if (fPtr == NULL || fTemp == NULL)
26      {
27         /* Thoat neu khong mo duoc tep tin */
28         printf("\nKhong the mo duoc tep tin.\n");
29         printf("Kiem tra va chon lai tep ton tai co the doc/ghi.\n");
30         exit(EXIT_SUCCESS);
31      }
32      /*
33       * Doc tung dong o tep nguon va ghi vao tep dich
34       * tep tin sau khi da thay the tu .
35       */
36      while ((fgets(buffer, BUFFER_SIZE, fPtr)) != NULL)
37      {
38        // Thay the tat ca tu can thay the xuat hien trong dong hien tai
39         replaceAll(buffer, oldWord, newWord);
40         // Sau thay the ghi no vao tep tin tam thoi
41         fputs(buffer, fTemp);
42      }
43      /* Dong tat ca cac tep tin co lien quan*/
44      fclose(fPtr);
45      fclose(fTemp);
46      /* Xoa tep nguon ban dau */
47      remove(path);
48      /* Doi ten tep tam thoi tro thanh ten cua tep ban dau */
49      rename("replace.tmp", path);
50      printf("\nThay the thanh cong tat ca cac tu can thay the '%s'
    bang tu '%s'.", oldWord, newWord);
51      return 0;
52   }
53   /**
54    * Ham de thay the tu can thay the trong tep bang mot tu khac.
55    */
56   void replaceAll(char *str, const char *oldWord, const char
    *newWord)
57   {
58      char *pos, temp[BUFFER_SIZE];
```

```
59      int index = 0;
60      int owlen;
61      owlen = strlen(oldWord);
62      /*
63       * Lap cho den khi tat ca tu can thay the trong tep duoc thay the.
64       */
65      while ((pos = strstr(str, oldWord)) != NULL)
66      {
67          // Sao chep dong hien tai
68          strcpy(temp, str);
69          // Chi so cua tu hien tai tim thay
70          index = pos - str;
71          str[index] = '\0';
72          // Ghep str voi tu moi
73          strcat(str, newWord);
74          // Ghep str voi cac tu con lai
75          //  sau khi tu cu duoc tim thay.
76          strcat(str, temp + index + owlen);
77      }
78  }
```

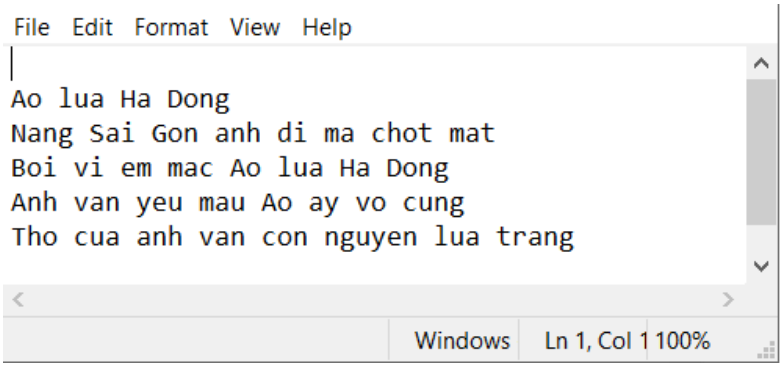
✍ Kết quả chương trình

Nhap tep van ban ban dau: test_7_3.txt

Nhap tu can thay the: ao

Duoc thay the bang tu: Ao

Thay the thanh cong tat ca cac tu can thay the 'ao' bang tu 'Ao'.



B. BÀI TẬP TỰ GIẢI

🕮 Bài tập 7.10

Viết chương trình so sánh 2 tệp văn bản.

🕮 Bài tập 7.11

Viết chương trình xóa 1 từ trong tệp văn bản.

🕮 Bài tập 7.12

Viết chương trình đếm số lần xuất hiện của 1 từ được nhập từ bàn phím trong một tệp văn bản.

🕮 Bài tập 7.13

Viết chương trình sao chép nội dung từ một tệp văn bản này sang một tệp văn bản khác.

Gợi ý:

Các bước để thực hiện việc sao chép nội dung từ một tệp văn bản sang một tệp văn bản khác:

- Khởi tạo tệp nguồn và tệp đích;
- Mở tệp nguồn ở chế độ đọc (r) và tệp đích ở chế độ ghi (w). Kiểm tra sự tồn tại của các tệp;
- Đọc từng ký tự từ tập nguồn và ghi từng ký tự ở tệp đích sử dụng hàm fputc().
- Thực hiện lặp bước 3 cho đến khi đọc đến ký tự cuối cùng ở tệp nguồn và ghi sang tệp đích.
- Đóng cả hai tệp nguồn và đích.

🕮 Bài tập 7.14

Viết chương trình thay thế chuyển tất cả các ký tự viết hoa trong tệp văn bản thành ký tự viết thường.

🕮 Bài tập 7.15

Viết chương trình thay thế 1 dòng được chỉ định trong tệp văn bản bằng một dòng văn bản khác.

🕮 Bài tập 7.16

Viết chương trình ghi vào tệp test_7_16.txt n số nguyên (được nhập từ bàn phím). Đọc các số từ tệp test_7_16.txt ghi vào một mảng. Sắp xếp mảng tăng dần theo giá trị và ghi các số của mảng vào một tệp test_sap_xep.txt.

🕮 Bài tập 7.17

Viết chương trình nhập vào thông tin của n sinh viên, biết thông tin của một sinh viên bao gồm mã sinh viên (chuỗi ký tự), tên sinh viên (chuỗi ký tự), điểm tổng kết năm học (số thực), đánh giá (chuỗi ký tự). Đánh giá của sinh viên được dựa vào điểm tổng kết năm học.

Nếu điểm tổng kết ≥ 5: "Lên lớp"

Nếu điểm tổng kết < 5: "Thi lại"

Sắp xếp danh sách sinh viên theo điểm tổng kết và ghi vào tệp Dslop.txt

Đọc tệp Dslop.txt, lọc các sinh viên có đánh giá "Lên lớp" ghi vào một tệp "Lenlop.txt", lọc các sinh viên có đánh giá "Thi lại" ghi vào một tệp "Thilai.txt"

🕮 Bài tập 7.18

Viết chương trình nhập vào thông tin của n nhân viên của một công ty. Thông tin của một nhân viên bao gồm: Mã nhân viên, tên nhân viên, địa chỉ, bậc lương, lương.

Biết rằng lương nhân viên = 1.3 *(bậc lương * lương cơ bản) (Lương cơ bản = 1.390.000);

a. Ghi thông tin các nhân viên vào tệp nhanvien.bin;

b. Đọc thông tin từ tệp nhanvien.bin ghi vào mảng cấu trúc nhanvien[];

c. Sắp xếp lại danh sách nhân viên theo lương, ghi vào tệp nhanviensx.bin.

Lưu ý:

- Mỗi ý xây dựng bằng một hàm.

🕮 Bài tập 7.19

Viết chương trình cho phép nhập từ bàn phím thông tin của n sản phẩm (n được nhập từ bàn phím), mỗi sản phẩm bao gồm các thông tin: mã hàng (chuỗi ký tự, độ dài tối đa 10 ký tự), tên hàng (chuỗi ký tự, độ dài tối đa 30 ký tự), số lượng(kiểu số nguyên), đơn giá (kiểu số thực), số tiền (kiểu số thực). Biết rằng số tiền = đơn giá * số lượng.

a. Viết hàm tính số tiền của mỗi mặt hàng;

b. Ghi thông tin của các sản phẩm vào tệp dssp.bin;

c. Đọc nội dung từ tệp dssp.bin ghi vào một mảng;

d. Cho biết mặt hàng có số tiền mua lớn nhất.

e. Nhập vào một mã hàng, cho biết mã hàng đó có trong cửa hàng hay không?

f. Sắp xếp các mặt hàng trong cửa hàng tăng dần theo số tiền thanh toán. Ghi thông tin đã sắp xếp vào tệp dssx.bin.

Lưu ý:

- Mỗi ý xây dựng bằng một hàm.

🕮 Bài tập 7.20

Cho mảng các số nguyên, tính tổng các phần tử của mảng.

Dữ liệu vào: tập tin văn bản ARRAY.INP gồm hai dòng.

- Dòng 1 chứa số nguyên n (n < = 10).

- Dòng 2 chứa n số nguyên.

Kết quả: Đưa ra tập tin văn bản ARRAY.OUT gồm 1 dòng ghi tổng các phần tử trong mảng.

🕮 Bài tập 7.21

Cho mảng các số nguyên, hãy liệt kê các phần tử là số nguyên tố.

Dữ liệu vào: tập tin văn bản NT.INP gồm hai dòng.

- Dòng 1 chứa số nguyên n (n < = 100);

- Dòng 2 chứa n số nguyên.

Kết quả: Đưa ra tập tin văn bản NT.OUT gồm hai dòng:

- Dòng 1 chứa số lượng các phần tử nguyên tố trong mảng.

- Dòng 2 liệt kê các số nguyên tố đó.

🕮 Bài tập 7.22*

Tạo file văn bản có tên là "INPUT.TXT" có cấu trúc như sau:

- Dòng đầu tiên ghi N (N là số nguyên dương nhập từ bàn phím).

- Trong các dòng tiếp theo ghi N số nguyên ngẫu nhiên trong phạm vi từ 0 đến 100, mỗi dòng 10 số (các số cách nhau ít nhất một khoảng trắng).

Hãy đọc dữ liệu của file "INPUT.TXT" và lưu vào mảng một chiều A.

Thực hiện các công việc sau:

a. Tìm giá trị lớn nhất của mảng A.
b. Đếm số lượng số chẵn, số lượng số lẻ của mảng A.
c. Hãy sắp xếp các phần tử theo thứ tự tăng dần.

Hãy ghi các kết quả vào file văn bản có tên OUTPUT.TXT theo mẫu sau:

INPUT.TXT
18 87 39 78 19 89 4 40 98 29 65 20 43 1 99 38 34 58 4

OUTPUT.TXT
Cau a: 99 Cau b: 9 9 Cau c: 1 4 4 19 20 29 34 38 39 40 43 58 65 78 87 89 98 99

🕮 Bài tập 7.23*

Viết chương trình nhập và lưu hồ sơ của sinh viên vào một file có tên là "DSSV.TXT". Sau đó đọc file "DSSV.TXT" và cất vào mảng, hãy sắp xếp các hồ sơ sinh viên theo thứ tự giảm dần theo điểm trung bình môn học rồi in ra màn hình hồ sơ các sinh viên theo thứ tự đó ra màn hình có thông tin như sau :

- Mã số sinh viên.

- Họ và tên sinh viên.

- Điểm trung bình kiểm tra.

- Điểm thi hết môn.

- Điểm trung bình môn học (tính bằng (điểm TBKT+điểm thi)/2).

🕮 Bài tập 7.24*

Tạo một file text có tên là "INPUT.TXT" có cấu trúc như sau:

- Dòng đầu tiên ghi hai số M và N (M, N là hai số nguyên dương nhập từ bàn phím).

- Trong M dòng tiếp theo mỗi dòng ghi N số nguyên ngẫu nhiên trong phạm vi từ 0 đến 100 (các số này cách nhau ít nhất một khoảng trắng).

INPUT.TXT
6 6 41 17 33 23 12 1 44 24 23 49 5 24 33 20 17 25 33 19 0 48 45 48 41 32 10 24 36 19 19 24 30 4 23 26 27 36

OUTPUT.TXT
Cau a: 49 Cau b: 17 19 Cau c: 127 169 147 214 132 146

Hãy đọc dữ liệu từ file trên và lưu vào mảng hai chiều. Rồi thực hiện các công việc sau:

a. Tìm giá trị lớn nhất của ma trận.
b. Đếm số lượng số chẵn, lẻ, nguyên tố có trong ma trận.
c. Hãy tính tổng các phần tử trên mỗi dòng của ma trận.

Hãy ghi kết quả này vào filetext có tên là "OUTPUT.TXT"

🕮 Bài tập 7.25*

Xét dãy số a1, a2,…, aN. Một đoạn con của dãy là dãy các phần tử liên tiếp nhau được xác định bởi chỉ số của số bắt đầu (L) và chỉ số của số cuối cùng (R). Tổng các số trên đoạn được gọi là tổng đoạn.

Yêu cầu: Cho dãy (a_N), hãy tìm đoạn con có tổng đoạn lớn nhất (T).

Dữ liệu được cho trong tập tin văn bản SUMMAX.INP.

- Dòng thứ nhất chứ số nguyên N (0 < N <= 30000).

- N dòng tiếp theo, mỗi dòng chứa một số là các số của dãy đã cho theo đúng thứ tự. Giá trị tuyệt đối của mọi số không vượt quá 30000.

Kết quả tìm được ghi vào tập tin văn bản SUMMAX.OUT gồm 1 dòng ghi 3 số T, L, R.

Ví dụ:

SUMMAX.INP
5
-1
5
-3
2
4

SUMMAX.OUT
8 2 5

🕮 Bài tập 7.26 *

Cho dãy (a_N), hãy tìm đoạn con tăng dần có tổng lớn nhất.

Dữ liệu: được cho trong tập tin AMAX.INP.

- Dòng 1 chứa số nguyên N (0 < N <= 30000).

- N dòng tiếp theo, mỗi dòng chứa một số là các số của dãy đã cho theo đúng thứ tự. Giá trị tuyệt đối của mỗi số không vược quá 30000.

Kết quả tìm được ghi vàp tin văn bản AMAX.OUT gồm hai dòng:

- *Dòng 1 ghi tổng của dãy con.*

- *Dòng 2 ghi mảng con tăng dần có tổng lớn nhất.*

🕮 Bài tập 7.27

Viết chương trình nhập lý lịch một nhân viên vào danh sách các nhân viên. Khi không nhập nữa bấm phím Esc và ghi vào tập tin NHANVIEN.DAT sau đó:

a. Đọc từ tập tin NHANVIEN.DAT vừa tạo và in danh sách các nhân viên lên màn hình.

b. Tìm và in lý lịch một nhân viên bằng các nhập và họ tên hoặc mã số nhân viên.

CHƯƠNG 8
ĐỒ HỌA

Lập trình đồ họa trong C được sử dụng để vẽ các hình dạng hình học khác nhau (hình chữ nhật, nhật thực hình tròn, v.v.), sử dụng chức năng toán học trong vẽ đường cong, tô màu một đối tượng với màu sắc và hoa văn khác nhau và các chương trình hoạt hình đơn giản như nhảy bóng và di chuyển ô tô.

8.1. GIỚI THIỆU

8.1.1. Chương trình đầu tiên

🕮 **Ví dụ 8.1**

```
1  #include<graphics.h>
2  #include<stdio.h>
3  #include<conio.h>
4  int main()
5  {
6     int gd = DETECT, gm;
7     int x1 = 200, y1 = 200;
8     int x2 = 300, y2 = 300;
9     clrscr();
10
11    initgraph(&gd, &gm, "");
12    line(x1, y1, x2, y2);
13    getch();
14    closegraph();
15 }
```

8.1.2. Giải thích về các lệnh

Bước đầu tiên trong bất kỳ chương trình đồ họa nào khai báo thư viện *"Graphics.h".* Thư viện *"Graphics.h"* cung cấp quyền truy cập vào thư viện đồ họa đơn giản cho phép vẽ các đường, hình chữ nhật, hình bầu dục, cung, đa giác, hình ảnh và chuỗi trên cửa sổ đồ họa.

Bước thứ hai là khởi tạo trình điều khiển đồ họa trên máy tính bằng phương thức initgraph của thư viện Graphics.h.

```
void initgraph(int *graphicsDriver, int *graphicsMode, char *driverDirectoryPath);
```

Hàm initgraph khởi tạo hệ thống đồ họa bằng cách tải trình điều khiển đồ họa, thay đổi hệ thống sang chế độ đồ họa. Nó cũng đặt lại hoặc khởi tạo tất cả các cài đặt đồ họa như màu sắc, bảng màu, vị trí hiện tại, v.v. về các giá trị mặc định của chúng. Hàm initgraph có ba tham số đầu vào:

GraphicsDriver: là một con trỏ tới một số nguyên, chỉ định trình điều khiển đồ họa sẽ được sử dụng. Nó báo cho trình biên dịch biết sử dụng trình điều khiển đồ họa nào hoặc tự động phát hiện ổ đĩa. Trong tất cả các chương trình, chúng ta sẽ sử dụng macro DETECT của thư viện Graphics.h hướng dẫn trình biên dịch để tự động phát hiện trình điều khiển đồ họa.

GraphicsMode: là một con trỏ tới một số nguyên, chỉ định chế độ đồ họa sẽ được sử dụng. Nếu *gdriver được thiết lập thành DETECT, thì initgraph thiết lập * gmode thành độ phân giải cao nhất có sẵn cho trình điều khiển.

driverDirectoryPath: chỉ định đường dẫn thư mục chứa các tệp trình điều khiển đồ họa (tệp BGI). Nếu đường dẫn thư mục không được cung cấp, thì nó sẽ tìm kiếm các tệp trình điều khiển trong thư mục làm việc hiện tại. Trong tất cả các chương trình đồ họa mẫu, chúng ta phải thay đổi đường dẫn của thư mục BGI cho phù hợp với trình biên dịch Turbo C++ được cài đặt. (Nếu sử dụng DevC, việc thiết lập đường dẫn này sẽ phức tạp hơn).

Bước tiếp theo, khai báo các biến để theo dõi điểm bắt đầu và điểm kết thúc

```
int x1=200, y1=200;
int x2=300, y2=300;
```

Truyền 4 tham số vào hàm line()

```
line(x1,y1,x2,y2);
```

Hàm line() vẽ một đường thẳng từ điểm (x1, y1) đến điểm (x2,y2)

Cú pháp hàm line()

line(x1,y1,x2,y2);

Trong đó:

x1, y1 là tung độ và hoành độ của điểm thứ nhất;

x2, y2 là tung độ và hoành độ của điểm thứ hai.

Cuối chương trình đồ họa, chúng ta phải dỡ trình điều khiển đồ họa và đặt màn hình trở lại chế độ văn bản bằng cách gọi hàm closegraph();

8.2. CÁC HÀM ĐỒ HỌA

8.2.1. Thiết lập chế độ đồ họa

Nếu chúng ta muốn tạo bất kỳ hình ảnh đồ họa nào trên màn hình đầu ra của trình biên dịch C thì việc đầu tiên cần làm là khai báo tệp tiêu đề "Graphics.h". Thư viện "Graphics.h" chứa tất cả các định nghĩa và giải thích về các hàm đồ họa mà người dùng cần trong lập trình đồ họa.

Graphics Driver

Để hiểu khái niệm về trình điều khiển đồ họa, chúng ta phải hiểu khái niệm về trình điều khiển thiết bị. Trình điều khiển thiết bị là một số chương trình nhỏ giao tiếp trực tiếp với phần cứng. Người dùng không thể giao tiếp trực tiếp với máy, mà phải thông qua các chương trình nhỏ cần thiết để giao tiếp, được gọi là trình điều khiển thiết bị.

Trình điều khiển đồ họa là một chương trình nhỏ của trình điều khiển thiết bị, được áp dụng trên chế độ đồ họa. Trong các chương trình về đồ họa, có một biến nguyên gd đại diện cho trình điều khiển đồ họa, người dùng cũng có thể chọn bất kỳ tên biến nào khác thay vì gd cho trình điều khiển đồ họa.

Hàm initgraph() tự động chọn trình điều khiển đồ họa. Đối số đầu tiên của phương thức initgraph() đại diện cho địa chỉ của trình điều khiển đồ họa.

Graphics Mode

Để tạo ra các hình ảnh độ họa trong C, chúng ta phải chuyển qua chế độ đồ họa. Số lượng chế độ đồ họa trong hệ thống máy tính phụ thuộc vào bộ điều hợp (adapter) và màn hình của máy tính.

Mỗi chế độ đồ họa có một số cụ thể, trong tất cả các chế độ có sẵn của hệ thống máy tính, chúng ta có thể chuyển sang chế độ đồ họa cung cấp độ phân giải tốt nhất. Để chuyển qua chế độ đồ họa cung cấp độ phân giải phù hợp, chúng ta cần gọi hàm initgraph(). Hàm này phát hiện chế độ đồ họa có độ phân giải phù hợp và đặt số tương ứng với chế độ đó trong

biến gm. Trong ví dụ 8.1, chúng tôi đã khai báo một biến số nguyên gm đại diện cho chế độ đồ họa. Bạn cũng có thể chọn bất kỳ tên biến nào khác thay vì gm cho chế độ đồ họa.

Trong phương thức initgraph() ở ví dụ 8.1, bao gồm tham số đầu tiên gd là địa chỉ của trình điều khiển đồ họa, tham số thứ hai gm là chế độ đồ họa.

Phương thức initgraph()

Phương thức initgraph() được sử dụng để quyết định trình điều khiển đồ họa phù hợp, khởi tạo chế độ đồ họa để đặt độ phân giải màn hình tối đa và chọn đường dẫn tệp BGI thích hợp nhất. Trong ví dụ 8.1, phương thức initgraph() có ba đối số, trong đó đối số thứ ba được sử dụng để chọn đường dẫn của tệp BGI, các tệp BGI là phần mở rộng của trình điều khiển đồ họa.

Phương thức initgraph() được sử dụng để quyết định trình điều khiển đồ họa phù hợp, khởi tạo chế độ đồ họa để đặt độ phân giải màn hình tối đa và chọn đường dẫn tệp BGI thích hợp.

Trong chương trình trên trong phương thức initgraph() có ba đối số. Chúng tôi đã thảo luận về hai đối số đầu tiên là địa chỉ của trình điều khiển đồ họa (& gd) và địa chỉ của chế độ đồ họa (& gm).

Đối số thứ ba của phương thức initgraph() được sử dụng để chọn đường dẫn có tệp BGI của bạn. Các tệp BGI này là phần mở rộng của trình điều khiển đồ họa.

Phương thức closegraph()

Phương thức closegraph() được sử dụng để để đóng (thoát) chế độ đồ họa.

Phương thức restorecrtmode ()

Phương thức restorectmode() được sử dụng để khôi phục chế độ đồ họa gốc. Hàm này có thể được sử dụng cùng với setgraphmode() để chuyển đổi quan lại giữa chế độ văn bản và chế độ đồ họa.

8.2.2. Thiết lập màu sắc

Có 16 màu được khai báo trong tệp tiêu đề Graphics.h. Chúng ta sử dụng màu sắc để đặt màu vẽ hiện tại, thay đổi màu nền, thay đổi màu văn bản, để tô màu hình kín, v.v. Để chỉ định màu, chúng ta có thể sử dụng các hằng số màu như setcolor (RED) hoặc mã số nguyên tương ứng của chúng như setcolor (4). Dưới đây là mã màu theo thứ tự tăng dần.

Bảng 8.1. Bảng màu được sử dụng trong chương trình đồ họa

Tên hằng	Giá trị	Backgroup Màu nền	Foregroup Màu ký tự	Màu hiển thị
BLACK	0	Có	Có	Đen
BLUE	1	Có	Có	Xanh da trời
GREEN	2	Có	Có	Xanh lá cây
CYAN	3	Có	Có	Xanh lơ
RED	4	Có	Có	Đỏ
MAGENTA	5	Có	Có	Tím
BROWN	6	Có	Có	Nâu
LIGHTGRAY	7	Có	Có	Xám nhạt
DARKGRAY	8	Không	Có	Xám đậm
LIGHTBLUE	9	Không	Có	Xanh da trời nhạt
LIGHTGREEN	10	Không	Có	Xanh da trời đậm
LIGHTCYAN	11	Không	Có	Xanh lơ nhạt
LIGHTRED	12	Không	Có	Đỏ nhạt
LIGHTMAGENTA	13	Không	Có	Tím nhạt
YELLOW	14	Không	Có	Vàng
WHITE	15	Không	Có	Trắng

Để thiết lập màu nền sử dụng hàm setbkcolor()

Cú pháp:

```
void setbkcolor(int color);
```

Hàm setbkcolor đặt nền thành màu được chỉ định bởi tên hằng màu hoặc giá trị của màu đó (tham chiếu Bảng 8.1, những tên hằng màu hoặc giá trị này được định nghĩa trong Graphics.h). Những màu này cũng có thể được sử dụng để thiết lập textcolor (màu của văn bản) hoặc điền vào các hình dạng khác nhau mà bạn tạo trong chương trình của mình.

🕮 **Ví dụ 8.2**

```
1  #include<graphics.h> /* header file */
2  #include<conio.h>
3  main()
4  {
```

```
5    /* hai dòng sau đây là cú pháp để viết một chương trình đồ
     họa.*/
6      int gd = DETECT, gm;
7      initgraph(&gd, &gm, "");
8      setbkcolor (GREEN);
9      getch();
10     closegraph();
11     return 0;
12   }
```

✍ Kết quả chương trình

Thiết lập màu nền màu xanh lá cây

8.2.3. Các hàm đồ họa để vẽ hình

Khi chúng ta muốn vẽ một hình bất kỳ nào đó trên màn hình, chúng ta cần các hàm khác nhau được xác định trong thư viện Graphics.h. Các hàm này được sử dụng để vẽ các hình ảnh khác nhau trên màn hình và cũng để cung cấp các chế độ đồ họa hấp dẫn theo yêu cầu của người sử dụng.

Các hàm đồ họa được sử dụng để phát triển các chương trình đồ họa, vẽ hình, tạo hình ảnh và phát triển trò chơi và hoạt hình, v.v.

Các hàm chứa trong thư viện Graphics.h được sử dụng để vẽ các hình dạng như hình tròn, hình elip, hình chữ nhật, đường thẳng và các hình dạng hình học khác. Chúng ta có thể tô màu cho các hình ảnh này và màu sắc cũng có thể được thay đổi bằng cách sử dụng các hàm thiết lập màu sắc.

Sau đây là bảng cung cấp tổng quan về các hàm trong thư viện Graphics.h thường được sử dụng để vẽ các hình đồ họa:

Bảng 8.2.Các hàm trong Graphics.h được sử dụng để vẽ đồ hoạ

Tên hàm	Chức năng	Ví dụ minh họa
putpixel(x, y)	Vẽ một pixel tại vị trí lưới (x, y).	putpixel(320, 240): vẽ một pixel với màu, đặt ở chính giữa màn hình.
getpixel(x, y)	Lấy màu của pixel nằm ở vị trí (x, y).	getpixel(320, 240): lấy màu ở chính giữa màn hình có độ phân giải 640 x 480.
line(x1, y1, x2, y2)	Vẽ một đường thẳng từ điểm (x1, y1) đến điểm (x2, y2).	line(0, 0, 320, 240): vẽ một đường thẳng từ bên trái màn hình đến trung tâm màn hình.
circle(xc, yc, r)	Vẽ một vòng tròn có tâm (xc, yc) và bán kính = r.	circle(320, 240, 10): Vẽ một vòng tròn có tâm ở chính giữa màn hình, có bán kính = 10 đơn vị
rectangle(x1, y1, x2, y2)	Vẽ một hình chữ nhật với 2 góc có tọa độ (x1, y1) và (x2, y2).	rectangle(320, 240, 400, 300): vẽ một hình chữ nhật sao cho (320, 240) là tọa độ trên cùng bên trái và (400, 300) là tọa độ dưới cùng bên phải.
ellipse(xc, yc, s, e, xr, yr)	Vẽ một hình elip với (xc, yc) làm tâm, xr, yr là các trục lớn và trục bé. Nếu s = 0, e = 180, chỉ có nửa trên của hình elip được vẽ.	ellipse(320, 240, 0, 360, 100, 50): vẽ toàn bộ hình elip có tâm (320, 240) - ở chính giữa màn hình, có trục lớn và trục bé tương ứng là 100 và 50.
arc(xc, yc, s, e, r)	Vẽ một cung có tâm (xc, yc), bán kính = r, góc bắt đầu và kết thúc là s và e tương ứng.	arc(320, 240, 45, 135, 100): vẽ một cung có tâm ở chính giữa màn hình (320, 240), bán kính 100 đơn vị, góc bắt đầu là 45 độ, góc kết thúc là 135 độ.

Tên hàm	Chức năng	Ví dụ minh họa
moveto(x, y)	Di chuyển con trỏ đến tọa độ (x, y).	moveto(320, 240): di chuyển con trỏ đến chính giữa màn hình (có tọa độ (320, 240)) từ vị trí hiện tại của nó.
lineto(x, y)	Vẽ một đường thẳng từ vị trí hiện tại đến điểm (x, y).	lineto(320, 240): vẽ một đường thẳng từ vị trí hiện tại đến chính giữa màn hình.
moverel(xr, yr)	Di chuyển con trỏ đến điểm mới có tọa độ tương đối, theo hướng trục x thêm xr đơn vị, hướng trục y (xr + yr) đơn vị.	moverel(10, 5): di chuyển con trỏ theo hướng trục x thêm 10 đơn vị và theo hướng trục y thêm 15 đơn vị.
linerel(xd, yd)	vẽ đường thẳng từ vị trí con trỏ vẽ hiện tại (giả sử con trỏ vẽ hiện tại có toạ độ (x, y)) đến điểm có toạ độ (x + dx, y + dy), vẽ xong con trỏ tới điểm mới.	linerel(10, 5): vẽ một đường thẳng từ vị trí hiện tại đến một điểm có khoảng cách từ điểm hiện tại thêm 10 đơn vị theo trục xvà 5 đơn vị theo trục y.
outtextxy(x, y,“string”)	Hiển thị chuỗi ký tự trong dấu “ ”, tại vị trí (x, y).	outtextxy(320, 240, “DHCNGTVT”): Hiển thị chuỗi DHCNGTVT ở chính giữa màn hình (có tọa độ (320, 240))
setlinestyle(x,y,z)	Vẽ các đường thẳng với các kiểu của đường thẳng được chỉ định. x: kiểu đường thẳng. x = 0: nét liền. x = 1: nét chấm chấm.	setlinestyle(0, 1,1): thiết lập hệ thống vẽ một đường có nét liền, có độ dày 1 và với một mẫu cụ thể.

Tên hàm	Chức năng	Ví dụ minh họa
	x = 2: đường trung tâm. x = 3: nét đứt. x = 4: do người dùng định nghĩa. y: là mẫu. z: độ dày.	
setcolor(x)	Thiết lập màu cho các đường thẳng, có tất cả 16 màu.	setcolor(RED): thiết lập màu cho bút vẽ là màu đỏ.
setfillstyle(x,y)	Đặt mẫu tô (x) và mầu tô (y) cho các hình đặc và miền đóng. Màu có 16 giá trị từ 0 đến 15, mẫu có 12 giá trị từ 0 đến 12.	setfillstyle(SOLID_FILL, RED): lấp đầy vùng theo mẫu đặc với màu đỏ.
fillellipse(xc, yc, xrad, yrad)	Điền đầy vào hình elip có tâm (xc, yc), có hai trục tại xrad và yrad.	fillellipse(320, 240, 100, 50): vẽ hình elip có tâm ở giữa màn hình và hai trục tại x = 100 và y = 50.
drawpoly(x, y)	Vẽ hình đa giác trong đó x là số điểm được sử dụng để vẽ đa giác, y: địa chỉ cơ sở của mảng chứa các điểm phối hợp.	drawpoly(4, array): vẽ một đa giác chứa 4 điểm và địa chỉ cơ sở được chứa trong một mảng array[].
fillpoly(x, y)	Lấp đầy một đa giác, với x: số lượng điểm được sử dụng để xây dựng đa giác và y: địa chỉ cơ sở của mảng chứa các điểm phối hợp.	fillpoly(4, array): vẽ một đa giác chứa 4 điểm và địa chỉ cơ sở được chứa trong một mảng array[]. Nó cũng lấp đầy đa giác bằng cách sử dụng kiểu và màu tô hiện tại.
settextstyle(x, y, z)	Thiết lập kiểu font ký	settextstyle(DEFAULT_f

Tên hàm	Chức năng	Ví dụ minh họa
	tự: x: Kiểu font y: hướng font z: kích thước	ont, HORIZ_DIR, 4): Thiết lập theo font chữ hiện tại, hướng từ trái qua phải, kích thước 4.
sound(x)	Kích hoạt loa ở một thời điểm cụ thể, đơn vị tính là x Hz.	sound(7): Kích hoạt âm thanh ở 7Hz.
nosound()	Tắt âm thanh kích hoạt loa trước đó.	
delay(x)	Cho phép lệnh thực hiện trước đó vẫn được kích hoạt trong một đơn vị thời gian là x (msec)	Delay(6000): cho phép lệnh thực hiện trong 6s.
getimage(int x1, int y1, int x2, int y2, void *p)	Chép các điểm ảnh trong vùng chữ nhật (x1, y1, x2, y2) vào vùng nhớ do con trỏ p trỏ tới (vùng nhớ này và con trỏ p cho bởi hàm malloc, độ lớn vùng nhớ cho bởi hàm imagesize).	
bar(left, top, right, bottom)	Vẽ một thanh hình chữ nhật hai chiều đầy. Hàm không vẽ ranh giới thanh, hình được vẽ sử dụng mẫu và màu hiện tại.	bar(20, 30, 40, 70): Vẽ hình chữ nhật đầy hai chiều, có góc trên bên trái là (20, 30) và góc dưới bên phải là (40, 70).

8.3. MỘT SỐ VÍ DỤ

🕮 Ví dụ 8.3

Vẽ các hình chữ nhật, hình tròn, thanh bar, đường thẳng, hình elip và hiển thị trong chương trình “Chuong trinh ve cac hinh don gian” trên màn hình.

```
1  #include<graphics.h>
2  #include<conio.h>
3   main()
4  {
5  int gd = DETECT, gm;
   int left=100, top=100, right=200, bottom=200;
   int x= 300, y=150, radius=50;
6    initgraph(&gd, &gm, "");// tham so thu ba de trong, la thu muc
   chua BGI, tuy vao viec cai tool ho tro
7     rectangle(left, top, right, bottom);
8    circle(x, y, radius);
9    bar(left + 300, top, right + 300, bottom);
10   line(left - 10, top + 150, left + 410, top + 150);
11   ellipse(x, y + 200, 0, 360, 100, 50);
12    outtextxy(left + 100, top + 325, "Chuong trinh ve hinh don
   gian");
13    getch();
14   closegraph();
15   return 0;
16 }
```

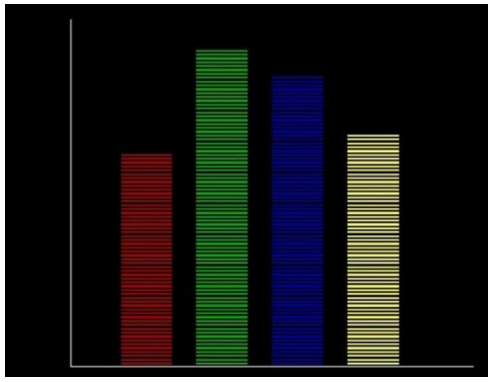
🕮 Ví dụ 8.4

Viết chương trình vẽ các thanh biểu đồ.

```
1   #include <graphics.h>
2   #include <conio.h>
3   #include <dos.h>
4   #include <stdlib.h>
5   void main() {
6           //Khoi tao trinh dieu khien do hoa
7           // va bien che do do hoa
8           int graphicdriver=DETECT, graphicmode;
9           //goi ham initgraph voi cac tham so
10          initgraph(&graphicdriver,&graphicmode," ");
11          //In thong diep tren man hinh
12          outtextxy(10, 10 + 10, "Chuong trinh ve thanh bieu do
   trong C ");
```

```
13          // khoi tao duong thang
14          line(100,420,100,60);
15          line(100,420,500,420);
16          //tao thanh bieu do
17          setfillstyle(LINE_FILL,RED);
18          bar(150,200,200,419);
19          setfillstyle(LINE_FILL,GREEN);
20          bar(225,90,275,419);
21          setfillstyle(LINE_FILL,BLUE);
22          bar(300,120,350,419);
23          setfillstyle(LINE_FILL,YELLOW);
24          bar(375,180,425,419);
25          getch();
26  }
```

✍ **Kết quả chương trình:**



🕮 **Ví dụ 8.5**

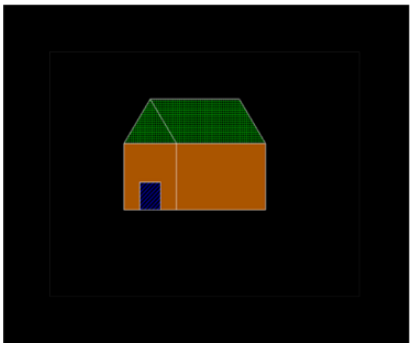
Sử dụng các hàm đồ họa vẽ ngôi nhà.

✍ **Chương trình:**

```
1   #include<graphics.h>
2   #include<conio.h>
3    int main(){
4           int gd = DETECT,gm;
5           initgraph(&gd, &gm, "");
6           /* Ve ngoi nha */
```

```
7           setcolor(WHITE);
8           rectangle(150,180,250,300);
9           rectangle(250,180,420,300);
10          rectangle(180,250,220,300);
11
12          line(200,100,150,180);
13          line(200,100,250,180);
14          line(200,100,370,100);
15          line(370,100,420,180);
16
17          /* Do mau */
18          setfillstyle(SOLID_FILL, BROWN);
19          floodfill(152, 182, WHITE);
20          floodfill(252, 182, WHITE);
21          setfillstyle(SLASH_FILL, BLUE);
22          floodfill(182, 252, WHITE);
23          setfillstyle(HATCH_FILL, GREEN);
24          floodfill(200, 105, WHITE);
25          floodfill(210, 105, WHITE);
26          getch();
27          closegraph();
28          return 0;
29  }
```

✍ Kết quả chương trình



🕮 Ví dụ 8.6

Vẽ xe ô tô chuyển động.

✍ Chương trình:

```
1  #include <graphics.h>
2  #include <dos.h>
3  int main()
4  {
5    int i, j = 0, gd = DETECT, gm;
6    initgraph(&gd,&gm," ");
7    settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
8    outtextxy(25,240,"Nhan mot phim de xem xe chuyen dong!");
9    getch();
10   for( i = 0 ; i <= 420 ; i = i + 10, j++)
11   {
12     rectangle(50+i,275,150+i,400);
13     rectangle(150+i,350,200+i,400);
14     circle(75+i,410,10);
15     circle(175+i,410,10);
16     setcolor(j);
17     delay(100);
18     if( i == 420)
19       break;
20     if ( j == 15)
21       j = 2;
22     cleardevice(); // xoa man hinh
23   }
24   getch();
25   closegraph();
26   return 0;
27 }
```

🕮 Ví dụ 8.7

Sử dụng hàm bar3D() vẽ thanh hình chữ nhật ba chiều.

Gợi ý: Hàm bar3d(int left, int top, int right, int bottom, int depth, int topflag): có 6 tham số trong đó:

left: chỉ định tọa độ X của góc trên bên trái;

top: chỉ định tọa độ Y của góc trên bên trái;

right: chỉ định tọa độ X của góc dưới bên phải;

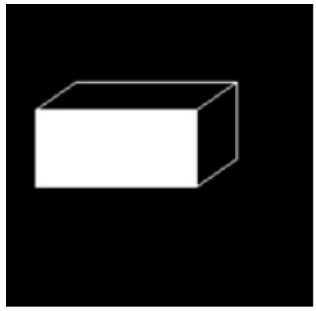
bottom: chỉ định tọa độ Y của góc dưới bên phải;

depth: chỉ định độ sâu của thanh tính theo pixel.

topflag: là một tham số quan trọng xác định rằng đỉnh 3D có nên được đặt trên thanh hay không. Nếu giá trị của topflag = 1, thì nó sẽ đặt đỉnh trên thanhvà ngược lại nếu khác 1, nó sẽ không đặt đỉnh trên thanh.

```
1  #include <stdio.h>
2  #include <conio.h>
3  #include <graphics.h>
4  void main()
5  {
6  int gd = DETECT,gm;
7  initgraph(&gd,&gm, " ");
8  bar3d( 100,150,200,200,25,1);
9  getch();
10 closegraph();
11 }
```

✍ Kết quả chương trình:



🕮 Ví dụ 8.8

Vẽ bầu trời đầy sao, dừng màn hình lấy giá trị màu tại điểm ảnh có tọa độ (100, 100).

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <stdlib.h>
void main()
{
int driver = 0, mode = 0, maloi;
initgraph(&driver,&mode," ");
if ( (maloi = graphresult()) != 0 )
{
    printf("khong the khoi dong do hoa \n");
    printf("ma loi : &d \nnguyen nhan loi %s ",maloi, grapherrormsg(maloi) );
    getch();
    exit(1);
 }
randomize(); // khởi tạo hàm radom (cho số ngẫu nhiên)
int I = 0;
while (i<200)
{
// random(639): so ngau nhien trong khoang tu 0 den 639
// ve mot diem co toa do (x, y) cho ngau nhien voi mau trang
putpixel(random(639), random(476), WHITE);
i++;
}
getch();
// lay mau cua diem anh tren man hinh
// tai vi tri co toa do (100, 100)
int mau = getpixel(100,100); /* closegraph();
printf("mau cua diem anh tai vi tri co toa do (100,100) la : %d", mau);
getch();
}
```

🕮 Ví dụ 8.9

Cho quả bóng chuyển động ngang trên màn hình, hết chiều ngang màn hình bóng quay lại.

Gợi ý:

- Hàm unsigned imagesize(int x1, int y1, int x2, int y2) cho số byte cần thiết để lưu ảnh trong phạm vi hình chữ nhật (x1, y1, x2, y2).
- Hàm void getimage(int x1, int y1, int x2, int y2, void *p): chép các điểm ảnh trong vùng chữ nhật (x1,y1,x2,y2) vào vùng nhớ do con trỏ p trỏ tới (vùng nhớ này và con trỏ p cho bởi hàm malloc(), độ lớn vùng nhớ cho bởi hàm imagesize()).
- Hàm void putimage(int x, int y, void *p, int kiểu_chép): chép ảnh tử vùng nhớ do con trỏ p trỏ tới ra màn hình sao cho góc trên trái vùng ảnh có toạ độ (x, y).

Tạo ảnh chuyển động không có màn hình nền:

Bước 1: Tạo ảnh bằng màu vẽ.

Bước 2: Xoá ảnh bằng màu nền.

Bước 3: Tạo lại ảnh bằng màu vẽ tại vị trí mới.

✍ Chương trình:

```
1   #include <stdio.h>
2   #include <conio.h>
3   #include <graphics.h>
4   #include <stdlib.h>
5   #include <dos.h>
6   void bong(int x, int y, int r, int mau)
7   {
8   setcolor(mau);
9   setfillstyle(1,mau);
10  pieslice(x,y,0,360,r);
11  }
12  void main()
13  {
14  int driver = 0, mode = 0, maloi;
15  initgraph(&driver,&mode," ");
16  if ( (maloi = graphresult()) != 0)
17  {
```

```
18  printf("khong the khoi dong do hoa \n");
19  printf("ma loi: %d \nnguyen nhan loi %s", maloi,
    grapherrormsg(maloi));
20  getch();
21  exit(1);
22  }
23  setbkcolor(CYAN);
24  int x = 30,y = 100;
25  do
26  {
27  if (x >= 640-30) x = 50;
28  bong(x,y,30,CYAN);
29  x + = 4;
30  bong(x,y,30,RED);
31  delay(200); // t tam dung 20 mili giay
32  } while (!kbhit()); // lap vo han toi khi nhan 1 phim bat ky thi
    dung
33  getch();
34  closegraph();
35  }
```

🕮 Ví dụ 8.10

Vẽ quả bóng màu đỏ và nhớ vào vùng nhớ p, xoá màn hình, vẽ bầu trời sao, cho quả bóng chuyển động ngẫu nhiên trên màn hình đến khi ấn một phím bất kỳ.

Gợi ý:

Tạo ảnh chuyển động mà không xoá màn hình nền.

Vẽ ảnh trong vùng chữ nhật (x1, y1, x2, y2), dùng hàm imagesize() tính số byte cần thiết để lưu ảnh trong vùng chữ nhật vừa vẽ (giả sử là n byte), dùng hàm malloc() để cấp phát vùng nhớ n byte và cho con trỏ p trỏ và vùng nhớ này, dùng hàm getimage() chép ảnh từ vùng chữ nhật vào vùng nhớ p, xoá màn hình, tạo màn hình nền, dùng lệnh putimage() với kiểu chép XOR_PUT để in ảnh lưu trong P ra màn hình với toạ độ góc trên bên trái là(x, y), tạm dừng chương trình một lát, dùng putimage() với kiểu chép XOR_PUT in lại ảnh lưu trong P ra màn hình cũng tại (x, y) để xoá ảnh cũ (vì 1 xor 1 cho 0), thay đổi các thành phần của toạ độ (x, y) in ảnh tại toạ độ mới, v.v.

✍ Chương trình:

```
1  #include <stdio.h>
2  #include <conio.h>
3  #include <graphics.h>
4  #include <stdlib.h>
5  #include <dos.h>
6  void bong(int x, int y, int r, int mau)
7  {
8  setcolor(mau);
9  setfillstyle(1, mau);
10 pieslice(x , y, 0, 360, r);
11 }
12 void main()
13 {
14 int driver=0, mode = 0, maloi;
15 initgraph(&driver,&mode,"");
16 if ((maloi=graphresult()) !=0)
17 {
18 printf("khong the khoi dong do hoa \n");
19 printf("ma loi: %d \nnguyen nhan loi %s", maloi,
   grapherrormsg(maloi));
20 getch();
21 exit(1);
22 }
23 setcolor(RED);
24 setfillstyle(1,RED);
25 pieslice(50,50,0,360,50);
26 char *p;
27 int n= imagesize(0, 0, 100, 100);
28 p = (char *)malloc(n);
29 getimage(0, 0, 100, 100, p);
30 getch();
31 cleardevice();
32 randomize();
33 for (int i=0; i<=300; i++)
```

```
{
int k = random(16);
setcolor(k);
setfillstyle(1,k);
pieslice(random(640), random(480), 0, 360, 2);
}
int x=random(640-100), y=random(480-100),x1,y1;
do
{
// in bong mau do voi phep XOR
// ve bong tren man hinh, vi tri nao da co mau lan truoc
// thi in de len bong
putimage(x,y,p,1);
delay(100); // tam dung
// in voi phep XOR mot lan nua
// xoa bong vua in tren man hinh do bong lan truoc in la mau do
// in lai mau do theo phep XOR, 1 XOR 1 = 0
// tuc la hai mau do de len nhau se ra mau nen cu truoc do
putimage(x,y,p,1);
  do
{
// muc dich cua vong lap do while nay
// nham khong cho qua bong hien thi ngoai vung man hinh
// (0, 0, 540, 340) và tao gia tri toa do ngau nhien moi cho bong
x1 =x+random(50) -25;
y1 =y+random(50) -25;
} while ( (x1<0) || ( x1> (639-100) ) || (y1<0) || (y1>(439-100) ) )
;
x=x1;
y=y1;
}while(!kbhit()) ;// lap vo han cho den khi nhan 1 phim bat ky
getch();
closegraph();
}
```

🕮 Ví dụ 8.11

Sử dụng ngôn ngữ C vẽ khuôn mặt cười xuất hiện tại các vị trí ngẫu nhiên trên màn hình.

✍ Chương trình:

```
1  #include<graphics.h>
2  #include<conio.h>
3  #include<stdlib.h>
4   main()
5  {
6  int gd = DETECT, gm, area, temp1, temp2, left = 25, top = 75;
7  void *p;
8  initgraph(&gd, &gm, "C:\\TC\\BGI"); // chay bang tool Tubro C,
   thu vien chua BGI
9  setcolor(YELLOW);
10 circle(50, 100, 25);
11 setfillstyle(SOLID_FILL, YELLOW);
12 floodfill(50, 100, YELLOW);
13 setcolor(BLACK);
14 setfillstyle(SOLID_FILL, BLACK);
15 fillellipse(44, 85, 2, 6);
16 fillellipse(56, 85, 2, 6);
17 ellipse(50, 100, 205, 335, 20, 9);
18 ellipse(50, 100, 205, 335, 20, 10);
19 ellipse(50, 100, 205, 335, 20, 11);
20 area = imagesize(left, top, left + 50, top + 50);
21 p = malloc(area);
22 setcolor(WHITE);
23 settextstyle(SANS_SERIF_FONT, HORIZ_DIR, 2);
24 outtextxy(155, 451, "Mat cuoi");
25 setcolor(BLUE);
```

```
26  rectangle(0, 0, 639, 449);
27  while(!kbhit())
28  {
29  temp1 = 1 + random (588);
30  temp2 = 1 + random (380);
31  getimage(left, top, left + 50, top + 50, p);
32  putimage(left, top, p, XOR_PUT);
33  putimage(temp1, temp2, p, XOR_PUT);
34  delay(100);
35  left = temp1;
36  top = temp2;
37  }
38  getch();
39  closegraph();
40  return 0;
41  }
```

BÀI TẬP CHƯƠNG 8

A. BÀI TẬP CÓ LỜI GIẢI

🕮 Bài tập 8.1

Viết chương trình bằng ngôn ngữ C in dòng chữ Hello Word ra màn hình.

✍Chương trình

```
1   #include<stdio.h>
2   #include<conio.h>
3   int main()
4   {
5           textcolor(RED);
6           clrcsr();
7           cprintf("HELLO WORLD\N");
8           getch();
9           return 0;
10  }
```

✍Kết quả chương trình

🕮 Bài tập 8.2

Viết chương trình in ra các vòng tròn xoắn ốc.

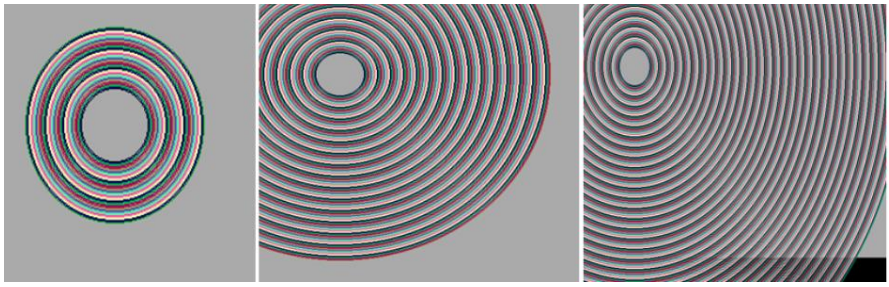
✍Chương trình

```
1   #include<stdio.h>
2   #include<conio.h>
3   #include<graphics.h>
4   #include<dos.h>
```

```
5   void main()
6   {
7           int gd=DETECT, gm; int i;
8           initgraph (&gd,&gm," ");
9           clrscr();
10          for(i=0;i<500;i++)
11          {
12                  setcolor(i);
13                  circle(100,100,30 + i);
14                  delay(30);
15          }
16          getch();
17          closegraph();
18  }
```

✍ Kết quả chương trình

Các vòng tròn tiếp tục được phát triển cho đến khi bằng số lượng vòng tròn đã chỉ định.



B. BÀI TẬP TỰ GIẢI

🕮 Bài tập 8.3

Viết chương trình bằng ngôn ngữ C để vẽ hai hình chữ nhật và điền màu trắng vào một trong hai hình đó.

🕮 Bài tập 8.4

Viết chương trình bằng ngôn ngữ C để vẽ hình tròn.

🕮 Bài tập 8.5

Viết chương trình bằng ngôn ngữ C để thay đổi màu nền của màn hình từ màu đen sang màu trắng.

🕮 Bài tập 8.6

Viết chương trình vẽ các điểm ảnh với màu đã cho trên màn hình.

🕮 Bài tập 8.7

Viết chương trình bằng ngôn ngữ C để vẽ biểu đồ hình tròn, cho thấy tỷ lệ phần trăm của các thành phần khác nhau được vẽ với các kiểu và màu sắc khác nhau.

🕮 Bài tập 8.8*

Viết một chương trình vẽ hình trong C, bằng cách sử dụng chuột vẽ các hình dạng khác nhau như đường thẳng, hình tròn và nhiều hình dạng khác. Người dùng có thể thay đổi màu sắc, xóa màn hình.

PHỤ LỤC 1: TỪ KHÓA TRONG C

Nhóm 1: Từ khóa khai báo kiểu dữ liệu

Từ khóa	Ý nghĩa
char	Kiểu số nguyên 1 byte, ký tự
int	Kiểu số nguyên 2 byte
short	Kiểu số nguyên 2 byte
long	Kiểu số nguyên 4 byte
unsigned	Số nguyên dương (đặt trước các từ khóa số nguyên để tăng phạm vi biểu diễn số nguyên dương)
signed	Có dấu (mặc định số nguyên là signed)
float	Kiểu số thực 4 byte
double	Kiểu số thực 8 byte
void	Dùng với hàm không có giá trị trả về hoặc với con trỏ chưa xác định được giá trị trả về là kiểu gì

Nhóm 2: Từ khóa điều kiện, lựa chọn

Từ khóa	Ý nghĩa
if	Nếu … thì
else	Trường hợp còn lại (thường dùng với if)
switch	Với các trường hợp của giá trị (rẽ nhiều nhánh)
case	Với 1 trường hợp (thường dùng với switch)
default	mặt định (thường dùng với switch)

Nhóm 3: Từ khóa lặp

Từ khóa	Ý nghĩa
for	Lặp từ … đến
do	Lặp trước kiểm tra sau
while	Kiểm tra trước, lặp sau

Nhóm 4: Từ khóa lệnh nhảy

Từ khóa	Ý nghĩa
continue	Tiếp tục (bỏ qua lần lặp này, chạy tiếp lần lặp sau)
break	Nhảy ra ngoài khối lệnh
return	Kết thúc hàm và trả về kết quả
goto	Nhảy đến nhãn …

Nhóm 5: Từ khóa cấu trúc

Từ khóa	Ý nghĩa
enum	Kiểu liệt kê
typedef	Định nghĩa kiểu mới
struct	Cấu trúc
union	Cho phép tạo các kiểu dữ liệu khác nhau trong cùng một vùng nhớ

Nhóm 6: Các từ khóa khác

Từ khóa	Ý nghĩa
asm	Khai báo bắt đầu mã Assembly
const	Khai báo hằng (read only)
static	Khai báo tĩnh (vùng nhớ cố định)
extern	Tham chiếu đến biến cùng tên nào đó, đã được định nghĩa bên ngoài
register	Đặt trước các biến nhằm yêu cầu trình biên dịch duy trì giá trị biến đó trên thanh ghi của máy tính nhằm tăng tốc độ thực hiện
auto	Dùng để khai báo các biến cục bộ (tuy nhiên rất ít dùng vì mặc định khi khai báo biến cục bộ thì mặc định nó là auto rồi)
sizeof	Trả về kích thước (byte) vùng nhớ

Lưu ý: Các từ khóa viết thường, không được đặt định danh trùng với từ khóa.

PHỤ LỤC 2: TỔNG HỢP MỘT SỐ THƯ VIỆN C CHUẨN

STT	Thư viện chuẩn	Các hàm thường sử dụng
1	<assert.h>	Hàm assert() đánh giá biểu thức đầu vào
2	<ctype.h>	Hàm isalnum() kiểm tra kí tự là alphanumeric Hàm isalpha() kiểm tra kí tự là chữ Hàm iscntrl( ) kiểm tra kí tự là kí tự điều khiển Hàm isdigit() kiểm tra kí tự là kiểu số Hàm isgraph() kiểm tra kí tự in được Hàm islower() kiểm tra kí tự là chữ in thường Hàm isupper() kiểm tra kí tự là chữ in hoa Hàm isprint() kiểm tra kí tự in được Hàm ispunct() kiểm tra kí tự in được (trừ số và chữ) Hàm isspace() kiểm tra kí tự trắng Hàm isxdigit() kiểm tra kí tự là chữ số hexa Hàm tolower() convert in hoa thành in thường Hàm toupper() convert in thường thành in hoa
3	<math.h>	Hàm exp() tính e mũ x Hàm log() tính logarit Hàm pow() tính x mũ y Hàm sqrt() tính căn bậc 2 Hàm ceil() làm tròn số nguyên nhỏ nhất lớn hơn hoặc bằng X Hàm floor() làm tròn số nguyên lớn nhất & nhỏ hơn X Hàm fabs() tính trị tuyệt đối của kiểu double Hàm ldexp() trả về giá trị X nhân với 2 mũ Y Hàm frexp() convert số dấu phẩy động về dạng mũ Hàm modf() tách phần nguyên và phần thập phân Hàm fmod() tính số dư của phép chia x/y Hàm sin(), cos(), tan() Hàm acsin(), accos(), actan() Hàm atan2() tính arctan của x/y Hàm tính hypebol sinx(), cosx(), tanx()

STT	Thư viện chuẩn	Các hàm thường sử dụng
4	<stdio.h>	Hàm printf() xuất dữ liệu ra màn hình Hàm putchar() in kí tự ra màn hình Hàm puts() in một chuỗi kí tự ra màn hình Hàm scanf() đọc dữ liệu từ bàn phím Hàm sscanf() đọc dữ liệu theo định dạng Hàm getchar() đọc kí tự đầu vào Hàm gets() đọc string từ bàn phím Hàm fflush() làm sạch bộ đệm Hàm fopen() mở file Hàm fclose() để close file Hàm fprintf() ghi nội dung theo định dạng vào file text Hàm fscanf() đọc dữ liệu có định dạng từ file Hàm fputc() ghi kí tự xuống file Hàm fgetc() đọc một kí tự từ file Hàm fputs() ghi một chuỗi kí tự xuống file Hàm fgets() đọc chuỗi kí tự từ file Hàm fwrite() ghi nội dung xuống file binary Hàm fread() đọc dữ liệu từ file binary Hàm fseek() di chuyển con trỏ file tới vị trí bất kì Hàm ftell() trả về vị trí con trỏ file Hàm rewind() di chuyển con trỏ file về đầu file Hàm fsetpos() di chuyển con trỏ file về vị trí mong muốn Hàm fgetpos() lấy vị trí con trỏ file hiện tại Hàm feof() kiểm tra con trỏ file ở cuối file
5	<stdlib.h>	Hàm abs() tính giá trị tuyệt đối Hàm div() lấy thương và số dư của phép chia Hàm atoi() convert string thành số nguyên Hàm atof() convert string thành số thập phân Hàm strtod() tách số từ string

STT	Thư viện chuẩn	Các hàm thường sử dụng
		Hàm strtoul() tách số từ string dựa vào cơ số Hàm malloc() cấp phát bộ nhớ động Hàm realloc() thay đổi kích thước của vùng nhớ đã cấp phát Hàm calloc() cấp phát và khởi tạo cho vùng nhớ Hàm free() giải phóng vùng nhớ đã cấp phát Hàm abort() kết thúc chương trình đang chạy Hàm exit() kết thúc chương trình đang chạy Hàm rand() trả về số nguyên ngẫu nhiên
6	<string.h>	Hàm strcpy() copy chuỗi nguồn sang chuỗi đích Hàm strncpy() copy n byte từ chuỗi nguồn sang chuỗi đích Hàm strcat() nối hai chuỗi Hàm strncat() nối n byte chuỗi nguồn vào chuỗi đích Hàm strcmp() so sánh hai chuỗi Hàm strncmp() so sánh hai chuỗi con Hàm strchr() tìm kiếm kí tự trong chuỗi Hàm strrchr() tìm kiếm kí tự c trong chuỗi Hàm strspn() tìm kiếm bộ kí tự trong chuỗi Hàm strpbrk() trả về con trỏ tới vị trí xuất hiện đầu tiên của một trong các kí tự Hàm strstr() tìm kiếm chuỗi con Hàm strlen() tính độ dài của chuỗi Hàm strtok() tìm chuỗi con dựa vào bộ kí tự Hàm strxfrm() copy n kí tự chuỗi nguồn vào chuỗi đích Hàm memcpy() copy n byte vùng nhớ nguồn vào vùng nhớ đích Hàm memmove() copy n byte từ nguồn vào đích Hàm memcmp() so sánh n byte của 2 vùng nhớ

STT	Thư viện chuẩn	Các hàm thường sử dụng
		Hàm memchr() tìm kiếm kí tự c trong giới hạn n byte kí tự Hàm memset() gán giá trị c cho n byte đầu tiên của vùng nhớ
7	<time.h>	Hàm clock() tính thời gian thực hiện chương trình Hàm time() trả về số giây tính từ 01/01/1970 đến thời điểm gọi hàm time Hàm localtime() convert timer thành thời gian Hàm asctime() convert struct tm thành định dạng hh:mm:ss Hàm ctime() convert timer thành thời gian định dạng hh:mm:ss Hàm gmtime() convert timer thành thời gian UTC Hàm mktime() convert thời gian định dạng tm sang định dạng time_t Hàm difftime() trả về khoảng thời gian giữa 2 mốc thời gian Hàm strftime() trả về số kí tự được đặt trong vùng nhớ

PHỤ LỤC 3: BẢNG MÃ ASCII

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	

PHỤ LỤC 4: MỨC ĐỘ ƯU TIÊN CỦA CÁC TOÁN TỬ

Loại	Toán tử	Thứ tự ưu tiên
Postfix	() [] -> . ++ - -	Trái sang phải
Unary	+ - ! ~ ++ - - (type)* & sizeof	Phải sang trái
Tính nhân	* / %	Trái sang phải
Tính cộng	+ -	Trái sang phải
Dịch chuyển	<< >>	Trái sang phải
Quan hệ	< <= > >=	Trái sang phải
Cân bằng	== !=	Trái sang phải
Phép AND bit	&	Trái sang phải
Phép XOR bit	^	Trái sang phải
Phép OR bit	|	Trái sang phải
Phép AND logic	&&	Trái sang phải
Phép OR logic	||	Trái sang phải
Điều kiện	?:	Phải sang trái
Gán	= += -= *= /= %=>>= <<= &= ^= |=	Phải sang trái
Dấu phảy	,	Trái sang phải

PHỤ LỤC 5: MÃ ĐỊNH DẠNG

Stt	Mã	Diễn giải	Kiểu dữ liệu
1	%c	Ký tự	char unsigned char
2	%d	Số nguyên có dấu	short unsigned short int long
3	%e , %E	Số chấm động (ký hiệu có số mũ)	float double
4	%f	Số chấm động (ký hiệu thập phân)	float
5	%g , %G	Tương tự %e, %E	float double
6	%hi	Số nguyên ngắn có dấu	short
7	%hu	Số nguyên ngắn không âm	unsigned short
8	%i	Số nguyên có dấu	short unsigned short int long
9	%l, %ld, %li	Số nguyên có dấu (chỉ số nguyên dài)	long
10	%lf	Số chấm động (dài)	double
11	%Lf	Số chấm động (dài)	long double
12	%lu	Số nguyên thập phân không dấu (dài)	unsigned int unsigned long
13	%lli, %lld	Số nguyên (dài)	long long
14	%llu	Số nguyên thập phân không dấu (dài)	unsigned long long
15	%o	Số nguyên bát phân không dấu	short unsigned short int unsigned int long
16	%p	Con trỏ	void *
17	%s	Chuỗi ký tự	char *
18	%u	Số nguyên thập phân không dấu	unsigned int unsigned long
19	%x, %X	Số nguyên hexa không dấu	short unsigned short int unsigned int long
20	%n	Không in gì cả	
21	%%	In ký tự %	

TÀI LIỆU THAM KHẢO

[1] PGS.TS. Phạm Văn Ất, Giáo trình Kỹ thuật lập trình C cơ sở và nâng cao, NXB Giao thông vận tải, 2006.

[2] Nguyễn Linh Giang, Giáo trình Kỹ thuật lập trình C, NXB Giáo dục Việt Nam, 2010.

[3] Quách Tuấn Ngọc, Ngôn ngữ lập trình C, NXB Thống kê, 2002.

[4] Nguyễn Thanh Thủy, Nguyễn Quang Huy, Bài tập ngôn ngữ lập trình C, NXB Khoa học và Kỹ thuật, 2005.

[5] Th.S Trần Thị Hoa, Giáo trình lập trình căn bản, Lưu hành nội bộ, Học viện Kỹ thuật mật mã, 2013.

[6] Vũ Bá Duy, Giáo trình tin học cơ sở, phần lập trình trên ngôn ngữ C, Lưu hành nội bộ, Đại học Quốc gia Hà Nội, 2003.

[7] Brian W. Kernighan, Dennis M. Ritchie, The C programming language, Second Edition, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[8] https://www.programiz.com/c-programming (22.07.2019).

[9] https://www.w3resource.com/c-programming-exercises (22.07.2019).