MỤC LỤC

Bài 1 :

NGÔN NGỮ LẬP TRÌNH & PHƯƠNG PHÁP LẬP TRÌNH

1.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:

- Ý nghĩa, các bước lập trình.
- Xác định dữ liệu vào, ra.
- Phân tích các bài toán đơn giản.
- Khái niệm so sánh, lặp.
- Thể hiện bài toán bằng lưu đồ.

1.2 Lý thuyết

1.2.1 Ngôn ngữ lập trình (Programming Language)

Phần này chúng ta sẽ tìm hiểu một số khái niệm căn bản về thuật toán, chương trình, ngôn ngữ lập trình. Thuật ngữ "thuật giải" và "thuật toán" dĩ nhiên có sự khác nhau song trong nhiều trường hợp chúng có cùng nghĩa.

1.2.1.1 Thuật giải (Algorithm)

Là một dãy các thao tác xác định trên một đối tượng, sao cho sau khi thực hiện một số hữu hạn các bước thì đạt được mục tiêu. Theo R.A.Kowalski thì bản chất của thuật giải:

Thuật giải = Logic + Điều khiển

*** *Logic***: Đây là phần khá quan trọng, nó trả lời câu hỏi "Thuật giải làm gì, giải quyết vấn đề gì?", những yếu tố trong bài toán có quan hệ với nhau như thế nào v.v… Ở đây bao gồm những kiến thức chuyên môn mà bạn phải biết để có thể tiến hành giải bài toán.

Ví dụ 1: Để giải một bài toán tính diện tích hình cầu, mà bạn không còn nhớ công thức tính hình cầu thì bạn không thể viết chương trình cho máy để giải bài toán này được.

* ***Điều khiển***: Thành phần này trả lời câu hỏi: giải thuật phải làm như thế nào?. Chính là cách thức tiến hành áp dụng thành phần logic để giải quyết vấn đề.

1.2.1.2 Chương trình (Program)

Là một tập hợp các mô tả, các phát biểu, nằm trong một hệ thống qui ước về ý nghĩa và thứ tự thực hiện, nhằm điều khiển máy tính làm việc. Theo Niklaus Wirth thì:

Chương trình = Thuật toán + Cấu trúc dữ liệu

Các thuật toán và chương trình đều có cấu trúc dựa trên ***3 cấu trúc điều khiển cơ bản***:

* ***Tuần tự*** (Sequential): Các bước thực hiện tuần tự một cách chính xác từ trên xuống, mỗi bước chỉ thực hiện đúng một lần.

* ***Chọn lọc*** (Selection): Chọn 1 trong 2 hay nhiều thao tác để thực hiện.

* ***Lặp lại*** (Repetition): Một hay nhiều bước được thực hiện lặp lại một số lần.

Muốn trở thành lập trình viên chuyên nghiệp bạn hãy làm đúng trình tự để có thói quen tốt và thuận lợi sau này trên nhiều mặt của một người làm máy tính. Bạn hãy làm theo các bước sau:

Tìm, xây dựng thuật giải (trên giấy) → viết chương trình trên máy
→ dịch chương trình → chạy và thử chương trình

1.2.1.3 Ngôn ngữ lập trình (Programming language)

Ngôn ngữ lập trình là hệ thống các ký hiệu tuân theo các qui ước về ngữ pháp và ngữ nghĩa, dùng để xây dựng thành các chương trình cho máy tính.

Một chương trình được viết bằng một ngôn ngữ lập trình cụ thể (ví dụ Pascal, C…) gọi là chương trình nguồn, chương trình dịch làm nhiệm vụ dịch chương trình nguồn thành chương trình thực thi được trên máy tính.

1.2.2 Các bước lập trình

Bước 1: Phân tích vấn đề và xác định các đặc điểm. (xác định I-P-O)
Bước 2: Lập ra giải pháp. (đưa ra thuật giải)
Bước 3: Cài đặt. (viết chương trình)
Bước 4: Chạy thử chương trình. (dịch chương trình)
Bước 5: Kiểm chứng và hoàn thiện chương trình. (thử nghiệm bằng nhiều số liệu và đánh giá)

1.2.3 Kỹ thuật lập trình

1.2.3.1 I-P-O Cycle (Input-Pprocess-Output Cycle) (Quy trình nhập-xử lý-xuất)

Quy trình xử lý cơ bản của máy tính gồm I-P-O.

Ví dụ 2: Xác định Input, Process, Output của việc làm 1 ly nước chanh nóng
Input : ly, đường, chanh, nước nóng, muỗng.
Process : - cho hỗn hợp đường, chanh, nước nóng vào ly.
- dùng muỗng khuấy đều.
Output : ly chanh nóng đã sẵn sàng để dùng.

Ví dụ 3: Xác định Input, Process, Output của chương trình tính tiền lương công nhân tháng 10/2002 biết rằng lương = lương căn bản * ngày công
Input : lương căn bản, ngày công
Process : nhân lương căn bản với ngày công
Output : lương

Ví dụ 4: Xác định Input, Process, Output của chương trình giải phương trình bậc nhất $ax + b = 0$
Input : hệ số a, b
Process : chia – b cho a
Output : nghiệm x

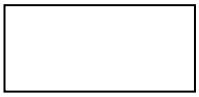
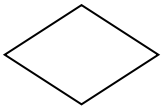
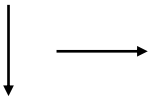
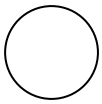
Ví dụ 5: Xác định Input, Process, Output của chương trình tìm số lớn nhất của 2 số a và b.
Input : a, b
Process : Nếu a > b thì *Output* = a lớn nhất
Ngược lại *Output* = b lớn nhất

Bài tập

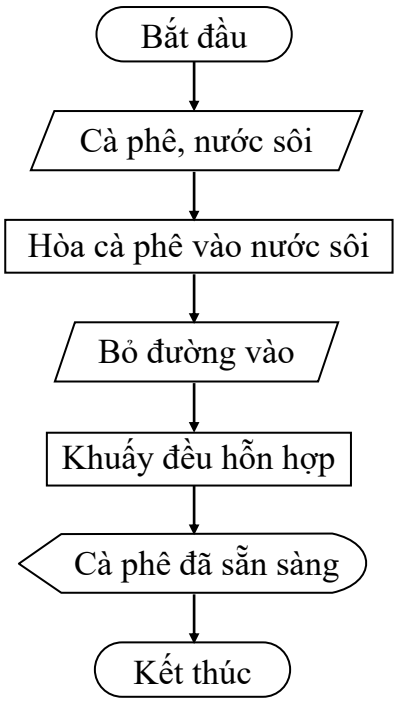
Xác định Input, Process, Output của các chương trình sau:
1. Đổi từ tiền VND sang tiền USD.
2. Tính điểm trung bình của học sinh gồm các môn Toán, Lý, Hóa.
3. Giải phương trình bậc 2: $ax^2 + bx + c = 0$
4. Đổi từ độ sang radian và đổi từ radian sang độ
(công thức $\alpha/\pi = a/180$, với α: radian, a: độ)
5. Kiểm tra 2 số a, b giống nhau hay khác nhau.

1.2.3.2 Sử dụng lưu đồ (Flowchart)

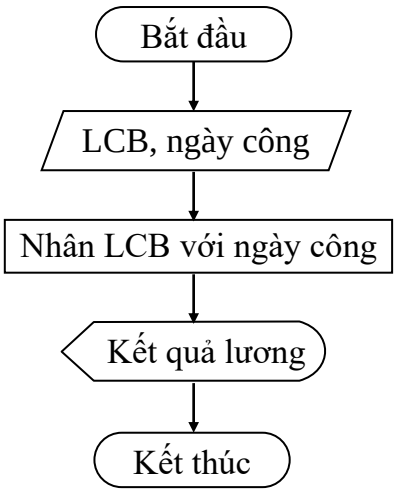
Để dễ hơn về quy trình xử lý, các nhà lập trình đưa ra dạng lưu đồ để minh họa từng bước quá trình xử lý một vấn đề (bài toán).

Hình dạng (symbol)	Hành động (Activity)
	Dữ liệu vào (Input)
	Xử lý (Process)
	Dữ liệu ra (Output)
	Quyết định (Decision), sử dụng điều kiện
	Luồng xử lý (Flow lines)
	Gọi CT con, hàm… (Procedure, Function…)
	Bắt đầu, kết thúc (Begin, End)
	Điểm ghép nối (Connector)

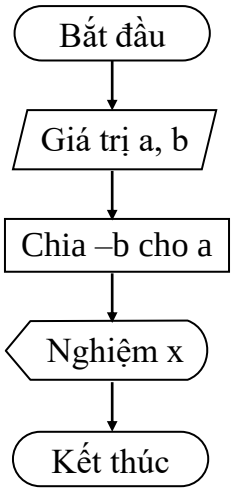
Ví dụ 6: Chuẩn bị cà phê



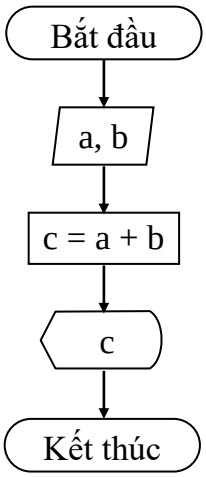
Ví dụ 7: Mô tả ví dụ 3



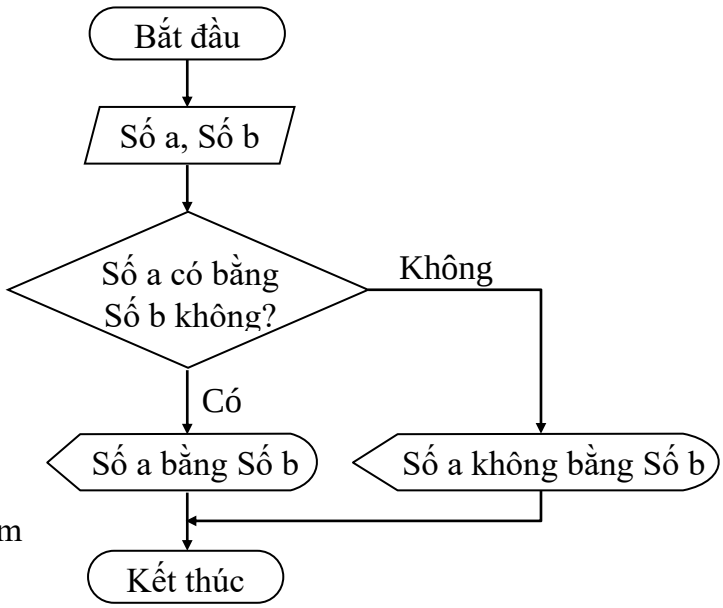
Ví dụ 8: Mô tả ví dụ 4



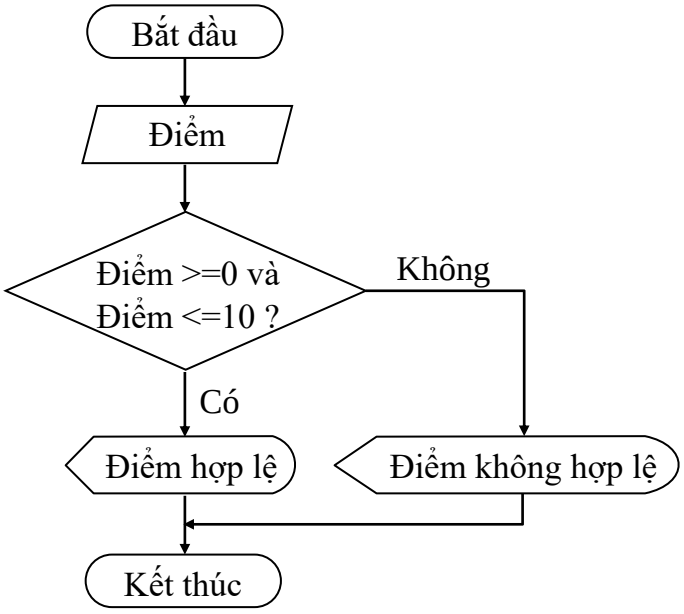
Ví dụ 9: Cộng 2 số



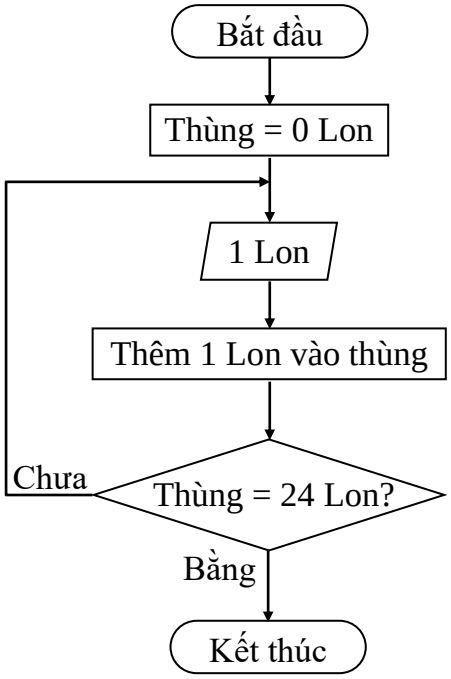
Ví dụ 10: so sánh 2 số



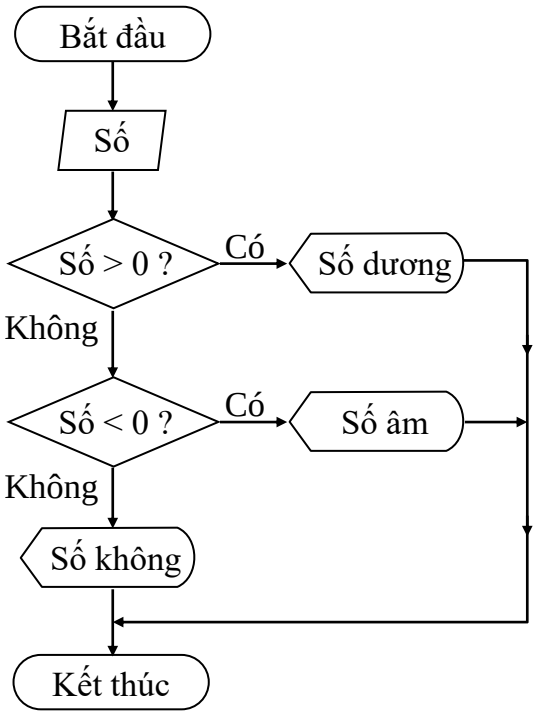
Ví dụ 11: Kiểm tra tính hợp lệ của điểm



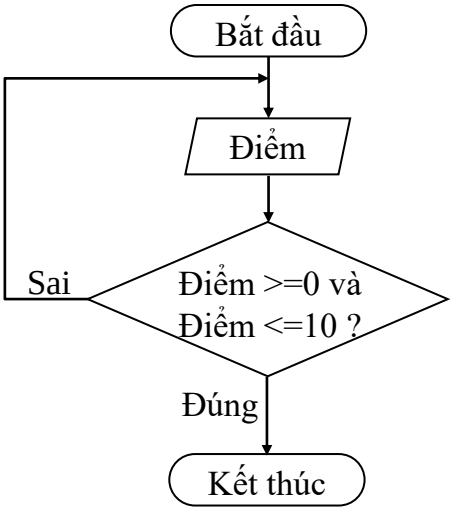
Ví dụ 12: Xếp lon vào thùng



Ví dụ 13: Kiểm tra loại số



Ví dụ 14: Kiểm tra tính hợp lệ của điểm



Bài tập

Vẽ lưu đồ cho các chương trình sau:

1. Đổi từ tiền VND sang tiền USD.
2. Tính điểm trung bình của học sinh gồm các môn Toán, Lý, Hóa.
3. Giải phương trình bậc 2: $ax^2 + bx + c = 0$
4. Đổi từ độ sang radian và đổi từ radian sang độ
   (công thức $\alpha/\pi = a/180$, với α: radian, a: độ)
5. Kiểm tra 2 số a, b giống nhau hay khác nhau.

Bài 2 :

LÀM QUEN LẬP TRÌNH C QUA CÁC VÍ DỤ ĐƠN GIẢN

2.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:

- Ngôn ngữ C.
- Một số thao tác cơ bản của trình soạn thảo C.
- Cách lập trình trên C.
- Tiếp cận một số lệnh đơn giản thông qua các ví dụ.
- Nắm bắt được một số kỹ năng đơn giản.

2.2 Nội dung

2.2.1 Khởi động và thoát BorlandC

2.2.1.1 Khởi động

- Nhập lệnh tại dấu nhắc DOS: **gõ BC ↵ (Enter)** (nếu đường dẫn đã được cài đặt bằng lệnh **path** trong đó có chứa đường dẫn đến thư mục chứa tập tin BC.EXE). Nếu đường dẫn chưa được cài đặt ta tìm xem thư mục BORLANDC nằm ở ổ đĩa nào. Sau đó ta gõ lệnh sau:

<ổ đĩa>:\BORLANDC\BIN\BC ↵ (Enter)

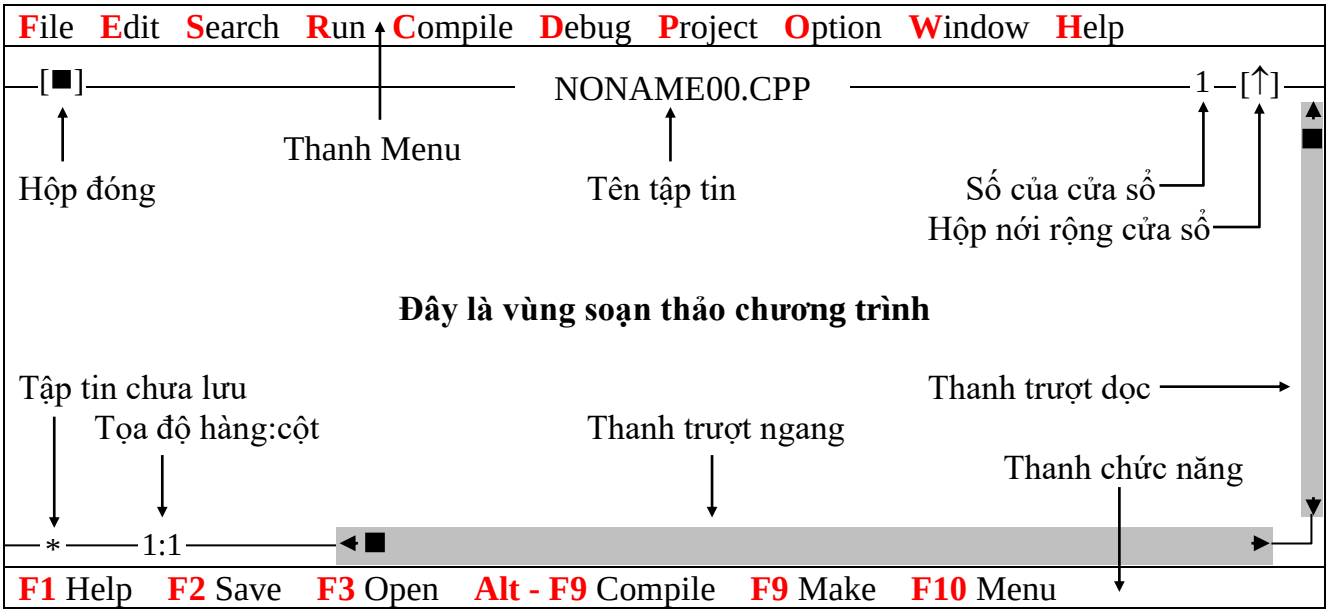
Nếu bạn muốn vừa khởi động BC vừa soạn thảo chương trình với một tập tin có tên do chúng ta đặt, thì gõ lệnh: **BC [đường dẫn]<tên file cần soạn thảo>**, nếu tên file cần soạn thảo đã có thì được nạp lên, nếu chưa có sẽ được tạo mới.

- Khởi động tại Windows: Bạn vào menu Start, chọn Run, bạn gõ vào hộp Open 1 trong các dòng lệnh như nhập tại DOS. Hoặc bạn vào Window Explorer, chọn ổ đĩa chứa thư mục BORLANDC, vào thư mục BORLANDC, vào thư mục BIN, khởi động tập tin BC.EXE.

Ví dụ: Bạn gõ D:\BORLANDC\BIN\BC E:\BAITAP_BC\VIDU1.CPP

Câu lệnh trên có nghĩa khởi động BC và nạp tập tin VIDU1.CPP chứa trong thư mục BAITAP_BC trong ổ đĩa E. Nếu tập tin này không có sẽ được tạo mới.

Màn hình sau khi khởi động thành công



2.2.1.2 Thoát

- Ấn phím **F10** (kích hoạt Menu), chọn menu **File**, chọn **Quit**;
- Hoặc ấn tổ hợp phím **Alt – X**.

2.2.2 Các ví dụ đơn giản

2.2.2.1 Ví dụ 1

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Chuong trinh in ra cau bai hoc C dau tien */
2	#include <stdio.h>
3	
4	void main(void)
5	{
6	printf("Bai hoc C dau tien.");
7	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

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

- Dòng thứ 1: bắt đầu bằng **/*** và kết thúc bằng ***/** cho biết hàng này là hàng diễn giải (chú thích). Khi dịch và chạy chương trình, dòng này không được dịch và cũng không thi hành lệnh gì cả. Mục đích của việc ghi chú này giúp chương trình rõ ràng hơn. Sau này bạn đọc lại chương trình biết chương trình làm gì.
- Dòng thứ 2: chứa phát biểu tiền xử lý ***#include <stdio.h>***. Vì trong chương trình này ta sử dụng hàm thư viện của C là ***printf***, do đó bạn cần phải có khai báo của hàm thư viện này để báo cho trình biên dịch C biết. **Nếu không khai báo chương trình sẽ báo lỗi**.
- Dòng thứ 3: hàng trắng viết ra với ý đồ làm cho bảng chương trình thoáng, dễ đọc.
- Dòng thứ 4: ***void main(void)*** là thành phần chính của mọi chương trình C (bạn có thể viết main() hoặc void main() hoặc main(void)). Tuy nhiên, bạn nên viết theo dạng void main(void) để chương trình rõ ràng hơn. Mọi chương trình C đều bắt đầu thi hành từ hàm main. Cặp dấu ngoặc () cho biết đây là khối hàm (function). Hàm void main(void) có từ khóa void đầu tiên cho biết hàm này không trả về giá trị, từ khóa void trong ngoặc đơn cho biết hàm này không nhận vào đối số.
- Dòng thứ 5 và 7: cặp dấu ngoặc móc {} giới hạn thân của hàm. Thân hàm bắt đầu bằng dấu { và kết thúc bằng dấu }.
- Dòng thứ 6: ***printf("Bai hoc C dau tien.");***, chỉ thị cho máy in ra chuỗi ký tự nằm trong nháy kép (""). Hàng này được gọi là một câu lệnh, kết thúc một câu lệnh trong C phải là dấu chấm phẩy (;).

☹ **Chú ý**:

- ✓ Các từ include, stdio.h, void, main, printf phải viết bằng chữ thường.
- ✓ Chuỗi trong nháy kép cần in ra "Bạn có thể viết chữ HOA, thường tùy, ý".
- ✓ Kết thúc câu lệnh phải có dấu chấm phẩy.
- ✓ Kết thúc tên hàm không có dấu chấm phẩy hoặc bất cứ dấu gì.
- ✓ Ghi chú phải đặt trong cặp /* …. */.
- ✓ Thân hàm phải được bao bởi cặp { }.
- ✓ Các câu lệnh trong thân hàm phải viết thụt vào.

Bạn nhập đoạn chương trình trên vào máy. Dịch, chạy và quan sát kết quả.

Ctrl – F9: Dịch và chạy chương trình. Alt – F5: Xem màn hình kết quả.

Sau khi bạn nhập xong đoạn chương trình vào máy. Bạn Ấn và giữ phím Ctrl, gõ F9 để dịch và chạy chương trình. Khi đó bạn thấy chương trình chớp rất nhanh và không thấy kết quả gì cả. Bạn Ấn và giữ phím Alt, gõ F5 để xem kết quả, khi xem xong, bạn ấn phím bất kỳ để quay về màn hình soạn thảo chương trình.

Bây giờ bạn sửa lại dòng thứ 6 bằng câu lệnh printf("Bai hoc C dau tien.\n");, sau đó dịch và chạy lại chương trình, quan sát kết quả.

☞ ***Kết quả in ra màn hình***

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

Ở dòng bạn vừa sửa có thêm \n, \n là ký hiệu xuống dòng sử dụng trong lệnh printf. Sau đây là một số ký hiệu khác.

+ ***Các kí tự điều khiển***:

\n : Nhảy xuống dòng kế tiếp canh về cột đầu tiên.
\t : Canh cột tab ngang.
\r : Nhảy về đầu hàng, không xuống hàng.
\a : Tiếng kêu bip.

+ ***Các kí tự đặc biệt***:

\\ : In ra dấu \
\" : In ra dấu "
\' : In ra dấu '

Bây giờ bạn sửa lại dòng thứ 6 bằng câu lệnh printf("\tBai hoc C dau tien.\a\n");, sau đó dịch và chạy lại chương trình, quan sát kết quả.

☞ ***Kết quả in ra màn hình***

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

Khi chạy chương trình bạn nghe tiếng bip phát ra từ loa.

Mỗi khi chạy chương trình bạn thấy rất bất tiện trong việc xem kết quả phải ấn tổ hợp phím Alt – F5. Để khắc phục tình trạng này bạn sửa lại chương trình như sau:

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Chuong trinh in ra cau bai hoc C dau tien */
2	#include <stdio.h>
3	#include <conio.h>
4	
5	void main(void)
6	{
7	printf("\t\tBai hoc C \rdau tien.\n");
8	getch();
9	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

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

- Dòng thứ 3: chứa phát biểu tiền xử lý ***#include <conio.h>***. Vì trong chương trình này ta sử dụng hàm thư viện của C là ***getch***, do đó bạn cần phải có khai báo của hàm thư viện này để báo cho trình biên dịch C biết. **Nếu không khai báo chương trình sẽ báo lỗi**.
- Dòng thứ 8: ***getch();***, chờ nhận 1 ký tự bất kỳ từ bàn phím, nhưng không in ra màn hình. Vì thế ta sử dụng hàm này để khi chạy chương trình xong sẽ dừng lại ở màn hình kết quả, sau đó ta ấn phím bất kỳ sẽ quay lại màn hình soạn thảo.

Bạn nhập đoạn chương trình trên vào máy. Dịch, chạy và quan sát kết quả.

2.2.2.2 Ví dụ 2

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Chuong trinh nhap va in ra man hinh gia tri bien*/
2	#include <stdio.h>
3	#include <conio.h>
4	
5	void main(void)
6	{
7	int i;
8	printf("Nhap vao mot so: ");
9	scanf("%d", &i);
10	printf("So ban vua nhap la: %d.\n", i);
11	getch();
12	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao mot so: 15
So ban vua nhap la: 15.
_
```

- Dòng thứ 7: ***int i;*** là lệnh khai báo, mẫu tự i gọi là tên biến. Biến là một vị trí trong bộ nhớ dùng lưu trữ giá trị nào đó mà chương trình sẽ lấy để sử dụng. Mỗi biến phải thuộc một kiểu dữ liệu. Trong trường hợp này ta sử dụng biến i kiểu số nguyên (integer) viết tắt là int.
- Dòng thứ 9: ***scanf("%d", &i)***. Sử dụng hàm scanf để nhận từ người sử dụng một trị nào đó. Hàm scanf trên có 2 đối mục. Đối mục ***"%d"*** được gọi là chuỗi định dạng, cho biết loại dữ kiện mà người sử dụng sẽ nhập vào. Chẳng hạn, ở đây phải nhập vào là số nguyên. Đối mục thứ 2 ***&i*** có dấu & đi đầu gọi là address operator, dấu & phối hợp với tên biến cho hàm scanf biến đem trị gõ từ bàn phím lưu vào biến i.
- Dòng thứ 10: ***printf("So ban vua nhap la: %d.\n", i);***. Hàm này có 2 đối mục. Đối mục thứ nhất là một chuỗi định dạng có chứa chuỗi văn bản ***So ban vua nhap la:*** và ***%d*** (ký hiệu khai báo chuyển đổi dạng thức) cho biết số nguyên sẽ được in ra. Đối mục thứ 2 là i cho biết giá trị lấy từ biến i để in ra màn hình.

Bạn nhập đoạn chương trình trên vào máy. Dịch, chạy và quan sát kết quả.

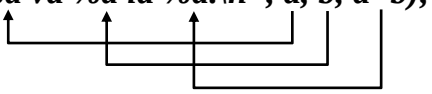
2.2.2.3 Ví dụ 3

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Chuong trinh nhap vao 2 so a, b in ra tong*/
2	#include <stdio.h>
3	#include <conio.h>
4	
5	void main(void)
6	{
7	int a, b;
8	printf("Nhap vao so a: ");
9	scanf("%d", &a);
10	printf("Nhap vao so b: ");
11	scanf("%d", &b);
12	printf("Tong cua 2 so %d va %d la %d.\n", a, b, a+b);
13	getch();
14	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao so a: 4
Nhap vao so b: 14
Tong cua 2 so 4 va 14 la 18.
_
```

- Dòng thứ 12: ***printf("Tong cua 2 so %d va %d la %d.\n", a, b, a+b);***



Bạn nhập đoạn chương trình trên vào máy. Dịch, chạy và quan sát kết quả.

2.2.2.4 Ví dụ 4

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Chuong trinh nhap vao ban kinh hinh tron. Tinh dien tich */
2	#include <stdio.h>
3	#include <conio.h>
4	
5	#define PI 3.14
6	
7	void main(void)
8	{
9	float fR;
10	printf("Nhap vao ban kinh hinh tron: ");
11	scanf("%f", &fR);
12	printf("Dien tich hinh tron: %.2f.\n", 2*PI*fR);
13	getch();
14	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao ban kinh hinh tron: 1
Dien tich hinh tron: 6.28
_
```

- Dòng thứ 5: ***#define PI 3.14***, dùng chỉ thị define để định nghĩa hằng số PI có giá trị 3.14. *Trước define phải có dấu # và cuối dòng không có dấu chấm phẩy*.
- Dòng thứ 12: ***printf("Dien tich hinh tron: %.2f.\n", 2*PI*fR);***. Hàm này có 2 đối mục. Đối mục thứ nhất là một chuỗi định dạng có chứa chuỗi văn bản ***Dien tich hinh tron:*** và ***%.2f*** (ký hiệu khai báo chuyển đổi dạng thức) cho biết dạng số chấm động sẽ được in ra, trong đó ***.2*** nghĩa là in ra với 2 số lẻ. Đối mục thứ 2 là biểu thức hằng ***2*PI*fR***;

Bạn nhập đoạn chương trình trên vào máy. Dịch, chạy và quan sát kết quả.

Bài 3 :
CÁC THÀNH PHẦN TRONG NGÔN NGỮ C

3.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:
- Khái niệm từ khóa
- Các kiểu dữ liệu
- Cách ghi chú
- Đặt tên biến
- Khai báo biến.
- Phạm vi sử dụng biến.

3.2 Nội dung

3.2.1 Từ khóa

Từ khóa là từ có ý nghĩa xác định dùng để khai báo dữ liệu, viết câu lệnh… Trong C có các từ khóa sau:

asm	const	else	for	interrupt	return	sizeof	void
break	continue	enum	goto	long	short	switch	volatile
case	default	extern	huge	near	static	typedef	while
cdecl	do	far	if	pascal	struct	union	
char	double	float	int	register	signed	unsigned	

☞ Các từ khóa phải viết bằng ***chữ thường***

3.2.2 Tên

Khái niệm tên 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. Tên thường được đặt cho hằng, biến, mảng, con trỏ, nhãn… Chiều dài tối đa của tên là 32 ký tự.

Tên biến hợp lệ là một chuỗi ký tự liên tục gồm: ***Ký tự chữ, số và dấu gạch dưới***. Ký tự đầu của tên phải là ***chữ hoặc dấu gạch dưới***. Khi đặt tên không được đặt trùng với các từ khóa.

Ví dụ 1 :

- Các tên đúng: delta, a_1, Num_ODD, Case
- Các tên sai:

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

Lưu ý: Trong C, tên phân biệt chữ hoa, chữ thường

Ví dụ 2 : number khác Number
case khác Case
(case là từ khóa, do đó bạn đặt tên là Case vẫn đúng)

3.2.3 Kiểu dữ liệu

Có 4 kiểu dữ liệu cơ bản trong C là: char, int, float, double.

TT	Kiểu dữ liệu (Type)	Kích thước (Length)	Miền giá trị (Range)
1	unsigned char	1 byte	0 đến 255
2	char	1 byte	– 128 đến 127
3	enum	2 bytes	– 32,768 đến 32,767
4	unsigned int	2 bytes	0 đến 65,535
5	short int	2 bytes	– 32,768 đến 32,767
6	int	2 bytes	– 32,768 đến 32,767
7	unsigned long	4 bytes	0 đến 4,294,967,295
8	long	4 bytes	– 2,147,483,648 đến 2,147,483,647
9	float	4 bytes	$3.4 * 10^{-38}$ đến $3.4 * 10^{38}$
10	double	8 bytes	$1.7 * 10^{-308}$ đến $1.7 * 10^{308}$
11	long double	10 bytes	$3.4 * 10^{-4932}$ đến $1.1 * 10^{4932}$

3.2.4 Ghi chú

Trong khi lập trình cần phải ghi chú để giải thích các biến, hằng, 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 ghi chú sau: // hoặc /* nội dung ghi chú */

Ví dụ 3 :

```
void main()
{
   int a, b;          //khai bao bien t kieu int
   a = 1;             //gan 1 cho a
   b =3;              //gan 3 cho b
   /* thuat toan tim so lon nhat la
      neu a lon hon b thi a lon nhat
      nguoc lai b lon nhat */
   if (a > b) printf("max: %d", a);
   else printf("max: %d", b);
}
```

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

Tóm lại, đối với ghi chú dạng // dùng để ghi chú một hàng và dạng /* …. */ có thể ghi chú một hàng hoặc nhiều hàng.

3.2.5 Khai báo biến

3.2.5.1 Tên biến

Cách đặt tên biến như mục 2.

3.2.5.2 Khai báo biến

- Cú pháp

Kiểu dữ liệu *Danh sách tên biến*;

☞ Kiểu dữ liệu: 1 trong các kiểu ở mục 3

Danh sách tên biến: gồm các tên biến có cùng kiểu dữ liệu, mỗi tên biến cách nhau dấu phẩy (,), cuối cùng là dấu chấm phẩy (;).

☞ Khi khai báo biến nên đặt tên biến theo **quy tắc Hungarian Notation**

Ví dụ 4 :

```
int ituoi;              //khai báo biến ituoi có kiểu int
float fTrongluong;      //khai báo biến fTrongluong có kiểu long
char ckitu1, ckitu2;    //khai báo biến ckitu1, ckitu2 có kiểu char
```

Các biến khai báo trên theo quy tắc Hungarian Notation. Nghĩa là biến ituoi là kiểu int, bạn thêm chữ **i** (kí tự đầu của kiểu) vào đầu tên biến **tuoi** để trong quá trình lập trình hoặc sau này xem lại, sửa chữa… bạn dễ dàng nhận ra biến **ituoi có kiểu int** mà không cần phải di chuyển đến phần khai báo mới biết kiể.u của biến này. Tương tự cho biến **fTrongluong**, bạn nhìn vào là biết ngay biến này có kiểu **float**.

3.2.5.3 Vừa khai báo vừa khởi gán

Có thể kết hợp việc khai báo với toán tử gán để biến nhận ngay giá trị cùng lúc với khai báo.
Ví dụ 5 :

- ***Khai báo trước, gán giá trị sau***:

```
void main()
{
   int a, b, c;
   a = 1;
   b = 2;
   c = 5;
   …
}
```

- ***Vừa khai báo vừa gán giá trị***:

```
void main()
{
   int a = 1, b = 2, c = 5;
   …
}
```

3.2.5.4 Phạm vi của biến

Khi lập trình, bạn 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 bạn 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... Các biến này có ảnh hưởng đến toàn bộ chương trình. Chu trình sống của nó là bắt đầu chạy chương trình đến lúc kết thúc chương trình.

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

Bài 4 :

NHẬP / XUẤT DỮ LIỆU

4.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:
- Ý nghĩa, cách sử dụng hàm printf, scanf
- Sử dụng khuôn dạng, ký tự đặc biệt, ký tự điều khiển trong printf, scanf.

4.2 Nội dung

4.2.1 Hàm printf

Kết xuất dữ liệu được định dạng.

- ***Cú pháp***

printf ("chuỗi định dạng"[, đối mục 1, đối mục 2,…]);

☞ *Khi sử dụng hàm phải khai báo tiền xử lý* ***#include <stdio.h>***

- printf: tên hàm, ***phải viết bằng chữ thường.***
- đối mục 1,…: là các mục dữ kiện cần in ra màn hình. Các đối mục này có thể là biến, hằng hoặc biểu thức phải được định trị trước khi in ra.
- chuỗi định dạng: được đặt trong cặp nháy kép (" "), gồm 3 loại:

+ Đối với chuỗi kí tự ghi như thế nào in ra giống như vậy.

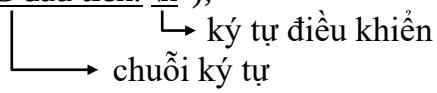
+ Đối với những kí tự chuyển đổi dạng thức cho phép kết xuất giá trị của các đối mục ra màn hình tạm gọi là mã định dạng. Sau đây là các dấu mô tả định dạng:

%c : Ký tự đơn
%s : Chuỗi
%d : Số nguyên thập phân có dấu
%f : Số chấm động (ký hiệu thập phân)
%e : Số chấm động (ký hiệu có số mũ)
%g : Số chấm động (%f hay %g)
%x : Số nguyên thập phân không dấu
%u : Số nguyên hex không dấu
%o : Số nguyên bát phân không dấu
l : Tiền tố dùng kèm với %d, %u, %x, %o để chỉ số nguyên dài (ví dụ %ld)

+ Các ký tự điều khiển và ký tự đặc biệt

\n : Nhảy xuống dòng kế tiếp canh về cột đầu tiên.
\t : Canh cột tab ngang.
\r : Nhảy về đầu hàng, không xuống hàng.
\a : Tiếng kêu bip.
\\ : In ra dấu \
\" : In ra dấu "
\' : In ra dấu '
%%: In ra dấu %

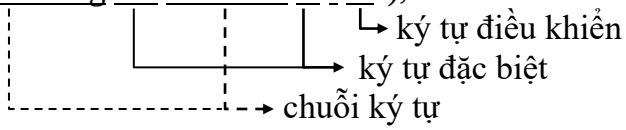
Ví dụ 1: printf("Bai hoc C dau tien. \n");



☞ ***Kết quả in ra màn hình***

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

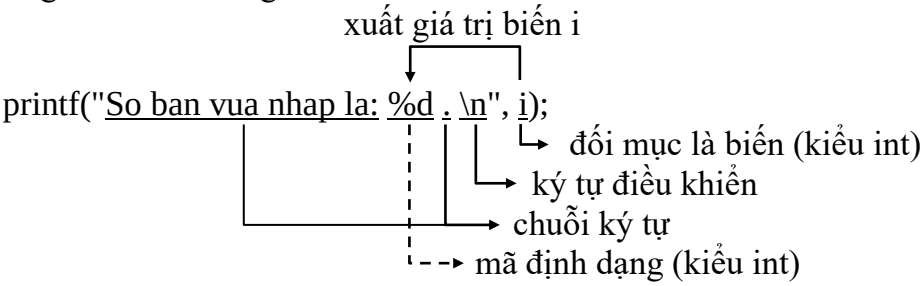
Ví dụ 2: printf("Ma dinh dang \\\" in ra dau \" . \n");



☞ ***Kết quả in ra màn hình***

```
Ma dinh dang \" in ra dau ".
_
```

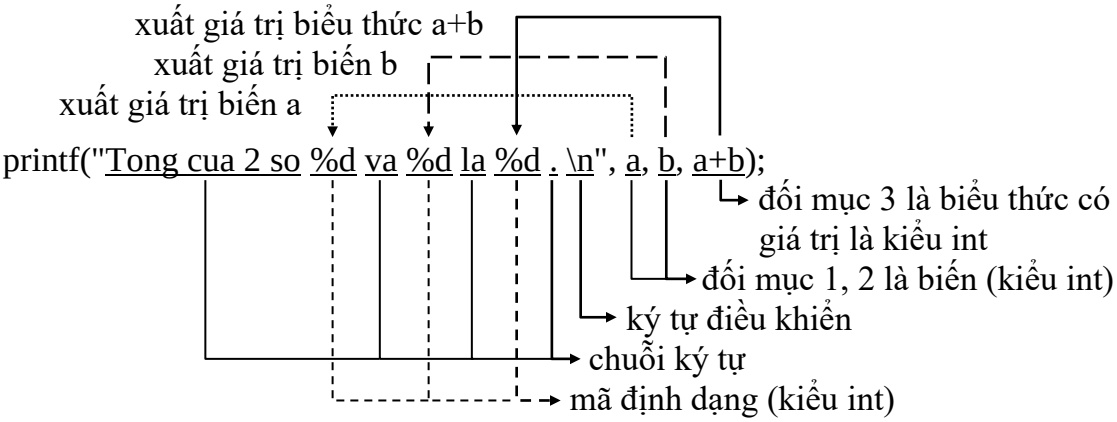
Ví dụ 3: giả sử biến i có giá trị = 5



☞ ***Kết quả in ra màn hình***

```
So ban vua nhap la: 5.
_
```

Ví dụ 4: giả sử biến a có giá trị = 7 và b có giá trị = 4

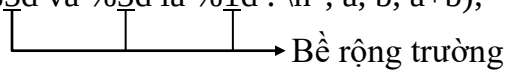


☞ ***Kết quả in ra màn hình***

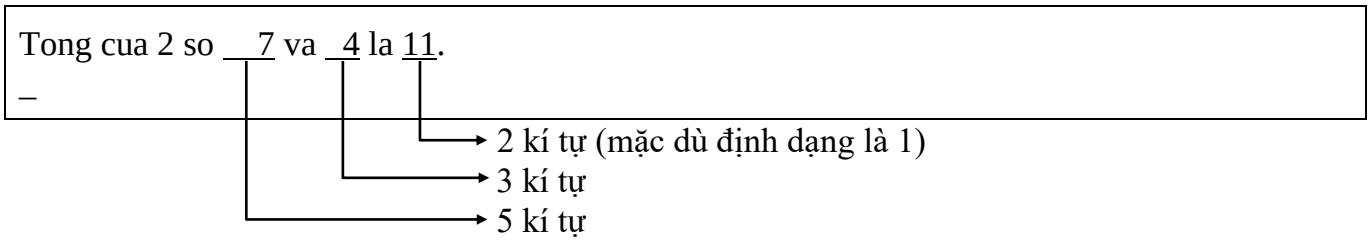
```
Tong cua 2 so 7 va 4 la 11.
_
```

Ví dụ 5: sửa lại ví dụ 4

printf("Tong cua 2 so %5d va %3d la %1d . \n", a, b, a+b);



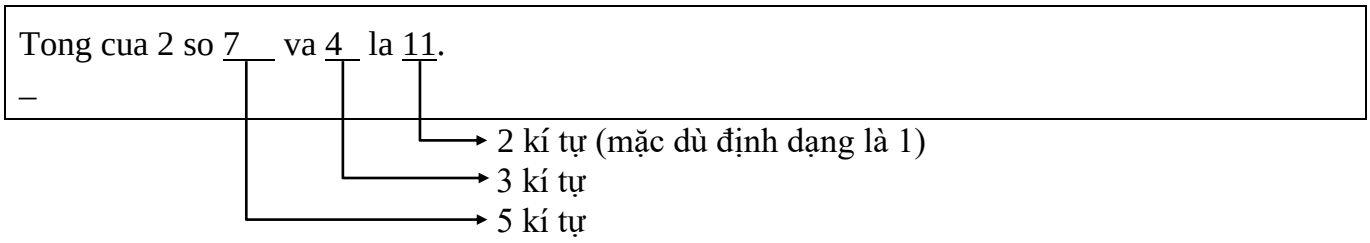
☞ ***Kết quả in ra màn hình***



Ví dụ 6: sửa lại ví dụ 5

```
printf("Tong cua 2 so %-5d va %-3d la %-1d . \n", a, b, a+b);
```

☞ ***Kết quả in ra màn hình***

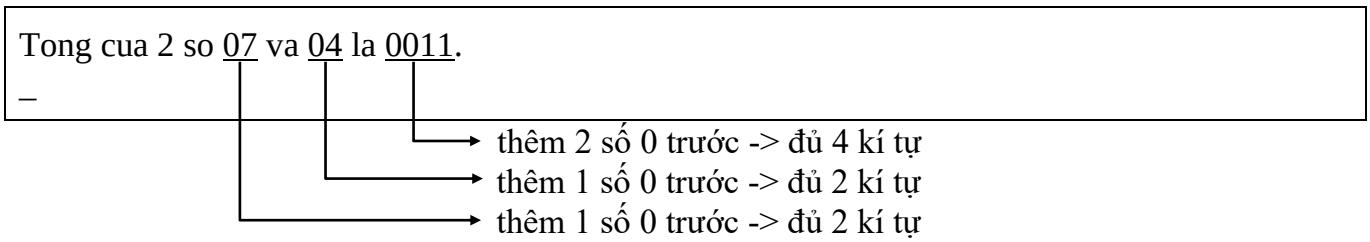


☞ **Dấu trừ trước bề rộng trường sẽ kéo kết quả sang trái**

Ví dụ 7: sửa lại ví dụ 4

```
printf("Tong cua 2 so %02d va %02d la %04d . \n", a, b, a+b);
```

☞ ***Kết quả in ra màn hình***



Ví dụ 8: giả sử int a = 6, b = 1234, c = 62

```
printf("%7d%7d%7d.\n", a, b, c);
printf("%7d%7d%7d.\n", 165, 2, 965);
```

☞ ***Kết quả in ra màn hình***

6 1234 62 165 2 965 –	Số canh về bên phải bề rộng trường.

```
printf("%-7d%-7d%-7d.\n", a, b, c);
printf("%-7d%-7d%-7d.\n", 165, 2, 965);
```

☞ ***Kết quả in ra màn hình***

6 1234 62 165 2 965 –	Số canh về bên trái bề rộng trường.

Ví dụ 9: giả sử float a = 6.4, b = 1234.56, c = 62.3

```
printf("%7.2d%7.2d%7.2d.\n", a, b, c);
                              số số lẻ
```

☞ ***Kết quả in ra màn hình***

```
  6.40  1234.56  62.30
_
```

Số canh về bên phải bề rộng trường.

7 kí tự

☞ **Bề rộng trường bao gồm: phần nguyên, phần lẻ và dấu chấm động**

Ví dụ 10: giả sử float a = 6.4, b = 1234.55, c = 62.34

```
printf("%10.1d%10.1d%10.1d.\n", a, b, c);
printf("%10.1d%10.1d%10.1d.\n", 165, 2, 965);
```

☞ ***Kết quả in ra màn hình***

```
     6.4    1234.6      62.3
   165.0       2.0     965.0
_
```

Số canh về bên phải bề rộng trường.

```
printf("%-10.2d%-10.2d%-10.2d.\n", a, b, c);
printf("%-10.2d%-10.2d%-10.2d.\n", 165, 2, 965);
```

☞ ***Kết quả in ra màn hình***

```
6.40      1234.55   62.34
165.00    2.00      965.00
_
```

Số canh về bên trái bề rộng trường.

4.2.2 Hàm scanf

Định dạng khi nhập liệu.

- ***Cú pháp***

scanf ("chuỗi định dạng"[, đối mục 1, đối mục 2,…]);

☞ *Khi sử dụng hàm phải khai báo tiền xử lý* ***#include <stdio.h>***

- scanf: tên hàm, ***phải viết bằng chữ thường.***
- khung định dạng: được đặt trong cặp nháy kép (" ") là hình ảnh dạng dữ liệu nhập vào.
- đối mục 1,…: là danh sách các đối mục cách nhau bởi dấu phẩy, mỗi đối mục sẽ tiếp nhận giá trị nhập vào.

Ví dụ 11: scanf("%d", &i);

đối mục 1

mã định dạng

☞ Nhập vào 12abc, biến i chỉ nhận giá trị 12. Nhập 3.4 chỉ nhận giá trị 3.

Ví dụ 12: scanf("%d%d", &a, &b);

☞ Nhập vào 2 số a, b phải cách nhau bằng **khoảng trắng** hoặc **enter**.

Ví dụ 13: scanf("%d/%d/%d", &ngay, &thang, &nam);

☞ Nhập vào ngày, tháng, năm theo dạng ngay/thang/nam (20/12/2002)

Ví dụ 14: scanf("%d%*c%d%*c%d", &ngay, &thang, &nam);

☞ Nhập vào ngày, tháng, năm với dấu phân cách /, -,…; ngoại trừ số.

Ví dụ 15: scanf("%2d%2d%4d", &ngay, &thang, &nam);

☞ Nhập vào ngày, tháng, năm theo dạng dd/mm/yyyy.

4.3 Bài tập

1. ***Viết chương trình đổi một số nguyên hệ 10 sang hệ 2.***
2. ***Viết chương trình đổi một số nguyên hệ 10 sang hệ 16.***
3. ***Viết chương trình đọc và 2 số nguyên và in ra kết quả của phép (+), phép trừ (-), phép nhân (*), phép chia (/). Nhận xét kết quả chia 2 số nguyên.***
4. ***Viết chương trình nhập vào bán kính hình cầu, tính và in ra diện tích, thể tích của hình cầu đó.***

   Hướng dẫn: $S = 4\pi R^2$ và $V = (4/3)\pi R^3$.

5. ***Viết chương trình nhập vào một số a bất kỳ và in ra giá trị bình phương (a^2), lập phương (a^3) của a và giá trị a^4.***
6. ***Viết chương trình đọc từ bàn phím 3 số nguyên biểu diễn ngày, tháng, năm và xuất ra màn hình dưới dạng "ngay/thang/nam" (chỉ lấy 2 số cuối của năm).***
7. ***Viết chương trình nhập vào số giây từ 0 đến 86399, đổi số giây nhập vào thành dạng "gio:phut:giay", mỗi thành phần là một số nguyên có 2 chữ số.***

   Ví dụ: 02:11:05

Bài 5 :

CẤU TRÚC RẼ NHÁNH CÓ ĐIỀU KIỆN

(Cấu trúc chọn)

5.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:

- Ý nghĩa lệnh, khối lệnh.
- Cú pháp, ý nghĩa, cách sử dụng lệnh if, lệnh switch.
- Một số bài toán sử dụng lệnh if, switch thông qua các ví dụ.
- So sánh, đánh giá một số bài toán sử dụng lệnh if hoặc switch.
- Cách sử dụng các cấu trúc lồng nhau.

5.2 Nội dung

5.2.1 Lệnh và khối lệnh

5.2.1.1 Lệnh

Là một tác vụ, biểu thức, hàm, cấu trúc điều khiển…

Ví dụ 1:

```
x = x + 2;
printf("Day la mot lenh\n");
```

5.2.1.2 Khối lệnh

Là một dãy các câu lệnh được bọc bởi cặp dấu { }, các lệnh trong khối lệnh phải viết thụt vô 1 tab so với cặp dấu { }

Ví dụ 2:

```
{  //dau khoi
   a = 5;
   b = 6;                                  } viết thụt vô 1 tab so với cặp { }
   printf("Tong %d + %d = %d", a, b, a+b);
}  //cuoi khoi
```

☞ **Quên dùng cặp dấu { } bao bọc khi sử dụng khối lệnh, hoặc mở dấu { và quên đóng dấu }**

5.2.2 Lệnh if

Câu lệnh if cho phép lựa chọn một trong hai nhánh tùy thuộc vào giá trị của biểu thức luận lý là đúng (true) hay sai (false) hoặc khác không hay bằng không.

5.2.2.1 Dạng 1 (if thiếu)

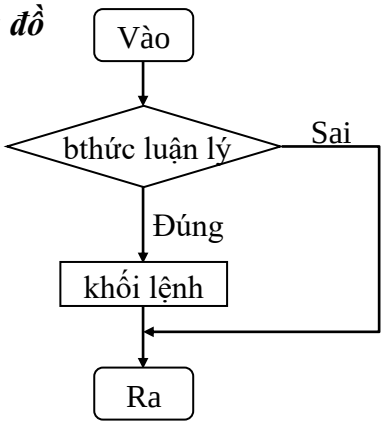
Quyết định sẽ thực hiện hay không một khối lệnh.

- ***Cú pháp lệnh***

```
if (biểu thức luận lý)
    khối lệnh;
```

☞ từ khóa **if** phải viết bằng chữ thường
☞ kết quả của **biểu thức luận lý** phải là **đúng (≠ 0)** hoặc **sai (= 0)**

- ***Lưu đồ***



☞ nếu **biểu thức luận lý** đúng thì
thực hiện khối lệnh và thoát khỏi if,
ngược lại
không làm gì cả và thoát khỏi if.

✍ Nếu **khối lệnh** bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

Diễn giải:

+ Khối lệnh là một lệnh ta viết lệnh if như sau:

```
if (biểu thức luận lý)
   lệnh;
```

+ Khối lệnh bao gồm nhiều lệnh: lệnh 1, lệnh 2..., ta viết lệnh if như sau:

```
if (biểu thức luận lý)
{
   lệnh 1;
   lệnh 2;
   ...
}
```

☞ **Không đặt dấu chấm phẩy sau câu lệnh if.**
Ví dụ: if(biểu thức luận lý);
→ trình biên dịch không báo lỗi nhưng khối lệnh không được thực hiện cho dù điều kiện đúng hay sai.

Ví dụ 3: Viết chương trình nhập vào 2 số nguyên a, b. Tìm và in ra số lớn nhất.

a. Phác họa lời giải

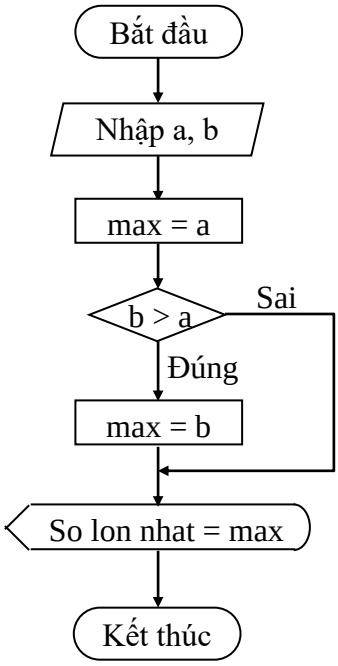
Trước tiên ta cho giá trị ***a là giá trị lớn nhất bằng cách gán a cho max*** (max là biến được khai báo cùng kiểu dữ liệu với a, b). Sau đó so sánh b với a, ***nếu b lớn hơn a ta gán b cho max*** và cuối cùng ta ***được kết quả max là giá trị lớn nhất***.

b. Mô tả quy trình xử lý (giải thuật)

Ngôn ngữ tự nhiên	Ngôn ngữ C
- Khai báo 3 biến a, b, max kiểu số nguyên	- int ia, ib, imax;
- Nhập vào giá trị a	- printf("Nhap vao so a: "); scanf("%d", &ia);
- Nhập vào giá trị b	- printf("Nhap vao so b: "); scanf("%d", &ib);
- Gán a cho max	- imax = ia;
- Nếu b > a thì gán b cho max	- if (ib > ia) imax = ib;
- In ra kết quả max	- printf("So lon nhat = %d.\n", imax);

☞ **Biểu thức luận lý phải đặt trong cặp dấu ( ). if ib > ia → báo lỗi**

c. Mô tả bằng lưu đồ



d. Viết chương trình

File **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
/* Chuong trinh tim so lon nhat tu 2 so nguyen a, b */

#include <stdio.h>
#include <conio.h>

void main(void)
{
   int ia, ib, imax;
   printf("Nhap vao so a: ");
   scanf("%d", &ia);
   printf("Nhap vao so b: ");
   scanf("%d", &ib);
   imax = ia;
   if (ib>ia)
      imax = ib;
   printf("So lon nhat = %d.\n", imax);
   getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so a : 10 Nhap vao so b : 8 So lon nhat = 10. _	Cho chạy lại chương trình và thử lại với: a = 7, b = 9 a = 5, b = 5 Quan sát và nhận xét kết quả

Ví dụ 4: Viết chương trình nhập vào 2 số nguyên a, b. Nếu a lớn hơn b thì hoán đổi giá trị a và b, ngược lại không hoán đổi. In ra giá trị a, b.

a. Phác họa lời giải

Nếu giá trị a lớn hơn giá trị b, bạn phải hoán chuyển 2 giá trị này cho nhau (nghĩa là a sẽ mang giá trị b và b mang giá trị a) bằng cách đem ***giá trị a gởi (gán) cho biến tam*** (biến tam

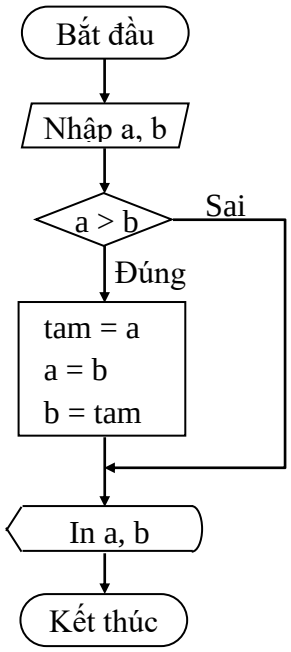
được khai báo theo kiểu dữ liệu của a, b), kế đến bạn ***gán giá trị b cho a*** và cuối cùng bạn ***gán giá trị tam cho b***, rồi in ra a, b.

b. Mô tả quy trình thực hiện (giải thuật)

Ngôn ngữ tự nhiên	Ngôn ngữ C

- Khai báo 3 biến a, b, tam kiểu số nguyên
- Nhập vào giá trị a
 	
- Nhập vào giá trị b
 	
- Nếu a > b thì
tam = a;
a = b;
b = tam;
 	
- In ra a, b
	- int ia, ib, itam;
	- printf("Nhap vao so a: ");
	scanf("%d", &ia);
	- printf("Nhap vao so b: ");
	scanf("%d", &ib);
	- if (ia > ib)
	{
	itam = ia;
	ia = ib;
	ib = itam;
	}
	- printf("%d, %d\n", ia, ib);

c. Mô tả bằng lưu đồ



d. Viết chương trình

File **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
/* Chuong trinh hoan vi 2 so a, b neu a > b */

#include <stdio.h>
#include <conio.h>

void main(void)
{
   int ia, ib, itam;
   printf("Nhap vao so a: ");
   scanf("%d", &ia);
   printf("Nhap vao so b: ");
   scanf("%d", &ib);
```

```
    if (ia>ib)
    {
        itam = ia;      //hoan vi a va b
        ia = ib;
        ib = itam;
    }
    printf("%d, %d.\n", ia, ib);
    getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so a : 10 Nhap vao so b : 8 8, 10 _	Cho chạy lại chương trình và thử lại với: a = 1, b = 8 a = 2, b = 2 Quan sát và nhận kết quả

5.2.2.2 Dạng 2 (if đ ủ)

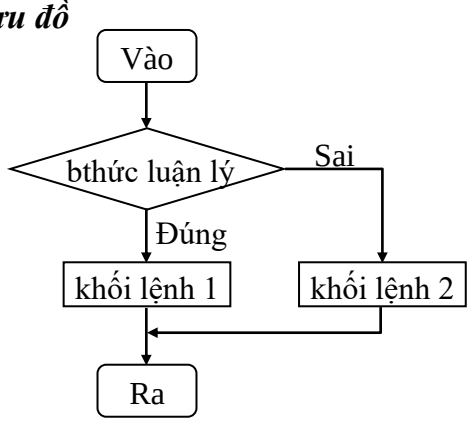
Quyết định sẽ thực hiện 1 trong 2 khối lệnh cho trước.

- ***Cú pháp lệnh***

```
if (biểu thức luận lý)
    khối lệnh 1;
else
    khối lệnh 2;
```

☞ từ khóa **if, else** phải viết bằng chữ thường
☞ kết quả của **biểu thức luận lý** phải là **đúng (≠ 0)** hoặc **sai (= 0)**

- ***Lưu đồ***



☞ nếu **biểu thức luận lý** đúng thì
thực hiện khối lệnh 1 và thoát khỏi if
ngược lại
thực hiện khối lệnh 2 và thoát khỏi if.

✍ Nếu **khối lệnh 1, khối lệnh 2** bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

Ví dụ 5: Viết chương trình nhập vào 2 số nguyên a, b. In ra thông báo "a bằng b" nếu a = b, ngược lại in ra thông báo "a khác b".

a. Phác họa lời giải

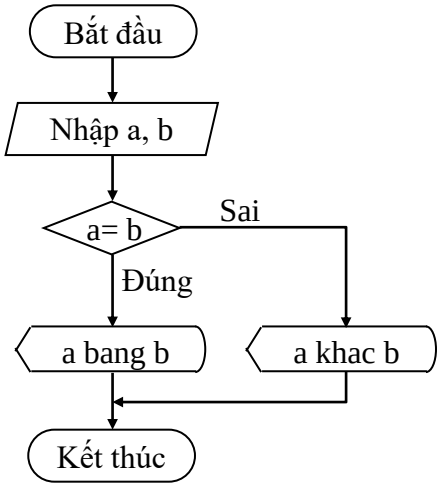
So sánh a với b, nếu a bằng b thì in ra câu thông báo "a bằng b", ngược lại in ra thông báo "a khác b".

b. Mô tả quy trình xử lý (giải thuật)

Ngôn ngữ tự nhiên	Ngôn ngữ C
- Khai báo 2 biến a, b kiểu số nguyên - Nhập vào giá trị a - Nhập vào giá trị b	- int ia, ib; - printf("Nhap vao so a: "); scanf("%d", &ia); - printf("Nhap vao so b: ");

- Nếu a = b thì
in ra thông báo "a bằng b"
Ngược lại (còn không thì)
in ra thông báo "a khác b"
	scanf("%d", &ib);
	- if (ia == ib)
	printf("a bang b\n");
	else
	printf("a khac b\n");

c. Mô tả bằng lưu đồ



d. Viết chương trình

File **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
/* Chuong trinh in ra thong bao "a bang b" neu a = b, nguoc lại in ra "a khac b" */

#include <stdio.h>
#include <conio.h>

void main(void)
{
   int ia, ib;
   printf("Nhap vao so a: ");
   scanf("%d", &ia);
   printf("Nhap vao so b: ");
   scanf("%d", &ib);
   if (ia == ib)
      printf("a bang b\n");
   else
      printf("a khac b\n");
   getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so a : 10 Nhap vao so b : 8 a khac b. _	Cho chạy lại chương trình và thử lại với: a = 6, b = 6 a = 1, b = 5 Quan sát và nhận xét kết quả

☞ **Sau else không có dấu chấm phẩy.**

Ví dụ: else; printf('a khac b\n");

→ trình biên dịch không báo lỗi, lệnh printf("a khac b\n"); không thuộc else

Ví dụ 6: 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".

a. Phác họa lời giải

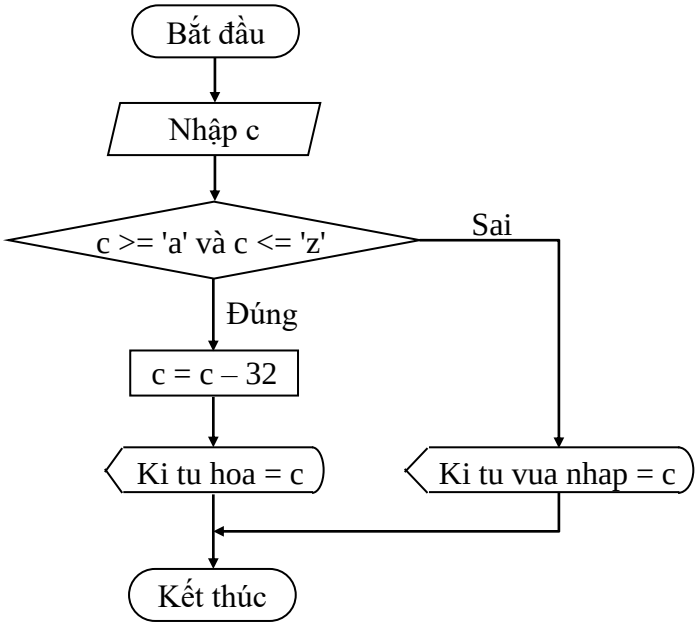
Trước tiên bạn phải kiểm tra xem nếu kí tự c thuộc khoảng 'a' và 'z' thì đổi kí tự c thành chữ in hoa bằng cách lấy kí tự c – 32 rồi gán lại cho chính nó (c = c – 32) (vì giữa kí tự thường và in hoa trong bảng mã ASCII cách nhau 32, ví dụ: A trong bảng mã ASCII là 65, B là 66…, còn a là 97, b là 98…),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

b. Mô tả quy trình xử lý (giải thuật)

Ngôn ngữ tự nhiên	Ngôn ngữ C

- Khai báo biến c kiểu kí tự
- Nhập vào kí tự c
 	
- Nếu c >= a và c <= z thì
c = c – 32
in c ra màn hình
 	
Ngược lại
in ra thông báo " Kí tự bạn vừa nhập là: c "
	- char c;
	- printf("Nhap vao 1 ki tu: ");
	scanf("%c", &c);
	- 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);

c. Mô tả bằng lưu đồ



d. Viết chương trình

File Edit Search Run Compile Debug Project Option Window Help

```
/* Chuong trinh nhap vao ky tu c, neu c la chu thuong in ra chu IN HOA */

#include <stdio.h>
#include <conio.h>

void main(void)
{
   char c;
```

```
    printf("Nhap vao 1 ki tu: ");
    scanf("%c", &c);
    if (c >= 'a' && c <= 'z')          //hoac if(c >= 97 && c <= 122)
    {
        c = c – 32;                    //doi thanh chu in hoa
        printf("Ki tu hoa la: %c.\n", c);
    };
    else
        printf("Ki tu ban vua nhap la: %c.\n", c);
    getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao mot ki tu: g Ki tu hoa la: G. _	Cho chạy lại chương trình và thử lại với: c = '!', c = '2', c = 'A', c = 'u' Quan sát và nhận xét kết quả

5.2.2.3 Cấu trúc else if

Quyết định sẽ thực hiện 1 trong n khối lệnh cho trước.

- ***Cú pháp lệnh***

```
if (biểu thức luận lý 1)
    khối lệnh 1;
else if (biểu thức luận lý 2)
    khối lệnh 2;
…
else if (biểu thức luận lý n-1)
    khối lệnh n-1;
else
    khối lệnh n;
```

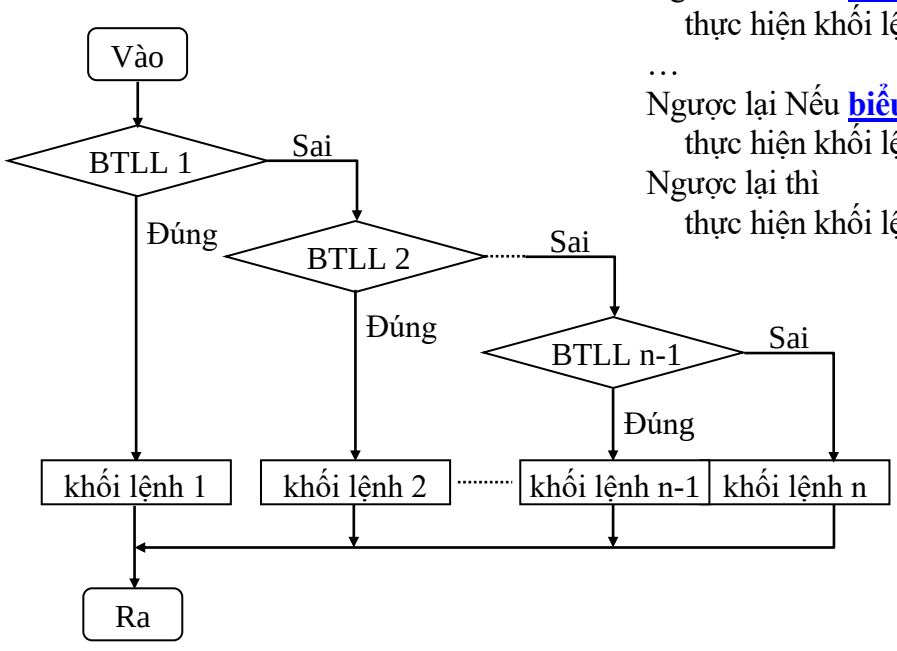
☞ từ khóa **if, else if, else** phải viết bằng chữ thường

☞ kết quả của **biểu thức luận lý 1, 2..n** phải là **đúng (≠ 0)** hoặc **sai (= 0)**

✍ Nếu **khối lệnh 1, 2…n** bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

Nếu **biểu thức luận lý 1** đúng thì
    thực hiện khối lệnh 1 và thoát khỏi cấu trúc if
Ngược lại Nếu **biểu thức luận lý 2** đúng thì
    thực hiện khối lệnh 2 và thoát khỏi cấu trúc if
…
Ngược lại Nếu **biểu thức luận lý n-1** đúng thì
    thực hiện khối lệnh n-1 và thoát khỏi cấu trúc if
Ngược lại thì
    thực hiện khối lệnh n.

- ***Lưu đồ***



Ví dụ 7: Viết chương trình nhập vào 2 số nguyên a, b. In ra thông báo "a lớn hơn b" nếu a>b, in ra thông báo "a nhỏ hơn b" nếu a<b, in ra thông báo "a bằng b" nếu a=b.

a. Phác họa lời giải

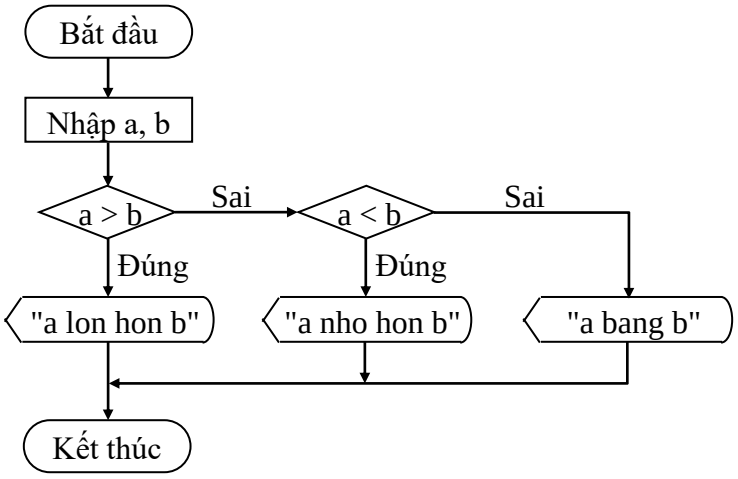
Trước tiên so sánh a với b. Nếu a > b thì in ra thông báo "a lớn hơn b", ngược lại nếu a < b thì in ra thông báo "a nhỏ hơn b", ngược với 2 trường hợp trên thì in ra thông báo "a bằng b".

b. Mô tả quy trình thực hiện (giải thuật)

Ngôn ngữ tự nhiên	Ngôn ngữ C

- Khai báo 2 biến a, b kiểu số nguyên
- Nhập vào giá trị a
 	
- Nhập vào giá trị b
 	
- Nếu a > b thì
in ra thông báo "a lớn hơn b"
Ngược lại Nếu a < b thì
in ra thông báo "a nhỏ hơn b"
Ngược lại thì
in ra thông báo "a bằng b"
	- int ia, ib;
	- printf("Nhap vao so a: ");
	scanf("%d", &ia);
	- printf("Nhap vao so b: ");
	scanf("%d", &ib);
	- if (ia > ib)
	printf("a lon hon b.\n");
	else if (ia < ib)
	printf("a nho hon b.\n");
	else
	printf("a bang b.\n");

c. Mô tả bằng lưu đồ



d. Viết chương trình

File Edit Search Run Compile Debug Project Option Window Help

```
/* Chuong trinh nhap vao 2 so nguyen a, b. In ra thong bao a > b, a < b, a = b */

#include <stdio.h>
#include <conio.h>

void main(void)
{
   int ia, ib;
   printf("Nhap vao so a: ");
   scanf("%d", &ia);
   printf("Nhap vao so b: ");
   scanf("%d", &ib);
   if (ia>ib)
      printf("a lon hon b.\n");
   else if (ia<ib)
```

```
        printf("a nho hon b.\n");
    else
        printf("a bang b.\n");
    getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so a : 5 Nhap vao so b : 7 a nho hon b _	Cho chạy lại chương trình và thử lại với: a = 8, b = 4 a = 2, b = 2 Quan sát và nhận xét kết quả

Ví dụ 8: 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, nếu kí tự in hoa trong khoảng A đến Z thì đổi sang chữ thường và in ra, nếu kí tự là số từ 0 đến 9 thì in ra câu "Kí tự bạn vừa nhập là số …(in ra kí tự c)", còn lại không phải 3 trường hợp trên in ra thông báo "Bạn đã nhập kí tự …(in ra kí tự c)".

a. Phác họa lời giải

Nhập kí tự c vào, kiểm tra xem nếu kí tự c thuộc khoảng 'a' và 'z' 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, ví dụ: A trong bảng mã ASCII là 65, B là 66…, còn a là 97, b là 98…), sau khi đổi xong bạn in kí tự c ra. Ngược lại Nếu kí tự c thuộc khoảng 'A' và 'Z', đổi kí tự c thành chữ thường (theo cách ngược lại) và in ra. Ngược lại Nếu kí tự c thuộc khoảng '0' và '9' thì in ra thông báo "Kí tự bạn vừa nhập là số…". Ngược lại, in câu thông báo "Bạn đã nhập kí tự…".

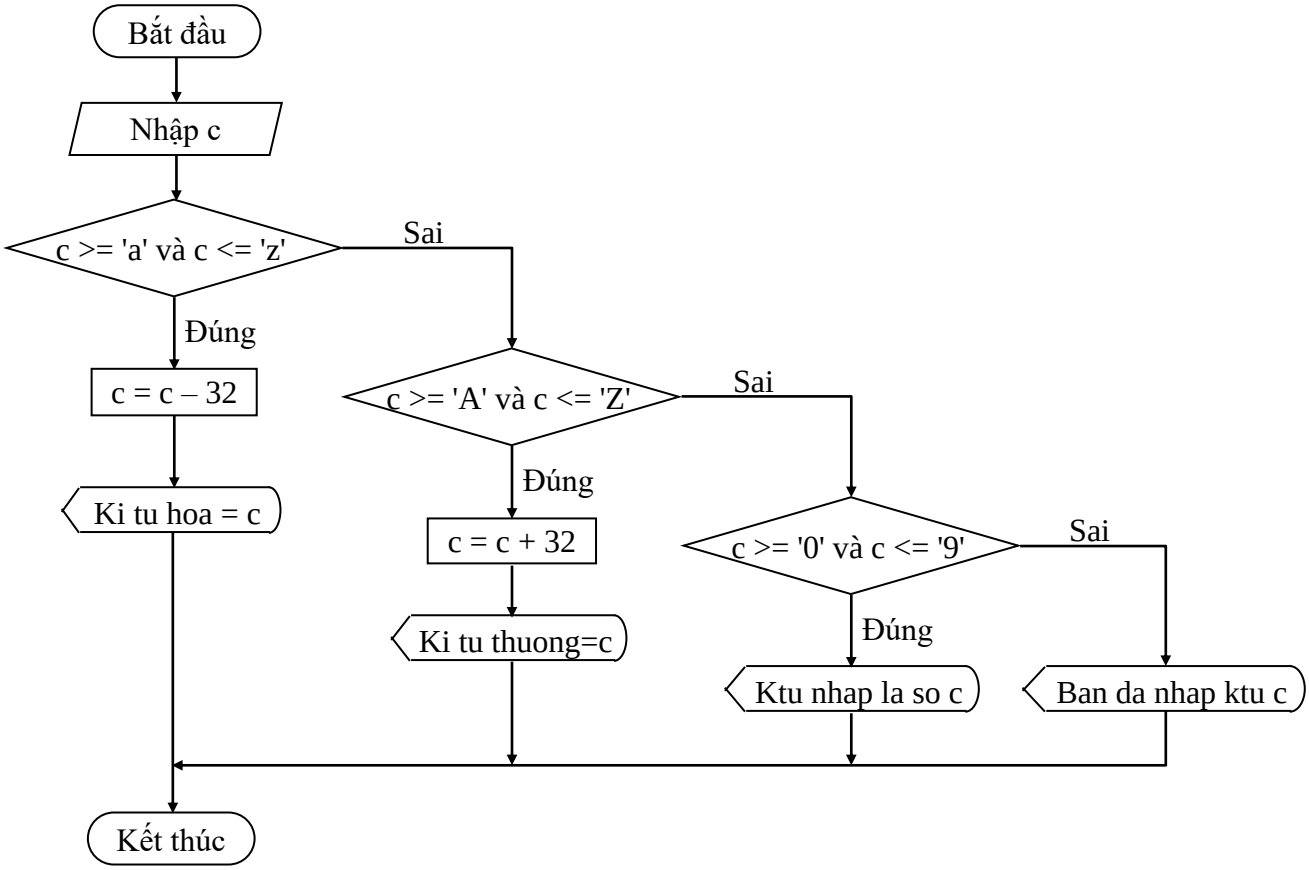
b. Mô tả quy trình xử lý (giải thuật)

Ngôn ngữ tự nhiên	Ngôn ngữ C

- Khai báo biến c kiểu kí tự
- Nhập vào kí tự c
 	
- Nếu c >= a và c <= z thì
c = c – 32
in c ra màn hình
 	
Ngược lại Nếu c >= A và c <= Z thì
c = c + 32
in c ra màn hình
 	
Ngược lại Nếu c >= 0 và c <= 9 thì
in thông báo "Kí tự bạn vừa nhập là số c"
Ngược lại thì
in thông báo "Bạn đã nhập kí tự c"
	- char c;
	- printf("Nhap vao 1 ki tu: ");
	scanf("%c", &c);
	- if (c >= 'a' && c <= 'z')
	{
	c = c – 32;
	printf("Ki tu hoa la: %c.\n", c);
	};
	else if(c >= 'A' && c <= 'Z')
	{
	c = c + 32;
	printf("Ki tu thuong la: %c.\n", c);
	};
	else if(c >= '0' && c <= '9')
	printf("Ki tu Ban vua nhap la so %c.\n", c);
	else
	printf("Ban da nhap ki tu %c.\n", c);

☞ **Cũng như if, không đặt dấu chấm phẩy sau câu lệnh else if.**
Ví dụ: else if(c >= 'A' && c <= 'Z');
→ trình biên dịch không báo lỗi nhưng khối lệnh sau else if không được thực hiện.

c. Mô tả bằng lưu đồ



e. Viết chương trình

File **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
/* Chuong trinh nhap vao ki tu c. Doi ra hoa, thuong */

#include <stdio.h>
#include <conio.h>

void main(void)
{
   char c;
   printf("Nhap vao 1 ki tu: ");
   scanf("%c", &c);
   if (c >= 'a' && c <= 'z')          //hoac if(c >= 97 && c <= 122)
   {
      c = c – 32;                     //doi thanh chu in hoa
      printf("Ki tu hoa la: %c.\n", c);
   };
   else if(c >= 'A' && c <= 'Z')      //hoac if(c >= 65 && c <= 90)
   {
      c = c + 32;                     //doi thanh chu thuong
      printf("Ki tu thuong la: %c.\n", c);
   };
   else if(c >= '0' && c <= '9')      //hoac if(c >= 48 && c <= 57)
      printf("Ki tu Ban vua nhap la so %c.\n", c);
```

```
    else
        printf("Ban da nhap ki tu %c.\n", c);
    getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao mot ki tu: g Ki tu hoa la: G. _	Cho chạy lại chương trình và thử lại với: c = '!', c = '2', c = 'a', c = 'Z' Quan sát và nhận xét kết quả

5.2.2.4 Cấu trúc if lồng

Quyết định sẽ thực hiện 1 trong n khối lệnh cho trước.

- ***Cú pháp lệnh***

Cú pháp là một trong 3 dạng trên, nhưng trong 1 hoặc nhiều khối lệnh bên trong phải chứa ít nhất một trong 3 dạng trên gọi là cấu trúc if lồng nhau. Thường cấu trúc if lồng nhau càng nhiều cấp độ phức tạp càng cao, chương trình chạy càng chậm và trong lúc lập trình dễ bị nhầm lẫn.

Lưu ý: Các lệnh **if…else** lồng nhau thì **else** sẽ luôn luôn kết hợp với **if** nào chưa có else gần nhất. Vì vậy khi gặp những lệnh if không có else, Bạn phải đặt chúng trong những **khối lệnh rõ ràng** để tránh bị hiểu sai câu lệnh.

Ví dụ 9: Bạn viết các dòng lệnh sau:

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

Mặc dù Bạn 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 Bạn viết lại như sau:

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

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

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

- ***Lưu đồ***

Tương tự 3 dạng trên. Nhưng trong mỗi khối lệnh có thể có một (nhiều) cấu trúc if ở 3 dạng trên.

Ví dụ 10: Viết chương trình nhập vào điểm của một học sinh. In ra xếp loại học tập của học sinh đó. (Cách xếp loại. Nếu điểm >= 9, Xuất sắc. Nếu điểm từ 8 đến cận 9, Giỏi. Nếu điểm từ 7 đến cận 8, Khá. Nếu điểm từ 6 đến cận 7, TBKhá. Nếu điểm từ 5 đến cận 6, TBình. Còn lại là Yếu).

a. Phác họa lời giải

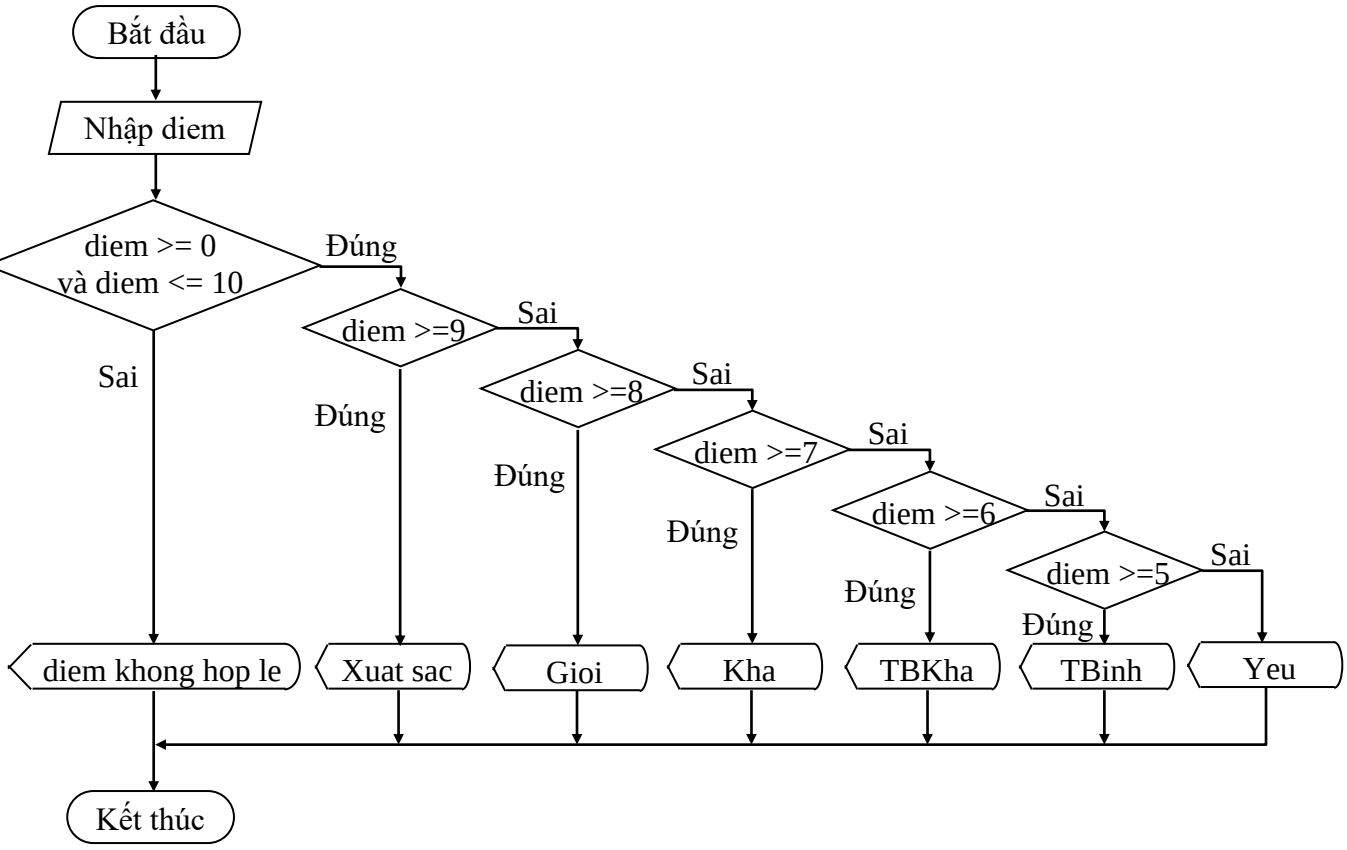
Điểm số nhập vào nếu hợp lệ (0 <= điểm <= 10), bạn tiếp tục công việc xếp loại, ngược lại thông báo "Nhập điểm không hợp lệ". Việc xếp loại bạn sử dụng cấu trúc else if.

b. Mô tả quy trình xử lý (giải thuật)

Ngôn ngữ tự nhiên	Ngôn ngữ C

- Khai báo biến diem kiểu số thực
- Nhập vào điểm số
 	
- Nếu diem >= 0 và diem <= 10 thì
- Nếu diem >= 9 thì
in ra xếp loại = Xuất sắc
Ngược lại Nếu diem >= 8 thì
in ra xếp loại = Giỏi
Ngược lại Nếu diem >= 7 thì
in ra xếp loại = Khá
Ngược lại Nếu diem >= 6 thì
in ra xếp loại = TBKhá
Ngược lại Nếu diem >= 5 thì
in ra xếp loại = TBình
Ngược lại thì
in ra xếp loại = Yếu
Ngược lại thì
in ra "Bạn nhập điểm không hợp lệ"
	- float fdiem;
	- printf("Nhap vao diem so: ");
	scanf("%f", &fdiem);
	- if (fdiem >= 0 && fdiem <= 10)
	- if (fdiem >= 9)
	printf("Xep loai = Xuat sac.\n");
	else if (fdiem >= 8)
	printf("Xep loai = Gioi.\n");
	else if (fdiem >= 7)
	printf("Xep loai = Kha.\n");
	else if (fdiem >= 6)
	printf("Xep loai = TBKha.\n");
	else if (fdiem >= 5)
	printf("Xep loai = TBinh.\n");
	else
	printf("Xep loai = Yeu.\n");
	else
	printf("Ban nhap diem khong hop le.\n");

c. Mô tả bằng lưu đồ



d. Viết chương trình

File **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
/* Chuong trinh nhap vao 2 so nguyen a, b. In ra thong bao a > b, a < b, a = b */

#include <stdio.h>
#include <conio.h>

void main(void)
{
   float fdiem;
   printf("Nhap vao diem so: ");
   scanf("%f", &fdiem);
   if (fdiem >=0 && fdiem <=10)
      if (fdiem >=9)
         printf("Xep loai = Xuat sac.\n");
      else if (fdiem >=8)
         printf("Xep loai = Gioi.\n");
      else if (fdiem >=7)
         printf("Xep loai = Kha.\n");
      else if (fdiem >=6)
         printf("Xep loai = TBKha.\n");
      else if (fdiem >=5)
         printf("Xep loai = TBinh.\n");
      else
         printf("Xep loai = Yeu.\n");
   else          //if (fdiem>=0 && fdiem<=10)
      printf("Nhap diem khong hop le.\n");
   getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao diem so: 6.5 Xep loai = TBKha. _	Cho chạy lại chương trình và thử lại với: diem = 4, diem = 9, diem = 7, diem = 12 Quan sát và nhận xét kết quả

e. Bàn thêm về chương trình

Trong chương trình trên cấu trúc **else if được lồng vào trong cấu trúc dạng 2**, trong cấu trúc else if ta không cần đặt trong khối vì tất cả các if trong cấu trúc này đều có else, nên else printf("Nhap diem khong hop le.\n") đương nhiên là thuộc về if (fdiem >= 0 && fdiem <= 10). Giả sử trong cấu trúc else if không có dòng else printf("Xep loai = Yeu.\n") thì khi đó dòng else printf("Nhap diem khong hop le.\n") sẽ thuộc về cấu trúc else if chứ không thuộc về if (fdiem >=0 && fdiem <= 10). Đối với trường hợp đó bạn cần phải đặt cấu trúc else if vào trong {}, thì khi đó dòng else printf("Nhap diem khong hop le.\n) sẽ thuộc về if (fdiem >= 0 && fdiem <= 10).

Ví dụ 11: Viết chương trình nhập vào 3 số nguyên a, b, c. Tìm và in ra số lớn nhất.

a. Phác họa lời giải

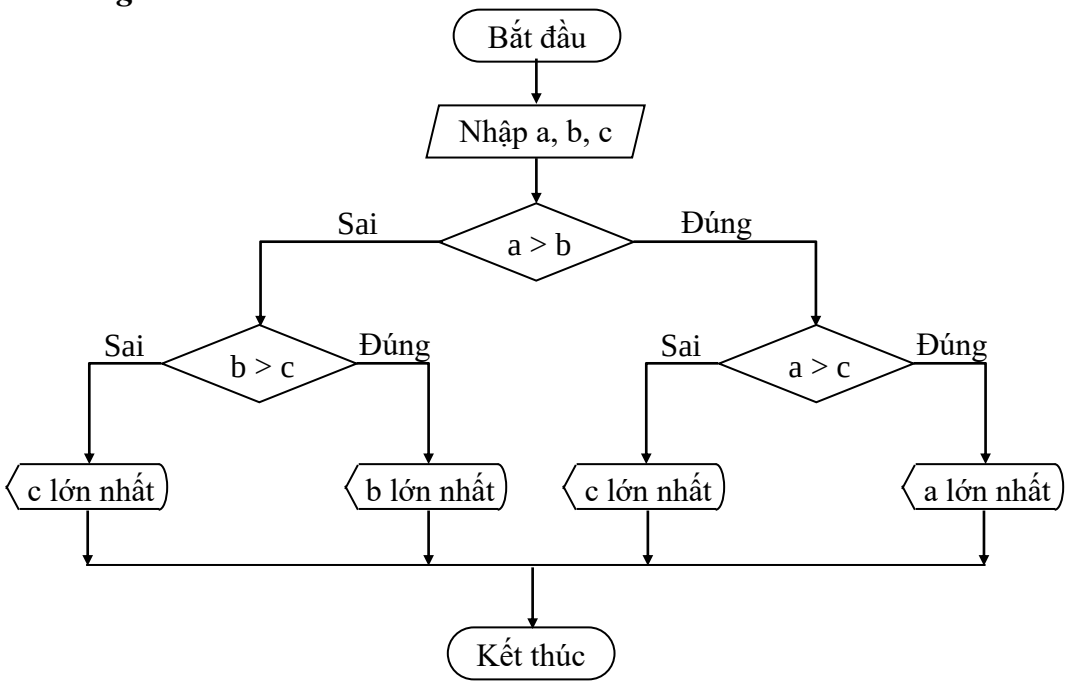
Trước tiên bạn so nếu a>b, mà a>c thì a lớn nhất, ngược lại c lớn nhất, còn nếu a<=b, mà c>b thì b lớn nhất, ngược lại c lớn nhất.

b. Mô tả quy trình xử lý (giải thuật)

Ngôn ngữ tự nhiên	Ngôn ngữ C

- Khai báo 3 biến a, b, c kiểu số nguyên
- Nhập vào số a
 	
- Nhập vào số b
 	
- Nhập vào số c
 	
- Nếu a > b thì
  - Nếu a > c thì
  a lớn nhất
  Ngược lại thì
  c lớn nhất
Ngược lại
  - Nếu b > c thì
  b lớn nhất
  Ngược lại thì
  c lớn nhất
	- int ia, ib, ic;
	- printf("Nhap vao so a: ");
	  scanf("%d", &ia);
	- printf("Nhap vao so b: ");
	  scanf("%d", &ib);
	- printf("Nhap vao so c: ");
	  scanf("%d", &ic);
	- if (ia > ib)
	  - if (ia > ic)
	  printf("%d lon nhat.\n", ia);
	  else
	  printf("%d lon nhat.\n", ic);
	else
	  - if (ib > ic)
	  printf("%d lon nhat.\n", ib);
	  else
	  printf("%d lon nhat.\n", ic);

c. Mô tả bằng lưu đồ



d. Viết chương trình

File **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
/* Chuong trinh nhap vao 2 so nguyen a, b, c. Tim, in ra so lon nhat */

#include <stdio.h>
#include <conio.h>

void main(void)
{
   int ia, ib, ic;
   printf("Nhap vao so a: ");
```

```
    scanf("%d", &ia);
    printf("Nhap vao so b: ");
    scanf("%d", &ib);
    printf("Nhap vao so c: ");
    scanf("%d", &ic);
    if (ia > ib)
        if (ia > ic)
            printf("%d lon nhat.\n", ia);
        else
            printf("%d lon nhat.\n", ic);
    else
        if (ib > ic)
            printf("%d lon nhat.\n", ib);
        else
            printf("%d lon nhat.\n", ic);
    getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so a: 4 Nhap vao so b: 5 Nhap vao so c: 3 5 lon nhat. _	Cho chạy lại chương trình và thử lại với: a = 5, b = 4, c = 2 a = 2, b = 1, c = 10 a = 5, b = 5, c = 5 Quan sát và nhận xét kết quả

e. Bàn thêm về chương trình

Trong chương trình trên cấu trúc **dạng 2 được lồng vào trong cấu trúc dạng 2.**

5.2.3 Lệnh switch

Lệnh switch cũng giống cấu trúc else if, nhưng nó mềm dẻo hơn và linh động hơn nhiều so với sử dụng if. Tuy nhiên, nó cũng có mặt hạn chế là kết quả của biểu thức phải là giá trị hằng nguyên (có giá trị cụ thể). 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.

5.2.3.1 Cấu trúc switch…case (switch thiếu)

Chọn thực hiện 1 trong n lệnh cho trước.

- ***Cú pháp lệnh***

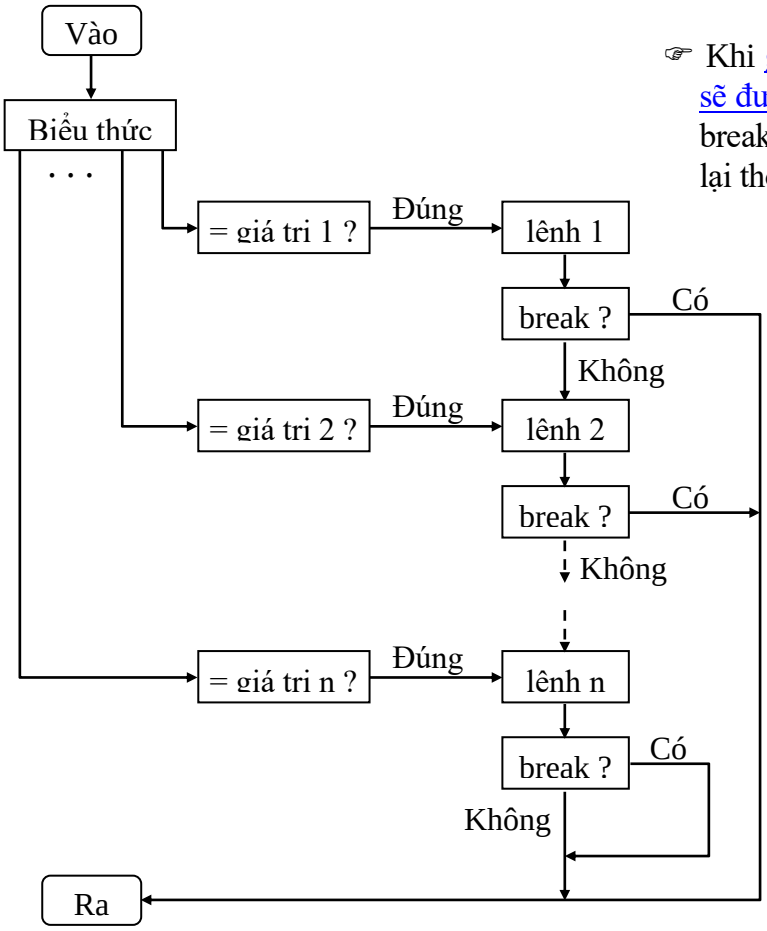
```
switch (biểu thức)
{
   case giá trị 1  : lệnh 1;
                     break;
   case giá trị 2  : lệnh 2;
                     break;
   …
   case giá trị n  : lệnh n;
                     [break;]
}
```

☞ từ khóa **switch, case, break** phải viết bằng chữ thường

☞ **biểu thức** phải là có kết quả là **giá trị hằng nguyên (char, int, long,…)**

☞ **Lệnh 1, 2…n** có thể gồm nhiều lệnh, nhưng không cần đặt trong cặp dấu { }

- ***Lưu đồ***



☞ Khi giá trị của biểu thức bằng giá trị i thì lệnh i sẽ được thực hiện. Nếu sau lệnh i không có lệnh break thì sẽ tiếp tục thực hiện lệnh i + 1…Ngược lại thoát khỏi cấu trúc switch.

Ví dụ 12: Viết chương trình nhập vào số 1, 2, 3. In ra tương ứng 1, 2, 3 sao.

a. Viết chương trình

File Edit Search Run Compile Debug Project Option Window Help

```
/* Chuong trinh nhap vao so 1, 2, 3. In ra so sao tuong ung */
#include <stdio.h>
#include <conio.h>
void main(void)
{
   int i;
   printf("Nhap vao so 1, 2 hoặc 3: ");
   scanf("%d", &i);
   switch(i)
   {
      case 3:  printf("*");
      case 2:  printf("*");
      case 1:  printf("*");
   };
   printf("An phim bat ky de ket thuc!\n");
   getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so 1, 2 hoặc 3: 2 ** _	Cho chạy lại chương trình và thử lại với: i = 1, i = 3, i = 0, i = 4 Quan sát và nhận xét kết quả

b. Bàn thêm về chương trình

Trong chương trình trên khi nhập vào i = 2 lệnh printf("*") ở dòng case 2 được thi hành, nhưng do không có lệnh break sau đó nên lệnh printf("*") ở dòng case 1 tiếp tục được thi hành. Kết quả in ra **.

☞ **Không đặt dấu chấm phẩy sau câu lệnh switch.**
Ví dụ: switch(i);
→ trình biên dịch không báo lỗi nhưng các lệnh trong switch không được thực hiện.

Ví dụ 13: Viết chương trình nhập vào tháng và in ra quý. (tháng 1 -> quý 1, tháng 10 -> quý 4)

a. Phác họa lời giải

Nhập vào giá trị tháng, kiểm tra xem tháng có hợp lệ (trong khoảng 1 đến 12). Nếu hợp lệ in ra quý tương ứng (1->3: quý 1, 4->6: quý 2, 7->9: quý 3, 10->12: quý 4).

b. Viết chương trình

File **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
/* Chuong trinh nhap vao thang. In ra quy tuong ung */

#include <stdio.h>
#include <conio.h>

void main(void)
{
   int ithang;
   printf("Nhap vao thang: ");
   scanf("%d", &ithang);
   if (ithang > 0 && ithang <= 12)
      switch(ithang)
      {
         case 1:
         case 2:
         case 3:  printf("Quy 1.\n");
                  break;
         case 4:
         case 5:
         case 6:  printf("Quy 2.\n");
                  break;
         case 7:
         case 8:
         case 9:  printf("Quy 3.\n");
                  break;
         case 10:
         case 11:
         case 12:printf("Quy 4.\n");
                  break;
      };
   else
      printf("Thang khong hop le.\n");
   getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao thang: 4 Quy 2. _	Cho chạy lại chương trình và thử lại với: thang = 7, thang = 1, thang = 13, thang = -4 Quan sát và nhận xét kết quả

c. Bàn thêm về chương trình

Trong chương trình trên cấu trúc **switch…case được lồng vào trong cấu trúc if dạng 2.**

5.2.3.2 Cấu trúc switch…case…default (switch đủ)

Chọn thực hiện 1 trong n + 1 lệnh cho trước.

- ***Cú pháp lệnh***

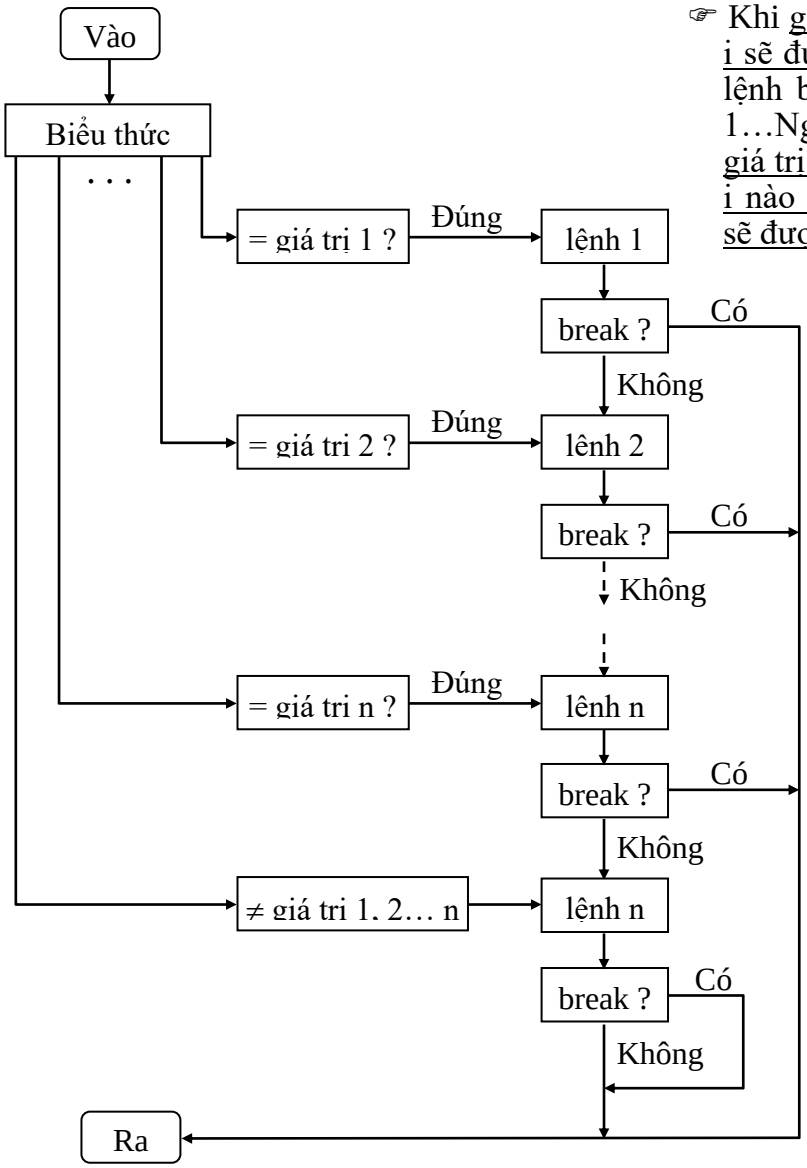
```
switch (biểu thức)
{
   case giá trị 1  : lệnh 1;
                     break;
   case giá trị 2  : lệnh 2;
                     break;
   …
   case giá trị n  : lệnh n;
                     break;
   default         : lệnh;
                     [break;]
}
```

☞ từ khóa **switch, case, break, default** phải viết bằng chữ thường

☞ **biểu thức** phải là có kết quả là **giá trị nguyên (char, int, long,…)**

☞ **Lệnh 1, 2…n** có thể gồm nhiều lệnh, nhưng không cần đặt trong cặp dấu { }

- ***Lưu đồ***



☞ Khi giá trị của biểu thức bằng giá trị i thì lệnh i sẽ được thực hiện. Nếu sau lệnh i không có lệnh break thì sẽ tiếp tục thực hiện lệnh i + 1…Ngược lại thoát khỏi cấu trúc switch. Nếu giá trị biểu thức không trùng với bất kỳ giá trị i nào thì lệnh tương ứng với từ khóa default sẽ được thực hiện.

Ví dụ 14: Viết lại chương trình ở **Ví dụ 12**

a. Viết chương trình

File **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
/* Chuong trinh nhap vao so 1, 2, 3. In ra so sao tuong ung */

#include <stdio.h>
#include <conio.h>

void main(void)
{
   int i;
   printf("Nhap vao so 1, 2 hoặc 3: ");
   scanf("%d", &i);
   switch(i)
   {
      case 3:  printf("*");
      case 2:  printf("*");
      case 1:  printf("*");
               break;
      default: printf("Ban nhap phai nhap vao so 1, 2 hoac 3.\n");
   };
   getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so 1, 2 hoặc 3: 3 *** _	Cho chạy lại chương trình và thử lại với: i = 1, i = 3, i = 0, i = 4 Quan sát kết quả

b. Bàn thêm về chương trình

Trong chương trình trên. Nếu bạn nhập vào 1, 2, 3 sẽ in ra số sao tương ứng. Ngoài các số này chương trình sẽ in ra câu thông báo "Bạn phải nhập vào số 1, 2 hoặc 3".

Ví dụ 15: Viết lại chương trình ở **Ví dụ 13**

a. Viết chương trình

File **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
/* Chuong trinh nhap vao thang. In ra quy tuong ung */

#include <stdio.h>
#include <conio.h>

void main(void)
{
   int ithang;
   printf("Nhap vao thang: ");
   scanf("%d", &ithang);
   switch(ithang)
   {
```

```
        case 1: case 2: case 3 :      printf("Quy 1.\n");
                                      break;
        case 4: case 5: case 6:       printf("Quy 2.\n");
                                      break;
        case 7: case 8: case 9:       printf("Quy 3.\n");
                                      break;
        case 10: case 11: case 12:    printf("Quy 4.\n");
                                      break;
        default                  :    printf("Ban phai nhap vao so trong khoang 1..12\n");
    };
    getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao thang: 4 Quy 2. _	Cho chạy lại chương trình và thử lại với: thang = 7, thang = 1, thang = 13, thang = -4 Quan sát kết quả

c. Bàn thêm về chương trình

Trong chương trình trên. Nếu bạn nhập vào 1 đến 12 sẽ in quý tương ứng. Ngoài các số này chương trình sẽ in ra câu thông báo "Bạn phải nhập vào số trong khoảng 1..12".

5.2.3.3 Cấu trúc switch lồng

Quyết định sẽ thực hiện 1 trong n khối lệnh cho trước.

- ***Cú pháp lệnh***

Cú pháp là một trong 2 dạng trên, nhưng trong 1 hoặc nhiều lệnh bên trong phải chứa ít nhất một trong 2 dạng trên gọi là cấu trúc switch lồng nhau. Thường cấu trúc switch lồng nhau càng nhiều cấp độ phức tạp càng cao, chương trình chạy càng chậm và trong lúc lập trình dễ bị nhầm lẫn.

- ***Lưu đồ***

Tương tự 2 dạng trên. Nhưng trong mỗi lệnh có thể có một (nhiều) cấu trúc switch ở 2 dạng trên.

Ví dụ 16: Viết chương trình menu 2 cấp

a. Viết chương trình

File **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
/* Chuong trinh menu 2 cap */

#include <stdio.h>
#include <conio.h>

void main(void)
{
    int imenu, isubmenu;
    printf("------------------------\n");
    printf("     MAIN MENU     \n");
    printf("------------------------\n");
    printf("1. File\n");
```

```
    printf("2. Edit\n");
    printf("3. Search\n");
    printf("Chon muc tuong ung: ");
    scanf("%d", &imenu);
    switch(imenu)
    {
       case 1:  printf("------------------------\n");
                printf("      MENU FILE      \n");
                printf("------------------------\n");
                printf("1. New\n");
                printf("2. Open\n");
                printf("Chon muc tuong ung: ");
                scanf("%d", &isubmenu);
                switch(isubmenu)
                {
                   case 1:  printf("Ban da chon chuc nang New File\n");
                            break;
                   case 2:  printf("Ban da chon chuc nang Open File\n");
                }
                break;       //break cua case 1 – switch(imenu)
       case 2:  printf("Ban da chon chuc nang Edit\n");
                break;
       case 3:  printf("Ban da chon chuc nang Search\n");
    };
    getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
-------------------------
    MAIN MENU
-------------------------
1. File
2. Edit
3. Search
Chon muc tuong ung: 1
-------------------------
     MENU FILE
-------------------------
1. New
2. Open
Chon muc tuong ung: 2
Ban da chon chuc nang Open File
_
```

Cho chạy lại chương trình và thử lại với: mục chọn chức năng khác
Quan sát kết quả.
* Thêm các thành phần sau vào chương trình:
- Thêm mục Save vào menu File.
- Tạo menu Edit gồm 4 chức năng: Copy, Cut, Paste, Clear.
- Tạo menu Search gồm 2 chức năng: Find, Replace.

Chạy lại chương trình và thử với nhiều mục chọn khác nhau.
Quan sát kết quả.

5.3 Bài tập

5.3.1 Sử dụng lệnh if

1. ***Viết lại chương trình ví dụ 3, sử dụng cấu trúc if dạng 2.***
2. ***Viết lại chương trình ví dụ 11, sử dụng cấu trúc if dạng 1.***

3. Viết lại chương trình ví dụ 11, sử dụng cấu trúc if dạng 2.

4. Viết chương trình nhập vào số nguyên dương, in ra thông báo số chẵn hay lẻ.

Hướng dẫn: Nhập vào số nguyên dương x. Kiểm tra nếu x chia chẵn cho hai thì x là số chẵn (hoặc chia cho 2 dư 0) ngược lại là số lẻ.

5. Viết chương trình nhập vào 4 số nguyên. Tìm và in ra số lớn nhất.

Hướng dẫn: Ta có 4 số nguyên a, b, c, d. Tìm 2 số nguyên lớn nhất x, y của 2 cặp (a, b) và (c, d). Sau đó so sánh 2 số nguyên x, y để tìm ra số nguyên lớn nhất.

6. Viết chương trình giải phương trình bậc 2: ax2 + bx + c = 0, với a, b, c nhập vào từ bàn phím.

Hướng dẫn: Nhập vào 3 biến a, b, c.

```
Tính Delta = b*b - 4*a*c
Nếu Delta < 0 thì
    Phương trình vô nghiệm
Ngược lại
    Nếu Delta = 0 thì
       x1 = x2 = - b/(2*a)
    Ngược lại
       x1 = (- b - sqrt(Delta))/(2*a)
       x2 = (- b + sqrt(Delta))/(2*a)
    Hết Nếu
Hết Nếu
```

7. Viết chương trình nhập vào giờ phút giây (hh:mm:ss). Cộng thêm số giây nhập vào và in ra kết quả dưới dạng hh:mm:ss.

Hướng dẫn: Nhập vào giờ phút giây vào 3 biến gio, phut, giay và nhập và giây công thêm vào biến them:

```
Nếu giay + them < 60 thì
    giay = giay + them
Ngược lại
    giay = (giay + them) - 60
    phut = phut + 1
    Nếu phut >= 60 thì
       phut = phut - 60
       gio = gio + 1
    Hết nếu
Hết nếu
```

5.3.2 Sử dụng lệnh switch

8. Viết chương trình nhập vào tháng, in ra tháng đó có bao nhiêu ngày.

Hướng dẫn: Nhập vào tháng

Nếu là tháng 1, 3, 5, 7, 8, 10, 12 thì có 30 ngày

Nếu là tháng 4, 6, 9, 11 thì có 31 ngày

Nếu là tháng 2 và là năm nhuận thì có 29 ngày ngược lại 28 ngày

(Năm nhuận là năm chia chẵn cho 4)

9. Viết chương trình trò chơi One-Two-Three ra cái gì ra cái này theo điều kiện:

- Búa (B) thắng Kéo, thua Giấy.
- Kéo (K) thắng Giấy, thua Búa.
- Giấy (G) thắng Búa, thua Kéo.

Hướng dẫn: Dùng lệnh switch lồng nhau

10. Viết chương trình xác định biến ký tự color rồi in ra thông báo

- RED, nếu color = 'R' hoặc color = 'r'
- GREEN, nếu color = 'G' hoặc color = 'g'
- BLUE, nếu color = 'B' hoặc color = 'b'
- BLACK, nếu color có giá trị khác.

11. Viết chương trình nhập vào 2 số x, y và 1 trong 4 toán tử +, -, *, /. Nếu là + thì in ra kết quả x + y, nếu là – thì in ra x – y, nếu là * thì in ra x * y, nếu là / thì in ra x / y (nếu y = 0 thì thông báo không chia được)

5.4 Bài tập làm thêm

12. Viết lại bài tập 8, 9, 10, 11 sử dụng lệnh if.

13. Viết chương trình nhập vào điểm 3 môn thi: Toán, Lý, Hóa của học sinh. Nếu tổng điểm >= 15 và không có môn nào dưới 4 thì in kết quả đậu. Nếu đậu mà các môn đều lớn hơn 5 thì in ra lời phê "Học đều các môn", ngược lại in ra "Học chưa đều các môn", các trường hợp khác là "Thi hỏng".

14. Viết chương trình nhập vào ngày tháng năm (dd:mm:yy), cho biết đó là thứ mấy trong tuần.

15. Viết chương trình nhập số giờ làm và lương giờ rồi tính số tiền lương tổng cộng. Nếu số giờ làm lớn hơn 40 thì những giờ làm dôi ra được tính 1,5 lần.

16. Viết chương trình nhập vào 3 giá trị nguyên dương a, b, c. Kiểm tra xem a, b, c có phải là 3 cạnh của tam giác không? Nếu là 3 cạnh của tam giác thì tính diện tích của tam giác theo công thức sau:

17. $S = \sqrt{p*(p-a)*(p-b)*p-c)}$ ***, với p là 1/2 chu vi của tam giác.***

Hướng dẫn: a, b, c là 3 cạnh của tam giác phải thỏa điều kiện sau:
(a + b) > c và (a + c) > b và (b + c) > a

18. Viết chương trình nhập vào 3 số nguyên rồi in ra màn hình theo thứ tự tăng dần.

19. Viết chương trình tính tiền điện gồm các khoản sau:

- Tiền thuê bao điện kế: 1000đ/tháng
- Định mức sử dụng điện cho mỗi hộ là: 50 KW với giá 230đ/KW
- Nếu phần vượt định mức <= 50KW thì tính giá 480đ/KW
- Nếu 50KW < phần vượt định mức < 100KW thì tính giá 700đ/KW
- Nếu phần vượt định mức <= 100KW thì tính giá 900đ/KW

Chỉ số mới và cũ được nhập vào từ bàn phím

- In ra màn hình chỉ số cũ, chỉ số mới, tiền trả định mức, tiền trả vượt định mức, tổng tiền phải trả.

Bài 6 :

CẤU TRÚC VÒNG LẶP

6.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:
- Ý nghĩa, cách hoạt động của vòng lặp.
- Cú pháp, ý nghĩa, cách sử dụng lệnh for, while, do…while.
- Ý nghĩa và cách sử dụng lệnh break, continue.
- Một số bài toán sử dụng lệnh for, while, do…while thông qua các ví dụ.
- So sánh, đánh giá một số bài toán sử dụng lệnh for, while hoặc do…while.
- Cấu trúc vòng lặp lồng nhau.

6.2 Nội dung

6.2.1 Lệnh for

Vòng lặp xác định thực hiện lặp lại một số lần xác định của một (chuỗi hành động)

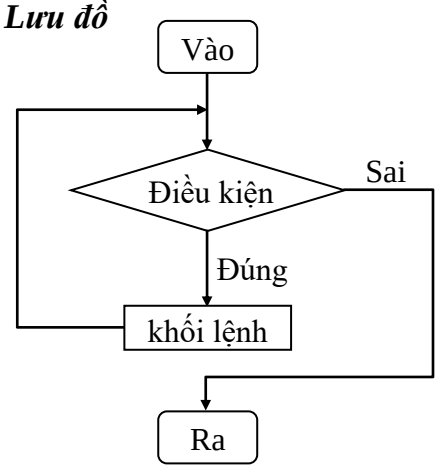
- ***Cú pháp lệnh***

```
for (biểu thức 1; biểu thức 2; biểu thức 3)
    khối lệnh;
```

☞ từ khóa **for** phải viết bằng chữ thường

✍ Nếu **khối lệnh** bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

- ***Lưu đồ***



☞ kiểm tra **điều kiện**
nếu **đúng** đúng thì
thực hiện khối lệnh;
lặp lại kiểm tra điều kiện
nếu **sai**
thoát khỏi vòng lặp.

Giải thích:

+ Biểu thức 1: khởi tạo giá trị ban đầu cho biến điều khiển.
+ Biểu thức 2: là quan hệ logic thể hiện điều kiện tiếp tục vòng lặp.
+ Biểu thức 3: phép gán dùng thay đổi giá trị biến điều khiển.

Nhận xét:

+ Biểu thức 1 bao giờ cũng chỉ được tính toán một lần khi gọi thực hiện for.
+ Biểu thức 2, 3 và thân for có thể thực hiện lặp lại nhiều lần.

Lưu ý:

+ **Biểu thức 1, 2, 3 phải phân cách bằng dấu chấm phẩy (;)**

+ Nếu biểu thức 2 không có, vòng for được xem là luôn luôn **đúng**. Muốn thoát khỏi vòng lặp for phải dùng một trong 3 lệnh **break**, **goto** hoặc **return**.

+ Với mỗi biểu thức có thể viết thành một dãy biểu thức con phân cách nhau bởi dấu phẩy. Khi đó các biểu thức con được xác định từ trái sang phải. Tính đúng sai của dãy biểu thức con trong biểu thức thứ 2 được xác định bởi biểu thức con cuối cùng.

+ Trong thân for (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.

+ Khi gặp lệnh **break**, cấu trúc lặp sâu nhất sẽ thoát ra.

+ Trong thân for có thể dùng lệnh **goto** để thoát khỏi vòng lặp đến vị trí mong muốn.

+ Trong thân for có thể sử dụng **return** để trở về một hàm nào đó.

+ Trong thân for có thể sử dụng lệnh continue để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

Ví dụ 1: Viết chương trình in ra câu "Vi du su dung vong lap for" 3 lần.

Dòng **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
1  /* Chuong trinh in ra cau "Vi du su dung vong lap for" 3 lan */
2
3  #include <stdio.h>
4  #include <conio.h>
5
6  #define MSG "Vi du su dung vong lap for.\n"
7
8  void main(void)
9  {
10    int i;
11    for(i = 1; i<=3; i++)          /hoac for(i = 1; i<=3; i+=1)
12       printf("%s", MSG);
13    getch();
14 }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Vi du su dung vong lap for.
Vi du su dung vong lap for.
Vi du su dung vong lap for.
_
```

Bạn thay 2 dòng 11 và 12 bằng câu lệnh
for(i=1; i<=3; i++, printf("%s", MSG));
Chạy lại chương trình, quan sát và nhận xét kết quả.

☞ **Có dấu chấm phẩy sau lệnh for(i=1; i<=3; i++); → các lệnh thuộc vòng lặp for sẽ không được thực hiện.**

Ví dụ 2: Viết chương trình nhập vào 3 số nguyên. Tính và in ra tổng của chúng.

Dòng **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
1  /* Chuong trinh nhap vao 3 so va tinh tong */
2
3  #include <stdio.h>
4  #include <conio.h>
5
6  void main(void)
7  {
8     int i, in, is;
9     is = 0;
10    for(i = 1; i<=3; i++)
11    {
```

```
12          printf("Nhap vao so thu %d :", i);
13          scanf("%d", &in);
14          is = is + in;
15      }
16      printf("Tong: %d", is);
17      getch();
18  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so thu 1: 5 Nhap vao so thu 2: 4 Nhap vao so thu 3: 2 Tong: 11. _	Bạn thay các dòng từ 9 đến 15 bằng câu lệnh: **for(is=0, i=1; i<=3; printf("Nhap vao so thu %d: ", i), scanf("%d", &in), i++, is=is+in);** Chạy lại chương trình, quan sát và nhận xét kết quả.

☞ **Trong vòng lặp for có sử dụng từ 2 lệnh trở lên, nhớ sử dụng cặp ngoặc { } để bọc các lệnh đó lại. Dòng 12, 13, 14 thuộc vòng for dòng 10 do được bọc bởi cặp ngoặc { }. Nếu 3 dòng này không bọc bởi cặp ngoặc { }, thì chỉ dòng 12 thuộc vòng lặp for, còn 2 dòng còn lại không thuộc vòng lặp for.**

Ví dụ 3: Viết chương trình nhập vào số nguyên n. Tính tổng các giá trị lẻ từ 0 đến n.

Dòng **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
1   /* Chuong trinh nhap vao 3 so va tinh tong */
2
3   #include <stdio.h>
4   #include <conio.h>
5
6   void main(void)
7   {
8       int i, in, is = 0;
9       printf("Nhap vao so n: ");
10      scanf("%d", &in);
11      is = 0;
12      for(i = 0; i<=in; i++)
13      {
14          if (i % 2 != 0)          //neu i la so le
15              is = is + i;         //hoac is += i;
16      }
17      printf("Tong: %d", is);
18      getch();
19  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so n : 5 Tong: 9. _	Bạn thay các dòng từ 11 đến 16 bằng câu lệnh: **for(is=0, i=1; i<=n; is=is+i, i+=2);** Chạy lại chương trình, quan sát và nhận xét kết quả.

☞ **Bạn có thể viết gộp các lệnh trong thân for vào trong lệnh for. Tuy nhiên, khi lập trình bạn nên viết lệnh for có đủ 3 biểu thức đơn và các lệnh thực hiện trong thân for mỗi lệnh một dòng để sau này có thể đọc lại dễ hiểu, dễ sửa chữa.**

Ví dụ 4: Một vài ví dụ thay đổi biến điều khiển vòng lặp.

- Thay đổi biến điều khiển từ 1 đến 100, mỗi lần tăng 1:
  for(i = 1; i <= 100; i++)
- Thay đổi biến điều khiển từ 100 đến 1, mỗi lần giảm 1:
  for(i = 100; i >= 1; i--)
- Thay đổi biến điều khiển từ 7 đến 77, mỗi lần tăng 7:
  for(i = 7; i <= 77; i += 7)
- Thay đổi biến điều khiển từ 20 đến 2, mỗi lần giảm 2:
  for(i = 20; i >= 2; i –= 2)

Ví dụ 5: Đọc vào một loạt kí tự trên bàn phím. Kết thúc khi gặp dấu chấm '.' .

Dòng | File Edit Search Run Compile Debug Project Option Window Help

```
1  /* Doc vao 1 loat ktu tren ban phim. Ket thuc khi gap dau cham */
2
3  #include <stdio.h>
4
5  #define  DAU_CHAM    '.'
6
7  void main(void)
8  {
9     char c;
10    for(; (c = getchar()) != DAU_CHAM; )
11       putchar(c);
12 }
```

F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt - F9 Compile F9 Make F10 Menu

☞ ***Kết quả in ra màn hình***

```
a
a
4
4
.
_
```

Bạn thay các dòng từ 10 đến 11 bằng câu lệnh:
for(; (c = getchar()) != DAU_CHAM; putchar(c));
Chạy lại chương trình, quan sát và nhận xét kết quả.

☞ Vòng lặp for vắng mặt biểu thức 1 và 3.

Ví dụ 6: Đọc vào một loạt kí tự trên bàn phím, đếm số kí tự nhập vào. Kết thúc khi gặp dấu chấm '.' .

Dòng | File Edit Search Run Cmpile Debug Project Option Window Help

```
1  /* Doc vao 1 loat ktu tren ban phim, dem so ktu nhap vao. Ket thuc khi gap dau cham */
2
3  #include <stdio.h>
4  #include <conio.h>
5
6  #define  DAU_CHAM    '.'
7
8  void main(void)
9  {
10    char c;
11    int idem;
12    for(idem = 0; (c = getchar()) != DAU_CHAM; )
13       idem++;
```

```
14      printf("So ki tu: %d.\n", idem);
15      getch();
16  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

afser. So ki tu: 5. _	Bạn thay các dòng từ 12 đến 13 bằng câu lệnh: **for(idem = 0; (c = getchar()) != DAU_CHAM; idem++);** Chạy lại chương trình, quan sát và nhận xét kết quả.

☞ Vòng lặp for vắng mặt biểu thức 3.

Ví dụ 7: Đọc vào một loạt kí tự trên bàn phím, đếm số kí tự nhập vào. Kết thúc khi gặp dấu chấm '.' .

```
1   /* Doc vao 1 loat ktu tren ban phim, dem so ktu nhap vao. Ket thuc khi gap dau cham */
2
3   #include <stdio.h>
4   #include <conio.h>
5
6   #define  DAU_CHAM     '.'
7
8   void main(void)
9   {
10      char c;
11      int idem = 0;
12      for(; ;)
13      {
14          c = getchar();
15          if (c == DAU_CHAM)      //nhap vao dau cham
16              break;              //thoat vong lap
17          idem++;
18      }
19      printf("So ki tu: %d.\n", idem);
20      getch();
21  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

afser. So ki tu: 5. _	Chạy lại chương trình, quan sát và nhận xét kết quả.

☞ Vòng lặp for vắng mặt cả ba biểu thức.

Ví dụ 8: Nhập vào 1 dãy số nguyên từ bàn phím đến khi gặp số 0 thì dừng. In ra tổng các số nguyên dương.

```
1   /* Nhap vao 1 day so nguyen tu ban phim den khi gap so 0 thi dung. In ra tong cac so
2   nguyen duong */
3
4   #include <stdio.h>
5   #include <conio.h>
```

```
 6
 7  void main(void)
 8  {
 9     int in, itong = 0;
10     for(; ;)
11     {
12        printf("Nhap vao 1 so nguyen: ");
13        scanf("%d", &in);
14        if (in < 0)
15           continue;       //in < 0 quay nguoc len dau vong lap
16        if (in == 0)
17           break;          //in = 0 thoat vong lap
18        itong += in;
19     }
20     printf("Tong: %d.\n", itong);
21     getch();
22  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao 1 so nguyen: -8 Nhap vao 1 so nguyen: 9 Nhap vao 1 so nguyen: -7 Nhap vao 1 so nguyen: 3 Nhap vao 1 so nguyen: 0 Tong: 12 _	Chạy lại chương trình với số liệu khác Quan sát và nhận xét kết quả.

6.2.2 Lệnh break

Thông thường lệnh break dùng để thoát khỏi vòng lặp không xác định điều kiện dừng hoặc bạn muốn dừng vòng lặp theo điều kiện do bạn chỉ định. Việc dùng lệnh break để thoát khỏi vòng lặp thường sử dụng phối hợp với lệnh if. Lệnh break dùng trong for, while, do…while, switch. Lệnh break thoát khỏi vòng lặp chứa nó.

Ví dụ 9 : Như ví dụ 7, 8

Sử dụng lệnh break trong switch để nhảy bỏ các câu lệnh kế tiếp còn lại.

6.2.3 Lệnh continue

Được dùng trong vòng lặp for, while, do…while. Khi lệnh continue thi hành quyền điều khiển sẽ trao qua cho biểu thức điều kiện của vòng lặp gần nhất. Nghĩa là lộn ngược lên đầu vòng lặp, tất cả những lệnh đi sau trong vòng lặp chứa continue sẽ bị bỏ qua không thi hành.

Ví dụ 10 : Như ví dụ 8

6.2.4 Lệnh while

Vòng lặp thực hiện lặp lại trong khi biểu thức còn đúng.

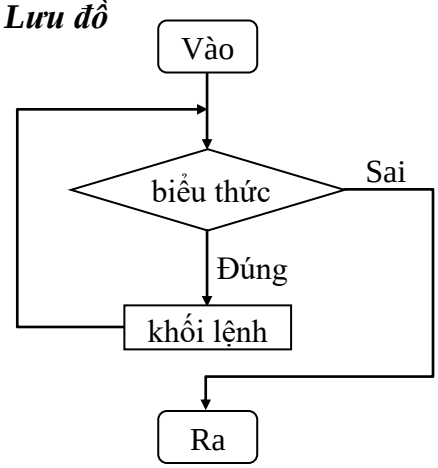
- ***Cú pháp lệnh***

```
while (biểu thức)
   khối lệnh;
```

☞ từ khóa **while** phải viết bằng chữ thường

✍ Nếu **khối lệnh** bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

- ***Lưu đồ***



☞ Trước tiên **biểu thức** được kiểm tra
nếu **sai** thì
kết thúc vòng lặp while
(khối lệnh không được thi hành 1 lần nào)
nếu **đúng**
thực hiện khối lệnh;
lặp lại kiểm tra biểu thức

+ Biểu thức: có thể là một biểu thức hoặc nhiều biểu thức con. Nếu là nhiều biểu thức con thì cách nhau bởi dấu phẩy (,) và tính đúng sai của biểu thức được quyết định bởi biểu thức con cuối cùng.

+ Trong thân while (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.

+ Trong thân while có thể sử dụng lệnh continue để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

+ Muốn thoát khỏi vòng lặp while tùy ý có thể dùng các lệnh **break, goto, return** như lệnh **for.**

Ví dụ 11: Viết chương trình in ra câu "Vi du su dung vong lap while" 3 lần.

Dòng	File Edit Search Run Compile Debug Project Option Window Help
1	/* Chuong trinh in ra cau "Vi du su dung vong lap while" 3 lan */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	
6	#define MSG "Vi du su dung vong lap while.\n"
7	
8	void main(void)
9	{
10	int i = 0;
11	while (i++ < 3)
12	printf("%s", MSG);
13	getch();
14	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Vi du su dung vong lap while. Vi du su dung vong lap while. Vi du su dung vong lap while. _	Bạn thay 2 dòng 11 và 12 bằng câu lệnh **while(printf("%s", MSG), ++i < 3);** Chạy lại chương trình và quan sát kết quả.

Ví dụ 12: Viết chương trình tính tổng các số nguyên từ 1 đến n, với n được nhập vào từ bàn phím.

Dòng	File Edit Search Run Compile Debug Project Option Window Help
1	/* Chuong trinh tính tong cac so nguyen tu 1 den n */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	

```
 6  void main(void)
 7  {
 8      int i = 0, in, is = 0;
 9      printf("Nhap vao so n: ");
10      scanf("%d", &in);
11      while (i++ < in)
12          is = is + i;            //hoac is += i;
13      printf("Tong: %d", is);
14      getch();
15  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so n : 5 Tong: 15. _	Bạn thay các dòng từ 11 đến 12 bằng câu lệnh: **while(is = is+i, i++ < in);** Chạy lại chương trình, quan sát và nhận xét kết quả.

Ví dụ 13: Thay dòng **for(; (c = getchar()) != DAU_CHAM;)** ở **ví dụ 5** thành dòng **while ((c = getchar()) != DAU_CHAM)**

☞ ***Chạy lại chương trình, quan sát và nhận xét kết quả.***

Ví dụ 14: Ở **ví dụ 6**, thay dòng **int dem;** thành dòng **int dem = 0**; , thay dòng **for(idem=0; (c = getchar()) != DAU_CHAM;)** thành dòng **while ((c = getchar()) != DAU_CHAM)**

☞ ***Chạy lại chương trình, quan sát và nhận xét kết quả.***

Ví dụ 15: Ở **ví dụ 7 và 8**, thay dòng **for(; ;)** thành dòng **while(1)**

☞ ***Chạy lại chương trình, quan sát và nhận xét kết quả.***

6.2.5 Lệnh do…while

Vòng lặp thực hiện lặp lại cho đến khi biểu thức sai.

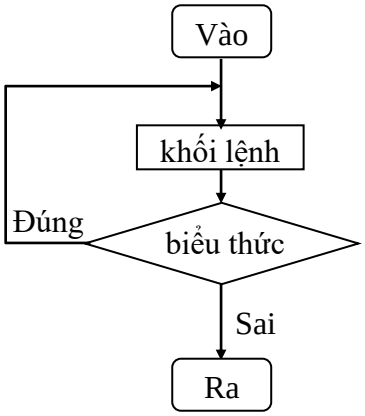
- ***Cú pháp lệnh***

```
do
    khối lệnh;
while (biểu thức);
```

☞ từ khóa **do, while** phải viết bằng chữ thường

✍ Nếu **khối lệnh** bao gồm từ 2 lệnh trở lên thì phải đặt trong dấu { }

- ***Lưu đồ***



☞ Thực hiện
khối lệnh
Kiểm tra biểu thức
Nếu **đúng** thì
lặp lại thực hiện khối lệnh
Nếu **sai** thì
kết thúc vòng lặp
(khối lệnh được thi hành 1 lần)

+ Biểu thức: có thể là một biểu thức hoặc nhiều biểu thức con. Nếu là nhiều biểu thức con thì cách nhau bởi dấu phẩy (,) và tính đúng sai của biểu thức được quyết định bởi biểu thức con cuối cùng.

+ Trong thân do…while (khối lệnh) có thể chứa một hoặc nhiều cấu trúc điều khiển khác.

+ Trong thân do…while có thể sử dụng lệnh continue để chuyển đến đầu vòng lặp (bỏ qua các câu lệnh còn lại trong thân).

+ Muốn thoát khỏi vòng lặp do…while tùy ý có thể dùng các lệnh **break, goto, return**.

Ví dụ 16: Viết chương trình kiểm tra password.

Dòng **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
1  /* Chuong trinh kiem tra mat khau */
2
3  #include <stdio.h>
4
5  # define PASSWORD    12345
6
7  void main(void)
8  {
9     int in;
10    do
11    {
12       printf("Nhap vao password: ");
13       scanf("%d", &in);
14    } while (in != PASSWORD)
15 }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao password: 1123
Nhap vao password: 12346
Nhap vao password: 12345
```

Bạn thay các dòng từ 10 đến 14 bằng câu lệnh:
do{}while(printf("Nhap vao password: "), scanf("%d", &in), in != PASSWORD);
Chạy lại chương trình và quan sát kết quả.

Ví dụ 17: Viết chương trình nhập vào năm hiện tại, năm sinh. In ra tuoi.

Dòng **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
1  /* Chuong trinh in tuoi */
2
3  #include <stdio.h>
4
5  # define CHUC      "Chuc ban vui ve (: >\n"
6
7  void main(void)
8  {
9     unsigned char choi;
10    int inamhtai, inamsinh;
11    do
12    {
13       printf("Nhap vao nam hien tai: ");
14       scanf("%d", inamhtai);
15       printf("Nhap vao nam sinh: ");
16       scanf("%d", inamsinh);
```

```
17          printf("Ban %d tuoi, %s", inamhtai – inamsinh, CHUC);
18          printf("Ban co muon tiep tuc? (Y/N)\n");
19          choi = getch();
20      } while (choi == 'y' || choi == 'Y');
21  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao nam hien tai: 2002 Nhap vao nam sinh: 1980 Ban 22 tuoi, chuc ban vui ve (:> Ban co muon tiep tuc? (Y/N) _ (nếu gõ y hoặc Y tiếp tục thực hiện chương trình, ngược lại gõ các phím khác chương trình sẽ thoát)	Bạn lại chương trình với số liệu khác. Quan sát, đánh giá và nhận xét kết quả.

6.2.6 Vòng lặp lồng nhau

Ví dụ 18: Vẽ hình chữ nhật đặc bằng các dấu '*'

```
1   /* Ve hinh chu nhat dac */
2
3   #include <stdio.h>
4   #include <conio.h>
5
6   void main(void)
7   {
8       int i, ij, idai, irong;
9       printf("Nhap vao chieu dai: ");
10      scanf("%d", &idai);
11      printf("Nhap vao chieu rong: ");
12      scanf("%d", &irong);
13      for (i = 1; i <= irong; i++)
14      {
15          for (ij = 1; ij <= idai; ij++)     //in mot hang voi chieu dai dau *
16              printf("*");
17          printf("\n");                      //xuong dong khi in xong 1 hang
18      }
19      getch();
20  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao chieu dai: 10 Nhap vao chieu rong: 5 * _	Bạn lại chương trình với số liệu khác. Quan sát, đánh giá và nhận xét kết quả.

Ví dụ 19: Vẽ hình chữ nhật đặc có chiều rộng = 10 hàng. Hàng thứ 1 = 10 số 0, hàng thứ 2 = 10 số 1…

Dòng | File Edit Search Run Compile Debug Project Option Window Help

```
1  /* Ve hinh chu nhat bang cac so tu 0 den 9 */
2
3  #include <stdio.h>
4  #include <conio.h>
5
6  void main(void)
7  {
8     int i = 0, ij;
9     while (i <= 9)
10    {
11       ij = 0;                    //khoi tao lai ij = 0 cho lan in ke tiep
12       while (ij++ <= 9)          //in 1 hang 10 so i
13          printf("%d", i);
14       printf("\n");              //xuong dong khi in xong 1 hang
15       i++;                       //tang i len 1 cho vong lap ke tiep
16    }
17    getch();
18 }
19
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9
_
```

Thay dòng 11, 12 thành câu lệnh
for (ij = 0; ij <= 9; ij++)
Chạy lại chương trình.
Quan sát, đánh giá và nhận xét kết quả.

☞ **Các lệnh lặp for, while, do…while có thể lồng vào chính nó, hoặc lồng vào lẫn nhau. Nếu không cần thiết không nên lồng vào nhiều cấp dễ gây nhầm lẫn khi lập trình cũng như kiểm soát chương trình.**

6.2.7 So sánh sự khác nhau của các vòng lặp

- Vòng lặp for thường sử dụng khi biết được số lần lặp xác định.
- Vòng lặp thường while, do…while sử dụng khi không biết rõ số lần lặp.
- Khi gọi vòng lặp while, do…while, nếu biểu thức sai vòng lặp while sẽ không được thực hiện lần nào nhưng vòng lặp do…while thực hiện được 1 lần.

☞ **Số lần thực hiện ít nhất của while là 0 và của do…while là 1**

6.3 Bài tập

1. ***Viết chương trình in ra bảng mã ASCII***
2. ***Viết chương trình tính tổng bậc 3 của N số nguyên đầu tiên.***

3. Viết chương trình nhập vào một số nguyên rồi in ra tất cả các ước số của số đó.

4. Viết chương trình vẽ một tam giác cân bằng các dấu *

5. Viết chương trình tính tổng nghịch đảo của N số nguyên đầu tiên theo công thức

$S = 1 + 1/2 + 1/3 + \ldots + 1/N$

6. Viết chương trình tính tổng bình phương các số lẻ từ 1 đến N.

7. Viết chương trình nhập vào N số nguyên, tìm số lớn nhất, số nhỏ nhất.

8. Viết chương trình nhập vào N rồi tính giai thừa của N.

9. Viết chương trình tìm USCLN, BSCNN của 2 số.

10. Viết chương trình vẽ một tam giác cân rỗng bằng các dấu *.

11. Viết chương trình vẽ hình chữ nhật rỗng bằng các dấu *.

12. Viết chương trình nhập vào một số và kiểm tra xem số đó có phải là số nguyên tố hay không?

13. Viết chương trình tính số hạng thứ n của dãy Fibonaci.

Dãy Fibonaci là dãy số gồm các số hạng p(n) với:
p(n) = p(n-1) + p(n-2) với n>2 và p(1) = p(2) = 1
Dãy Fibonaci sẽ là: 1 1 2 3 5 8 13 21 34 55 89 144…

14. Viết chương trình tính giá trị của đa thức

$P_n = a_nx^n + a_{n-1}x^{n-1} + \ldots + a_1x^1 + a_0$

Hướng dẫn đa thức có thể viết lại

$P_n = (\ldots(a_nx + a_{n-1})x + a_{n-2})x + \ldots + a_0$

Như vậy trước tiên tính $a_nx + a_{n-1,}$ lấy kết quả nhân với x, sau đó lấy kết quả nhân với x cộng thêm a_{n-2}, lấy kết quả nhân với x … n gọi là bậc của đa thức.

15. Viết chương trình tính xn với x, n được nhập vào từ bàn phím.

16. Viết chương trình nhập vào 1 số từ 0 đến 9. In ra chữ số tương ứng. Ví dụ: nhập vào số 5, in ra "Năm".

17. Viết chương trình phân tích một số nguyên N thành tích của các thừa số nguyên tố.

18. Viết chương trình lặp lại nhiều lần công việc nhập một ký tự và in ra mã ASCII của ký tự đó, khi nào nhập số 0 thì dừng.

19. Viết chương trình tìm ước số chung lớn nhất và bội số chung nhỏ nhất của 2 số nguyên.

20. Viết chương trình in lá cờ nước Mỹ.

21. Viết chương trình tính dân số của một thành phố sau 10 năm nữa, biết rằng dân số hiện nay là 6.000.000, tỉ lệ tăng dân số hàng năm là 1.8% .

22. Viết chương trình tìm các số nguyên gồm 3 chữ số sao cho tích của 3 chữ số bằng tổng 3 chữ số. Ví dụ: 1*2*3 = 1+2+3.

23. Viết chương trình tìm các số nguyên a, b, c, d khác nhau trong khoảng từ 0 tới 10 thỏa mãn điều kiện a*d*d = b*c*c*c

24. Viết chương trình tính tổ hợp N chập K (với K <= N)

C=((N-k+1) * (N-k+2)*…N)/1*2*3*…*k

Trong đó C là một tích gồm k phần tử với phần tử thứ I là (N-k+1)/I. Để viết chương trình này, bạn dùng vòng lặp For với biến điều khiển I từ giá trị đầu là 1 tăng đến giá trị cuối là k kết hợp với việc nhân dồn vào kết quả C.

25. Viết chương trình giải bài toán cổ điển sau:

Trăm trâu, trăm cỏ
Trâu đứng ăn năm
Trâu nằm ăn ba,
Ba trâu già ăn một
Hỏi mỗi loại trâu có bao nhiêu con.

26. Viết chương trình giải bài toán cổ điển sau:

Vừa gà vừa chó 36 con
Bó lại cho tròn, đếm đủ 100 chân
Hỏi có bao nhiêu gà, bao nhiêu chó

27. Viết chương trình in ra bảng cửu chương

28. Viết chương trình xác định xem một tờ giấy có độ dày 0.1 mm. Phải gấp đôi tờ giấy bao nhiêu lần để nó có độ dày 1m.

29. Viết chương trình tìm các số nguyên tố từ 2 đến N, với N được nhập vào.

30. Viết chương trình lặp đi lặp lại các công việc sau:

- Nhập vào một ký tự trên bàn phím.
- Nếu là chữ thường thì in ra chính nó và chữ HOA tương ứng.
- Nếu là chữ HOA thì in ra chính nó và chữ thường tường ứng.
- Nếu là ký số thì in ra chính nó.
- Nếu là một ký tự điều khiển thì kết thúc chương trình

31. Viết chương trình nhập vào x, n tính:

- $\sqrt{x+\sqrt{x+...+\sqrt{x}}}$ (n dấu căn)
- $1+\frac{x}{2}+...\frac{x^n}{n+1}$

32. Viết chương trình nhập vào N số nguyên, đếm xem có bao nhiêu số âm, bao nhiêu số dương và bao nhiêu số không.

Bài 7 :

HÀM

7.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:

- Khái niệm, cách khai báo về hàm.
- Cách truyền tham số, tham biến, tham trị.
- Sử dụng biến cục bộ, toàn cục trong hàm.
- Sử dụng tiền xử lý #define

7.2 Nội dung

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.

7.2.1 Các ví dụ về hàm

Ví dụ 1:

Dòng | **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
 1  #include <stdio.h>
 2  #include <conio.h>
 3
 4  // khai bao prototype
 5  void line();
 6
 7  // ham in 1 dong dau
 8  void line()
 9  {
10      int i;
11      for(i = 0; i < 19; i++)
12          printf("*");
13      printf("\n");
14  }
15
16  void main(void)
17  {
18      line();
19      printf("* Minh hoa ve ham *");
20      line();
21      getch();
22  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
******************
* Minh hoa ve ham *
******************

_
```

Giải thích chương trình

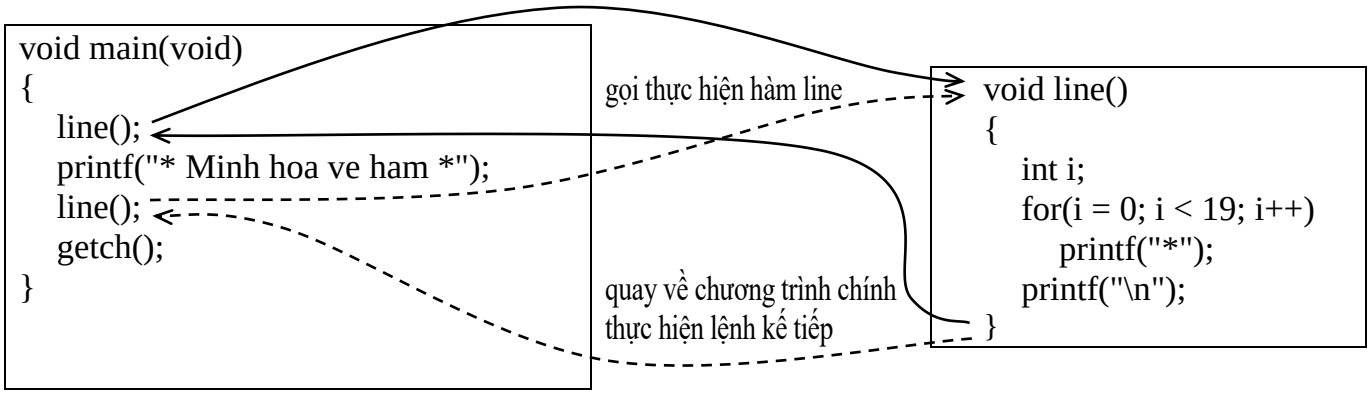
Dòng 8 đến dòng 14: định nghĩa hàm **line**, hàm này không trả về giá trị, thực hiện công việc in ra 19 dấu sao.

Dòng 5: khai báo prototype, sau tên hàm phải có dấu chầm phẩy

Trong hàm line có sử dụng biến i, biến i là biến cục bộ chỉ sử dụng được trong phạm vi hàm line.

Dòng 18 và 20: gọi thực hiện hàm line.

**** Trình tự thực hiện chương trình***



☹ **Không có dấu chấm phẩy sau tên hàm, phải có cặp dấu ngoặc ( ) sau tên hàm nếu hàm không có tham số truyền vào. Phải có dấu chấm phẩy sau tên hàm khai báo prototype. Nên khai báo prototype cho dù hàm được gọi nằm trước hay sau câu lệnh gọi nó.**

Ví dụ 2:

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	#include <stdio.h>
2	#include <conio.h>
3	
4	// khai bao prototype
5	int power(int, int);
6	
7	// ham tinh so mu
8	int power(int ix, int in)
9	{
10	int i, ip = 1;
11	for(i = 1; i <= in; i++)
12	ip *= ix;
13	return ip;
14	}
15	
16	void main(void)
17	{
18	printf("2 mu 2 = %d.\n", power(2, 2));
19	printf("2 mu 3 = %d.\n", power(2, 3));
20	getch();
	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

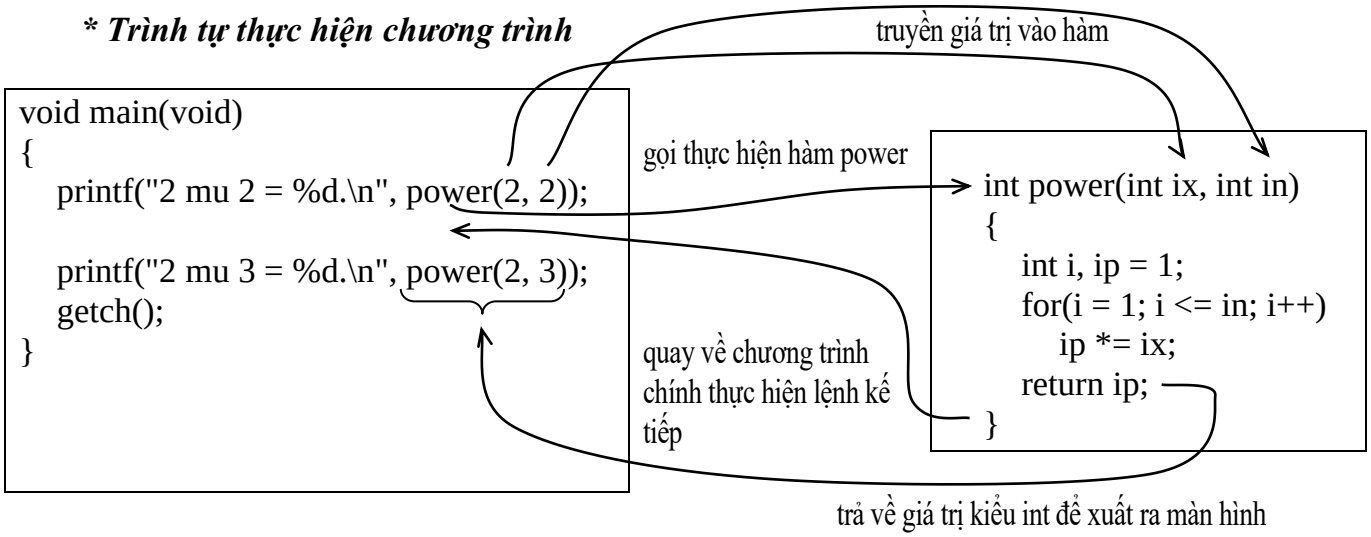
☞ ***Kết quả in ra màn hình***

```
2 mu 2 = 4.
2 mu 3 = 8.
_
```

✍ ***Giải thích chương trình***

Hàm **power** có hai tham số truyền vào là ix, in có kiểu int và kiểu trả về cũng có kiểu int.
Dòng 13: return ip: trả về giá trị sau khi tính toán
Dòng 18: đối mục 2 và 3 có kiểu trả về là int sau khi thực hiện gọi power.
Hai tham số ix, in của hàm power là dạng truyền tham trị.

**** Trình tự thực hiện chương trình***



☞ **Quy tắc đặt tên hàm giống tên biến, hằng… Mỗi đối số cách nhau = dấu phẩy kèm theo kiểu dữ liệu tương ứng.**

Ví dụ 3:

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	#include <stdio.h>
2	#include <conio.h>
3	
4	// khai bao prototype
5	void time(int & , int &);
6	
7	// ham doi phut thanh gio:phut
8	void time(int &ig, int &ip)
9	{
10	ig = ip / 60;
11	ip %= 60;
12	}
13	
14	void main(void)
15	{
16	int igio, iphut;

```
17      printf("Nhap vao so phut : ");
18      scanf("%d", &iphut);
19      time(igio, iphut);
20      printf("%02d:%02d\n", igio, iphut);
21      getch();
22  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

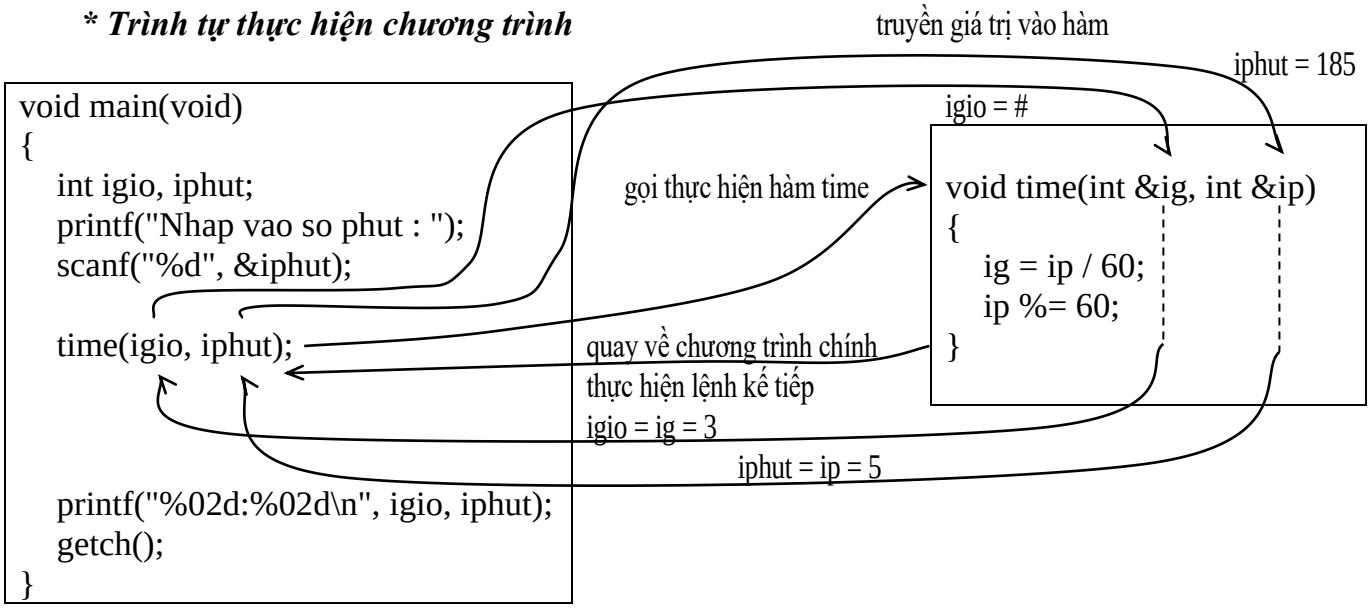
☞ ***Kết quả in ra màn hình***

```
Nhap vao so phut: 185
03:05
_
```

✍ ***Giải thích chương trình***

Hàm **time** có hai tham số truyền vào là ix, in có kiểu int. 2 tham số này có toán tử địa chỉ & đi trước cho biết 2 tham số này là dạng truyền tham biến.

**** Trình tự thực hiện chương trình***



7.2.2 Tham số dạng tham biến và tham trị

Ví dụ 4:

```
void thamtri(int ix, int iy)
{
    ix += 1;    //cong ix them 1
    iy += 1;    //cong iy them 1
}
void thambien(int &ix, int &iy)
{
    ix += 1;    //cong ix them 1
    iy += 1;    //cong iy them 1
}
```

```
void main(void)
{
    int ia = 5, ib = 5;
    thamtri(ia, ib);
    printf("a = %d, b = %d", ia, ib);
    thambien(ia, ib);
    printf("a = %d, b = %d", ai, ib);
}
```

Kết quả in ra:

```
a = 5, b = 5

a = 6, b = 6
```

☞ ***Đối với hàm sử dụng lệnh return bạn chỉ có thể trả về duy nhất 1 giá trị mà thôi. Để có thể trả về nhiều giá trị sau khi gọi hàm bạn sử dụng hàm truyền nhiều tham số dạng tham biến.***

7.2.3 Sử dụng biến toàn cục

Ví dụ 5:

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	#include <stdio.h>
2	#include <conio.h>
3	
4	// khai bao prototype
5	void oddeven();
6	void negative();
7	
8	//khai bao bien toan cuc
9	int inum;
10	
11	void main(void)
12	{
13	printf("Nhap vao 1 so nguyen : ");
14	scanf("%d", &inum);
15	oddeven();
16	negative();
17	getch();
18	}
19	
20	// ham kiem tra chan le
21	void oddeven()
22	{
23	if (inum % 2)
24	printf("%d la so le.\n", inum);
25	else
26	printf("%d la so chan.\n", inum);
27	}
28	
29	//ham kiem tra so am
30	void negative()
31	{
32	if (inum < 0)
33	printf("%d la so am.\n", inum);
34	else
35	printf("%d la so duong.\n", inum);
36	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao 1 so nguyen: 3
3 la so le.
3 la so duong.
_
```

Giải thích chương trình

Chương trình trên gồm 2 hàm **oddeven** và **negative**, 2 hàm này bạn thấy không có tham số để truyền biến inum vào xử lý nhưng vẫn cho kết quả đúng. Do chương trình sử dụng biến **inum** toàn cục (dòng.9) nên biến này có ảnh hưởng đến toàn bộ chương trình mỗi khi gọi và sử dụng nó. Xét tình huống sau: Giả sử trong hàm negative ta khai báo biến inum có kiểu int như sau:

```
void negative()
{
   int inum;
   ….
}
```

Khi đó chương trình sẽ cho kết quả sai! Do các câu lệnh trong hàm negative sử dụng biến inum sẽ sử dụng biến inum khai báo trong hàm negative và lúc này biến inum toàn cục không có tác dụng đối với các câu lệnh trong hàm này. Biến inum khai báo trong hàm negative chỉ có ảnh hưởng trong phạm vi hàm và chu trình sống của nó bắt đầu từ lúc gọi hàm đến khi thực hiện xong.

Cẩn thận khi đặt tên biến, xác định rõ phạm vi của biến khi sử dụng để có thể dễ dàng kiểm soát chương trình.

Ví dụ 6:

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	#include <stdio.h>
2	#include <conio.h>
3	
4	#define PI 3.14
5	
6	// khai bao prototype
7	float area();
8	
9	//khai bao bien toan cuc
10	float frad;
11	
12	void main(void)
13	{
14	printf("Nhap vao ban kinh hinh cau : ");
15	scanf("%f", &frad);
16	printf("Dien tich hinh cau: %10.3f.\n", area());
17	getch();
18	}
19	
20	// ham tinh dien tich hinh cau
21	float area()
22	{
23	return (4*PI*frad*frad);
24	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

Kết quả in ra màn hình

```
Nhap vao ban kinh hinh cau: 3.2
Dien tich hinh cau:   128.614
_
```

7.2.4 Dùng dẫn hướng #define

Sau đây là một vài ví dụ dùng dẫn hướng #define để định nghĩa hàm đơn giản

```
#define AREA_CIRCLE (frad)   (4*PI*frad*frad)    //tinh dien tich hinh cau
#define SUM (x, y)           (x + y)             //cong 2 so
#define SQR (x)              (x*x)               //tinh x binh phuong
#define MAX(x, y)            (x > y) ? x : y     //tim so lon nhat giua x va y
#define ERROR (s)            printf("%s.\n", s)  //in thong bao voi chuoi s
```

Ví dụ 7: Trong ví dụ 6 xóa từ dòng 20 đến dòng 24, xóa dòng 6, 7; thêm dòng **AREA_CIRCLE (frad) (4*PI+frad*frad)** vào sau dòng 5.

Sửa dòng **printf("Dien tich hinh cau: %10.3f.\n", area());** thành **printf("Dien tich hinh cau: %10.3f.\n", AREA_CIRCLE(frad));**

Chạy lại chương trình, quan sát và nhận xét kết quả.

Ví dụ 8:

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	#include <stdio.h>
2	#include <conio.h>
3	
4	#define MAX(x, y) (x > y) ? x : y
5	
6	void main(void)
7	{
8	float a = 4.5, b = 6.1;
9	printf("So lon nhat la: %5.2f.\n", MAX(a, b));
10	getch();
11	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

So lon nhat la: 6.10 –	Thêm vào dòng 8 giá trị c = 10 Sửa lại dòng 9: **MAX(a, b)** thành **MAX(MAX(a, b), c)** Chạy lại chương trình, quan sát và nhận xét kết quả

7.3 Bài tập

1. ***Viết hàm tính n!***
2. ***Viết hàm tính tổng S = 1+2+….+n.***
3. ***Viết hàm kiểm tra số nguyên tố.***
4. ***Viết hàm tính số hạng thứ n trong dãy Fibonacci.***
5. ***Viết hàm tìm số lớn nhất trong 2 số.***

Bài 8 :

MẢNG VÀ CHUỖI

8.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:

- Ý nghĩa, cách khai báo mảng, chuỗi.
- Nhập, xuất mảng, chuỗi.
- Khởi tạo mảng chuỗi.
- Một số kỹ thuật thao tác trên mảng, chuỗi.
- Dùng mảng làm tham số cho hàm.
- Một số hàm xử lý chuỗi

8.2 Nội dung

8.2.1 Mảng

Là tập hợp các phần tử có cùng dữ liệu. Giả sử bạn muốn lưu n số nguyên để tính trung bình, bạn không thể khai báo n biến để lưu n giá trị rồi sau đó tính trung bình.

Ví dụ 1 : bạn muốn tính trung bình 10 số nguyên nhập vào từ bàn phím, bạn sẽ khai báo 10 biến: a, b, c, d, e, f, g, h, i, j có kiểu int và lập thao tác nhập cho 10 biến này như sau:

printf("Nhap vao bien a: ");
scanf("%d", &a);

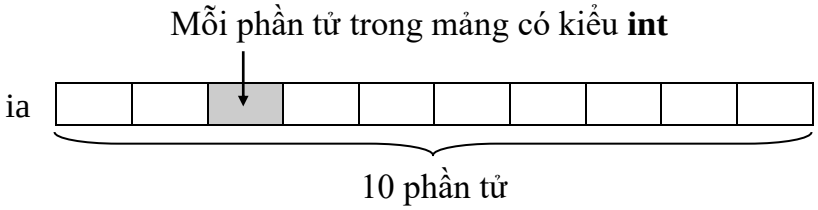
10 biến bạn sẽ thực hiện 2 lệnh trên 10 lần, sau đó tính trung bình:

(a + b + c + d + e + f + g + h + i + j)/10

☞ Điều này chỉ phù hợp với n nhỏ, còn đối với n lớn thì khó có thể thực hiện được. Vì vậy khái niệm mảng được sử dụng

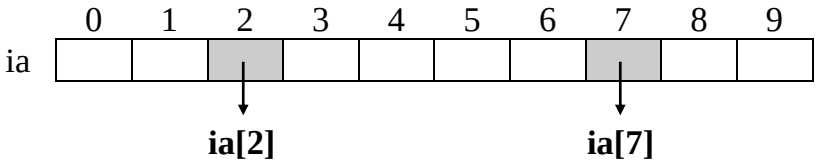
8.2.1.1 Cách khai báo mảng

Ví dụ 2 : **int ia[10];** với **int** là kiểu mảng, ia là tên mảng, 10 số phần tử mảng
Ý nghĩa: ***Khai báo một mảng số nguyên gồm 10 phần tử, mỗi phần tử có kiểu int***.



8.2.1.2 Tham chiếu đến từng phần tử mảng

Sau khi mảng được khai báo, mỗi phần tử trong mảng đều có chỉ số để tham chiếu. Chỉ số bắt đầu từ 0 đến n-1 (với n là kích thước mảng). Trong ví dụ trên, ta khai báo mảng 10 phần tử thì chỉ số bắt đầu từ 0 đến 9.



ia[2], ia[7]… là phần tử thứ 3, 8… trong mảng xem như là một biến kiểu **int**.

8.2.1.3 Nhập dữ liệu cho mảng

```
for (i = 0; i < 10; i++)        //vòng for có giá trị i chạy từ 0 đến 9
{
    printf("Nhap vao phan tu thu %d: ", i + 1);
    scanf("%d", &ia[i]);
}
```

8.2.1.4 Đọc dữ liệu từ mảng

```
for(i = 0; i < 10; i++)
    printf("%3d ", ia[i]);
```

Ví dụ 3 : Viết chương trình nhập vào n số nguyên. Tính và in ra trung bình cộng.

Dòng | **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
1  /* Tinh trung binh cong n so nguyen */
2
3  #include <stdio.h>
4  #include <conio.h>
5
6  void main(void)
7  {
8     int ia[50], i, in, isum = 0;
9     printf("Nhap vao gia tri n: ");
10    scanf("%d", &in);
11
12    //Nhap du lieu vao mang
13    for(i = 0; i < in; i++)
14    {
15       printf("Nhap vao phan tu thu %d: ", i + 1);
16       scanf("%d", &ia[i]);      //Nhap gia tri cho phan tu thu i
17    }
18
19    //Tinh tong gia tri cac phan tu
20    for(i = 0; i < in; i++)
21       isum += ia[i];              //cong don tung phan tu vao isum
22
23    printf("Trung binh cong: %.2f\n", (float) isum/in);
24    getch();
25 }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao gia tri n: 3
Nhap vao phan tu thu 1: 7
Nhap vao phan tu thu 2: 3
Nhap vao phan tu thu 3: 6
Trung binh cong: 5.33
_
```

Bạn có thể gộp 2 lệnh for thành một vừa nhập vừa tính tổng, đưa hàng 21 sau hàng 16 và bỏ các hàng 19, 20, 21.

Chạy và quan sát kết quả.

☹ Điều gì sẽ xảy ra cho đoạn chương trình trên nếu bạn nhập n > 50 trong khi bạn chỉ khai báo mảng ia tối đa là 50 phần tử. Bạn dùng lệnh if để ngăn chặn điều này trước khi vào thực hiện lệnh for. Thay dòng 9, 10 bằng đoạn lệnh sau :

```
do
{
   printf("Nhap vao gia tri n: ");
   scanf("%d", &in);
} while (in <= 0 || in > 50);        //chi chap nhan gia tri nhap vao trong khoang 1..50
```

☞ Chạy chương trình và nhập n với các giá trị -6, 0, 51, 6. Quan sát kết quả.

8.2.1.5 Sử dụng biến kiểu khác

Ngoài kiểu int, bạn có thể khai báo mảng kiểu char, float, double…

Ví dụ 4 : char cloai[20]; float ftemp[10]; cách tham chiếu, nhập dữ liệu, đọc dữ liệu như trên.

8.2.1.6 Kỹ thuật Sentinal

Sử dụng kỹ thuật này để nhập liệu giá trị cho các phần tử mảng mà không biết rõ số lượng phần tử sẽ nhập vào là bao nhiêu (không biết số n).

Ví dụ 5 : Viết chương trình nhập vào 1 dãy số dương rồi in tổng các số dương đó.

Phác họa lời giải: Chương trình yêu cầu nhập vào dãy số dương mà không biết trước số lượng phần tử cần nhập là bao nhiêu, vì vậy để chấm dứt nhập liệu khi thỏa mãn bằng cách nhập vào số âm hoặc không.

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Nhap vao day so nguyen duong, in ra day chan, day le */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	#define MAX 50
6	
7	void main(void)
8	{
9	float fa[MAX], fsum = 0;
10	int i = 0;
11	do
12	{
13	printf("Nhap vao phan tu thu %d: ", i + 1);
14	scanf("%f", &fa[i]); //Nhap gia tri cho phan tu thu i
15	} while (fa[i++] > 0); //con nhap lieu khi gia tri phan tu > 0
16	
17	i--; //giam i di 1 lan cuoi cung tang 1 truoc khi thoat
18	//Tinh tong
19	for(int ij = 0; ij < i; ij++)
20	fsum += fa[ij]; //cong don tung phan tu vao isum
21	
22	printf("Tong : %5.2f\n", fsum);
23	getch();
24	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao phan tu thu 1: 1.2 Nhap vao phan tu thu 2: 3 Nhap vao phan tu thu 3: 4.6 Nhap vao phan tu thu 4: -9 Tong : 8.80 _	Bạn chạy lại chương trình và thử lại với số liệu khác. Quan sát kết quả.

☹ **Điều gì sẽ xảy ra cho đoạn chương trình trên nếu bạn nhập số lượng phần tử vượt quá 50 trong khi bạn chỉ khai báo mảng fa tối đa là MAX = 50 phần tử. Bạn dùng lệnh break để thoát khỏi vòng lặp do…while trước khi bước sang phần tử thứ 51. Thêm đoạn lệnh sau vào trước dòng 13:**

```
if (i >= MAX)                  //kiem tra phan tu buoc sang 51
{
   printf("Mang da day!\n");   //thong bao "Mang da day"
   i++;                        //tang i len 1 do dong 17 giam i xuong 1
   break;                      //thoat khoi vong lap do…while
}
```

☞ Sửa dòng 5 thành #define MAX 4. Chạy chương trình và nhập các số 1.2, 3.5, 6.5, 4. Quan sát kết quả.

8.2.1.7 Khởi tạo mảng

Ví dụ 6 : Có 4 loại tiền 1, 5, 10, 25 và 50 đồng. Hãy viết chương trình nhập vào số tiền sau đó cho biết số số tiền trên gồm mấy loại tiền, mỗi loại bao nhiêu tờ.

Phác họa lời giải: Số tiền là 246 đồng gồm 4 tờ 50 đồng, 1 tờ 25 đồng, 2 tờ 10 đồng, 0 tờ 5 đồng và 1 tờ 1 đồng, Nghĩa là bạn phải xét loại tiền lớn trước, nếu hết khả năng mới xét tiếp loại kế tiếp.

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Nhap vao so tien va doi tien ra cac loai 50, 25, 10, 5, 1 */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	#define MAX 5
6	
7	void main(void)
8	{
9	int itien[MAX] = {50, 25, 10, 5, 1}; //Khai bao va khoi tao mang voi 5 phan tu
10	int i , isotien, ito;
11	printf("Nhap vao so tien: ");
12	scanf("%d", &isotien); //Nhap vao so tien
13	for (i = 0; i < MAX; i++)
14	{
15	ito = isotien/itien[i]; //Tim so to cua loai tien thu i
16	printf("%4d to %2d dong\n", ito, itien[i]);
17	isotien = isotien%itien[i]; //So tien con lai sau khi da loai tru cac loai tien da co
18	}

```
19      getch();
20  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so tien: 246 4 tờ 50 đồng 1 tờ 25 đồng 2 tờ 10 đồng 0 tờ 5 đồng 1 tờ 1 đồng _	Bạn chạy lại chương trình và thử lại với số liệu khác. Quan sát kết quả.

☹ **Điều gì sẽ xảy nếu số phần tử mảng lớn hơn số mục, số phần tử dôi ra không được khởi tạo sẽ điền vào số 0. Nếu số phần tử nhỏ hơn số mục khởi tạo trình biên dịch sẽ báo lỗi.**

Ví dụ 7 : int itien[5] = {50, 25}, phần tử itien[0] sẽ có giá trị 50, itien[1] có giá trị 25, itien[2], itien[3], itien[4] có giá trị 0.

int itien[3] = {50, 25, 10, 5, 1} → trình biên dịch báo lỗi

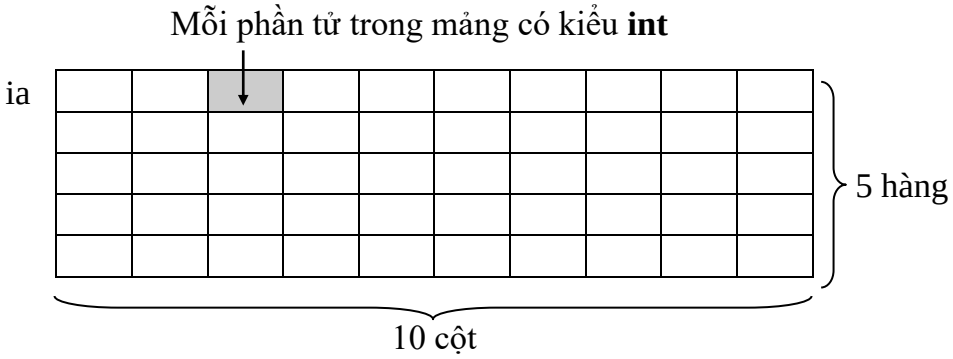
8.2.1.8 Khởi tạo mảng không bao hàm kích thước

Trong ví dụ trên giả sử ta khai báo int itien[] = {50, 25, 10, 5, 1}. Khi đó trình biên dịch sẽ đếm số mục trong danh sách khởi tạo và dùng con số đó làm kích thước mảng.

8.2.1.9 Mảng nhiều chiều

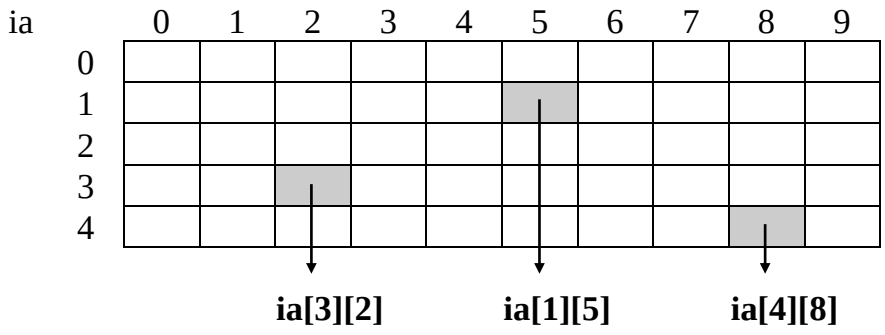
Ví dụ 8 : khai báo mảng 2 chiều **int ia[5][10];** với **int** là kiểu mảng, ia là tên mảng, số phần tử mảng là 5 x 10.

Ý nghĩa: ***Khai báo một mảng 2 chiều số nguyên gồm 50 phần tử, mỗi phần tử có kiểu int.***



8.2.1.10 Tham chiếu đến từng phần tử mảng 2 chiều

Sau khi được khai báo, mỗi phần tử trong mảng 2 chiều đều có 2 chỉ số để tham chiếu, chỉ số hàng và chỉ số cột. Chỉ số hàng bắt đầu từ 0 đến số hàng – 1 và chỉ số cột bắt đầu từ 0 đến số cột – 1. Tham chiếu đến một phần tử trong mảng 2 chiều ia: ia[chỉ số hàng][chỉ số cột]

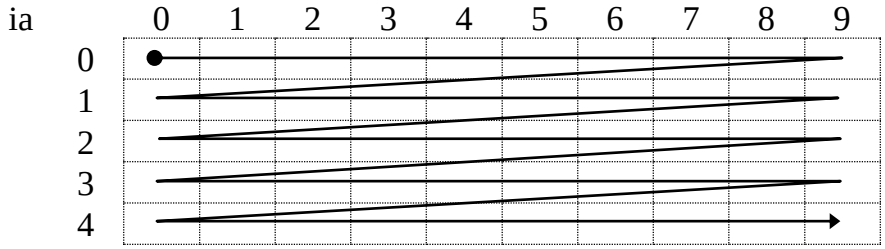


ia[3][2] là phần tử tại hàng 3 cột 2 trong mảng 2 chiều xem như là một biến kiểu **int**.

8.2.1.11 Nhập dữ liệu cho mảng 2 chiều

```
for (i = 0; i < 5; i++)          //vòng for có giá trị i chạy từ 0 đến 4 cho hàng
   for (ij = 0; ij < 10; ij++)   //vòng for có giá trị ij chạy từ 0 đến 9 cho cột
   {
      printf("Nhap vao phan tu ia[%d][%d]: ", i + 1, ij + 1);
      scanf("%d", &ia[i][ij]);
   }
```

* Thứ tự nhập dữ liệu vào mảng 2 chiều



8.2.1.12 Đọc dữ liệu từ mảng 2 chiều

Ví dụ 9 : in giá trị các phần tử mảng 2 chiều ra màn hình.

```
for (i = 0; i < 5; i++)          //vòng for có giá trị i chạy từ 0 đến 4 cho hàng
{
   for (ij = 0; ij < 10; ij++)   //vòng for có giá trị ij chạy từ 0 đến 9 cho cột
      printf("%3d ", ia[i][ij]);
   printf("\n");                 //xuống dòng để in hàng kế tiếp
}
```

Ví dụ 10 : Viết chương trình nhập vào 1 ma trận số nguyên n x n. In ra ma trận vừa nhập vào và ma trận theo thứ tự ngược lại.

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Tinh trung binh cong n so nguyen */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	#define MAX 50;
6	
7	void main(void)
8	{
9	int ia[MAX][MAX], i, ij, in;
10	printf("Nhap vao cap ma tran: ");
11	scanf("%d", &in);
12	
13	//Nhap du lieu vao ma tran
14	for (i = 0; i < in; i++) //vòng for có giá trị i chạy từ 0 đến in-1 cho hàng
15	for (ij = 0; ij < in; ij++) //vòng for có giá trị ij chạy từ 0 đến in-1 cho cột
16	{
17	printf("Nhap vao phan tu ia[%d][%d]: ", i + 1, ij + 1);
18	scanf("%d", &ia[i][ij]);
19	}
20	
21	//In ma tran
22	for (i = 0; i < in; i++) //vòng for có giá trị i chạy từ 0 đến in-1 cho hàng
23	{

```
24          for (ij = 0; ij < in; ij++)        //vòng for có giá trị ij chạy từ 0 đến in-1 cho cột
25              printf("%3d ", ia[i][ij]);
26          printf("\n");                      //xuống dòng để in hàng kế tiếp
27      }
28      printf("\n");                          //Tao khoang cach giua 2 ma tran
29
30      //In ma tran theo thu tu nguoc
31      for (i = in-1; i >= 0; i--)            //vòng for có giá trị i chạy từ in-1 đến 0 cho hàng
32      {
33          for (ij = in-1; ij >= 0 in; ij--)  //vòng for có giá trị ij chạy từ in-1 đến 0 cho cột
34              printf("%3d ", ia[i][ij]);
35          printf("\n");                      //xuống dòng để in hàng kế tiếp
36      }
37      getch();
38  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao cap ma tran: 2
Nhap vao phan tu ia[1][1]: 7
Nhap vao phan tu ia[1][2]: 4
Nhap vao phan tu ia[2][1]: 6
Nhap vao phan tu ia[2][2]: 15
  7   4
  6  15

 15   6
  4   7
_
```

Chạy và thử lại chương trình với n = 3, 5.
Quan sát kết quả.
- Sửa lại chương trình trên cho phép nhập vào ma trận m x n.. Nghĩa là ma trận có m hàng và n cột. Bạn sửa lại chương trình bằng cách cho nhập vào giá trị m và n và sửa lại vòng for cho hàng chạy m lần và vòng for cho cột chạy n lần.

☹ **Để khắc phục tình trạng người dùng nhập vào cấp ma trận > MAX, Bạn xem lại mục 3.1.4.**

8.2.1.13 Sử dụng biến kiểu khác trong mảng 2 chiều

Như mục 3.1.5.

8.2.1.14 Khởi tạo mảng 2 chiều

Ví dụ 11 : Vẽ chữ H lớn.

Dòng **File Edit Search Run Compile Debug Project Option Window Help**

```
1   /* Chuong trinh ve chu H lon */
2
3   #include <stdio.h>
4   #include <conio.h>
5   #define MAX   5
6
7   int H[MAX][MAX]  =  {{1, 0, 0, 0, 1},
8                        {1, 0, 0, 0, 1},
9                        {1, 1, 1, 1, 1},
10                       {1, 0, 0, 0, 1},
```

```
                                {1, 0, 0, 0, 1}};
void main(void)
{
   int i , ij;
   for (i = 0; i < MAX; i++)
   {
      for (ij = 0; ij < MAX; ij++)
         if (H[i][ij])
            printf("!");
         else
            printf(" ");
      printf("\n");
   }
   getch();
}
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
!   !
!   !
!!!!!
!   !
!   !

_
```

Bạn sửa lại chương trình để in ra chữa C, B…

8.2.1.15 Dùng mảng 1 chiều làm tham số cho hàm

Ví dụ 12 : Tìm số lớn nhất

Dòng **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
/* Chuong trinh tim so lon nhat su dung ham */

#include <stdio.h>
#include <conio.h>

#define MAX    20

//Khai bao prototype
int max(int, int);

//ham tim so lon nhat trong mang 1 chieu
int max(int ia[], int in)
{
   int i, imax;
   imax = ia[0];              //cho phan tu dau tien la max
   for (i = 1; i < in; i++)
      if (imax < ia[i])       //neu so dang xet > max
         imax = ia[i];        //gan so nay cho max
   return imax;               //tra ve ket qua so lon nhat
}

void main(void)
{
   int ia[MAX];
```

```
25      int i = 0, inum;
26      do
27      {
28          printf("Nhap vao mot so: ");
29          scanf("%d", &ia[i]);
30      } while (ia[i++] != 0);
31      i--;
32      inum = max(ia, i);
33      printf("So lon nhat la: %d.\n", inum);
34      getch();
35  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao mot so: 12
Nhap vao mot so: 45
Nhap vao mot so: 3
Nhap vao mot so: 0
So lon nhat la: 45
_
```

Chạy lại chương trình và thử lại với số liệu khác.

Thực hiện một số thay đổi sau:
- Di chuyển dòng **int a[MAX]**; lên sau **dòng số 10**
- Sửa dòng **int max(int, int);** thành **int max(int);**
- Sửa dòng **int max(int a[], int n);** thành **int max(int n);**
- Sửa dòng **num = max(a, i);** thành **num = max(i);**

Chạy lại chương trình, quan sát, nhận xét và đánh giá kết quả.

☞ ***Giải thích chương trình***

Chương trình ban đầu hàm max có hai tham số truyền vào và kết quả trả về là giá trị max có kiểu nguyên, một tham số là mảng 1 chiều kiểu int và một tham số có kiểu int. Với chương trình sau khi sửa hàm max chỉ còn một tham số truyền vào nhưng cho kết quả như nhau. Do sau khi sửa chương trình mảng a[MAX] được khai báo lại là biến toàn cục nên hàm max không cần truyền tham số mảng vào cũng có thể sử dụng được. Tuy vậy, khi lập trình bạn nên viết như chương trình ban đầu là truyền tham số mảng vào (dạng tổng quát) để hàm max có thể thực hiện được trên nhiều mảng khác nhau. Còn với chương trình sửa lại bạn chỉ sử dụng hàm max được với mảng a mà thôi.

Ví dụ 13 : Bạn khai báo các mảng sau ia[MAX], ib[MAX], ic[MAX]. Để tìm giá trị lớn nhất của từng mảng. Bạn chỉ cần gọi hàm

- imax_a = max(ia, i);
- imax_b = max(ib, i);
- imax_c = max(ic, i);

Với chương trình sửa lại bạn không thể tìm được số lớn nhất của mảng b và c.

☞ **Bạn lưu ý rằng khi truyền mảng sang hàm, không tạo bản sao mảng mới. Vì vậy mảng truyền sang hàm có dạng tham biến. Nghĩa là giá trị của các phần tử trong mảng sẽ bị ảnh hưởng nếu có sự thay đổi trên chúng.**

Ví dụ 14 : Tìm số lớn nhất của 3 mảng a, b, c

Dòng **File Edit Search Run Compile Debug Project Option Window Help**

```
1   /* Chuong trinh tim so lon nhat su dung ham */
2
3   #include <stdio.h>
4   #include <conio.h>
5
6   #define MAX    20
7
8   //Khai bao prototype
9   int max(int, int);
```

```
10  int input(int);
11
12  //ham tim phan tu lon nhat trong mang 1 chieu
13  int max(int ia[], int in)
14  {
15      int i, imax;
16      imax = ia[0];               //cho phan tu dau tien la max
17      for (i = 1; i < in; i++)
18          if (max < ia[i])        //neu so dang xet > max
19              max = ia[i];        //gan so nay cho max
20      return imax;                //tra ve ket qua so lon nhat
21  }
22
23  //ham nhap lieu vao mang 1 chieu
24  int input(int ia[])
25  {
26      int i = 0;
27      do
28      {
29          printf("Nhap vao mot so: ");
30          scanf("%d", &ia[i]);
31      } while (ia[i++] != 0);
32      i--;
33      return i;
34  }
35
36  void main(void)
37  {
38      int ia[MAX], ib[MAX], ic[MAX];
39      int inum1, inum2, inum3;
40      printf("Nhap lieu cho mang a: \n");
41      inum1 = max(ia, input(ia));
42      printf("Nhap lieu cho mang b: \n");
43      inum2 = max(ib, input(ib));
44      printf("Nhap lieu cho mang c: \n");
45      inum3 = max(ic, input(ic));
46      printf("So lon nhat cua mang a: %d, b: %d, c: %d.\n", inum1, inum2, inum3);
47      getch();
48  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap lieu cho mang a:
Nhap vao mot so: 12
Nhap vao mot so: 45
Nhap vao mot so: 3
Nhap vao mot so: 0
Nhap lieu cho mang b:
Nhap vao mot so: 5
Nhap vao mot so: 15
Nhap vao mot so: 0
Nhap lieu cho mang c:
Nhap vao mot so: 1
Nhap vao mot so: 5
Nhap vao mot so: 4
Nhap vao mot so: 0
So lon nhat cua mang a: 45, b: 15, c: 5.
_
```

Chạy lại chương trình và thử lại với số liệu khác.
Viết thêm hàm tìm số nhỏ nhất.

☞ ***Giải thích chương trình***

Hàm input có kiểu trả về là int thông qua biến i (cho biết số lượng phần tử đã nhập vào) và 1 tham số là mảng 1 chiều kiểu int. Dòng 41, 43, 45 lần lượt gọi hàm input với các tham số là mảng a, b, c. Khi hàm input thực hiện việc nhập liệu thì các phần tử trong mảng cũng được cập nhật theo.

8.2.1.16 Dùng mảng 2 chiều làm tham số cho hàm

Ví dụ 15 : Nhập vào 2 ma trận vuông cấp n số thập phân. Cộng 2 ma trận này lưu vào ma trận thứ 3 và tìm số lớn nhất trên ma trận thứ 3.

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* cong ma tran */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	
6	#define MAX 20
7	
8	//Khai bao prototype
9	void input(float);
10	void output(float);
11	void add(float, float, float);
12	float max(float);
13	
14	//khai bao bien toan cuc
15	int in;
16	
17	//ham tim so lon nhat trong mang 2 chieu
18	float max(float fa[][MAX])
19	{
20	float fmax;
21	fmax = fa[0][0]; //cho phan tu dau tien la max
22	for (int i = 0; i < in; i++)
23	for (int ij = 0; ij < in; ij++)
24	if (fmax < fa[i][ij]) //neu so dang xet > max
25	fmax = fa[i][ij]; //gan so nay cho max
26	return fmax; //tra ve ket qua so lon nhat
27	}
28	
29	//ham nhap lieu mang 2 chieu
30	void input(float fa[][MAX])
31	{
32	for (int i = 0; i < in; i++)
33	for (int ij = 0; ij < in; ij++)
34	{
35	printf("Nhap vao ptu[%d][%d]: ", i, ij);
36	scanf("%f", &fa[i, j]);
37	}
38	}
39	
40	//ham in mang 2 chieu ra man hinh
41	void output(float fa[][MAX])
42	{
43	for (int i = 0; i < in; i++)

```
44      {
45          for (int ij = 0; ij < n; ij++)
46              printf("%5.2f", fa[i][ij]);
47          printf("\n");
48      }
49  }
50
51  //ham cong 2 mang 2 chieu
52  void add(float fa[][MAX], float fb[][MAX], float fc[][MAX])
53  {
54      for (int i = 0; i < in; i++)
55          for (int ij = 0; ij < in; ij++)
56              fc[i, ij] = fa[i, ij] + fb[i, ij];
57  }
58
59  void main(void)
60  {
61      float fa[MAX][MAX], fb[MAX][MAX], fc[MAX][MAX];
62      printf("Nhap vao cap ma tran: ");
63      scanf("%d", &in);
64      printf("Nhap lieu ma tran a: \n");
65      input(fa);
66      printf("Nhap lieu ma tran b: \n");
67      input(fb);
68      printf("Nhap lieu ma tran c: \n");
69      input(fc);
70      add(fa, fb, fc);
71      printf("Ma tran a: \n");
72      output(fa);
73      printf("Ma tran b: \n");
74      output(fb);
75      printf("Ma tran c: \n");
76      output(fc);
77      printf("So lon nhat cua ma tran c la: %5.2f.\n", max(fc));
78      getch();
79  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao cap ma tran : 2
Nhap lieu ma tran a:
Nhap vao ptu[0][0] : 5.2
Nhap vao ptu[0][1] : 4
Nhap vao ptu[1][0] : 7.1
Nhap vao ptu[1][1] : 9
Nhap lieu ma tran b:
Nhap vao ptu[0][0] : 12
Nhap vao ptu[0][1] : 3.4
Nhap vao ptu[1][0] : 9.6
Nhap vao ptu[1][1] : 11
```

```
Ma tran a:
  5.20   4.00
  7.10   9.00
Ma tran b:
 12.00   3.40
  9.60  11.00
Ma tran c:
 17.20   7.40
 16.70  20.00
So lon nhat cua ma tran c la: 20.00
_
```

Chạy lại chương trình và thử lại với số liệu khác.

Viết thêm hàm tìm số nhỏ nhất.

☞ ***Giải thích chương trình***

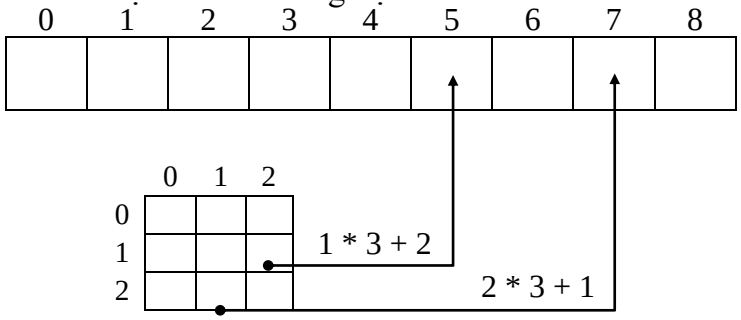
Trong chương trình khai báo biến in toàn cục do biến này sử dụng trong suốt quá trình chạy chương trình. Tham số truyền vào hàm là mảng hai chiều dưới dạng a[][MAX] vì hàm không dành chỗ cho mảng, hàm chỉ cần biết số cột để tham khảo đến các phần tử.

Ví dụ 16 : Mảng 2 chiều được khai báo int ia[3][3]

Truyền tham số vào hàm: ia[][3],
để tham khảo đến ptử[2][1],
hàm tính như sau:

2 * 3 + 1 = 7 **(chỉ số hàng * số cột + chỉ số cột)**

ia[3][3] gồm 9 phần tử được lưu trữ trong bộ nhớ như sau:



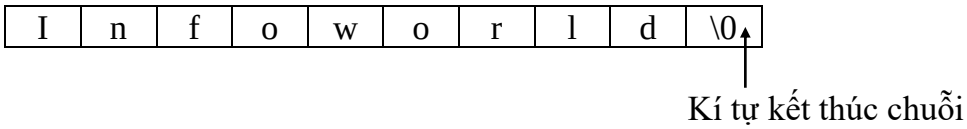
☞ Giống như mảng 1 chiều khi truyền mảng 2 chiều sang hàm cũng không tạo bản sao mới.

8.2.2 Chuỗi

Chuỗi được xem như là một mảng 1 chiều gồm các phần tử có kiểu char như mẫu tự, con số và bất cứ ký tự đặc biệt như +, -, *, /, $, #…

Theo quy ước, một chuỗi sẽ được kết thúc bởi ký tự **null** ('\0' : kí tự rỗng).

Ví dụ: chuỗi "Infoworld" được lưu trữ như sau:



8.2.2.1 Cách khai báo chuỗi

Ví dụ 17 : **char cname[30];**

Ý nghĩa: ***Khai báo chuỗi cname có chiều dài 30 kí tự.*** Do chuỗi kết thúc bằng kí tự null, nên khi bạn khai báo chuỗi có chiều dài 30 kí tự chỉ có thể chứa 29 kí tự.

Ví dụ 18 : Nhập vào in ra tên

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Chuong trinh nhap va in ra ten*/
2	
3	#include <stdio.h>
4	#include <conio.h>
5	
6	void main(void)
7	{
8	char cname[30];
9	printf("Cho biet ten cua ban: ");
10	scanf("%s", cname);

```
11      printf("Chao ban %s\n", cname);
12      getch();
13  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Cho biet ten cua ban: Minh Chao ban Minh _	Chạy lại chương trình và thử nhập tên: Mai Lan, Thanh Nhi Quan sát kết quả.

☞ **Lưu ý: không cần sử dụng toán tử địa chỉ & trong cname trong lệnh scanf("%s", fname), vì bản thân fname đã là địa chỉ.**

Dùng hàm scanf để nhập chuỗi có hạn chế như sau: Khi bạn thử lại chương trình trên với dữ liệu nhập vào là Mai Lan, nhưng khi in ra bạn chỉ nhận được Mai. Vì hàm scanf nhận vào dữ liệu đến khi gặp khoảng trắng thì kết thúc.

8.2.2.2 Hàm nhập (gets), xuất (puts) chuỗi

Sử dụng hàm gets, puts phải khai báo #include <stdio.h>

Dòng **File Edit Search Run Compile Debug Project Option Window Help**

```
1   /* Chuong trinh nhap va in ra ten*/
2
3   #include <stdio.h>
4   #include <conio.h>
5
6   void main(void)
7   {
8       char cname[30];
9       puts("Cho biet ten cua ban: ");
10      gets(cname);
11      puts("Chao ban ");
12      puts(cname);
13      getch();
14  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Cho biet ten cua ban: Mai Lan Chao ban Mai Lan _	Sửa dòng 9 thành printf("Cho biet ten cua ban: "); và từ dòng 11 đến 12 thành printf("Chao ban %s.\n", cname); Chạy lại chương trình vào thử nhập tên: Tuan Anh, Thanh Lan Quan sát kết quả.

☞ **Đối với hàm puts kí tự kết thúc chuỗi null (\0) được thay thế bằng kí tự newline (\n). Hàm gets và puts chỉ có 1 đối số và không sử dụng dạng thức trong nhập liệu cũng như xuất ra màn hình.**

8.2.2.3 Khởi tạo chuỗi

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Chuong trinh nhap va in ra ten*/
2	
3	#include <stdio.h>
4	#include <conio.h>
5	
6	void main(void)
7	{
8	char cname[30];
9	char chao[] = "Chao ban";
10	printf("Cho biet ten cua ban: ");
11	gets(cname);
12	printf("%s %s.\n", chao, cname);
13	getch();
14	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Cho biet ten cua ban: Mai Lan Chao ban Mai Lan _	Chạy lại chương trình vào thử nhập tên: Doan Trang Quan sát kết quả.

☞ **Chiều dài tối đa của chuỗi khởi tạo bằng số kí tự + 1 (kí tự null). Với chuỗi chao có chiều dài là 9.**

8.2.2.4 Mảng chuỗi

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Chuong trinh nhap thang (so) và in ra thang (chu) tuong ung*/
2	
3	#include <stdio.h>
4	#include <conio.h>
5	
6	void main(void)
7	{
8	char cthang[12][15] = {"January", "February", "March", "April",
9	"May", "June", "July", "August", "September",
10	"October", "November", "December"};
11	int ithang;
12	printf("Nhap vao thang (1-12): ");
13	scanf("%d", &ithang);
14	printf("%s.\n", cthang[ithang-1]);
15	getch();
16	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao thang (1-12): 2 February _	Chạy lại chương trình vào thử nhập vào các tháng khác Quan sát kết quả.

8.3 Bài tập

1. Viết hàm tìm số lớn nhất, nhỏ nhất trong một mảng n số nguyên.

2. Viết hàm sắp xếp tăng dần, giảm dần của một dãy số cho trước.

3. Viết hàm tách tên và họ lót từ một chuỗi cho trước.

4. Viết hàm cắt bỏ khoảng trắng thừa ở giữa, hai đầu.

5. Viết hàm chuyển đổi 1 chuỗi sang chữ thường và 1 hàm chuyển đổi sang chữ HOA.

6. Viết hàm chuyển đổi 1 chuỗi sang dạng Title Case (kí tự đầu của mỗi từ là chữ HOA, các kí tự còn lại chữ thường)

7. Viết chương trình nhập vào 1 chuỗi và in ra chuỗi đảo ngược.

Ví dụ: Nhập vào chuỗi "Lap trinh C can ban"
In ra "nab nac C hnirt paL"

8. Viết chương trình nhập vào một chuỗi ký tự rồi đếm xem trong chuỗi đó có bao nhiêu chữ 'th'.

9. Biết rằng năm 0 là năm Canh thân (năm kỵ nhau có chu kì là 3, năm hợp nhau có chu kì là 4). Hãy viết chương trình cho phép gõ vào năm dương lịch (ví dụ 1997), xuất ra năm âm lịch (Đinh sửu) và các năm kỵ và hợp.

Có 12 chi: Tý, Sửu, Dần, Mão, Thìn, Tỵ, Ngọ, Mùi, Thân, Dậu, Tuất, Hợi.
Có 10 can: Giáp, Ất, Bính, Đinh, Mậu, Kỷ, Canh, Tân, Nhâm, Quý.

10. Viết chương trình nhập vào 3 chữ số (305, 6, 28). Cho biết dòng chữ mô tả giá trị con số đó. Ví dụ 305 -> ba trăm lẻ năm.

11. Viết chương trình nhập vào một chuỗi sau đó in ra màn hình mỗi dòng là một từ. Ví dụ chuỗi "Lap trinh C". Kết quả in ra

Lap
trinh
C

12. Viết chương trình nhập vào một chuỗi các kí tự, ký số, khoảng trắng và dấu chấm câu. Cho biết chuỗi trên gồm bao nhiêu từ.

13. Viết chương trình nhập vào một chuỗi ký tự. Kiểm tra xem chuỗi đó có đối xứng không?

14. Viết chương trình nhập vào một chuỗi gồm các chữ cái (a -> z, A -> Z). Hãy đếm xem có bao nhiêu nguyên âm a, i, e, o, u.

15. Giả sử số phòng trong một khách sạn được cho bởi hằng số NUM_ROOM. Viết:

a. Một khai báo dãy thích hợp để theo dõi phòng nào còn trống.
b. Một hàm tìm phòng nào còn trống.
c. Viết chương trình đơn giản để quản lý phòng khách sạn theo dạng một trình đơn chọn công việc gồm có 4 mục như sau:

- Tìm phòng trống.
- Trả phòng.
- Liệt kê những phòng còn trống.
- Liệt kê những phòng đã thuê.
- Kết thúc.

16. Viết chương trình mô tả văn bản của một bức điện tín. Nhập liệu bao gồm 1 hay nhiều dòng chứa một số từ, mỗi từ cách nhau khoảng trắng. In ra hóa đơn tính tiền với mỗi từ giá 100 đồng, phí trả thêm 50 đồng cho từ dài quá 8 kí tự. Hóa đơn có dạng sau:

So tu : 10
So tu co kich thuoc binh thuong : 8 x 100 = 800 dong
So tu co kích thuoc > 8 ki tu : 2 x 150 = 300 dong
Tong cong : 1100 dong

17. Viết chương thống kê xem có bao nhiêu người họ "Ly", "Tran"… trong 1 danh sách cho trước. Nếu không có thông báo "Không có nguoi nao thuoc ho ….".

18. Viết chương trình nhập vào 1 chuỗi, sau đó chép sang chuỗi khác một chuỗi con từ chuỗi ban đầu có số kí tự chỉ định.

Ví dụ: Chuỗi ban đầu "Le Thuy Doan Trang". Nếu số kí tự chỉ định là 2 thì chuỗi đích sẽ là "Le"

19. Viết chương trình nhập vào 1 chuỗi, sau đó chép sang chuỗi khác một chuỗi con từ chuỗi ban đầu với vị trí bắt đầu và số kí tự chỉ định.

Ví dụ: Chuỗi ban đầu "Le Thuy Doan Trang". Nếu vị trí ban đầu là 14 và số kí tự chỉ định là 5 thì chuỗi đích sẽ là "Trang"

20. Viết chương trình nhập vào 1 chuỗi nguồn, ví dụ "Nguyen Minh Long", sau đó nhập vào 1 chuỗi con, ví dụ "Minh", chương trình sẽ xác định vị trí bắt đầu của chuỗi con ở vị trí nào trong chuỗi nguồn. Kết quả in ra màn hình như sau:

- Chuoi nguon la : Nguyen Minh Long
- Chuoi con la : Minh
- Vi tri bat dau cua chuoi con la : 8

21. Viết chương thực hiện các yêu cầu sau:

- Nhập vào 1 chuỗi bất kỳ, ví dụ : "Nguyen Minh Long
- Muốn xóa từ vị trí nào, ví dụ : 8
- Muốn xóa bao nhieu kí tự, ví dụ : 5

Kết quả in ra man hinh:

- Chuoi nguon la : Nguyen Minh Long
- Chuoi sau khi xoa : Nguyen Long

Bài 9 :

CON TRỎ

9.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:

- Ý nghĩa, cách khai báo con trỏ
- Sử dụng con trỏ trong mảng, chuỗi
- Truyền mảng và chuỗi giữa các hàm qua con trỏ
- Xử lý mảng dễ dàng qua con trỏ

9.2 Nội dung

9.2.1 Con trỏ?

Con trỏ dùng để truy cập biến thông qua địa chỉ biến và chương trình tham khảo biến gián tiếp qua địa chỉ này.

9.2.2 Khái báo biến con trỏ

Ví dụ 1:

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Cong hang so */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	
6	void main(void)
7	{
8	int ix = 6, iy = 7;
9	int *px, *py;
10	printf("x = %d, y = %d\n", ix, iy);
11	px = &ix;
12	py = &iy;
13	*px += 10;
14	*py += 10;
15	printf("x = %d, y = %d\n", ix, iy);
16	getch();
17	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

x = 6, y = 7 x = 16, y = 17 _	Chạy chương trình và quan sát kết quả.

☞ ***Giải thích chương trình***

Khai báo ở dòng 9 cấp phát 2 bytes để lưu giữ địa chỉ của biến nguyên và vùng nhớ đó có tên là px, tương tự cho py. Dấu * cho biết biến này chứa địa chỉ chứ không phải giá trị, int cho biết địa chỉ đó sẽ trỏ đến các biến nguyên.

☞ Ví dụ trên cho thấy *px và *py là 2 biến con trỏ trỏ đến địa chỉ của 2 biến ix và iy (dòng 11 và 12),vì vậy khi nội dung của biến con trỏ *px và *py thay đổi thì nội dung của ix, iy cũng thay đổi theo.

Địa chỉ vùng nhớ

1203	6	ix
1204		
1205		
1206		
1207	7	iy
1208		
1209		
1210		

Sau phép gán px = &ix và py = &iy thì giá trị của px = 1203 và py = 1207

Địa chỉ vùng nhớ

2015		
2016		
2017	1203	px
2018		
2019	1207	py
2020		
2021		
2022		

*px là nội dung của px và *py là nội dung của py. Nên khi thực hiện 2 phép cộng gán *px += 10 và *py += 10 thì giá trị của ix sẽ là 16 và giá trị iy sẽ là 17.

9.2.3 Truyền địa chỉ sang hàm

Ví dụ 2:

Dòng | **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
1  /* Khoi tao 2 so */
2
3  #include <stdio.h>
4  #include <conio.h>
5
6  void init (int, int);
7
8  void main(void)
9  {
10    int ix, iy;
11    init(&ix, &iy);
12    printf("x = %d, y = %d\n", ix, iy);
13    getch();
14 }
15
16 void init(int *px, int *py)
17 {
18    *px = 3;          //gan 3 cho noi dung cua px
19    *py = 5;          //gan 5 cho noi dung cua py
20 }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

x = 3, y = 5 _	Chạy chương trình và quan sát kết quả.

☞ ***Giải thích chương trình***

Ở dòng 9, gọi hàm init truyền 2 tham số là địa chỉ của biến ix và iy, nên khi nội dung của 2 biến con trỏ *px và *py thay đổi thì ix và iy của chương trình chính cũng thay đổi theo. Hàm main(void) đã sử dụng cách truy cập biến khác với hàm init, hàm main(void) gọi chúng là ix, iy còn hàm init gọi chúng là *px, *py. Hàm init đọc giá trị của biến con trỏ *px, *py từ vùng địa chỉ của chương trình gọi, sau khi thực hiện và trả kết quả về chương trình gọi.

9.2.4 Con trỏ và mảng

Ví dụ 3: Khai báo mảng sau int num[] = {23, 54, 16, 72, 16, 84};

Như đã nghiên cứu cách tham chiếu đến phần tử mảng thứ 5 là num[4], còn với kiểu con trỏ là *(num + 4). Nghĩa là num[4] tương đương với *(num + 4) và cách truy cập nội dung như nhau. Tương tự như vậy, cách tham khảo địa chỉ của phần tử mảng là &num[4] tương đương với num + 4.

9.2.5 Con trỏ trỏ đến mảng trong hàm

Ví dụ 4:

Dòng | **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
1  /* Cong hang so vao mang */
2
3  #include <stdio.h>
4  #include <conio.h>
5
6  #define SIZE 4
7
8  add(int *, int, int);
9
10 void main(void)
11 {
12    int iarray[] = {2, 5, 6, 9};
13    int i, ix = 10;
14    add(iarray, SIZE, ix);
15    for (i = 0; i < SIZE; i++);
16       printf("%d ", *(iarray + i));
17    getch();
18 }
19
20 void add(int *ptr, int inum, int ia)
21 {
22    int ij;
23    for (ij = 0; ij < inum; ij++)
24       *(ptr) = *(ptr++) + ia;
25 }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

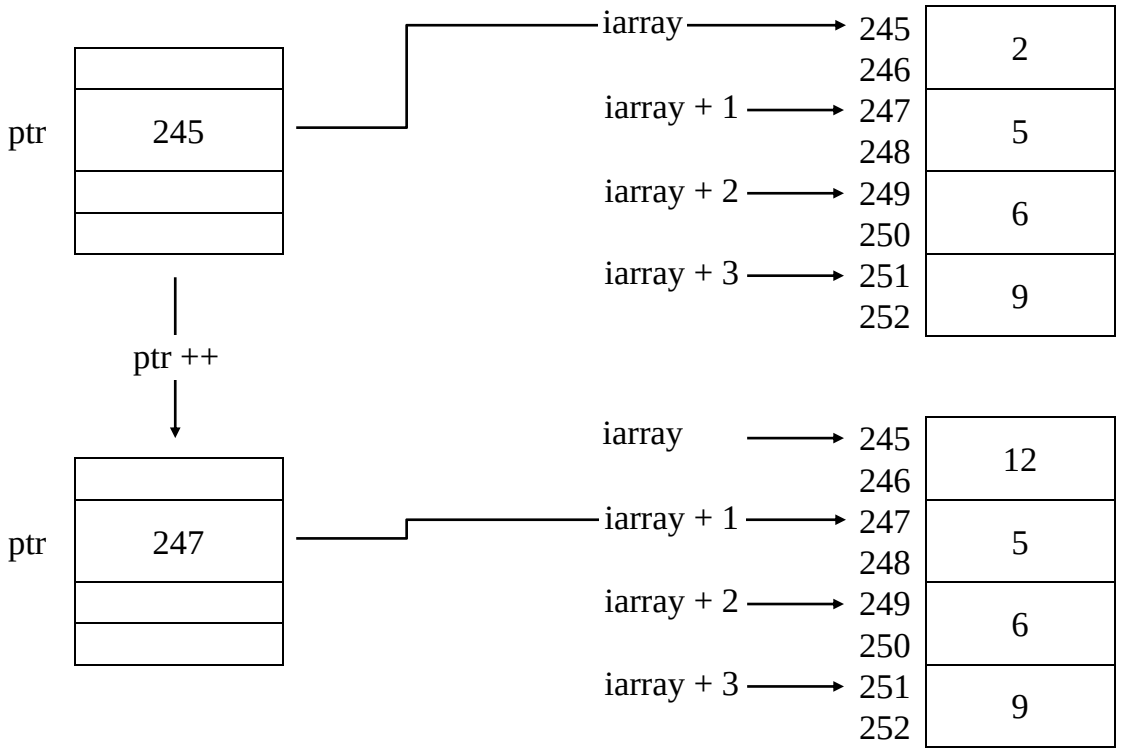
☞ ***Kết quả in ra màn hình***

12 15 16 19 _	Chạy chương trình và quan sát kết quả.

☞ ***Giải thích chương trình***

Hàm gán địa chỉ của mảng vào biến con trỏ ptr, kích thước vào biến inum và hằng số vào biến ia. Sau đó dùng vòng lặp để cộng hằng vào từng phần tử của mảng.

Giả sử địa chỉ của ia là 245 khi truyền vào hàm add qua ptr, ptr sẽ có giá trị = 245



9.2.6 Con trỏ và chuỗi

Ví dụ 5:

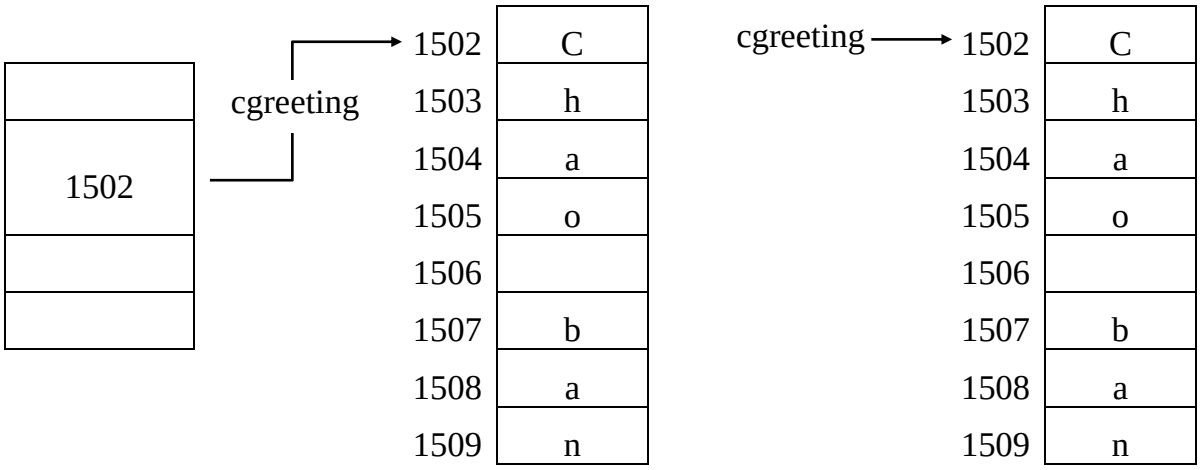
Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Chuong trinh nhap va in ra ten*/
2	
3	#include <stdio.h>
4	#include <conio.h>
5	
6	void main(void)
7	{
8	char *cgreeting = "Chao ban";
9	char cname[30];
10	puts("Cho biet ten cua ban: ");
11	gets(cname);
12	puts(cgreeting);
13	puts(cname);
14	getch();
15	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Cho biet ten cua ban: Minh Chao ban Minh _	Chạy chương trình và quan sát kết quả.

☞ ***Giải thích chương trình***

cgreeting là biến con trỏ được khởi tạo bằng phát biểu char *cgreeting = "Chao ban" thay vì char cgreeting[] = "Chao ban". Cả hai cách đều cho cùng kết quả và đều dành số byte cho chuỗi kèm theo kí tự null. Đối với mảng địa chỉ của kí tự đầu tiên của mảng sẽ là tên mảng, còn con trỏ sẽ có thêm biến con trỏ trỏ đến tên cgreeting.



char * cgreeting = "Chao ban" (biến con trỏ)

char cgreeting[] = "Chao ban" (hằng con trỏ)

9.2.7 Khởi tạo mảng con trỏ trỏ đến chuỗi

Ví dụ 6:

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Chuong trinh nhap thang (so) và in ra thang (chu) tuong ung*/
2	
3	#include <stdio.h>
4	#include <conio.h>
5	
6	void main(void)
7	{
8	char *cthang[12]= {"January", "February", "March", "April",
9	"May", "June", "July", "August", "September",
10	"October", "November", "December"};
11	int ithang;
12	printf("Nhap vao thang (1-12): ");
13	scanf("%d", &ithang);
14	printf("%s.\n", cthang[ithang-1]);
15	getch();
16	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao thang (1-12): 2 February _	Chạy lại chương trình vào thử nhập vào các tháng khác Quan sát kết quả.

☞ ***Giải thích chương trình***

Khai báo char *cthang[12] có ý nghĩa như sau: cthang là tên gọi, dấu * là kiểu con trỏ trỏ đến kí tự (char).

	Địa chỉ	0	1	2	3	4	5	6	7	8	9
cthang[0] →	1010	J	a	n	u	a	r	y	\0		
cthang[1] →	1020	F	e	b	r	u	a	r	y	\0	
cthang[2] →	1030	M	a	r	c	h	\0				
cthang[3] →	1040	A	p	r	i	l	\0				
cthang[4] →	1050	M	a	y	\0						
cthang[5] →	1060	J	u	n	e	\0					
cthang[6] →	1070	J	u	l	y	\0					
cthang[7] →	1080	A	u	g	u	s	t	\0			
cthang[8] →	1090	S	e	p	t	e	m	b	e	r	\0
cthang[9] →	1100	O	c	t	o	b	e	r	\0		
cthang[10] →	1110	N	o	v	e	m	b	e	r	\0	
cthang[11] →	1120	D	e	c	e	m	b	e	r	\0	

Mảng các chuỗi **char cthang[12][10]]**

cthang[0]	1010 →	1010	J	a	n	u	a	r	y	\0	
cthang[1]	1018 →	1018	F	e	b	r	u	a	r	y	\0
cthang[2]	1027 →	1027	M	a	r	c	h	\0			
cthang[3]	1033 →	1033	A	p	r	i	l	\0			
cthang[4]	1039 →	1039	M	a	y	\0					
cthang[5]	1043 →	1043	J	u	n	e	\0				
cthang[6]	1048 →	1048	J	u	l	y	\0				
cthang[7]	1053 →	1053	A	u	g	u	s	t	\0		
cthang[8]	1060 →	1060	S	e	p	t	e	m	b	e	r	\0
cthang[9]	1070 →	1070	O	c	t	o	b	e	r	\0	
cthang[10]	1078 →	1078	N	o	v	e	m	b	e	r	\0
cthang[11]	1087 →	1087	D	e	c	e	m	b	e	r	\0

Mảng các con trỏ trỏ đến các chuỗi **char *cthang[12]**

☞ Khởi tạo mảng các con trỏ trỏ đến các chuỗi chiếm ít bộ nhớ hơn khởi tạo mảng chuỗi.

9.2.8 Xử lý con trỏ trỏ đến chuỗi

Ví dụ 7:

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Nhap danh sach ten va sap xep theo thu tu tang dan*/
2	
3	#include <stdio.h>
4	#include <conio.h>
5	#inlcude <string.h>
6	
7	#define MAXNUM 5

```
 8  #define MAXLEN 10
 9
10  void main(void)
11  {
12      char cname[MAXNUM][MAXLEN];     //mang chuoi
13      char *cptr[MAXNUM];             //mang con tro tro den chuoi
14      char *ctemp;
15      int i, ij, icount = 0;
16
17      //nhap danh sach ten
18      while (icount < MAXNUM)
19      {
20          printf("Nhap vao ten nguoi thu %d: ", icount + 1);
21          gets(cname[icount]);
22          cptr[icount++] = cname[icount];          //con tro den ten
23      }
24
25      //sap xep danh sach theo thu tu tang dan
26      for (i = 0; i < icount – 1; i ++)
27          for (ij = i + 1; ij < icount; ij ++)
28              if (strcmp(cptr[i], cptr[ij]) > 0)
29              {
30                  ctemp = cptr[i];
31                  cptr[i] = cptr[ij];
32                  cptr[ij] = ctemp;
33              }
34
35      //In danh sach da sap xep
36      printf("Danh sach sau khi sap xep:\n");
37      for (i = 0; i < icount; i ++)
38          printf("Ten nguoi thu %d : %s\n", i + 1, cptr[i]);
39      getch();
40  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

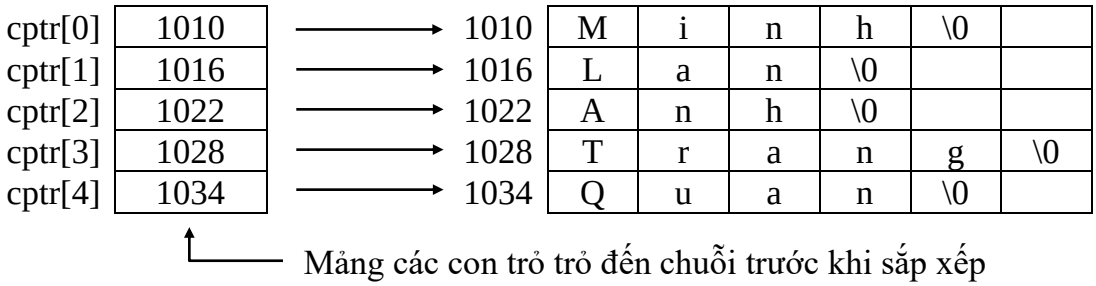
```
Nhap vao ten nguoi thu 1: Minh
Nhap vao ten nguoi thu 2: Lan
Nhap vao ten nguoi thu 3: Anh
Nhap vao ten nguoi thu 4: Trang
Nhap vao ten nguoi thu 5: Quan
Danh sach sau khi sap xep:
Ten nguoi thu 1:  Anh
Ten nguoi thu 2: Lan
Ten nguoi thu 3: Minh
Ten nguoi thu 4: Quan
Ten nguoi thu 5: Trang
_
```

Chạy lại chương trình và thử nhập với dữ liệu khác.

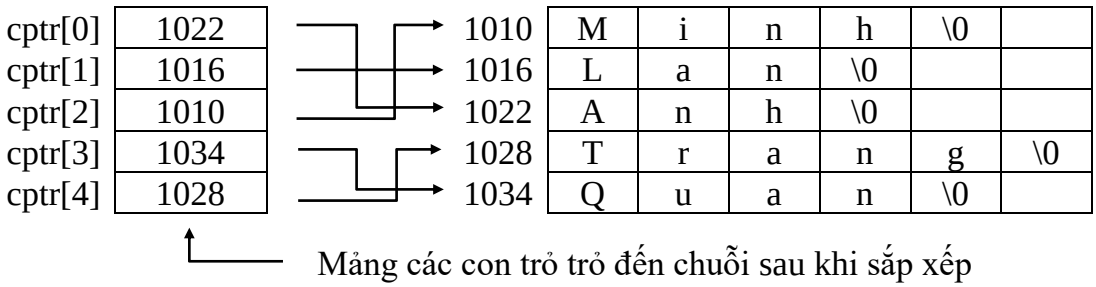
Quan sát kết quả.

☞ ***Giải thích chương trình***

Trong chương trình dùng cả mảng chuỗi char cname[MAXNUM][MAXLEN] và mảng con trỏ trỏ đến chuỗi char *cptr[MAXNUM];.



Mảng các con trỏ trỏ đến chuỗi trước khi sắp xếp



Mảng các con trỏ trỏ đến chuỗi sau khi sắp xếp

9.2.9 Con trỏ trỏ đến con trỏ

Ví dụ 8:

Dòng | **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
1  /* In ma trận*/
2
3  #include <stdio.h>
4  #include <conio.h>
5
6  #define ROWS 4
7  #define COLS 5
8
9  void main(void)
10 {
11    int itable[ROWS][COLS] =     {{10, 12, 14,  16, 18},
12                                  {11, 13, 15, 17, 19},
13                                  {20, 22, 24, 26, 28},
14                                  {21, 23, 25, 27, 29}};
15    int i, ij, ix = 10;
16
17    for (i = 0; i < ROWS; i ++)
18       for (ij = 0; ij < COLS; ij ++)
19          *(*(table + i) + ij) += ix;
20
21    for (i = 0; i < ROWS; i ++)
22    {
23       for (ij = 0; ij < COLS; ij ++)
24          printf("%4d", *(*(table + i) + ij));
25       printf("\n");
26    }
27    getch();
28 }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

20 22 24 26 28 21 23 25 27 29 30 32 34 36 38 31 33 35 37 39 _	Chạy chương trình và quan sát kết quả.

☞ ***Giải thích chương trình***

Trong chương trình dùng cả mảng chuỗi char cname[MAXNUM][MAXLEN] và mảng con trỏ trỏ đến chuỗi char *cptr[MAXNUM];.

9.3 Bài tập

Làm lại các bài tập ở bài Mảng và chuỗi sử dụng biến con trỏ.

Bài 10 :

CÁC KIỂU DỮ LIỆU TỰ TẠO

10.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:

- Ý nghĩa, cách khai structure, emum
- Nhập, xuất structure.
- Khởi tạo structure, enum
- Một số kỹ thuật thao tác trên structure, enum
- Dùng struct tham số cho hàm.

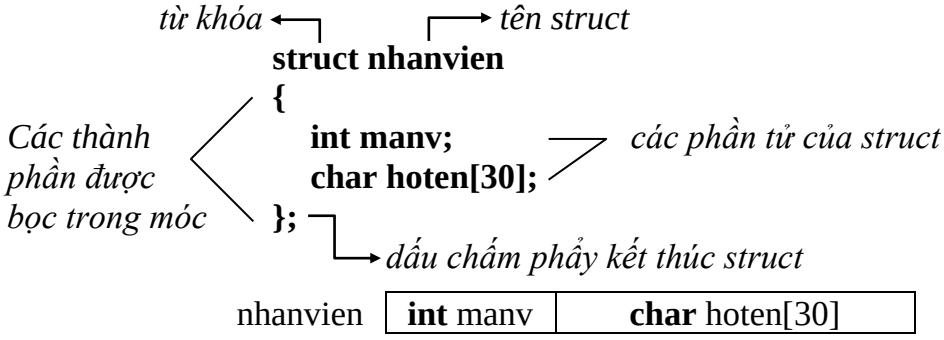
10.2 Nội dung

10.2.1 Structure

Đối với mảng, chỉ có thể lưu nhiều thông tin có cùng kiểu dữ liệu. Nhưng với structure ta có thể lưu thông tin như một mảng có nhiều kiểu dữ liệu khác nhau.

10.2.1.1 Khai báo kiểu structure

Ví dụ 1: khai báo một structure về thông tin nhân viên



Ví dụ trên định nghĩa kiểu dữ liệu mới có tên là **struct nhanviên**. Mỗi biến kiểu này gồm 2 phần tử: biến nguyên có tên là **manv** và biến chuỗi có tên **hoten**.

☞ **struct** phải viết bằng chữ thường

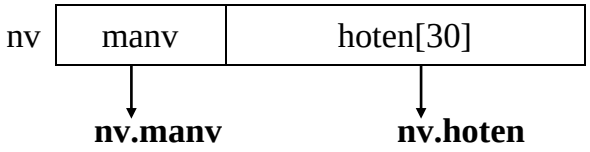
10.2.1.2 Cách khai báo biến có kiểu structure

Ví dụ 2: **struct nhanvien nv;** hoặc **nhanvien nv;**
Khai báo biến **nv** có kiểu **struct nhanvien**

☞ vừa tạo structure nhanvien vừa khai báo biến nv

```
struct nhanvien
{
   int manv;
   char hoten[30];
} nv;
```

10.2.1.3 Tham chiếu các phần tử trong structure



Để tham chiếu đến manv trong nv ta viết như sau: **nv.manv** (là biến có kiểu int)

☞ Đối với biến khai báo kiểu con trỏ **nhanvien *nv** thì tham chiếu đến phần tử manv: **nv -> manv.**

Ví dụ 3: Nhập và in danh sách nhân viên.

Dòng | **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
1  /* Danh sach nhan vien */
2
3  #include <stdio.h>
4  #include <conio.h>
5  #include <stdlib.h>
6
7  #define MAX 50
8
9  void main(void)
10 {
11    struct nhanvien
12    {
13       int manv;
14       char hoten[30];
15    };
16    nhanvien snv[MAX];
17    char ctam[10];
18    int i, in;
19    printf("Nhap vao so nhan vien: ");
20    gets(ctam);
21    in = atoi(ctam);
22
23    //Nhap danh sach nhan vien
24    for(i = 0; i < in; i++)
25    {
26       printf("Nhap vao ma nhan vien thu %d: ", i + 1);
27       gets(ctam);
28       snv[i].manv = atoi(ctam);
29       printf("Nhap vao ho ten: ");
30       gets(snv[i].hoten);
31    }
32
33    //in danh sach nhan vien
34    for(i = 0; i < in; i++)
35       printf("%5d %s\n", snv[i].manv, snv[i].hoten);
36    getch();
37 }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so nhan vien: 2 Nhap vao ma nhan vien thu 1: 123 Nhap vao ho ten: Le Thuy Doan Trang Nhap vao ma nhan vien thu 2: 35	Chạy và thử lại chương trình với dữ liệu khác. Quan sát kết quả.

Nhap vao ho ten: Le Nguyen Tuan Anh 123 Le Thuy Doan Trang 35 Le Nguyen Tuan Anh _	

☞ **Trong chương trình trên dùng tổ hợp 2 dòng 20 và 21 gồm 2 lệnh gets, atoi để nhập một số nguyên tránh lỗi do scanf và vùng đệm bàn phím gây ra.**

10.2.1.4 Khởi tạo structure

Ví dụ 4: Nhập vào bảng số xe, cho biết xe đó đăng kí ở tỉnh nào.

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Xac dinh bien so xe */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	#include <stdlib.h>
6	
7	#define MAX 6
8	
9	void main(void)
10	{
11	struct tinh
12	{
13	int ma;
14	char *ten;
15	};
16	tinh sds[MAX] = {{60, "Dong Nai"}, {61, "Binh Duong"}, {62, "Long An"},
17	{63, "Tien Giang"}, {64, "Vinh Long"}, {65, "Can Tho"}};
18	char ctam[10];
19	int i, in;
20	printf("Nhap vao bien so xe: ");
21	gets(ctam);
22	in = atoi(ctam);
23	
24	for(i = 0; i < MAX; i++)
25	if (sds[i].ma == in)
26	printf("Xe dang ki o tinh %s.\n", sds[i].ten);
27	getch();
28	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao bien so xe: 62F5-1152 Xe dang ki o tinh Long An _	Chạy và thử lại chương trình với 65H5-1246, 60F4-7712, 64F1-4542 Quan sát kết quả.

☞ **Dòng 22 đổi chuỗi sang số nguyên, ở ví dụ trên sau khi dòng này thực hiện giá trị của in = 62.**

10.2.1.5 Structure lồng nhau

Ví dụ 5: Nhập và in danh sách nhân viên.

Dòng	File Edit Search Run Compile Debug Project Option Window Help
1	/* Danh sach nhan vien */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	#include <stdlib.h>
6	
7	#define MAX 50
8	
9	void main(void)
10	{
11	struct giacanh
12	{
13	char vo_chong[30];
14	char con;
15	};
16	
17	struct nhanvien
18	{
19	int manv;
20	char hoten[30];
21	giacanh canhan;
22	};
23	nhanvien snv[MAX];
24	char ctam[10];
25	int i, in;
26	printf("Nhap vao so nhan vien: ");
27	gets(ctam);
28	in = atoi(ctam);
29	
30	//Nhap danh sach nhan vien
31	for(i = 0; i < in; i++)
32	{
33	printf("Nhap vao ma nhan vien thu %d: ", i + 1);
34	gets(ctam);
35	snv[i].manv = atoi(ctam);
36	printf("Nhap vao ho ten: ");
37	gets(snv[i].hoten);
38	printf("Cho biet ten vo (hoac chong): ");
39	gets(snv[i].canhan.vo_chong);
40	printf("So con: ");
41	gests(ctam);
42	}
43	
44	//in danh sach nhan vien

```
45      for(i = 0; i < in; i++)
46      {
47          printf("Ma so: %d\nHo ten: %s\n Ho ten vo (hoac chong): %s\nSo con: %d",
48                  snv[i].manv, snv[i].hoten, snv[i].canhan.vo_chong, snv[i].canhan.con);
49      getch();
50  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao so nhan vien: 3
Nhap vao ma nhan vien thu 1: 123
Nhap vao ho ten: Le Thuy Doan Trang
Nhap vao ma nhan vien thu 2: 35
Nhap vao ho ten: Le Nguyen Tuan Anh
 123  Le Thuy Doan Trang
  35  Le Nguyen Tuan Anh
_
```

Chạy và thử lại chương trình với dữ liệu khác.

Quan sát kết quả.

☞ **Trong chương trình trên dùng tổ hợp 2 dòng 20 và 21 gồm 2 lệnh gets, atoi để nhập một số nguyên tránh lỗi do scanf và vùng đệm bàn phím gây ra.**

10.2.1.6 Truyền structure sang hàm

Giống như mảng, bạn có thể truyền vào hàm qua tham biến.

Ví dụ 6: Sửa lại ví dụ 3, sử dụng hàm cho nhập và in danh sách

Dòng **File Edit Search Run Compile Debug Project Option Window Help**

```
1   /* Danh sach nhan vien */
2
3   #include <stdio.h>
4   #include <conio.h>
5   #include <stdlib.h>
6
7   #define MAX 50
8
9   //Khai bao structure toan cuc
10  struct nhanvien
11  {
12      int manv;
13      char hoten[30];
14  };
15
16  //Khai bao prototype
17  void  input(nhanvien, int);
18  void output(nhanvien, int);
19
20  //Ham nhap danh sach
21  void input(nhanvien snv[], int in)
22  {
```

```
23      char ctam[10];
24      for(int i = 0; i < in; i++)
25      {
26          printf("Nhap vao ma nhan vien thu %d: ", i + 1);
27          gets(ctam);
28          snv[i].manv = atoi(ctam);
29          printf("Nhap vao ho ten: ");
30          gets(snv[i].hoten);
31      }
32  }
33
34  //Ham in danh sach ra man hinh
35  void output(nhanvien snv[], int in)
36  {
37      for(i = 0; i < in; i++)
38          printf("%5d %s\n", snv[i].manv, snv[i].hoten);
39  }
40
41  void main(void)
42  {
43      nhanvien snv[MAX];
44      char ctam[10];
45      int i, in;
46      printf("Nhap vao so nhan vien: ");
47      gets(ctam);
48      in = atoi(ctam);
49      input(snv, in);
50      output(snv, in);
51      getch();
52  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao so nhan vien: 3
Nhap vao ma nhan vien thu 1: 123
Nhap vao ho ten: Le Thuy Doan Trang
Nhap vao ma nhan vien thu 2: 35
Nhap vao ho ten: Le Nguyen Tuan Anh
  123  Le Thuy Doan Trang
   35  Le Nguyen Tuan Anh
_
```

Chạy và thử lại chương trình với dữ liệu khác.

Quan sát kết quả.

☞ ***Giải thích chương trình***

Ở chương trình này ta phải khai báo struct nhanvien là biến toàn cục, vì khi định nghĩa hàm input và output có sử dụng kiểu dữ liệu struct nhanviên.

☞ **Bạn lưu ý rằng khi truyền struct sang hàm, không tạo bản sao mảng mới. Vì vậy struct truyền sang hàm có dạng tham biến. Nghĩa là giá trị của các phần tử trong struct sẽ bị ảnh hưởng nếu có sự thay đổi trên chúng.**

Ví dụ 7: Sửa lại ví dụ 6, từ dòng 20 đến dòng 32 như sau:

```
//Ham nhap tung nhan vien
nhanvien newnv()
{
    nhanvien snv;
    printf("Ma nhan vien:  ");
    gets(ctam);
    snv.manv = atoi(ctam);
    printf("Ho ten: ");
    gets(snv.hoten);
    return (snv);
}

//Ham nhap danh sach nhan vien
void input(nhanvien snv[], int in)
{
    for(int i = 0; i < in; i++)
    {
       printf("Nhap vao nhan vien thu %d: ", i + 1);
       snv[i] = newnv();
    }
}
```

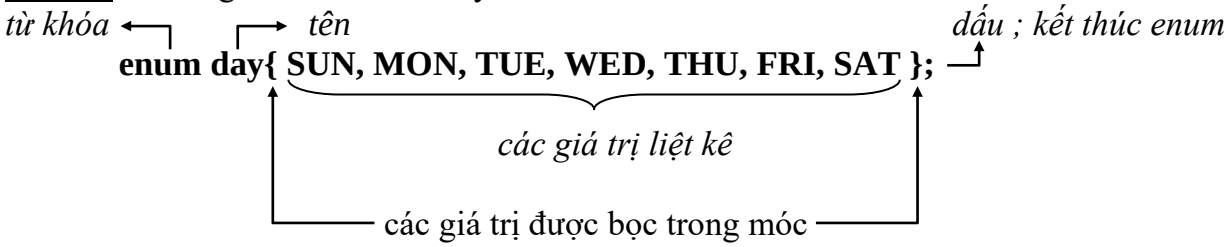
☞ **Hàm newnv có kiểu trả về là struct nhanvien**

10.2.2 Enum

Một biến là kiểu dữ liệu enum có thể nhận được một giá trị nào đó trong các giá trị được liệt kê.

10.2.2.1 Định nghĩa kiểu enum

Ví dụ 8: định nghĩa kiểu enum day



⇒ Các tên thứ (SUN, MON … SAT) trong day sẽ được đánh số lần lượt từ 0 đến 6 (SUN là 1, MON là 2… SAT là 6). Nếu bạn muốn bắt đầu bằng giá trị khác thì gán giá trị mong muốn vào và trị kế tiếp sẽ tăng lên 1.

☞ **enum** phải viết bằng chữ thường

10.2.2.2 Cách khai báo biến có kiểu enum

Ví dụ 9: **enum day ngay;** hoặc **day ngay;**
Khai báo biến **ngay** có kiểu **enum day.**

☞ vừa tạo enum day vừa khai báo biến ngay
enum day{ SUN, MON, TUE, WED, THU, FRI, SAT } **ngay**;

10.2.2.3 Sử dụng enum trong chương trình

Ví dụ 10: Tính tiền lương tuần cho nhân viên. Thứ bảy và Chủ nhật được tính phụ trội

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Tinh tien luong tuan cho nhan vien */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	
6	#define PHU_TROI_T7 1.5
7	#define PHU_TROI_CN 2.0
8	
9	//dinh nghia enum
10	enum tuan{CHU_NHAT, THU_HAI, THU_BA, THU_TU, THU_NAM, THU_SAU, THU_BAY};
11	typedef enum tuan ngay_tuan; //dinh nghia ngay_tuan la tuan
12	
13	void main(void)
14	{
15	int igio;
16	float fLuongCB, fLuongNgay, fTongLuong;
17	char cngay[][4] = {"Chu Nhat", "Thu Hai", "Thu Ba", "Thu Tu", "Thu Nam", "Thu
18	Sau", "Thu Bay"};
19	ngay_tuan engay;
20	ngay_tuan ngay_mai(ngay_tuan); //khai bao prototype
21	
22	printf("Nhap vao luong can ban: ");
23	scanf("%f", &fLuongCB);
24	
25	luong = 0.0;
26	printf("Nhap vao so gio lam viec tu Thu hai den Chu nhat:\n");
27	engay = CHU_NHAT;
28	do
29	{
30	engay = ngay_mai(engay);
31	printf("Nhap vao gio lam viec ngay %s :", cngay[engay]);
32	scanf("%d", &igio);
33	swith(engay)
34	{
35	case THU_HAI: case THU_BA: case THU_TU: case THU_NAM: case THU_SAU:
36	fLuongNgay = fLuongCB;
37	break;
38	case THU_BAY:
39	fLuongNgay = fLuongCB * PHU_TROI_T7;
40	break;
41	case CHU_NHAT:
42	fLuongNgay = fLuongCB * PHU_TROI_CN;
43	break;
44	}
45	fTongLuong += fLuongNgay * igio;
46	} while (ngay != CHU_NHAT);

```
47     printf("Tong luong tuan = %8.2f dong.\n", fTongLuong);
48     getch();
49  }
50
51  //ham chon ngay ke tiep
52  ngay_tuan ngay_mai(ngay_tuan en)
53  {
54     ngay_tuan engay_ke;
55     switch(en)
56     {
57        case CHU_NHAT : engay_ke = THU_HAI;      break;
58        case THU_HAI  : engay_ke = THU_BA;       break;
59        case THU_BA   : engay_ke = THU_TU;       break;
60        case THU_TU   : engay_ke = THU_NAM;      break;
61        case THU_NAM  : engay_ke = THU_SAU;      break;
62        case THU_SAU  : engay_ke = THU_BAY;      break;
63        case THU_BAY  : engay_ke = CHU_NHAT;     break;
64     }
65     return (engay_ke);
66  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao luong can ban: 250
Nhap vao so gio lam viec tu Thu hai den Chu nhat:
Nhap vao gio lam viec ngay Thu Hai: 7
Nhap vao gio lam viec ngay Thu Ba: 8
Nhap vao gio lam viec ngay Thu Tu: 6
Nhap vao gio lam viec ngay Thu Nam: 7
Nhap vao gio lam viec ngay Thu Sau: 8
Nhap vao gio lam viec ngay Thu Bay: 7
Nhap vao gio lam viec ngay Chu Nhat: 6
Tong luong tuan =  14625.00 dong.
_
```

Hàm chọn ngày kế tiếp trên khá dài, bạn thay từ dòng 54 đến 65 bằng câu lệnh

return (++en > 6 ? 0 : en); hoặc

return (++en % 7);

Chạy lại chương trình, quan sát, nhận xét và đánh giá kết quả với dữ liệu khác.

☞ ***Giải thích chương trình***

Ở chương trình này ta phải khai báo struct nhanvien là biến toàn cục, vì khi định nghĩa hàm input và output có sử dụng kiểu dữ liệu struct nhanviên.

10.3 Bài tập

1. Định nghĩa 1 dãy cấu trúc có thể được dùng làm danh bạ điện thoại, gồm có tên, địa chỉ, số điện thoại, với số mẫu tin tối đa là 40. Viết chương trình với các chức năng sau: nhập thông mới, tìm kiếm số điện thoại, in danh sách theo quận.

2. Viết chương trình đọc vào tên, địa chỉ, sắp xếp tên và địa chỉ theo thứ tự alphabet, sau đó hiển thị danh sách đã được sắp xếp.

3. Viết chương trình nhận vào các thông tin sau: Tên đội bóng, số trận thắng, số trận hòa, số trận thua. In ra đội bóng có số điểm cao nhất (với 1 trận thắng = 3 điểm, 1 trận hòa = 1 điểm và 1 trận thua = 0 điểm).

4. Xây dựng cấu trúc gồm: Họ tên, ngày sinh, trường, số báo danh, điểm thi. Trong đó, điểm thi là cấu trúc gồm 3 môn: Toán, Lý, Hóa. Nhập liệu vào khoảng 10 thí sinh, tìm và in ra các thí sinh có tổng điểm 3 môn >= 15.

5. Viết chương trình tạo lập và tìm kiếm dữ liệu. Nội dung yêu cầu gồm: Nhập họ và tên, địa chỉ (gồm: Quận, phường, tổ), tuổi, lương. Tìm kiếm những người ở Quận 3 có tuổi dưới 30 thu nhập từ 500.000đ trở lên và in ra màn hình.

Bài 11 :

TẬP TIN

11.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:

- Ý nghĩa của việc sử dụng tập tin (file)
- Mở, đóng file
- Ghi, đọc file số nguyên, mảng, chuỗi.
- Một số hàm xử lý tập tin.

11.2 Nội dung

11.2.1 Ví dụ ghi, đọc số nguyên

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Ghi n so nguyen vao file va doc ra tu file*/
2	
3	#include <stdio.h>
4	#include <conio.h>
5	#include <stdlib.h>
6	
7	void main(void)
8	{
9	FILE *f;
10	int in, i;
11	printf("Nhap vao so n: ");
12	scanf("%d", &in);
13	
14	//Ghi file
15	if((f = fopen("int_data.dat", "wb")) == NULL) //mo file
16	{
17	printf("Khong the mo file!.\n");
18	exit(0);
19	}
20	else
21	for(i = 1; i <= in; i++)
22	fwrite(&i, sizeof(int), 1, f); //ghi file
23	fclose(f); //dong file
24	
25	//Doc file
26	f = fopen("int_data.dat", "rb");
27	while(fread(&i, sizeof(int), 1, f) == 1)
28	printf("%d ", i);
29	fclose(f);
30	getch();
31	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so n: 10 1 2 3 4 5 6 7 8 9 10 _	Chạy và thử lại chương trình với dữ liệu khác. Quan sát kết quả.

☞ ***Giải thích chương trình***

- Dòng 9 : **FILE *f;** : khai báo biến con trỏ f có kiểu cấu trúc FILE.
- Dòng 15 : **if(f = fopen("int_data.dat", "wb") == NULL)** : là câu lệnh mở tập tin có tên int_data.dat ở mode "w" (ghi) dạng "b" (nhị phân), sau khi lệnh này thực hiện xong trả về dạng con trỏ FILE và gán cho f, nếu kết quả trả về = NULL thì không thể mở được tập tin, tập tin mở ở mode "w" nếu trên đĩa đã có sẵn tập tin này thì nội dung của nó sẽ bị ghi đè, nếu chưa có thì tập tin sẽ được tạo mới.
- Dòng 22 : **fwrite(&i, sizeof(int), 1, f);** : ghi thông tin vào tập tin, thông tin được ghi vào mỗi lần là một số nguyên i. Hàm này có 4 đối số: địa chỉ để ghi cấu trúc, kích thước của cấu trúc và số cấu trúc sẽ ghi, sau cùng là con trỏ để trỏ tới tập tin.
- Dòng 23 : **fclose(f);** : đóng tập tin
- Dòng 26 : **f = fopen("int_data.dat", "rb");** : mở tập tin có tên int_data.dat ở mode "r" (đọc) dạng "b" (nhị phân). Tập tin phải có sẵn trên đĩa.
- Dòng 27 : **while(fread(&i, sizeof(int), 1, f) == 1)** : đọc thông tin từ tập tin, mỗi lần đọc một số nguyên và lưu vào biến i. Mỗi lần đọc thành công giá trị trả về sẽ là số cấu trúc thực sự được đọc, nếu giá trị trả về = 0 báo hiệu kết thúc file.

☞ **Từ khóa FILE phải viết bằng chữ in hoa. Sử dụng fopen, fwrite, fread, fclose phải khai báo #include <stdio.h>, NULL phải viết hoa.**

11.2.2 Ghi, đọc mảng

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Ghi n so nguyen vao file va doc ra tu file*/
2	
3	#include <stdio.h>
4	#include <conio.h>
5	#include <stdlib.h>
6	
7	#define MAX 5
8	
9	void main(void)
10	{
11	FILE *f;
12	int i, ia[MAX], ib[MAX];
13	for (i = 0; i < 10; i++)
14	{
15	printf("Nhap vao mot so: ");
16	scanf("%d", &ia[i]);
17	}
18	
19	if((f = fopen("array.dat", "wb")) == NULL)
20	{
21	printf("Khong the mo file!\n");

```
22          exit(0);
23      }
24      fwrite(ia, sizeof(ia), 1, f);          //ghi mang vao file
25      fclose(f);
26
27      f = fopen("array.dat", "rb");
28      fread(ib, sizeof(ib), 1, f);           //doc mang tu file
29      for (i = 0; i < 10; i++)
30          printf("%d ", ib[i]);
31      fclose(f);
32      getch();
33  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao mot so: 3
Nhap vao mot so: 6
Nhap vao mot so: 8
Nhap vao mot so: 1
Nhap vao mot so: 9
3  6  8  1  9

_
```

Chạy và thử lại chương trình với dữ liệu khác.

Quan sát kết quả.

11.2.3 Ghi, đọc structure

Dòng **F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp

```
1   /* Danh sach nhan vien */
2
3   #include <stdio.h>
4   #include <conio.h>
5   #include <stdlib.h>
6
7   #define MAX 50
8
9   void main(void)
10  {
11      FILE *f;
12      struct nhanvien
13      {
14          int manv;
15          char hoten[30];
16      };
17      nhanvien snv[MAX], snv1[MAX];
18      char ctam[10];
19      int i, in;
20      printf("Nhap vao so nhan vien: ");
21      gets(ctam);
22      in = atoi(ctam);
23
24      //Nhap danh sach nhan vien va ghi vao file
25      if((f = fopen("struct.dat", "wb")) == NULL)
```

```
26      {
27          printf("Khong the mo file!\n");
28          exit(0);
29      }
30      fwrite(&in, sizeof(int), 1, f);                 //ghi so nhan vien vao file
31      for(i = 0; i < in; i++)
32      {
33          printf("Nhap vao ma nhan vien thu %d: ", i + 1);
34          gets(ctam);
35          snv[i].manv = atoi(ctam);
36          printf("Nhap vao ho ten: ");
37          gets(snv[i].hoten);
38          fwrite(&snv[i], sizeof(nhanvien), 1, f);    //ghi tung nhan vien vao file
39      }
40      fclose(f);
41
42      //doc danh sach nhan vien tu file va in ra
43      f = fopen("struct.dat", "rb");
44      fread(&in, sizeof(int), 1, f);                  //doc so nhan vien
45      for(i = 0; i < in; i++)
46      {
47          fread(&snv1[i], sizeof(nhanvien, 1, f);     //doc tung nhan vien in ra man hinh
48          printf("%5d %s\n", snv[i].manv, snv[i].hoten);
49      }
50      getch();
51  }
```

F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

```
Nhap vao so nhan vien: 2
Nhap vao ma nhan vien thu 1: 123
Nhap vao ho ten: Le Thuy Doan Trang
Nhap vao ma nhan vien thu 2: 35
Nhap vao ho ten: Le Nguyen Tuan Anh
  123  Le Thuy Doan Trang
   35  Le Nguyen Tuan Anh
_
```

Chạy và thử lại chương trình với dữ liệu khác.

Quan sát kết quả.

11.2.4 Các mode khác để mở tập tin

Ở 3 ví dụ trên chỉ sử dụng 2 mode "w" (ghi) và "r" (đọc), sau đây là một số mode khác:

- "a": mở để nối thêm, thông tin sẽ được ghi thêm vào cuối của tập tin đã có hoặc tạo tập tin mới nếu chưa có trên đĩa.
- "r+": mở để vừa đọc vừa ghi, tập tin phải có sẵn trên đĩa.
- "w+": mở để vừa đọc vừa ghi, nội dung của tập tin đã có trên đĩa sẽ bị ghi đè lên.
- "a+": mở để đọc và nối thêm, nếu trên đĩa chưa có tập tin nó sẽ được tạo mới.

11.2.5 Một số hàm thao tác trên file khác

Xem bài **Các hàm chuẩn**

11.3 Bài tập

Thêm chức năng ghi, đọc file ở các bài tập của bài Mảng và chuỗi, Các dữ liệu tự tạo.

Bài 12 :

ĐỆ QUY

12.1 Mục tiêu

Sau khi hoàn tất bài này học viên sẽ hiểu và vận dụng các kiến thức kĩ năng cơ bản sau:

- Ý nghĩa, phương pháp hoạt động của đệ quy.
- Có thể thay vòng lặp bằng đệ quy.

12.2 Nội dung

Bất cứ một hàm nào đó có thể triệu gọi hàm khác, nhưng ở đây một hàm nào đó có thể tự triệu gọi chính mình. Kiểu hàm như thế được gọi là **hàm đệ quy**.

Phương pháp đệ quy thường dùng phổ biến trong những ứng dụng mà cách giải quyết có thể được thể hiện bằng việc áp dụng liên tiếp cùng giải pháp cho những tập hợp con của bài toán.

Ví dụ 1: tính n!

n! = 1*2*3*…*(n-2)*(n-1)*n với n >= 1 và 0! = 1.

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Ham tinh giai thua */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	
6	void main(void)
7	{
8	int in;
9	long giaithua(int);
10	printf("Nhap vao so n: ");
11	scanf("%d", &in);
12	printf("%d! = %ld.\n", in, giaithua(in));
13	getch();
14	}
15	
16	long giaithua(int in)
17	{
18	int i;
19	long ltich = 1;
20	if (in == 0)
21	return (1L);
22	else
23	{
24	for (i = 1; i <= in; i++)
25	ltich *= i;
26	return (ltich);
27	}
28	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so n: 5 5! = 120. _	Thử lại chương trình với số liệu khác.

Với n! = 1*2*3*…*(n-2)*(n-1)*n,
ta viết lại như sau: (1*2*3*…*(n-2)*(n-1))*n = n*(n-1)! … = n*(n-1)*(n-2)!…

☞ Ta viết lại hàm giaithua bằng đệ quy như sau:

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Ham tinh giai thua */
2	
3	long giaithua(int in)
4	{
5	int i;
6	if (in == 0)
7	return (1L);
8	else
9	return (in * giaithua(in – 1));
10	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ Chạy lại chương trình, quan sát, nhận xét và đánh giá kết quả

☞ ***Giải thích hoạt động của hàm đệ quy giaithua***

Ví dụ giá trị truyền vào hàm giaithua qua biến in = 5.

- Thứ tự gọi thực hiện hàm giaithua

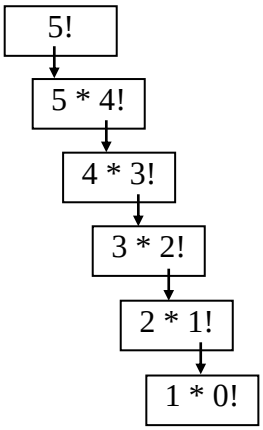
giaithua(in)	return(in * giaithua(in – 1))
5	5 * giaithua(4) = 5 * ?
4	4 * giaithua(3) = 4 * ?
3	3 * giaithua(2) = 3 * ?
2	2 * giaithua(1) = 2 * ?
1	1 * giaithua(0) = 1 * ?

Khi tham số in = 0 thì return về giá trị 1L (giá trị 1 kiểu long). Lúc này các giá trị ? bắt đầu định trị theo thứ tự ngược lại.

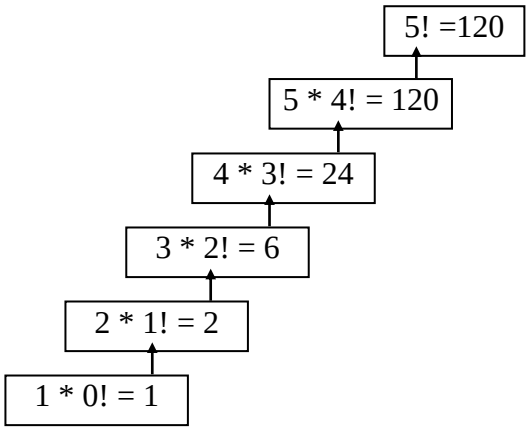
- Định trị theo thứ tự ngược lại

giaithua(in)	return(in * giaithua(in – 1))
1	1 * giaithua(0) = 1 * 1 = 1
2	2 * giaithua(1) = 2 * 1 = 2
3	3 * giaithua(2) = 3 * 2 = 6
4	4 * giaithua(3) = 4 * 6 = 24
5	5 * giaithua(4) = 5 * 24 = 120

Kết quả sau cùng ta có 5! = 120.



Thứ tự gọi đệ quy



Định trị theo thứ tự ngược lại

Ví dụ 2: Dãy số Fibonacci

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, … Bắt đầu bằng 0 và 1, các số tiếp theo bằng tổng hai số đi trước.

Dãy Fibonacci được khai báo đệ quy như sau:

$$\begin{cases} Fibonacci(0) = 0 \\ Fibonacci(1) = 1 \\ Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2) \end{cases}$$

Dòng	**F**ile **E**dit **S**earch **R**un **C**ompile **D**ebug **P**roject **O**ption **W**indow **H**elp
1	/* Tinh so fibonacci thu n */
2	
3	#include <stdio.h>
4	#include <conio.h>
5	
6	void main(void)
7	{
8	long in;
9	long fibonacci(long);
10	printf("Nhap vao so n: ");
11	scanf("%ld", &in);
12	printf("Fibonacci(%ld) = %ld.\n", in, fibonacci(in));
13	getch();
14	}
15	
16	long fibonacci(long in)
17	{
18	if (in == 0 || in == 1)
19	return in;
20	else
21	return fibonacci(in – 1) + fibonacci(in – 2);
22	}
	F1 Help **Alt-F8** Next Msg **Alt-F7** Prev Msg **Alt - F9** Compile **F9** Make **F10** Menu

☞ ***Kết quả in ra màn hình***

Nhap vao so n: 10 Fibonacci(10) = 55. –	Thử lại chương trình với số liệu khác.

? Sử dụng đệ quy hay vòng lặp

Phương pháp đệ quy không phải bao giờ cũng là giải pháp hữu hiệu nhất.Giải pháp vòng lặp có hiệu quả về mặt thời gian và vùng nhớ. Còn với đệ quy mỗi lần gọi đệ quy máy phải dành một số vùng nhớ để trữ các trị, thông số và biến cục bộ. Do đó, đệ quy tốn nhiều vùng nhớ, thời gian truyền đối mục, thiết lập vùng nhớ trung gian và trả về kết quả… Nhưng sử dụng phương pháp đệ quy trông chương trình đẹp mắt hơn vòng lặp và tính thuyết phục của nó. Điều cốt lõi khi thiết đặt chương trình phải làm thế nào hàm đệ quy có thể chấm dứt thông qua điều kiện cơ bản.

12.3 Bài tập

1. Viết hàm đệ quy tính tổng n số nguyên dương đầu tiên:
   tong (n) = n + tong (n – 1)

Bài 13 :

TRÌNH SOẠN THẢO CỦA BORLAND C

BC có hệ thống menu nhiều cấp. Để chọn một mục trong Menu bạn ấn phím **F10** (kích hoạt Menu), dùng các phím mũi tên di chuyển vệt sáng đến mục muốn chọn ấn **Enter** hoặc ấn phím có kí tự đổi màu (***phím chọn nhanh màu đỏ***). Có thể chọn nhanh mục menu trên thanh menu chính bạn ấn tổ hợp phím **Alt + phím có kí tự màu đỏ**. Ví dụ: ấn tổ hợp phím Alt + F kích hoạt menu File.

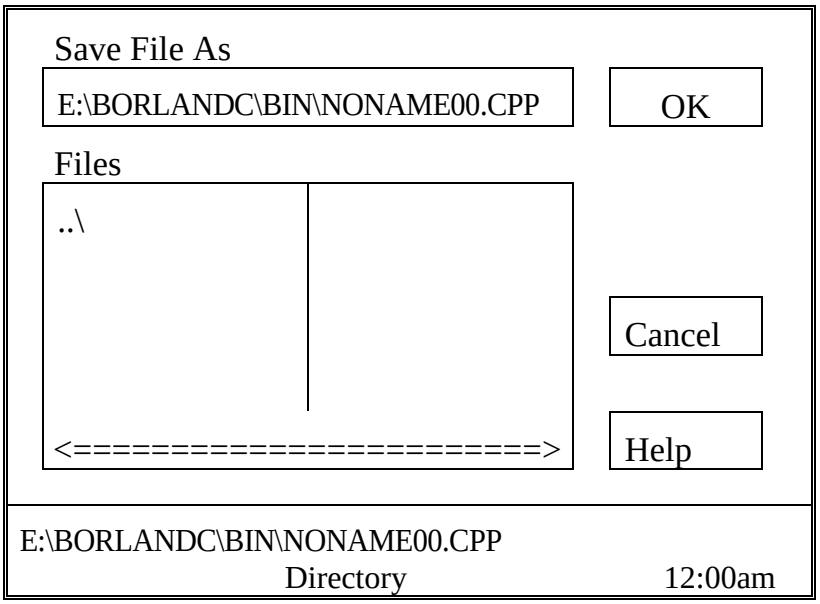
13.1 Mở tập tin soạn thảo mới

Chọn menu File -> chọn mục New -> tạo file soạn thảo mới có tên mặc định là NONAME00.CPP, NONAME01.CPP… tùy theo số lần mục New được chọn.

13.2 Lưu tập tin

13.2.1 Nếu là tập tin soạn thảo mới chưa lưu

Ấn phím F2 hoặc chọn menu File -> Save hoặc chọn menu File -> Save As sẽ xuất hiện hộp thoại Save File As



+ Chọn đường dẫn cần lưu tập tin ở hộp Files, chọn ..\ để trở về thư mục cha thư mục hiện tại.
+ Đặt tên tập tin ở hộp Save File As
+ Chọn OK
+ Hoặc có thể gõ [ổ đĩa:][đường dẫn]<tên tập tin>, chọn OK.

☞ **Dùng phím TAB để chuyển đổi vệt sáng giữa các mục trong hộp thoại.**

Ví dụ: muốn lưu tập tin có tên BT_IF1.CPP vào thư mục D:\BAITAPC

+ Bạn gõ vào hộp Save File As D: -> Enter (OK), danh sách các thư mục, tập tin của D hiển thị ở hộp Files, chọn thư mục BAITAPC ở hộp Files, gõ tên BT_IF1.CPP vào hộp Save File As, chọn OK.
+ Hoặc nếu bạn nhớ rõ đường dẫn, gõ vào hộp Save File As D:\BAITAPC\BT_IF1, chọn OK.

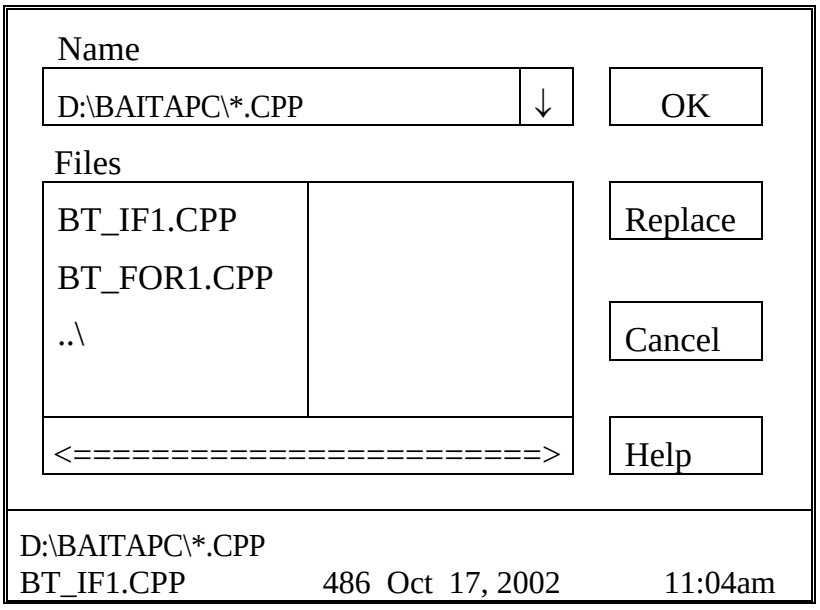
13.2.2 Nếu là tập tin đã lưu ít nhất 1 lần hoặc được mở bằng lệnh Open:

+ Ấn F2 hoặc chọn menu File -> Save, nội dung tập tin hiện hành sẽ được cập nhật nếu có thay đổi.
+ Chọn menu File -> Save As, xuất hiện hộp thoại Save File As, thực hiện các bước như mục 2.1 (nghĩa là bạn muốn lưu nội dung tập tin hiện hành với đường dẫn, tên tập tin khác).

☞ Để biết tập tin đang soạn thảo đã lưu hay chưa, bạn xem ở góc dưới trái cửa sổ, nếu có dấu hoa thị là văn bản của bạn có thay đổi và chưa được lưu.

13.3 Mở tập tin

Mở một tập tin đã có trên đĩa. Ấn phím F3 hoặc chọn menu File -> Open, hộp thoại Open a File xuất hiện:



+ Chọn đường dẫn cần mở tập tin ở hộp Files, chọn ..\ để trở về thư mục cha thư mục hiện tại.
+ Chọn tập tin cần mở ở hộp Files.
+ Chọn OK
+ Hoặc có thể gõ [ổ đĩa:][đường dẫn]<tên tập tin>, chọn OK.

Ví dụ: muốn mở tập tin có tên BT_IF1.CPP chứa trong thư mục D:\BAITAPC

+ Bạn gõ vào hộp Name D: -> Enter (OK), danh sách các thư mục, tập tin của D hiển thị ở hộp Files, chọn thư mục BAITAPC ở hộp Files, chọn tập tin BT_IF1.CPP, chọn OK.
+ Hoặc bạn có thể gõ vào hộp Name D:\BAITAPC*.CPP để hiển thị danh sách các tập tin có phần mở rộng CPP ở hộp Files, chọn tập tin BT_IF1.CPP, chọn OK.
+ Hoặc nếu bạn nhớ rõ đường dẫn, gõ vào hộp Name D:\BAITAPC\BT_IF1.CPP, chọn OK.

13.4 Các phím, tổ hợp phím thường dùng

13.4.1 Các phím di chuyển con trỏ

Phím / Tổ hợp phím	Chức năng
←	Di chuyển con trỏ sang trái một ký tự
→	Di chuyển con trỏ sang phải một ký tự
↑	Di chuyển con trỏ lên trên một dòng
↓	Di chuyển con trỏ xuống dưới một dòng
Home	Di chuyển con trỏ về đầu dòng
End	Di chuyển con trỏ về cuối dòng
PgUp (Page Up)	Lật lùi lại một trang màn hình
PgDn (Page Down)	Lật tới một trang màn hình
Ctrl – PgUp	Di chuyển con trỏ về đầu tập tin
Ctrl – PgDn	Di chuyển con trỏ về cuối tập tin
Backspace (←)	Xóa một ký tự bên trái con trỏ

Del (Delete)	Xóa một kỳ tự tại vị trí con trỏ
Ins (Insert)	Chuyển đổi giữa chế độ ghi chèn và ghi đè
Enter	Xuống một dòng

13.4.2 Các phím thao tác trên khối

Phím / Tổ hợp phím	Chức năng
Shift – →	Đánh dấu chọn một ký tự bên phải
Shift – ←	Đánh dấu chọn một ký tự bên trái
Shift – ↑	Đánh dấu chọn một hàng trên vị trí con trỏ
Shift – ↓	Đánh dấu chọn một hàng tại vị trí con trỏ
Shift – Home	Đánh dấu chọn từ đầu hàng đến vị trí con trỏ
Shift – End	Đánh dấu chọn từ vị trí con trỏ đến cuối hàng
Shift – PgUp	Đánh dấu chọn một trang lui màn hình
Shift – PgDn	Đánh dấu chọn một trang tới màn hình
Ctrl – Shift – ←	Đánh dấu chọn một từ bên trái
Ctrl – Shift – →	Đánh dấu chọn một từ bên phải
Ctrl – Shift – End	Đánh dấu chọn từ vị trí con trỏ đến cuối tập tin
Ctrl – Shift – Home	Đánh dấu chọn từ vị trí con trỏ đến đầu tập tin

13.4.3 Các thao tác xóa

Phím	Chức năng
Backspace (←)	Xóa một ký tự bên trái con trỏ
Del (Delete)	Xóa một kỳ tự tại vị trí con trỏ
Ctrl – Y	Xóa dòng tại vị trí con trỏ
Ctr – K – Y	Xóa khối
Ctrl – Q – Y	Xóa từ vị trí con trỏ đến cuối dòng
Ctrl – T	Xóa một từ tại vị trí con trỏ
Insert	Bật / tắt chế độ viết chèn / đè

13.4.4 Các thao tác copy, di chuyển

Phím / Tổ hợp phím	Chức năng
Ctrl – Insert	Sao chép khối chọn vào Clipboard
Shift – Delete	Cắt khối chọn vào Clipboard
Ctrl – Delete	Xóa khối chọn
Shift – Insert	Dán thông tin từ Clipboard vào vị trí con trỏ
Ctrl – K – R	Đọc thông tin từ tập tin vào cửa sổ soạn thảo
Ctrl – K – W	Ghi thông tin từ cửa sổ soạn thảo vào tập tin

13.4.5 Các thao tác khác

Phím / Tổ hợp phím	Chức năng
F3	Tạo tập tin mới hoặc nạp tập tin từ đĩa vào cửa sổ soạn thảo

Alt – F3	Đóng tập tin tại cửa sổ hiện hành
F2	Lưu tập tin hiện hành
F6	Chuyển đổi qua lại giữa các cửa sổ đang soạn thảo
F5	Chuyển đổi cửa sổ soạn thảo maximize ↔ restore
Alt – Backspace	Phục hồi lại thao tác trước đó
Ctrl – K – H	Ẩn / hiện dấu khối
Ctrl – Q – F	Tìm kiếm
Ctrl – L	Lập lại lần tìm kiếm sau cùng
Ctrl – Q – A	Tìm kiếm và thay thế
Ctrl – Q – [, Ctrl – Q –]	Xác định cặp ngoặc bao 1 khối lệnh
F1	Gọi giúp đỡ
Shift – F1	Hiện cửa sổ giúp đỡ theo mục
Ctrl – F1	Hiện cửa sổ giúp đỡ về hàm, toán tử… tương ứng tại vị trí con trỏ.

13.5 Ghi một khối ra đĩa

Đánh dấu chọn khối bằng các phím thao tác trên khối. Ấn tổ hợp phím Ctrl - K - W, xuất hiện hộp thoại Write Block to File, thực hiện các bước như lưu tập tin.

13.6 Chèn nội dung file từ đĩa vào vị trí con trỏ

Di chuyển con trỏ đến vị trí cần chèn nội dung, Ấn tổ hợp phím Ctrl - K - R, xuất hiện hộp thoại Read Block from File, thực hiện các bước như mở tập tin.

13.7 Tìm kiếm văn bản trong nội dung soạn thảo

Ấn tổ hợp phím Ctrl - Q - F hoặc chọn menu Search -> Find, hộp thoại Find Text xuất hiện:

+ Gõ nội dung cần tìm vào hộp Text to Find.
+ Nếu cần đánh dấu / bỏ chọn các mục sau:
  - Case-sensitive: phân biệt chữ hoa chữ thường.
  - Whole words only: tìm văn bản đứng riêng một từ.
  - Forward: Tìm xuôi.
  - Backward: Tìm ngược.
+ Chọn OK.

Khi tìm xong, muốn tìm tiếp ấn tổ hợp phím Ctrl - L hoặc chọn menu Search -> Search again.

13.8 Tìm và thay thế văn bản trong nội dung soạn thảo

Ấn tổ hợp phím Ctrl - Q - A hoặc chọn menu Search -> Replace, hộp thoại Find Text xuất hiện:

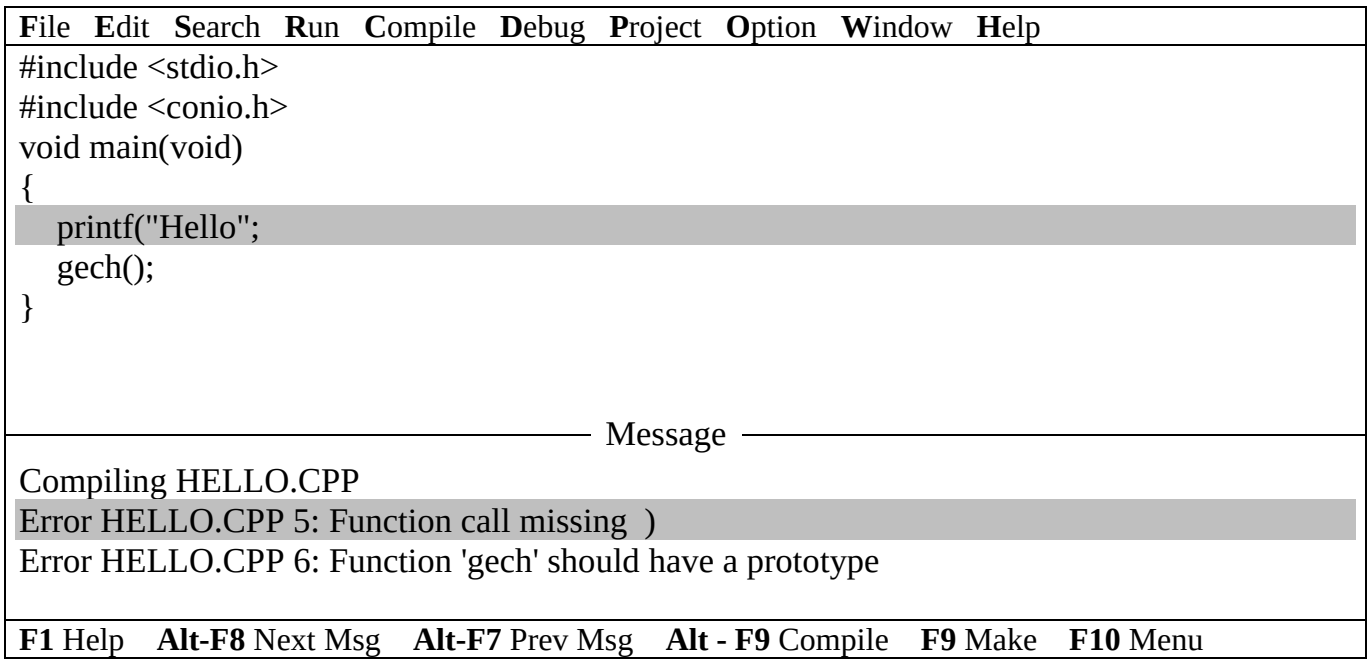
+ Gõ nội dung cần thay thế vào hộp Text to Find.
+ Gõ nội dung mới vào hộp New Text.
+ Nếu cần đánh dấu /bỏ chọn các mục sau:
  - Case-sensitive: phân biệt chữ hoa chữ thường.
  - Whole words only: tìm văn bản đứng riêng một từ.
  - Forward: Tìm xuôi.
  - Backward: Tìm ngược.
+ Chọn OK để thay thế từng văn bản được tìm thấy, chọn Change All để thay thế tất cả.

13.9 Sửa lỗi cú pháp

Khi biên dịch chương trình, nếu thành công bạn sẽ nhận được thông báo từ cửa sổ Compile (dòng cuối): **Success: Press any key**, ngược lại là thông báo lỗi **Error: Press any key**.

Nếu là thông báo lỗi, khi ấn phím bất kỳ cửa sổ Message xuất hiện chứa danh sách các lỗi. Thông báo lỗi đầu tiên được làm sáng và dòng có lỗi trong chương trình cũng được làm sáng.kèm theo dấu đỏ cho biết trình biên dịch phát hiện vị trí lỗi. Dùng phím mũi tên để di chuyển đến các thông báo lỗi khác, bạn sẽ thấy vệt sáng trong chương trình cũng sẽ chuyển đến dòng chứa lỗi tương ứng. Nếu bạn Enter tại dòng thông báo lỗi nào thì con trỏ sẽ chuyển vào cửa sổ soạn thảo tại dòng chứa lỗi tương ứng.

Ví dụ: In ra "Hello".



Vệt sáng nằm ở thông báo lỗi đầu tiên và dòng chứa lỗi tương ứng trong chương trình cũng được làm sáng: Lỗi ở dòng 5, không đóng ngoặc hàm printf.

13.10 Chạy từng bước

13.10.1 Mỗi lần 1 bước

Ở mỗi bước thực hiện ta phải bấn phím F7. Với lần bấm F7 đầu tiên, dòng đầu tiên trong chương trình (dòng main()) sẽ được làm sáng, dòng được làm sáng này được gọi là dòng chuẩn bị thực hiện, và nó sẽ được thực hiện ở lần bấm phím F7 tiếp theo. Mỗi lần bấm phím F7 dòng đang được làm sáng sẽ được thực hiện, sau đó trở về màu bình thường, và tùy theo nội dung của dòng đó mà một dòng lệnh tiếp theo nào đó sẽ được làm sáng để chuẩn bị thực hiện ở bước tiếp theo.

Ta cũng có thể dùng phím F8 thay cho F7 với những dòng không có lời gọi hàm được khai báo trong chương trình. Sự khác nhau giữa F7 và F8 chỉ xảy ra khi dòng được làm sáng có lời gọi hàm được khai báo trong chương trình.

Như vậy nhờ chạy từng bước, ta có thể dễ dàng nắm được các lỗi logic trong chương trình.

13.10.2 Tái lập lại quá trình gỡ rối

Bấm Ctrl-F2 hoặc vào menu Run chọn Program reset. Khi đó bộ nhớ dùng cho việc gỡ rối sẽ được giải tỏa, không có dòng nào được làm sáng và kết thúc quá trình gỡ rối.

13.10.3 Dùng cửa số Watch

Lần từng bước thường được dùng kèm với việc sử dụng cửa sổ Watch để theo dõi giá trị của biến trong mỗi bước thực hiện để dễ tìm ra nguyên nhân chương trình thực hiện sai.

Để làm điều đó ta phải nhập vào các biến cần theo dõi, bằng cách chọn mục Add watch của menu Break/Watch hoặc có thể bấm Ctrl-F7, sau đó nhập tên biến vào tại vị trí con trỏ trong cửa sổ Add watch và bấm Enter. Để nhập thêm tên biến vào cửa sổ này phải lập lại chức năng này hoặc bấm phím Insert.

Trong soạn thảo nếu chưa nhìn thấy cửa sổ Watch, ta bấm phím F5, khi đó trên màn hình sẽ đồng thời hiện ra cả 2 cửa sổ, để chuyển đổi giữa 2 cửa sổ này bấm phím F6. Mỗi biến trên cửa sổ Watch thực hiện trên 1 dòng. Khi cửa sổ Watch được chọn sẽ có 1 dòng được làm sáng để chỉ rằng biến đó đang được chọn. Giá trị trong cửa sổ Watch sẽ thay đổi theo kết quả của từng bước thực hiện.

13.11 Sử dụng Help (Giúp đỡ)

- Ấn phím F1 để kích hoạt màn hình Help chính.
- Muốn xem Help của hàm trong soạn thảo, di chuyển con trỏ đến vị trí hàm đó ấn tổ hợp phím Ctrl - F1
- Ấn tổ hợp phím Shift - F1 để xem danh sách các mục Help
- Ấn tổ hợp phím Alt - F1 để quay về màn hình Help trước đó.

Bài 14 :

CÁC HỆ ĐẾM

14.1 Khái niệm

Các chữ số cơ bản của một hệ đếm là các chữ số dùng để biểu diễn mọi số trong hệ đếm ấy. Hệ đếm thường gặp nhất là hệ thập phân (hệ 16). Nhưng do bản chất nhị phân của các thiết bị điện tử cho nên hầu hết dạng biểu diễn dữ liệu và các phép đại số đều thực hiện bằng hệ nhị phân (hệ 2). Hệ bát phân (hệ 8) rất ít dùng và hệ thập phân (hệ 10) là hệ chúng ta đang sử dụng để biểu diễn một con số nào đó trong cuộc sống hằng ngày.

Ví dụ 1: Hệ nhị phân gồm 2 chữ số : 0, 1
Hệ bát phân gồm 8 chữ số : 0, 1, 2, 3, 4, 5, 6, 7
Hệ thập phân gồm 10 chữ số : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hệ thập lục phân gồm 16 chữ số : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- Các chữ số trong một hệ đếm được sắp xếp theo quy tắc: Bất kỳ cơ số N nguyên dương nào, có N ký hiệu khác nhau để biểu diễn các số trong hệ thống. Giá trị của N ký hiệu này được sắp xếp từ 0 đến N-1.

Ví dụ 2: Hệ nhị phân có cơ số N = 2 : các chữ số được đánh từ 0..1
Hệ bát phân có cơ số N = 8 : các chữ số được đánh từ 0..7
Hệ thập phân có cơ số N = 10 : các chữ số được đánh từ 0..9
Hệ thập lục phân có cơ số N = 16 : các chữ số được đánh từ 0..9, A..F

- Do hệ thập lục phân có 16 chữ số, mà trong hệ thống chữ viết chỉ biểu diễn được 9 chữ số, vì vậy người ta chọn các ký tự A..F để biểu diễn 10..15 và nó cũng được xem như 1 chữ số (A, B…F) thay vì phải viết 10, 11…15 (2 chữ số)

14.2 Quy tắc

Để biểu diễn một số của một hệ đếm, ta dùng chỉ số đặt ở góc dưới phải số đó.

01101_2 : biểu thị số nhị phân.
082_8 : biểu thị số bát phân.
23_{16} : biểu thị số thập lục phân.

Đối với hệ thập phân ta có thể ghi chỉ số hoặc không ghi (nhầm hiểu), vì số thập phân là số mà ta sử dụng quen thuộc hằng ngày. Do đó, ta sử dụng công thức sau để chuyển đổi từ các hệ đếm sang hệ thập phân (cơ số 10) :

$$X = a_n a_{n-1} \dots a_1 a_0 = a_n b^n + a_{n-1} b^{n-1} + \dots + a_1 b^1 + a_0 b^0 \quad (*)$$

trong đó,

- **b** : là cơ số hệ đếm.
- $\mathbf{a_0 \dots a_n}$: là các chữ số trong một hệ đếm.
- **X** : là số thuộc một hệ đếm cơ số b.

- **Bảng các giá trị tương đương ở hệ thập phân, nhị phân, bát phân, thập lục phân. (**)**

Thập phân	Nhị phân	Bát phân	Thập lục phân
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6

7	**0111**	**7**	**7**
8	**1000**	**10**	**8**
9	**1001**	**11**	**9**
10	**1010**	**12**	**A**
11	**1011**	**13**	**B**
12	**1100**	**14**	**C**
13	**1101**	**15**	**D**
14	**1110**	**16**	**E**
15	**1111**	**17**	**F**

14.3 Chuyển đổi giữa các hệ

14.3.1 Chuyển đổi giữa hệ 2 và hệ 10

- Chuyển đổi từ hệ 2 sang hệ 10

Ví dụ 3: X = 01011_2 , để chuyển sang hệ 10 ta dùng công thức (*)

$X = 0*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$

$= 0 + 8 + 0 + 2 + 1$

$= 11$

Ví dụ 4: X = 1011010_2, để chuyển sang hệ 10 ta dùng công thức (*)

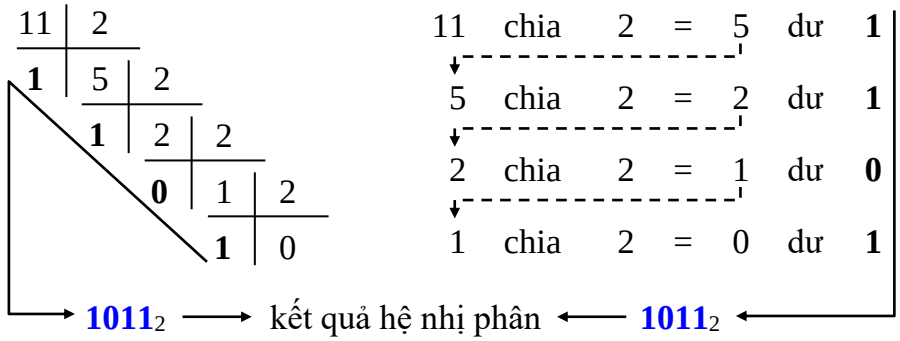
$X = 1*2^6 + 0*2^5 + 1*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0$

$= 64 + 0 + 16 + 8 + 0 + 2 + 0$

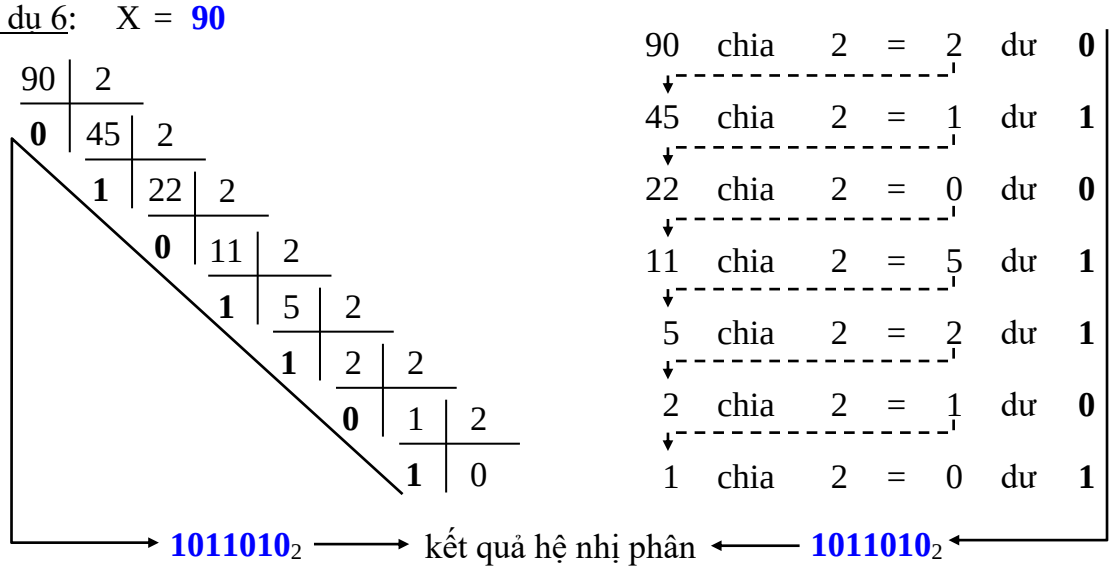
$= 90$

- Chuyển đổi từ hệ 10 sang hệ 2

Ví dụ 5: X = **11**



Ví dụ 6: X = **90**



14.3.2 Chuyển đổi giữa hệ 8 và hệ 10

- Chuyển đổi từ hệ 8 sang hệ 10

Ví dụ 7: X = 2106_8 , để chuyển sang hệ 10 ta dùng công thức (*)

X = $2*8^3 + 1*8^2 + 0*8^1 + 6*8^0$

= 1024 + 64 + 0 + 6

= 1094

Ví dụ 8: X = 130_8 , để chuyển sang hệ 10 ta dùng công thức (*)

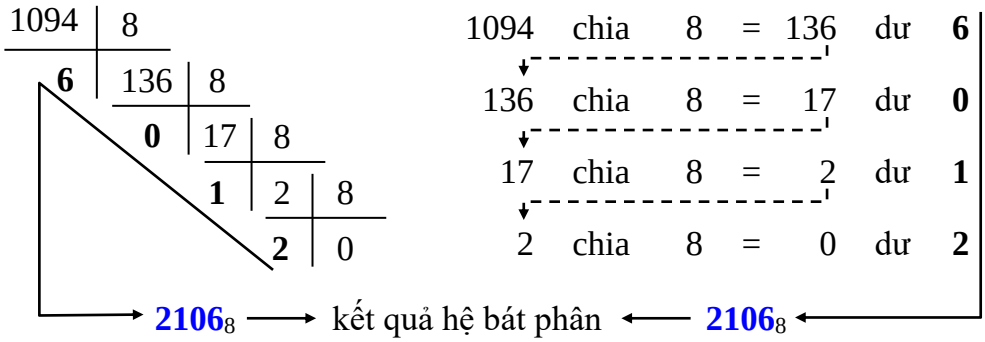
X = $1*8^2 + 3*8^1 + 0*8^0$

= 64 + 24 + 0

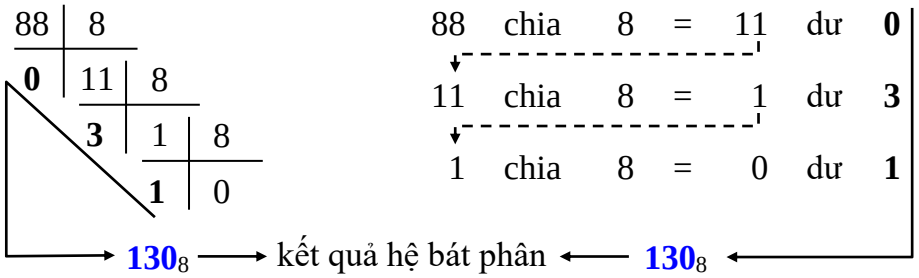
= 88

- Chuyển đổi từ hệ 10 sang hệ 8

Ví dụ 9: X = **1094**



Ví dụ 10: X = **88**



14.3.3 Chuyển đổi giữa hệ 16 và hệ 10

- Chuyển đổi từ hệ 16 sang hệ 10

Ví dụ 11: X = $F40_{16}$, để chuyển sang hệ 10 ta dùng công thức (*)

X = $15*16^2 + 4*16^1 + 0*16^0$

= 3840 + 64 + 0

= 3904

Ví dụ 12: X = $1D_{16}$, để chuyển sang hệ 10 ta dùng công thức (*)

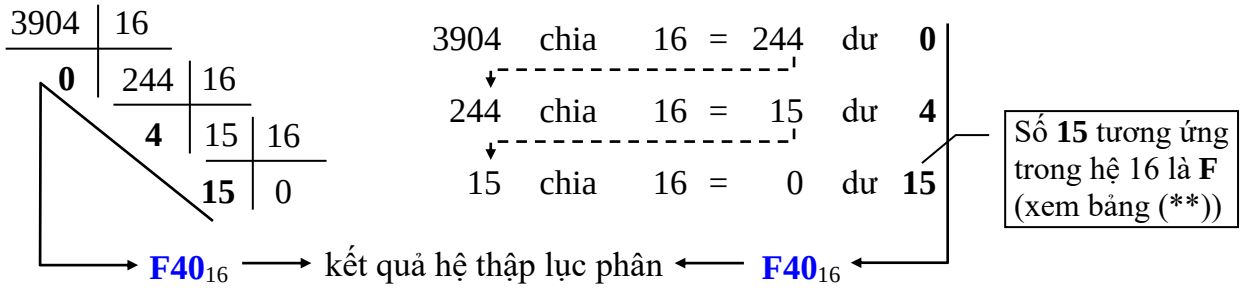
X = $1*16^1 + 13*16^0$

= 16 + 13

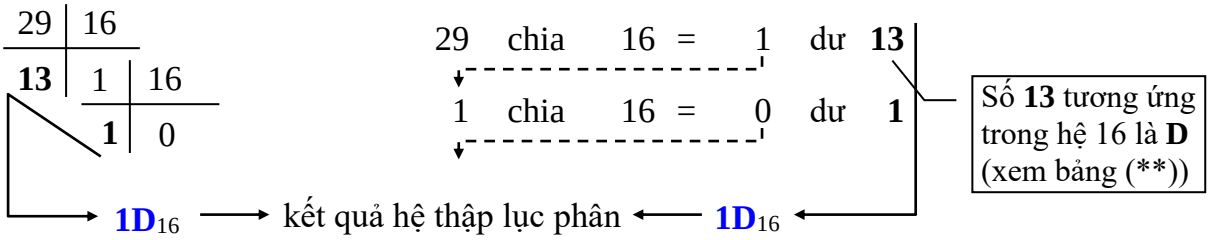
= 29

- Chuyển đổi từ hệ 10 sang hệ 16

Ví dụ 13: X = **3904**



Ví dụ 14: X = **29**



14.3.4 Chuyển đổi giữa hệ 2 và hệ 16

- Chuyển đổi từ hệ 2 sang hệ 16

Ví dụ 15: X = **01011**$_2$, để chuyển sang hệ 16 ta tra trong bảng (**)
→ X = **B**$_{16}$
Diễn giải : **0** **1011**$_2$
0 **B**$_{16}$ = **B**$_{16}$

Ví dụ 16: X = **1011010**$_2$, để chuyển sang hệ 16 ta tra trong bảng (**)
→ X = **5A**$_{16}$
Diễn giải : **101** **1010**$_2$
5 **A**$_{16}$ = **5A**$_{16}$

- Chuyển đổi từ hệ 16 sang hệ 2

Ví dụ 17: X = **B**$_{16}$, để chuyển sang hệ 2 ta tra trong bảng (**)
→ X = **1011**$_2$
Diễn giải : **B**$_{16}$
1011$_2$ = **1011**$_2$

Ví dụ 18: X = **5A**$_{16}$, để chuyển sang hệ 2 ta tra trong bảng (**)
→ X = **1011010**$_2$
Diễn giải : **5** **A**$_{16}$
0101 **1010**$_2$ = **1011010**$_2$

Bài 15 :

BIỂU THỨC VÀ PHÉP TOÁN

15.1 Biểu thức

Là sự phối hợp của những toán tử và toán hạng.

Ví dụ 1:

a + b
b = 1 + 5 * 2/i
a = 6 % (7 + 1)
x++ * 2/4 + 5 – power(i, 2)

Toán hạng sử dụng trong biểu thức có thể là hằng số, biến, hàm.

15.2 Phép toán

Trong C có 4 nhóm toán tử chính yếu sau đây:

15.2.1 Phép toán số học

+	:	cộng	áp dụng trên tất cả các toán hạng có kiểu dữ liệu char, int float, double (kể cả long, short, unsigned)
–	:	trừ	
*	:	nhân	
/	:	chia	
%	:	lấy phần dư	áp dụng trên các toán hạng có kiểu dữ liệu char, int, long

* Thứ tự ưu tiên: Đảo dấu +, – → ( ) → *, / , % → +, –

Ví dụ 2:

10%4 = 2 (10 chia 4 dư 2); 9%3 = 0 (9 chia 3 dư 0)
3 * 5 + 4 = 19
6 + 2 / 2 – 3 = 4
–7 + 2 * ((4 + 3) * 4 + 8) = 65

☞ chỉ sử dụng cặp ngoặc () trong biểu thức, cặp ngoặc đơn được thực hiện theo thứ tự ưu tiên từ trong ra ngoài.

15.2.2 Phép quan hệ

>	:	lớn hơn
>=	:	lớn hơn hoặc bằng
<	:	nhỏ hơn
<=	:	nhỏ hơn hoặc bằng
==	:	bằng
!=	:	khác

* Thứ tự ưu tiên: > , >= , < , <= → == , !=

☞ Kết quả của phép toán quan hệ là số nguyên kiểu int, bằng 1 nếu đúng, bằng 0 nếu sai. Phép toán quan hệ ngoài toán hạng được sử dụng là số còn được sử dụng với kiểu dữ liệu char.

* Thứ tự ưu tiên giữa toán tử số học và toán tử quan hệ

Toán tử số học → Toán tử quan hệ

Ví dụ 3:

4 > 10 → có giá trị 0 (sai)

4 >= 4 → có giá trị 1 (đúng)
3 == 5 → có giá trị 0 (sai)
2 <= 1 → có giá trị 0 (sai)
6 != 4 → có giá trị 1 (đúng)
6 – 3 < 4 → có giá trị 1 (đúng), tương đương (6 – 3) < 4
–2 * –4 < 3 + 2 → có giá trị 0 (sai), tương đương (–2 * –4) < (3 + 2)

15.2.3 Phép toán luận lý

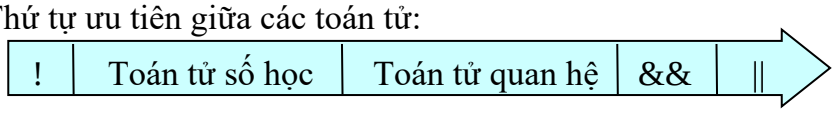
! : NOT (phép phủ định)
&&: AND (phép và)
|| : OR (phép hoặc)

Toán hạng a	Toán hạng b	!a	a && b	a || b
Khác 0	Khác 0	0 (sai)	1 (đúng)	1 (đúng)
Khác 0	Bằng 0	0 (sai)	0 (sai)	1 (đúng)
Bằng 0	Khác 0	1 (đúng)	0 (sai)	1 (đúng)
Bằng 0	Bằng 0	1 (đúng)	0 (sai)	0 (sai)

* Thứ tự ưu tiên: ! → && → ||

Ví dụ 4:
!(2 <= 1) → có giá trị 1 (đúng)
5 && 10 → có giá trị 1 (đúng)
!6 → có giá trị 0 (sai)
1 && 0 → có giá trị 0 (sai)
1 || 0 → có giá trị 1 (đúng)

* Thứ tự ưu tiên giữa các toán tử:



15.2.4 Phép toán trên bit (bitwise)

& : và (AND)
| : hoặc (OR)
^ : hoặc loại trừ (XOR)
>> : dịch phải
<< : dịch trái
~ : đảo

Bit a	Bit b	~a	a & b	a | b	a ^ b
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Ví dụ 5:
a = 13 → đổi ra hệ nhị phân → 1101
b = 10 → đổi ra hệ nhị phân → 1010

1101	1101	1101	
& 1010	| 1010	^ 1010	
= **1000**	= **1111**	= **0111**	
= **8**	= **15**	= **7**	(dạng thập phân)

a = 1235 → đổi ra hệ nhị phân → 0100 1101 0011
b = 465 → đổi ra hệ nhị phân → 0001 1101 0001

0100 1101 0011	0100 1101 0011	0100 1101 0011
& 0001 1101 0001	| 0001 1101 0001	^ 0001 1101 0001
= **0000 1101 0001**	= **0101 1101 0011**	= **0101 0000 0010**
= **209**	= **1491**	= **1282** (dạng thập phân)

15.2.5 Các phép toán khác

1. Phép toán gán

Phép gán là thay thế giá trị hiện tại của biến bằng một giá trị mới.
Các phép gán: =, +=, –=, *=, /=, %=, <<=, >>=, &=, |=, ^=.
Ví dụ 6: ta có giá trị i = 3

i = i + 3 → i = 6
i += 3 → i = 6 ≡ i = i + 3
i *= 3 → i = 9 ≡ i = i * 3

2. Phép toán tăng, giảm: ++, —

Toán tử ++ sẽ cộng thêm 1 vào toán hạng của nó, toán tử — sẽ trừ đi 1.
Ví dụ 7: ta có giá trị n = 6

+ Sau phép tính ++n hoặc n++, ta có n = 7.
+ Sau phép tính —n hoặc n— , ta có n = 5.

* Sự khác nhau giữa ++n và n++, —n và n—

+ Sau phép tính x = ++n + 2, ta có x = 9. (n tăng 1 cộng với 2 rồi gán cho x)
+ Sau phép tính x = n++ + 2, ta có x = 8. (n cộng với 2 gán cho x rồi mới tăng 1)

15.2.6 Độ ưu tiên của các phép toán

Độ ưu tiên	Các phép toán	Trình tự kết hợp
1	( ) [ ] ->	Trái sang phải
2	! ~ & * – ++ – – (type) sizeof	Phải sang trái
3	* / %	Trái sang phải
4	+ –	Trái sang phải
5	<< >>	Trái sang phải
6	< <= > >=	Trái sang phải
7	== !=	Trái sang phải
8	&	Trái sang phải
9	^	Trái sang phải
10	|	Trái sang phải
11	&&	Trái sang phải
12	||	Trái sang phải
13	? :	Phải sang trái
14	= += –= *= /= %= <<= >>= &= ^= |=	Phải sang trái
15	,	Trái sang phải

Lưu ý:

- Phép đảo (–) ở dòng 2, phép trừ (–)ở dòng 4
- Phép lấy địa chỉ (&) ở dòng 2, phép AND bit (&) ở dòng 8
- Phép lấy đối tượng con trỏ (*) ở dòng 2, phép nhân (*) ở dòng 3.

15.3 Bài tập

1. Giả sử a, b, c là biến kiểu int với a = 8, b = 3 và c = 5. Xác định giá trị các biểu thức sau:

a + b + c		a % c * 2		a * (a % b)	
a / b – c		2 * b + 3 * (a – c)		a * (b + (c – 4 * 3))	

a + c / a		c * (b / a)		5 * a – 6 / b	
a % b		(a * b) % c		5 % b % c	

2. Giả sử x, y, z là biến kiểu float với x = 8.8, y = 3.5 và z = 5.2. Xác định giá trị các biểu thức sau:

x + y + z		z / (y + x)		x / y – z * y	
5 * y + 6 * (x – z)		(z / y) + x		2.5 * x / z – (y + 6)	
x / z		2 * y / 3 * z		5 * 6 / ((x + y) / z)	
x % z		2 * y / (3 * z)		x / y*(6 + ((z–y)+3.4))	

3. Cho chương trình C với các khai báo và khởi tạo các biến như sau:

```
int i = 8, j = 5;
float x = 0.005, y = –0.01;
char c = 'c', d = 'd';
```

Hãy xác định giá trị trả về của các biểu thức sau:

(3 * i – 2 * j) % (4 * d – c)		c < d	
2 * ((i / 4) + (6 * (j – 3)) % (i + j – 4))		x >= 0	
(i – 7 * j) % (c + 3 * d) / (x – y)		x < y	
– (i + j) * –1		j != 6	
++i		c == 99	
i++		d != 100	
i++ + 5		5 * (i + j + 1) > 'd'	
++i + 5		(3 * x + y) == 0	
j– –		2 * x + (y == 0)	
– –j		!(i < j)	
j– – + i		!(d == 100)	
– –j – –5		!(x < 0)	
++x		(i > 0) && (j < 6)	
y--		(i > 0) !! (j < 5)	
i >= j		(x > y) && (i > 0) || (j < 5)	

4. Cho chương trình có các khai báo biến và khởi tạo như sau:

```
int i = 8, j = 5, k;
float x = 0.005, y = –0.01, z;
char a, b, c = 'c', d = 'd';
```

Xác định giá trị các biểu thức gán sau:

k = (i + j * 4)		z = i / j		i %= j	
x = (x + y * 1.2)		a = b = d		i += (j – 3)	
i = j		y –=x		k = (j = = 5) ? i : j	
k = (x + y)		x *= 2		k = (j > 5) ? i : j	
k = c		i /= j		i += j *= i /= 2	
i = j = 1.1		i += 2		a = (c < d) ? c : d	
z = k = x		z = (x >= 0) ? x : 0		i –= (j > 0) ? j : 0	
k = z = x		z = (y >= 0) ? y : 0		i = (i*9*(3+(8*j/3)))	

Bài 16 :

MỘT SỐ HÀM CHUẨN THƯỜNG DÙNG

16.1 Các hàm chuyển đổi dữ liệu

16.1.1 atof

double atof(const char *s); ☞ Phải khai báo **math.h** hoặc **stdlib.h**

Chuyển đổi 1 chuỗi sang giá trị double.

Ví dụ:
```
float f;
char *str = "12345.67";
f = atof(str);
Kết quả f = 12345.67;
```

16.1.2 atoi

int atoi(const char *s); ☞ Phải khai báo **stdlib.h**

Chuyển đổi 1 chuỗi sang giá trị int.

Ví dụ:
```
int i;
char *str = "12345.67";
i = atoi(str);
Kết quả i = 12345
```

16.1.3 itoa

char *itoa(int value, char *string, int radix); ☞ Phải khai báo **stdlib.h**

Chuyển đổi số nguyên value sang chuỗi string theo cơ số radix.

Ví dụ:
```
int number = 12345;
char string[25];
itoa(number, string, 10); //chuyển đổi number sang chuỗi theo cơ số 10
Kết quả string = "12345";
itoa(number, string, 2); //chuyển đổi number sang chuỗi theo cơ số 2
Kết quả string = "11000000111001";
```

16.1.4 tolower

int tolower(int ch); ☞ Phải khai báo **ctype.h**

Đổi chữ hoa sang chữ thường.

Ví dụ:
```
int len, i;
char *string  = "THIS IS A STRING";
len = strlen(string);
for (i = 0; i < len; i++)
   string[i] = tolower(string[i]); //đổi từ kí tự trong string thành chữ thường
```

16.1.5 toupper

int toupper(int ch); ☞ Phải khai báo **ctype.h**

Đổi chữ thường sang chữ hoa.

Ví dụ:
```
int len, i;
char *string  = "this is a string";
len = strlen(string);
for (i = 0; i < len; i++)
   string[i] = toupper(string[i]); //đổi từ kí tự trong string thành chữ thường
```

16.2 Các hàm xử lý chuỗi ký tự

16.2.1 strcat

char *strcat(char *dest, const char *src); ☞ Phải khai báo **string.h**

Thêm chuỗi src vào sau chuỗi dest.

16.2.2 strcpy

char *strcpy(char *dest, const char *src); ☞ Phải khai báo **string.h**

Chép chuỗi src vào dest.

```
Ví dụ:  char destination[25];
        char *blank = " ", *c = "C++", *borland = "Borland";
        strcpy(destination, borland);     //chép chuỗi borland vào destination
        strcat(destination, blank);       //thêm chuỗi blank vào sau chuỗi destination
        strcat(destination, c);           //thêm chuỗi c vào sau chuỗi destination
```

16.2.3 strcmp

int *strcmp(const char *s1, const char *s2); ☞ Phải khai báo **string.h**

So sánh chuỗi s1 với chuỗi s2. Kết quả trả về:

- < 0 nếu s1 < s2
- = 0 nếu s1 = s2
- > 0 nếu s1 > s2

```
Ví dụ:  char *buf1 = "aaa", *buf2 = "bbb", *buf3 = "aaa";
        strcmp(buf1, buf2);       //kết quả trả về - 1
        strcmp(buf1, buf3);       //kết quả trả về 0
        strcmp(buf2, buf3);       //kết quả trả về 1
```

16.2.4 strcmpi

int *strcmp(const char *s1, const char *s2); ☞ Phải khai báo **string.h**

So sánh chuỗi s1 với chuỗi s2 không phân biệt chữ hoa, chữ thường. Kết quả trả về:

- < 0 nếu s1 < s2
- = 0 nếu s1 = s2
- > 0 nếu s1 > s2

```
Ví dụ:  char *buf1 = "aaa", *buf2 = "AAA";
        strcmp(buf1, buf2);       //kết quả trả về 0
```

16.2.5 strlwr

char *strlwr(char *s); ☞ Phải khai báo **string.h**

Chuyển chuỗi s sang chữ thường

```
Ví dụ:  char *s = "Borland C";
        s = strlwr(s);            //kết quả s = "borland c"
```

16.2.6 strupr

char *strupr(char *s); ☞ Phải khai báo **string.h**

Chuyển chuỗi s sang chữ hoa

```
Ví dụ:  char *s = "Borland C";
        s = strlwr(s);            //kết quả s = "BORLAND C"
```

16.2.7 strlen

int strlen(const char *s); ☞ Phải khai báo **string.h**

Trả về độ dài chuỗi s.

```
Ví dụ:  char *s = "Borland C";
        int len_s;
        len_s = strlen(s);          //kết quả len_s = 9
```

16.3 Các hàm toán học

16.3.1 abs

int abs(int x); ☞ Phải khai báo **stblib.h**

Cho giá trị tuyệt đối của số nguyên x.

```
Ví dụ:  int num = - 123;
        num = abs(num);             //kết quả num = 123
```

16.3.2 labs

long int labs(long int x); ☞ Phải khai báo **stblib.h**

Cho giá trị tuyệt đối của số nguyên dài x.

```
Ví dụ:  int num = - 12345678L;
        num = labs(num);            //kết quả num = 12345678
```

16.3.3 rand

int rand(void); ☞ Phải khai báo **stblib.h**

Cho 1 giá trị ngẫu nhiên từ 0 đến 32767

```
Ví dụ:  int num;
        randomize();                //dùng hàm này để khởi đầu bộ số ngẫu nhiên
        num = rand();               //kết quả num = 1 con số trong khoảng 0..32767
```

16.3.4 random

int random(int num); ☞ Phải khai báo **stblib.h**

Cho 1 giá trị ngẫu nhiên từ 0 đến 32767

```
Ví dụ:  int n;
        randomize();
        n = random(100);            //kết quả n = 1 con số trong khoảng 0..99
```

16.3.5 pow

double pow(double x, double y); ☞ Phải khai báo **math.h**

Tính x mũ y

```
Ví dụ:  double x = 2.0, y = 3.0, z;
        z = pow(x, y);              //kết quả z = 8.0
```

16.3.6 sqrt

double sqrt(double x); ☞ Phải khai báo **math.h**

Tính căn bậc 2 của x.

```
Ví dụ:  double x = 4.0, y;
        y = sqrt(x);                //kết quả y = 2.0
```

16.4 Các hàm xử lý file

16.4.1 rewind

void rewind(FILE *stream); ☞ Phải khai báo **stdio.h**

Đưa con trỏ về đầu file.

16.4.2 ftell

long ftell(FILE *stream); ☞ Phải khai báo **stdio.h**

Trả về vị trí con trỏ file hiện tại.

16.4.3 fseek

int fseek(FILE *stream, long offset, int whence); ☞ Phải khai báo **stdio.h**

Di chuyển con trỏ file đến vị trí mong muốn

- **long offset**: chỉ ra số byte kể từ vị trí trước đó đến vị trí bắt đầu đọc
- **int whence**: chỉ ra điểm xuất phát để tính offset gồm các giá trị sau: SEEK_SET (đầu tập tin), SEEK_CUR (tại vị trí con trỏ hiện hành), SEEK_END (cuối tập tin).

CHươNG 1

CáC KHáI NIệM Cơ Sở của LậP TRìNH HướNG Đối TượNG

Chương 1 trình bày những vấn đề sau:

- *Thảo luận về cách tiếp cận hướng đối tượng, những nhược điểm của lập trình truyền thống và các đặc điểm của lập trình hướng đối tượng.*
- *Các khái niệm cơ sở của phương pháp hướng đối tượng:*
  - *Đối tượng*
  - *Lớp*
  - *Trừu tượng hóa dữ liệu và bao gói thông tin*
  - *Kế thừa*
  - *Tương ứng bội*
  - *Liên kết động*
  - *Truyền thông báo*
- *Các bước cần thiết để thiết kế chương trình theo hướng đối tượng*
- *Các ưu điểm của lập trình hướng đối tượng*
- *Các ngôn ngữ hướng đối tượng*
- *Một số ứng dụng của lập trình hướng đối tượng*

Bài 17 : 1.1. Giới thiệu

17.11.1.1. Tiếp cận hướng đối tượng

Trong thế giới thực, chung quanh chúng ta là những đối tượng, đó là các thực thể có mối quan hệ với nhau. Ví dụ các *phòng* trong một công ty kinh doanh được xem như những đối tượng. Các *phòng* ở đây có thể là: phòng quản lý, phòng bán hàng, phòng kế toán, phòng tiếp thị,... Mỗi *phòng* ngoài những cán bộ đảm nhiệm những công việc cụ thể, còn có những dữ liệu riêng như thông tin về nhân viên, doanh số bán hàng, hoặc các dữ liệu khác có liên quan đến bộ phận đó. Việc phân chia các phòng chức năng trong công ty sẽ tạo điều kiện dễ dàng cho việc quản lý các hoạt động. Mỗi nhân viên trong phòng sẽ điều khiển và xử lý dữ liệu của phòng đó. Ví dụ phòng kế toán phụ trách về lương bổng nhân viên trong công ty. Nếu bạn đang ở bộ phận tiếp thị và cần tìm thông tin chi tiết về lương của đơn vị mình thì sẽ gởi yêu cầu về phòng kế toán. Với cách làm này bạn được đảm bảo là chỉ có nhân viên của bộ phận kế toán được quyền truy cập dữ liệu và cung cấp thông tin cho bạn. Điều này cũng cho thấy rằng, không có người nào thuộc bộ phận khác có thể truy cập và thay đổi dữ liệu của bộ phận kế toán. Khái niệm như thế về đối tượng hầu như có thể được mở rộng đối với mọi lĩnh vực trong đời sống xã hội và hơn nữa - đối với việc tổ chức chương trình. Mọi ứng dụng có thể được định nghĩa như một tập các thực thể - hoặc các đối tượng, sao cho quá trình tái tạo những suy nghĩa của chúng ta là gần sát nhất về thế giới thực.

Trong phần tiếp theo chúng ta sẽ xem xét phương pháp lập trình truyền thống để từ đó thấy rằng vì sao chúng ta cần chuyển sang phương pháp lập trình hướng đối tượng.

17.21.1.2. Những nhược điểm của lập trình hướng thủ tục

Cách tiếp cận lập trình truyền thống là lập trình hướng thủ tục (LTHTT). Theo cách tiếp cận này thì một hệ thống phần mềm được xem như là dãy các công việc cần thực hiện như đọc dữ liệu, tính toán, xử lý, lập báo cáo và in ấn kết quả v.v... Mỗi công việc đó sẽ được thực hiện bởi một số hàm nhất định. Như vậy trọng tâm của cách tiếp cận này là các hàm chức năng. LTHTT sử dụng kỹ thuật phân rã hàm chức năng theo cách tiếp cận trên xuống (top-down) để tạo ra cấu trúc phân cấp. Các ngôn ngữ lập trình bậc cao như COBOL, FORTRAN, PASCAL, C, v.v..., là những ngôn ngữ lập trình hướng thủ tục.
Những nhược điểm chính của LTHTT là:

- Chương trình khó kiểm soát và khó khăn trong việc bổ sung, nâng cấp chương trình. Chương trình được xây dựng theo cách TCHTT thực chất là danh sách các câu lệnh mà theo đó máy tính cần thực hiện. Danh sách các lệnh đó được tổ chức thành từng nhóm theo đơn vị cấu trúc của ngôn ngữ lập trình và được gọi là hàm/thủ tục. Trong chương trình có nhiều hàm/thủ tục, thường thì có nhiều thành phần dữ liệu quan trọng sẽ được khai báo tổng thể (global) để các hàm/thủ tục có thể truy nhập, đọc và làm thay đổi giá trị của biến tổng thể. Điều này sẽ làm cho chương trình rất khó kiểm soát, nhất là đối với các chương trình lớn, phức tạp thì vấn đề càng trở nên khó khăn hơn. Khi ta muốn thay đổi, bổ sung cấu trúc dữ liệu dùng chung cho một số hàm/thủ tục thì phải thay đổi hầu như tất cả các hàm/thủ tục liên quan đến dữ liệu đó.
- Mô hình được xây dựng theo cách tiếp cận hướng thủ tục không mô tả được đầy đủ, trung thực hệ thống trong thực tế.
- Phương pháp TCHTT đặt trọng tâm vào hàm là hướng tới hoạt động sẽ không thực sự tương ứng với các thực thể trong hệ thống của thế giới thực.

17.31.1.3. Lập trình hướng đối tượng

Lập trình hướng đối tượng (Object Oriented Programming - LTHĐT) là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng thuật giải, xây dựng chương trình. Đối tượng được xây dựng trên cơ sở gắn cấu trúc dữ liệu với các phương thức (các hàm/thủ tục) sẽ thể hiện được đúng cách mà chúng ta suy nghĩ, bao quát về thế giới thực. LTHĐT cho phép ta kết hợp những tri thức bao quát về các quá trình với những khái niệm trừu tượng được sử dụng trong máy tính.

Điểm căn bản của phương pháp LTHĐT là thiết kế chương trình xoay quanh dữ liệu của hệ thống. Nghĩa là các thao tác xử lý của hệ thống được gắn liền với dữ liệu và như vậy khi có sự thay đổi của cấu trúc dữ liệu thì chỉ ảnh hưởng đến một số ít các phương thức xử lý liên quan.

LTHĐT không cho phép dữ liệu chuyển động tự do trong hệ thống. Dữ liệu được gắn chặt với từng phương thức thành các vùng riêng mà các phương thức đó tác động lên và nó được bảo vệ để cấm việc truy nhập tùy tiện từ bên ngoài. LTHĐT cho phép phân tích bài toán thành tập các thực thể được gọi là các đối tượng và sau đó xây dựng các dữ liệu cùng với các phương thức xung quanh các đối tượng đó.

Tóm lại LTHĐT có những đặc tính chủ yếu như sau:

1. Tập trung vào dữ liệu thay cho các phương thức.
2. Chương trình được chia thành các lớp đối tượng.
3. Các cấu trúc dữ liệu được thiết kế sao cho đặc tả được các đối tượng.
4. Các phương thức xác định trên các vùng dữ liệu của đối tượng được gắn với nhau trên cấu trúc dữ liệu đó.
5. Dữ liệu được bao bọc, che dấu và không cho phép các thành phần bên ngoài truy nhập tự do.
6. Các đối tượng trao đổi với nhau thông qua các phương thức.
7. Dữ liệu và các phương thức mới có thể dễ dàng bổ sung vào đối tượng nào đó khi cần thiết.
8. Chương trình được thiết kế theo cách tiếp cận bottom-up (dưới -lên).

Bài 18 : 1.2. Các khái niệm cơ bản của lập trình hướng đối tượng

Những khái niệm cơ bản trong LTHĐT bao gồm: Đối tượng; Lớp; Trừu tượng hóa dữ liệu, bao gói thông tin; Kế thừa; Tương ứng bội; Liên kết động; Truyền thông báo.

18.11.2.1. Đối tượng

Trong thế giới thực, khái niệm đối tượng được hiểu như là một thực thể, nó có thể là người, vật hoặc một bảng dữ liệu cần xử lý trong chương trình,... Trong LTHĐT thì đối tượng là biến thể hiện của *lớp*.

18.21.2.2. Lớp

Lớp là một khái niệm mới trong LTHĐT so với kỹ thuật LTHTT. Nó là một bản mẫu mô tả các thông tin cấu trúc dữ liệu và các thao tác hợp lệ của các phần tử dữ liệu. Khi một phần tử dữ liệu được khai báo là phần tử của một lớp thì nó được gọi là *đối tượng*. Các hàm được định nghĩa hợp lệ trong một lớp được gọi là các *phương thức* (method) và chúng là các hàm duy nhất có thể xử lý dữ liệu của các đối tượng của lớp đó. Mỗi đối tượng có riêng cho mình một bản sao các phần tử dữ liệu của lớp. Mỗi lớp bao gồm: danh sách các thuộc tính (attribute) và danh sách các phương thức để xử lý các thuộc tính đó. Công thức phản ánh bản chất của kỹ thuật LTHĐT là:

Đối tượng = Dữ liệu + Phương thức

Chẳng hạn, chúng ta xét lớp HINH_CN bao gồm các thuộc tính: (x1,y1) toạ độ góc trên bên trái, d,r là chiều dài và chiều rộng của HCN. Các phương thức nhập số liệu cho HCN, hàm tính diện tích, chu vi và hàm hiển thị. Lớp HINH_CN có thể được mô tả như sau:

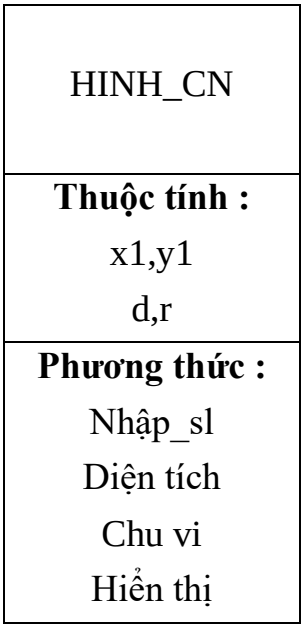
18.3

18.4

18.5

18.6

18.7



Hình 2.2 Mô tả lớp HINH_CN

Chú ý: Trong LTHĐT thì lớp là khái niệm tĩnh, có thể nhận biết ngay từ văn bản chương trình, ngược lại đối tượng là khái niệm động, nó được xác định trong bộ nhớ của máy tính, nơi đối tượng chiếm một vùng bộ nhớ lúc thực hiện chương trình. Đối tượng được tạo ra để xử lý thông tin, thực hiện nhiệm vụ được thiết kế, sau đó bị hủy bỏ khi đối tượng đó hết vai trò.

18.81.2.3. Trừu tượng hóa dữ liệu và bao gói thông tin

Trừu tượng hóa là cách biểu diễn những đặc tính chính và bỏ qua những chi tiết vụn vặt hoặc những giải thích. Khi xây dựng các lớp, ta phải sử dụng khái niệm trừu tượng hóa. Ví dụ ta có thể định nghĩa một lớp để mô tả các đối tượng trong không gian hình học bao gồm các thuộc tính trừu tượng như là kích thước, hình dáng, màu sắc và các phương thức xác định trên các thuộc tính này.

Việc đóng gói dữ liệu và các phương thức vào một đơn vị cấu trúc lớp được xem như một nguyên tắc *bao gói thông tin*. Dữ liệu được tổ chức sao cho thế giới bên ngoài (các đối tượng ở lớp khác) không truy nhập vào, mà chỉ cho phép các phương thức trong cùng lớp hoặc trong những lớp có quan hệ kế thừa với nhau mới được quyền truy nhập. Chính các phương thức của lớp sẽ đóng vai trò như là giao diện giữa dữ liệu của đối tượng và phần còn lại của chương trình. Nguyên tắc bao gói dữ liệu để ngăn cấm sự truy nhập trực tiếp trong lập trình được gọi là sự che giấu thông tin.

18.91.2.4. Kế thừa

Kế thừa là quá trình mà các đối tượng của lớp này được quyền sử dụng một số tính chất của các đối tượng của lớp khác. Sự kế thừa cho phép ta định nghĩa một lớp mới trên cơ sở các lớp đã tồn tại. Lớp mới này, ngoài những thành phần được kế thừa, sẽ có thêm những thuộc tính và các hàm mới. Nguyên lý kế thừa hỗ trợ cho việc tạo ra cấu trúc phân cấp các lớp.

18.10 1.2.5. Tương ứng bội

Tương ứng bội là khả năng của một khái niệm (chẳng hạn các phép toán) có thể sử dụng với nhiều chức năng khác nhau. Ví dụ, phép + có thể biểu diễn cho phép "cộng" các số nguyên (int), số thực (float), số phức (complex) hoặc xâu ký tự (string) v.v... Hành vi của phép toán tương ứng bội phụ thuộc vào kiểu dữ liệu mà nó sử dụng để xử lý.

Tương ứng bội đóng vai quan trọng trong việc tạo ra các đối tượng có cấu trúc bên trong khác nhau nhưng cùng dùng chung một giao diện bên ngoài (như tên gọi).

18.11 1.2.6. Liên kết động

Liên kết động là dạng liên kết các thủ tục và hàm khi chương trình thực hiện lời gọi tới các hàm, thủ tục đó. Như vậy trong liên kết động, nội dung của đoạn chương trình ứng với thủ tục, hàm sẽ không được biết cho đến khi thực hiện lời gọi tới thủ tục, hàm đó.

18.12 1.2.7. Truyền thông báo

Các đối tượng gửi và nhận thông tin với nhau giống như con người trao đổi với nhau. Chính nguyên lý trao đổi thông tin bằng cách truyền thông báo cho phép ta dễ dàng xây dựng được hệ thống mô phỏng gần hơn những hệ thống trong thế giới thực. *Truyền thông báo* cho một đối tượng là yêu cầu đối tượng thực hiện một việc gì đó. Cách ứng xử của đối tượng được mô tả bên trong lớp thông qua các phương thức.

Trong chương trình, thông báo gửi đến cho một đối tượng chính là yêu cầu thực hiện một công việc cụ thể, nghĩa là sử dụng những hàm tương ứng để xử lý dữ liệu đã được khai báo trong đối tượng đó. Vì vậy, trong thông báo phải chỉ ra được hàm cần thực hiện trong đối tượng nhận thông báo. Thông báo truyền đi cũng phải xác định tên đối tượng và thông tin truyền đi. Ví dụ, lớp CONGNHAN có thể hiện là đối tượng cụ thể được đại diện bởi Hoten nhận được thông báo cần tính lương thông qua hàm TINHLUONG đã được xác định trong lớp CONGNHAN. Thông báo đó sẽ được xử lý như sau:

Trong chương trình hướng đối tượng, mỗi đối tượng chỉ tồn tại trong thời gian nhất định. Đối tượng được tạo ra khi nó được khai báo và sẽ bị hủy bỏ khi chương trình ra khỏi miền xác định của đối tượng đó. Sự trao đổi thông tin chỉ có thể thực hiện trong thời gian đối tượng tồn tại.

Bài 19 : 1.3. Các bước cần thiết để thiết kế chương trình theo hướng đối tượng

Chương trình theo hướng đối tượng bao gồm một tập các đối tượng và mối quan hệ giữa các đối tượng với nhau. Vì vậy, lập trình trong ngôn ngữ hướng đối tượng bao gồm các bước sau:

1. Xác định các dạng đối tượng (lớp) của bài tóan.

2. Tìm kiếm các đặc tính chung (dữ liệu chung) trong các dạng đối tượng này, những gì chúng cùng nhau chia xẻ.

3. Xác định lớp cơ sở dựa trên cơ sở các đặc tính chung của các dạng đối tượng.

4. Từ lớp cơ sở, xây dựng các lớp dẫn xuất chứa các thành phần, những đặc tính không chung còn lại của các dạng đối tượng. Ngoài ra, ta còn đưa ra các lớp có quan hệ với các lớp cơ sở và lớp dẫn xuất.

Bài 20 : 1.4. Các ưu điểm của lập trình hướng đối tượng

Cách tiếp cận hướng đối tượng giải quyết được nhiều vấn đề tồn tại trong quá trình phát triển phần mềm và tạo ra được những sản phẩm phần mềm có chất lượng cao. Những ưu điểm chính của LTHĐT là:

1. Thông qua nguyên lý kế thừa, có thể loại bỏ được những đoạn chương trình lặp lại trong quá trình mô tả các lớp và mở rộng khả năng sử dụng các lớp đã được xây dựng.

2. Chương trình được xây dựng từ những đơn thể (đối tượng) trao đổi với nhau nên việc thiết kế và lập trình sẽ được thực hiện theo quy trình nhất định chứ không phải dựa vào kinh nghiệm và kỹ thuật như trước. Điều này đảm bảo rút ngắn được thời gian xây dựng hệ thống và tăng năng suất lao động.

3. Nguyên lý che giấu thông tin giúp người lập trình tạo ra được những chương trình an toàn không bị thay bởi những đoạn chương trình khác.

4. Có thể xây dựng được ánh xạ các đối tượng của bài toán vào đối tượng của chương trình.

5. Cách tiếp cận thiết kế đặt trọng tâm vào đối tượng, giúp chúng ta xây dựng được mô hình chi tiết và gần với dạng cài đặt hơn.

6. Những hệ thống hướng đối tượng dễ mở rộng, nâng cấp thành những hệ lớn hơn.

7. Kỹ thuật truyền thông báo trong việc trao đổi thông tin giữa các đối tượng giúp cho việc mô tả giao diện với các hệ thống bên ngoài trở nên đơn giản hơn.

8. Có thể quản lý được độ phức tạp của những sản phẩm phần mềm.

Không phải trong hệ thống hướng đối tượng nào cũng có tất cả các tính chất nêu trên. Khả năng có các tính chất đó còn phụ thuộc vào lĩnh vực ứng dụng của dự án tin học và vào phương pháp thực hiện của người phát triển phần mềm.

Bài 21 : 1.5. Các ngôn ngữ hướng đối tượng

Lập trình hướng đối tượng không là đặc quyền của một ngôn ngữ nào đặc biệt. Cũng giống như lập trình có cấu trúc, những khái niệm trong lập trình hướng đối tượng có thể cài đặt trong những ngôn ngữ lập trình như C hoặc Pascal,... Tuy nhiên, đối với những chương trình lớn thì vấn đề lập trình sẽ trở nên phức tạp. Những ngôn ngữ được thiết kế đặc biệt, hỗ trợ cho việc mô tả, cài đặt các khái niệm của phương pháp hướng đối tượng được gọi chung là ngôn ngữ đối tượng. Dựa vào khả năng đáp ứng các khái niệm về hướng đối tượng, ta có thể chia ra làm hai loại:

1. Ngôn ngữ lập trình dựa trên đối tượng
2. Ngôn ngữ lập trình hướng đối tượng

Lập trình dựa trên đối tượng là kiểu lập trình hỗ trợ chính cho việc bao gói, che giấu thông tin và định danh các đối tượng. Lập trình dựa trên đối tượng có những đặc tính sau:

- Bao gói dữ liệu
- Cơ chế che giấu và truy nhập dữ liệu
- Tự động tạo lập và xóa bỏ các đối tượng
- Phép toán tải bội

Ngôn ngữ hỗ trợ cho kiểu lập trình trên được gọi là ngôn ngữ lập trình dựa trên đối tượng. Ngôn ngữ trong lớp này không hỗ trợ cho việc thực hiện kế thừa và liên kết động, chẳng hạn Ada là ngôn ngữ lập trình dựa trên đối tượng.

Lập trình hướng đối tượng là kiểu lập trình dựa trên đối tượng và bổ sung thêm nhiều cấu trúc để cài đặt những quan hệ về kế thừa và liên kết động. Vì vậy đặc tính của LTHĐT có thể viết một cách ngắn gọn như sau:

Các đặc tính dựa trên đối tượng + kế thừa + liên kết động.

Ngôn ngữ hỗ trợ cho những đặc tính trên được gọi là ngôn ngữ LTHĐT, ví dụ như C++, Smalltalk, Object Pascal v.v...

Việc chọn một ngôn ngữ để cài đặt phần mềm phụ thuộc nhiều vào các đặc tính và yêu cầu của bài toán ứng dụng, vào khả năng sử dụng lại của những chương trình đã có và vào tổ chức của nhóm tham gia xây dựng phần mềm.

Bài 22 : 1.6. Một số ứng dụng của LTHĐT

LTHĐT đang được ứng dụng để phát triển phần mềm trong nhiều lĩnh vực khác nhau. Trong số đó, có ứng dụng quan trọng và nổi tiếng nhất hiện nay là hệ điều hành Windows của hãng Microsoft đã được phát triển dựa trên kỹ thuật LTHĐT. Một số những lĩnh vực ứng dụng chính của kỹ thuật LTHĐT bao gồm:

+ Những hệ thống làm việc theo thời gian thực.
+ Trong lĩnh vực mô hình hóa hoặc mô phỏng các quá trình
+ Các cơ sở dữ liệu hướng đối tượng.
+ Những hệ siêu văn bản, multimedia

+ Lĩnh vực trí tuệ nhân tạo và các hệ chuyên gia.

+ Lập trình song song và mạng nơ-ron.

+ Những hệ tự động hóa văn phòng và trợ giúp quyết định.

...

Chương 2

các mở rộng của ngôn ngữ C++

Chương 2 trình bày những vấn đề sau đây:

- *Giới thiệu chung về ngôn ngữ C++*
- *Một số mở rộng của ngôn ngữ C++ so với ngôn ngữ C*
- *Các đặc tính của C++ hỗ trợ lập trình hướng đối tượng*
- *Vào ra trong C++*
- *Cấp phát và giải phóng bộ nhớ*
- *Biến tham chiếu, hằng tham chiếu*
- *Truyền tham số cho hàm theo tham chiếu*
- *Hàm trả về giá trị tham chiếu*
- *Hàm với tham số có giá trị mặc định*
- *Các hàm nội tuyến (inline)*
- *Hàm tải bội*

Bài 23 : 2.1. Giới thiệu chung về C++

C++ là ngôn ngữ lập trình hướng đối tượng và là sự mở rộng của ngôn ngữ C. Vì vậy mọi khái niệm trong C đều dùng được trong C++. Phần lớn các chương trình C đều có thể chạy được trong C++. Trong chương này chỉ tập trung giới thiệu những khái niệm, đặc tính mới của C++ hỗ trợ cho lập trình hướng đối tượng. Một số kiến thức có trong C++ nhưng đã có trong ngôn ngữ C sẽ không được trình bày lại ở đây.

Bài 24 : 2.2. Một số mở rộng của C++ so với C

24.12.2.1. Đặt lời chú thích

Ngoài kiểu chú thích trong C bằng /* ... */ , C++ đưa thêm một kiểu chú thích thứ hai, đó là chú thích bắt đầu bằng //. Kiểu chú thích /*...*/ được dùng cho các khối chú thích lớn gồm nhiều dòng, còn kiểu // được dùng cho các chú thích trên một dòng. Chương trình dịch sẽ bỏ qua tất cả các chú thích trong chương trình.

Ví dụ:
```
/* Đây là
   câu chú thích trên nhiều dòng  */
// Đây là chú thích trên một dòng
```

24.2 2.2.2. Khai báo biến

Trong C tất cả các câu lệnh khai báo biến, mảng cục bộ phải đặt tại đầu khối. Vì vậy vị trí khai báo và vị trí sử dụng của biến có thể ở cách khá xa nhau, điều này gây khó khăn trong việc kiểm soát chương trình. C++ đã khắc phục nhược điểm này bằng cách cho phép các lệnh khai báo biến có thể đặt bất kỳ chỗ nào trong chương trình trước khi các

biến được sử dụng. Phạm vi hoạt động của các biến kiểu này là khối trong đó biến được khai báo.

Ví dụ 2.1 Chương trình sau đây nhập một dãy số thực rồi sắp xếp theo thứ tự tăng dần:

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>
void main()
 {
    int n;
    printf("\n So phan tu cua day N=");
    scanf("%d",&n);
    float *x=(float*)malloc((n+1)*sizeof(float));
    for (int i=0;i<n;i++)
     {
    printf("\n X[%d]=",i);
    scanf("%f",x+i);
      }
    for(i=0;i<n-1;++i)
     for (int j=i+1;j<n;++j)
       if (x[i]>x[j])
   {
     float tg=x[i];
     x[i]=x[j];
     x[j]=tg;
    }
    printf("\n Day sau khi sap xep\n");
    for (i=0;i<n;++i)
      printf("%0.2f ",x[i]);
    getch();
 }
```

24.32.2.3. Phép chuyển kiểu bắt buộc

Ngoài phép chuyển kiểu bắt buộc được viết trong C theo cú pháp:

(kiểu) biểu thức

C++ còn sử dụng một phép chuyển kiểu mới như sau:

Kiểu(biểu thức)

Phép chuyển kiểu này có dạng như một hàm số chuyển kiểu đang được gọi. Cách chuyển kiểu này thường được sử dụng trong thực tế.

Ví dụ 2.2 Chương trình sau đây tính sau tổng S = $1+\frac{1}{2}+\frac{1}{3}+...+\frac{1}{n}$

Với n là một số nguyên dương nhập từ bàn phím.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n;
    printf("\n So phan tu cua day N=");
    scanf("%d",&n);
    float s=0.0;
    for (int i=1;i<=n;++i)
      s+= float(1)/float(i); //chuyen kieu theo C++
    printf("S=%0.2f",s);
    getch();
}
```

24.42.2.4. Lấy địa chỉ các phần tử mảng thực 2 chiều

Trong C không cho phép dùng phép toán & để lấy địa chỉ của các phần tử mảng thực 2 chiều. Vì vậy khi nhập một ma trận thực (dùng hàm scanf()) ta phải nhập qua một biến trung gian sau đó mới gán cho các phần tử mảng.

C++ cho phép dùng phép toán & để lấy địa chỉ các phần tử mảng thực 2 chiều, do đó thể dùng hàm scanf() để nhập trực tiếp vào các phần tử mảng.

Ví dụ 2.3 Chương trình sau đây cho phép nhập một mảng thực cấp 20x20 và tìm các phần tử có giá trị lớn nhất.

```
#include <conio.h>
#include <stdio.h>
void main()
{
    float a[20][20],smax;
    int m,n,i,j,imax,jmax;
    clrscr();
   puts(" Cho biet so hang va so cot cua ma tran: ");
    scanf("%d%d",&m,&n);
    for (i=0;i<m;++i)
```

```
   for (j=0;j<n;++j)
    {  printf("\n a[%d][%d]=",i,j);
    scanf("%f",&a[i][j]);
     }
    smax=a[0][0];
    imax=0;
    jmax=0;
    for (i=0;i<m;++i)
    for(j=0;j<n;++j)
      if(smax<a[i][j])
       {
       smax=a[i][j];
       imax=i;
      jmax=j;
       }
    puts("\n\n Ma tran");
    for (i=0;i<m;++i)
   for (j=0;j<n;++j)
    {
      if (j==0) puts("");
    printf("%6.1f",a[i][j]);
       }
    puts("\n\n Phan tu max:");

    printf("\n Co gia tri=%6.1f", smax);
    printf("\n\n Tai hang %d cot %d",imax,jmax);
    getch();
}
```

Bài 25 : 2.3. Vào ra trong C++

Để xuất dữ liệu ra màn hình và nhập dữ liệu từ bàn phím, trong C++ vẫn có thể dùng hàm printf() và scanf(), ngoài ra trong C++ ta có thể dùng dòng xuất/nhập chuẩn để nhập/xuất dữ liệu thông qua hai biến đối tượng của dòng (stream object) là **cout** và **cin**.

25.12.3.1. Xuất dữ liệu

Cú pháp: cout << biểu thức 1<<. . .<< biểu thức N;

Trong đó *cout* được định nghĩa trước như một đối tượng biểu diễn cho thiết bị xuất chuẩn của C++ là màn hình, *cout* được sử dụng kết hợp với toán tử chèn << để hiển thị giá trị các biểu thức 1, 2,..., N ra màn hình.

25.22.3.2. Nhập dữ liệu

Cú pháp: cin >>biến 1>>. . . >>biến N;

Toán tử *cin* được định nghĩa trước như một đối tượng biểu diễn cho thiết bị vào chuẩn của C++ là bàn phím, *cin* được sử dụng kết hợp với toán tử trích >> để nhập dữ liệu từ bàn phím cho các biến 1, 2, ..., N.

Chú ý:

- Để nhập một chuỗi không quá n ký tự và lưu vào mảng một chiều a (kiểu char) có thể dùng hàm cin.get như sau: cin.get(a,n);
- Toán tử nhập cin>> sẽ để lại ký tự chuyển dòng '\n' trong bộ đệm. Ký tự này có thể làm trôi phương thức cin.get. Để khắc phục tình trạng trên cần dùng phương thức cin.ignore(1) để bỏ qua một ký tự chuyển dòng.
- Để sử dụng các loại toán tử và phương thức nói trên cần khai báo tập tin dẫn hướng iostream.h

25.32.3.3. Định dạng khi in ra màn hình

- Để quy định số thực được hiển thị ra màn hình với p chữ số sau dấu chấm thập phân, ta sử dụng đồng thời các hàm sau:

```
setiosflags(ios::showpoint);    // Bật cờ hiệu showpoint(p)
setprecision(p);
```

Các hàm này cần đặt trong toán tử xuất như sau:

```
cout<<setiosflag(ios::showpoint)<<setprecision(p);
```

Câu lệnh trên sẽ có hiệu lực đối với tất cả các toán tử xuất tiếp theo cho đến khi gặp một câu lệnh định dạng mới.

- Để quy định độ rộng tối thiểu để hiển thị là k vị trí cho giá trị (nguyên, thực, chuỗi) ta dùng hàm: setw(k)

Hàm này cần đặt trong toán tử xuất và nó chỉ có hiệu lực cho một giá trị được in gần nhất. Các giá trị in ra tiếp theo sẽ có độ rộng tối thiểu mặc định là 0, như vậy câu lệnh:

```
cout<<setw(6)<<"Khoa"<<"CNTT"
```

sẽ in ra chuỗi " KhoaCNTT".

Ví dụ 2.4 Chương trình sau cho phép nhập một danh sách không quá 100 thí sinh. Dữ liệu mỗi thí sinh gồm họ tên, các điểm thi môn 1, môn 2, môn 3. Sau đó in danh sách thí sinh theo thứ tự giảm dần của tổng điểm.

```
#include <iostream.h>
```

```
#include <conio.h>
#include <iomanip.h>
void main()
{
  struct
  {
    char ht[25];
    float d1,d2,d3,td;
   }ts[100],tg;
    int n,i,j;
    clrscr();
    cout << "So thi sinh:";
    cin >> n;
    for (i=0;i<n;++i)
    {
    cout << "\n Thi sinh:"<<i;
    cout << "\n Ho ten:";
    cin.ignore(1);
    cin.get(ts[i].ht,25);
    cout << "Diem cac mon thi :";
    cin>>ts[i].d1>>ts[i].d2>>ts[i].d3;
    ts[i].td=ts[i].d1+ts[i].d2+ts[i].d3;
    }
    for (i=0;i<n-1;++i)
      for(j=i+1;j<n;++j)
       if(ts[i].td<ts[j].td)
    {
       tg=ts[i];
       ts[i]=ts[j];
       ts[j]=tg;
    }
    cout<< "\ Danh sach thi sinh sau khi sap xep :";
    for (i=0;i<n;++i)
     {
        cout<< "\n" <<
        setw(25)<<ts[i].ht<<setw(5)<<ts[i].td;
```

```
    }
   getch();}
```

Ví dụ 2.5 Chương trình sau đây cho phép nhập một mảng thực cấp 50x50. Sau đó in ma trận dưới dạng bảng và tìm một phần tử lớn nhất.

```
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
void main()
{
   float a[50][50],smax;
   int m,n,i,j,imax,jmax;
   clrscr();
   cout<< "Nhap so hang va cot:";
   cin>>m>>n;
   for (i=0;i<m;++i)
  for (j=0;j<n;++j)
     {

     cout<< "a["<<i<<","<<j<<"]=";
     cin>> a[i][j];
     }
   smax= a[0][0];
   imax=0;
   jmax=0;
   for (i=0;i<m;++i)
  for (j=0;j<n;++j)
     if (smax<a[i][j])
        {
         smax=a[i][j];
         imax=i;
         jmax=j;
        }
   cout << "\n\n Mang da nhap";
  cout << setiosflags(ios::showpoint)<<setprecision(1);
   for (i=0;i<m;++i)
  for (j=0;j<n;++j)
```

```
      {
      if (j==0) cout<<"\n";
      cout << setw(6)<<a[i][j];
      }
    cout << "\n\n"<< "Phan tu max:"<< "\n";
    cout << "co gia tri ="<<setw(6)<<smax;
    cout<<"\nTai hang"<<imax<< " cot "<<jmax;
    getch();
}
```

Bài 26 : 2.4. Cấp phát và giải phóng bộ nhớ

Trong C có thể sử dụng các hàm cấp phát bộ nhớ như malloc(), calloc() và hàm free() để giải phóng bộ nhớ được cấp phát. C++ đưa thêm một cách thức mới để thực hiện việc cấp phát và giải phóng bộ nhớ bằng cách dùng hai toán tử **new** và **delete.**

26.12.4.1. Toán tử new để cấp phát bộ nhớ

Toán tử new thay cho hàm malloc() và calloc() của C có cú pháp như sau:

new Tên kiểu ;

hoặc new (Tên kiểu);

Trong đó Tên kiểu là kiểu dữ liệu của biến con trỏ, nó có thể là: các kiểu dữ liệu chuẩn như int, float, double, char,... hoặc các kiểu do người lập trình định nghĩa như mảng, cấu trúc, lớp,...

Chú ý: Để cấp phát bộ nhớ cho mảng một chiều, dùng cú pháp như sau:

Biến con trỏ = new kiểu[n];

Trong đó n là số nguyên dương xác định số phần tử của mảng.

Ví dụ: float *p = new float; //cấp phát bộ nhớ cho biến con trỏ p có kiểu int

int *a = new int[100]; //cấp phát bộ nhớ để lưu trữ mảng một chiều a
// gồm 100 phần tử

Khi sử dụng toán tử new để cấp phát bộ nhớ, nếu không đủ bộ nhớ để cấp phát, new sẽ trả lại giá trị NULL cho con trỏ. Đoạn chương trình sau minh họa cách kiểm tra lỗi cấp phát bộ nhớ:

```
    double *p;
    int n;
    cout<< "\n So phan tu : ";
    cin>>n;
    p = new double[n]
    if (p == NULL)
  {
```

```
      cout << "Loi cap phat bo nho";
      exit(0);
     }
```

26.22.4.2. Toán tử delete

Toán tử delete thay cho hàm free() của C, nó có cú pháp như sau:

delete con trỏ ;

Ví dụ 2.6 Chương trình sau minh hoạ cách dùng new để cấp phát bộ nhớ chứa n thí sinh. Mỗi thí sinh là một cấu trúc gồm các trường ht(họ tên), sobd(số báo danh), và td(tổng điểm). Chương trình sẽ nhập n, cấp phát bộ nhớ chứa n thí sinh, kiểm tra lỗi cấp phát bộ nhớ, nhập n thí sinh, sắp xếp thí sinh theo thứ tự giảm của tổng điểm, in danh sách thí sinh sau khi sắp xếp, giải phóng bộ nhớ đã cấp phát.

```
#include <iomanip.h>
#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct TS
{
  char ht[20];
  long sobd;
  float td;
};
void main(void)
{
   TS *ts;
   int n;
   cout<<"\nSo thi sinh n = ";
   cin>>n;
   ts = new TS[n+1];
   if (ts == NULL)
     {
    cout << "\n Loi cap phat vung nho";
    getch();
    exit(0);
     }
```

```
    for (int i=0;i<n;++i)
    {
        cout << "\n Thi sinh thu "<<i;
        cout<< "\n Ho ten";
        cin.ignore(1);
        cin.get (ts[i].ht,20);
        cout << "so bao danh";
        cin>> ts[i].sobd;
        cout<< "tong diem:";
        cin>>ts[i].td;
     }
    for (i=0;i<n-1;++i)
     for (int j=i+1;j<n;++j)
      if  (ts[i].td<ts[j].td)
       {
    TS  tg=ts[i];
        ts[i]=ts[j];
         ts[j]=tg;
       }
    cout << setiosflags(ios::showpoint)<<setprecision(1);
    for (i=0;i<n;++i)
    cout << "\n" << setw(20)<<ts[i].ht<<setw(6)<<ts[i].td;
    delete ts;
    getch();
}
```

Bài 27 : 2.5. Biến tham chiếu

Trong C có 2 loại biến là: Biến giá trị dùng để chứa dữ liệu (nguyên, thực, ký tự,...) và biến con trỏ dùng để chứa địa chỉ. Các biến này đều được cung cấp bộ nhớ và có địa chỉ. C++ cho phép sử dụng loại biến thứ ba là biến tham chiếu. Biến tham chiếu là một tên khác (bí danh) cho biến đã định nghĩa trước đó. Cú pháp khai báo biến tham chiếu như sau:

Kiểu &Biến tham chiếu = Biến;

Biến tham chiếu có đặc điểm là nó được dùng làm bí danh cho một biến (kiểu giá trị) nào đó và sử dụng vùng nhớ của biến này.

Ví dụ: Với câu lệnh: int a, &tong=a; thì *tong* là bí danh của biến a và biến *tong* dùng chung vùng nhớ của biến a. Lúc này, trong mọi câu lệnh, viết *a* hay viết *tong* đều có ý

nghĩa như nhau, vì đều truy nhập đến cùng một vùng nhớ. Mọi sự thay đổi đối với biến *tong* đều ảnh hưởng đối với biến a và ngược lại.

Ví dụ: int a, &tong = a;

```
tong =1;       //a=1
cout<< tong;   //in ra số 1
tong++;        //a=2
++a;           //a=3
cout<<tong;    //in ra số 3
```

Chú ý:

- Trong khai báo biến tham chiếu phải chỉ rõ tham chiếu đến biến nào.
- Biến tham chiếu có thể tham chiếu đến một phần tử mảng, nhưng không cho phép khai báo mảng tham chiếu.
- Biến tham chiếu có thể tham chiếu đến một hằng. Khi đó nó sử dụng vùng nhớ của hằng và có thể làm thay đổi giá trị chứa trong vùng nhớ này.
- Biến tham chiếu thường được sử dụng làm đối của hàm để cho phép hàm truy nhập đến các tham biến trong lời gọi hàm

Bài 28 : 2.6. Hằng tham chiếu

Cú pháp khai báo hằng tham chiếu như sau:

const Kiểu dữ liệu &Biến = Biến/Hằng;

Ví dụ :

```
int n = 10;
const int &m = n;
const int &p = 123;
```

Hằng tham chiếu có thể tham chiếu đến một biến hoặc một hằng.

Chú ý:

- Biến tham chiếu và hằng tham chiếu khác nhau ở chỗ: không cho phép dùng hằng tham chiếu để làm thay đổi giá trị của vùng nhớ mà nó tham chiếu.

Ví dụ:

```
int    y=12, z;
const int &p = y //Hằng tham chiếu p tham chiếu đến biến y
p = p + 1;       //Sai, trình biên dịch sẽ thông báo lỗi
```

- Hằng tham chiếu cho phép sử dụng giá trị chứa trong một vùng nhớ, nhưng không cho phép thay đổi giá trị này.
- Hằng tham chiếu thường được sử dụng làm tham số của hàm để cho phép sử dụng giá trị của các tham số trong lời gọi hàm, nhưng tránh làm thay đổi giá trị tham số.

Bài 29 : 2.7. Truyền tham số cho hàm theo tham chiếu

Trong C chỉ có một cách truyền dữ liệu cho hàm là truyền theo theo giá trị. Chương trình sẽ tạo ra các bản sao của các tham số thực sự trong lời gọi hàm và sẽ thao tác trên các bản sao này chứ không xử lý trực tiếp với các tham số thực sự. Cơ chế này rất tốt nếu khi thực hiện hàm trong chương trình không cần làm thay đổi giá trị của biến gốc. Tuy nhiên, nhiều khi ta lại muốn những tham số đó thay đổi khi thực hiện hàm trong chương trình. C++ cung cấp thêm cách truyền dữ liệu cho hàm theo tham chiếu bằng cách dùng đối là tham chiếu. Cách làm này có ưu điểm là không cần tạo ra các bản sao của các tham số, do dó tiết kiệm bộ nhớ và thời gian chạy máy. Mặt khác, hàm này sẽ thao tác trực tiếp trên vùng nhớ của các tham số, do đó dễ dàng thay đổi giá trị các tham số khi cần.

Ví dụ 2.7 Chương trình sau sẽ nhập dãy số thực, sắp xếp dãy theo thứ tự tăng dần và hiển thị ra màn hình.

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
void nhapds(double *a,int n)
{
  for(int i=0;i<n;++i)
    {
     cout<<"\n Phan tu thu "<<i<<":";
     cin>>a[i];
    }
 }
 void hv(double &x,double &y)
    {
      double tam=x;x=y;y=tam;
    }
void sapxep(double *a,int n)
 {
  for(int i=0;i<n-1;++i)
     for(int j=i+1;j<n;++j)
        if(a[i]>a[j])
         hv(a[i],a[j]);
 }
void main()
{
```

```
    double x[100];
    int i,n;
    clrscr();
    cout<<"\n nhap so phan tu N = ";
    cin>>n;
    nhapds(x,n);
    sapxep(x,n);
    cout<<"\nCac phan tu mang sau khi sap xep :";
    for(i=0;i<n;++i)
  printf("\n%6.2f",x[i]);
  getch();
}
```

Ví dụ 2.8 Chương trình sẽ nhập dữ liệu một danh sách thí sinh bao gồm họ tên, điểm các môn 1, môn 2, môn 3 và in danh sách thí sinh:

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <iomanip.h>
 struct TS
  {
      char ht[20];
    float d1,d2,d3,td;
  };
 void ints(const TS &ts)
 {
  cout<<setiosflags(ios::showpoint)<<setprecision(1);
  cout<<"\n ho ten"<<setw(20)<<ts.ht<<setw(6)<<ts.td;
 }

void nhapsl(TS *ts,int n)
  {
  for(int i=0;i<n;++i)
  {
    cout<<"\n Thi sinh"<<i;
    cout<<"\n ho ten  ";
    cin.ignore(1);
```

```
    cin.get(ts[i].ht,25);
    cout<<"Nhap diem cac mon thi : ";
      cin>>ts[i].d1>>ts[i].d2>>ts[i].d3;
    ts[i].td=ts[i].d1+ts[i].d2+ts[i].d3;
   }
  }
  void hvts(TS &ts1,TS &ts2)
  {
    TS tg=ts1;
    ts1=ts2;
    ts2=tg;
  }
  void sapxep(TS *ts,int n)
  {
   for(int i=0;i<n-1;++i)
    for(int j=i+1;j<n;++j)
     if(ts[i].td<ts[j].td)
       hvts(ts[i],ts[j]) ;
    }
void main()
{
  TS ts[100];
  int n,i;
  clrscr();
  cout<<"So thi sinh : ";
  cin>>n;
  nhapsl(ts,n);
  sapxep(ts,n);
  float dc;
  cout<<"\n\nDanh sach thi sinh \n";
  for(i=0;i<n;++i)
    if(ts[i].td>=dc)
      ints(ts[i]);
    else
     break;
  getch();
```

```
}
```

Ví dụ 2.9 Chương trình sau sẽ nhập một mảng thực kích thước 20x20, in mảng đã nhập và in các phần tử lớn nhất và nhỏ nhất trên mỗi hàng của mảng.

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <iomanip.h>
void nhapmt(float a[20][20],int m,int n)
   {
    for(int i=0;i<m;++i)
      for(int j=0;j<n;++j)
     {
       cout<<"\n a["<<i<<","<<j<<"]=";
       cin>> a[i][j];
     }
    }
void inmt(float a[20][20],int m,int n)
 {
  cout<<setiosflags(ios::showpoint)<<setprecision(1);
  cout<<"\nMang da nhap : ";
  for(int i=0;i<m;++i)
  for(int j=0;j<n;++j)
   {
    if(j==0) cout<<"\n";
    cout<<setw(6)<<a[i][j];
   }
 }
 void maxminds(float *x,int n,int &vtmax,int &vtmin)
    {
    vtmax=vtmin=0;
    for(int i=1;i<n;++i)
    {
      if(x[i]>x[vtmax])    vtmax=i;
      if(x[i]<x[vtmin])    vtmin=i;
     }
   }
```

```
void main()
{
    float a[20][20];
    int m,n;
    clrscr();
    cout<<"\n Nhap so hang va so cot : ";
    cin>>m>>n;
    nhapmt(a,m,n);
  inmt(a,m,n);
    float *p=(float*) a;
    int vtmax,vtmin;
    for(int i=0;i<m;++i)
  {
    p=((float*)a)+i*20;
    maxminds(p,n,vtmax,vtmin);
    printf("\n Hang %d phan tu max=%6.1f tai cot
              %d",i,p[vtmax],vtmax);
     printf("\n Phan tu min=%6.1f tai cot
            %d",p[vtmin],vtmin);
  }
    getch();
   }
```

Bài 30 : 2.8. Hàm trả về giá trị tham chiếu

C++ cho phép hàm trả về giá trị là một tham chiếu, lúc này định nghĩa của hàm có dạng như sau :

```
Kiểu    &Tên hàm(...)
  {    //thân hàm
 return   <biến phạm vi toàn cục>;
  }
```

Trong trường hợp này biểu thức được trả lại trong câu lệnh *return* phải là tên của một biến xác định từ bên ngoài hàm, bởi vì khi đó mới có thể sử dụng được giá trị của hàm. Khi ta trả về một tham chiếu đến một biến cục bộ khai báo bên trong hàm, biến cục bộ này sẽ bị mất đi khi kết thúc thực hiện hàm. Do vậy tham chiếu của hàm sẽ không còn ý nghĩa nữa.

Khi giá trị trả về của hàm là tham chiếu, ta có thể gặp các câu lệnh gán hơi khác thường, trong đó vế trái là một lời gọi hàm chứ không phải là tên của một biến. Điều này hoàn toàn hợp lý, bởi vì bản thân hàm đó có giá trị trả về là một tham chiếu. Nói cách khác, vế trái của lệnh gán có thể là lời gọi đến một hàm có giá trị trả về là một tham chiếu.Xem các ví dụ sau:

Ví dụ 2.10

```
#include <iostream.h>
#include <conio.h>
 int z;
 int &f()// ham tra ve mot bi danh cua bien toan bo z
   {
    return z;
   }
void main()
{
   f()=50;//z=50
   cout<<"\nz="<<z;
   getch();
 }
```

Ví dụ 2.11

```
    #include <iostreams.h>
    #include <stdio.h>
    #include <string.h>
    #include <conio.h>
    int & max(int& a, int& b);
    void main()
    {
      clrscr();
      int b =10, a= 7, c= 20;
      cout << "Max a,b : "<<max(b,a) << endl;
      max(b,a)++;
      cout << "Gia tri b va a :"<< b <<"  "<<a <<endl;
      max(b,c)=5;
      cout << "Gia tri b va a va c :"<<b<<" "<<a
           <<"  "<<c<< endl;
     }
```

```
int &max(int &a, int &b)
{
  return a>b ? a:b;
}
```

Kết quả trên màn hình sẽ là :

```
Max a,b : 10
Gia tri cua b va a : 11   7
Gia tri cua b va a va c : 11   7    5
```

Bài 31 : 2.9. Hàm với tham số có giá trị mặc định

C++ cho phép xây dựng hàm với các tham số được khởi gán giá trị mặc định.

Quy tắc xây dựng hàm với tham số mặc định như sau:

- Các đối có giá trị mặc định cần là các tham số cuối cùng tính từ trái qua phải.
- Nếu chương trình sử dụng khai báo nguyên mẫu hàm thì các tham số mặc định cần được khởi gán trong nguyên mẫu hàm, không được khởi gán khởi gán lại cho các đối mặc định trong dòng đầu của định nghĩa hàm.

```
void f(int a, float x, char *st="TRUNG TAM", int b=1, double y = 1.234);
void f(int a, float x, char *st="TRUNG TAM", int b=1, double y = 1.234)
    {
   //Các câu lệnh
    }
```

- Khi xây dựng hàm, nếu không khai báo nguyên mẫu, thì các đối mặc định được khởi gán trong dòng đầu của định nghĩa hàm, ví dụ:

```
void f(int a, float x, char *st="TRUNG TAM", int b=1, double y = 1.234)
    {
     //Các câu lệnh
    }
```

Chú ý: Đối với các hàm có tham số mặc định thì lời gọi hàm cần viết theo quy định: Các tham số vắng mặt trong lời gọi hàm tương ứng với các tham số mặc định cuối cùng (tính từ trái sang phải), ví dụ với hàm:

```
void f(int a, float x, char *st="TRUNG TAM", int b=1, double y = 1.234);
```

thì các lời gọi hàm đúng:

```
f(3,3.4,"TIN HOC",10,1.0);//Đầy đủ tham số
f(3,3.4,"ABC");          //Thiếu 2 tham số cuối
f(3,3.4);                //Thiếu 3 tham số cuối
```

Các lời gọi hàm sai:

f(3);
f(3,3.4, ,10);

Ví dụ 2.12

```
#include <iostream.h>
#include <conio.h>
 void ht(char *dc="TRUNG TAM",int n=5);
 void ht(char *dc,int n)
 {
 for(int i=0;i<n;++i)
  cout<<"\n" <<dc;
 }
  void main()
  {
  ht();// in dong chu "TRUNG TAM"tren 5 dong
  ht("ABC",3);// in dong chu "ABC"tren 3 dong
  ht("DEF");// in dong chu "DEF"tren 5 dong
  getch();
}
```

Bài 32 : 2.10. Các hàm nội tuyến (inline)

Việc tổ chức chương trình thành các hàm có ưu điểm chương trình được chia thành các đơn vị độc lập, điều này giảm được kích thước chương trình, vì mỗi đoạn chương trình thực hiện nhiệm vụ của hàm được thay bằng lời gọi hàm. Tuy nhiên hàm cũng có nhược điểm là làm là chậm tốc độ thực hiện chương trình vì phải thực hiện một số thao tác có tính thủ tục mỗi khi gọi hàm như: cấp phát vùng nhớ cho các tham số và biến cục bộ, truyền dữ liệu của các tham số cho các đối, giải phóng vùng nhớ trước khi thoát khỏi hàm.

C++ cho khả năng khắc phục được nhược điểm nói trên bằng cách dùng hàm nội tuyến. Để biến một hàm thành hàm nội tuyến ta viết thêm từ khóa **inline** vào trước khai báo nguyên mẫu hàm.

Chú ý: Trong mọi trường hợp, từ khóa inline phải xuất hiện trước các lời gọi hàm thì trình biên dịch mới biết cần xử lý hàm theo kiểu inline.

Ví dụ hàm f() trong chương trình sau sẽ không phải là hàm nội tuyến vì inline viết sau lời gọi hàm.

Ví dụ 2.13

```
#include <iostream.h>
```

```
#include <conio.h>
void main()
{
  int s ;
  s=f(5,6);
  cout<<s;
  getch();
}
  inline int f(int a,int b)
  {
   return a*b;
  }
```

Chú ý:

- Chương trình dịch các hàm inline như tương tự như các macro, nghĩa là nó sẽ thay đổi lời gọi hàm bằng một đoạn chương trình thực hiện nhiệm vụ hàm. Cách làm này sẽ tăng tốc độ chương trình do không phải thực hiện các thao tác có tính thủ tục khi gọi hàm nhưng lại làm tăng khối lượng bộ nhớ chương trình (nhất là đối với các hàm nội tuyến có nhiều câu lệnh). Vì vậy chỉ nên dùng hàm inline đối với các hàm có nội dung đơn giản.
- Không phải khi gặp từ khoá inline là chương trình dịch nhất thiết phải xử lý hàm theo kiểu nội tuyến. Từ khoá inline chỉ là một từ khoá gợi ý cho chương trình dịch chứ không phải là một mệnh lệnh bắt buộc.

Ví dụ 2.14 Chương trình sau sử dụng hàm inline để tính chu vi và diện tích hình chữ nhật.

```
#include <iostream.h>
#include <conio.h>
inline void dtcvhcn(int a,int b,int &dt,int &cv)
 {
  dt=a*b;
  cv=2*(a+b);
 }
void main()
{
  int a[20],b[20],cv[20],dt[20],n;
  cout<<"\n So hinh chu nhat";
  cin>>n;
  for(int i=0;i<n;++i)
```

```
  {
   cout<<"\n Nhap 2 canh cua hinh chu nhat"<<i<<":";
   cin>>a[i]>>b[i];
   dtcvhcn(a[i],b[i],dt[i],cv[i]);
  }
  clrscr();
  for(i=0;i<n;++i)
   {
    cout<<"\n Hinh chu nhat thu "<<i+1<<":";
    cout<<"\n Do dai hai canh "<<a[i]<<"va"<<b[i];
    cout<<"\n dien tich "<<dt[i];
    cout<<"\n chu vi "<<cv[i];
   }
 getch();
}
```

Ví dụ 2.15 Một cách viết khác của chương trình trong ví dụ 2.14

```
#include <iostream.h>
#include <conio.h>
inline void dtcvhcn(int a,int b,int &dt,int &cv);
void main()
{
  int a[20],b[20],cv[20],dt[20],n;
  cout<<"\n So hinh chu nhat";
  cin>>n;
  for(int i=0;i<n;++i)
  {
   cout<<"\n Nhap 2 canh cua hinh chu nhat"<<i<<":";
   cin>>a[i]>>b[i];
   dtcvhcn(a[i],b[i],dt[i],cv[i]);
  }
  clrscr();
  for(i=0;i<n;++i)
   {
    cout<<"\n Hinh chu nhat thu "<<i+1<<":";
    cout<<"\n Do dai hai canh "<<a[i]<<"va"<<b[i];
    cout<<"\n dien tich "<<dt[i];
```

```
    cout<<"\n chu vi "<<cv[i];
   }
 getch();
 }
void dtcvhcn(int a,int b,int &dt ,int &cv)
 {
  dt=a*b;
  cv=2*(a+b);
 }
```

Bài 33 : 2.11. Hàm tải bội

Các hàm tải bội là các hàm có cùng một tên và có tập đối khác nhau (về số lượng các đối hoặc kiểu). Khi gặp lời gọi các hàm tải bội thì trình biên dịch sẽ căn cứ vào số lượng và kiểu các tham số để gọi hàm có đúng tên và đúng các tham số tương ứng.

Ví dụ 2.16 Chương trình tìm max của một dãy số nguyên và max của một dãy số thực. Trong chương trình có 6 hàm: hai hàm dùng để nhập dãy số nguyên và dãy số thực có tên chung là nhapds, bốn hàm: tính max 2 số nguyên, tính max 2 số thực, tính max của dãy số nguyên, tính max của dãy số thực được đặt chung một tên là max.

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void nhapds(int *x,int n);
void nhapds(double *x,int n);
int max(int x,int y);
double max(double x,double y);
void nhapds(int *x,int n)
 {
  for(int i=0;i<n;++i)
    {
     cout<<"Phan tu "<<i<<" = ";
     cin>>x[i];
    }
 }
 void nhapds(double *x,int n)
  {
   for (int i=0;i<n;i++)
   {
```

```
    cout<<"Phan tu "<<i<<" = ";
    cin>>x[i];
   }
  }
int max(int x,int y)
  {
   return x>y?x:y;
  }
double max(double x,double y)
  {
   return x>y?x:y;
  }
int max(int *x,int n)
  {
   int s=x[0];
   for(int i=1;i<n;++i)
    s=max(s,x[i]);
    return s;
  }
double max(double *x,int n)
    {
     double s=x[0];
   for(int i=1;i<n;++i)
    s=max(s,x[i]);
   return s;
   }
void main()
 {
   int a[20],n,ni,nd,maxi;
   double x[20],maxd;
   clrscr();
   cout<<"\n So phan tu nguyen n: ";
   cin>>ni;
   cout<<"\n Nhap day so nguyen: ";
   nhapds(a,ni);
   cout<<"\n So phan tu so thuc: ";
```

```
   cin>>nd;
   cout<<"\n Nhap day so thuc: ";
   nhapds(x,nd);
   maxi=max(a,ni);
   maxd=max(x,nd);
   cout<<"\n Max day so nguyen ="<<maxi;
   cout<<"\n Max day so thuc="<<maxd;
   getch();
 }
```

Chú ý: Nếu hai hàm trùng tên và trùng đối thì trình biên dịch không thể phân biệt được. Ngay cả khi hai hàm này có cùng kiểu khác nhau thì trình biên dịch vẫn báo lỗi. Ví dụ sau xây dựng hai hàm cùng có tên là f và cùng một đối nguyên a, nhưng kiểu hàm khác nhau. Hàm thứ nhất có kiểu nguyên(trả về a*a), hàm thứ hai có kiểu void. Chương trình sau sẽ bị thông báo lỗi khi biên dịch.

Ví dụ 2.17

```
#include <iostream.h>
#include <conio.h>
int f(int a);
void f(int a);
int f(int a)
 {
    return a*a;
 }
  void f(int a)
  {
     cout<<"\n"<<a;
  }
void main()
{
   int b = f(5);
   f(b);
   getch();
 }
```

Bài tập

1. Viết chương trình thực hiện các yêu cầu sau đây:
   - Nhập dữ liệu cho các sinh viên (dùng cấu trúc danh sách liên kết đơn), các thông tin của sinh viên bao gồm: mã sinh viên, họ tên, lớp, điểm trung bình.
   - Chương trình có sử dụng toán tử new và delete.
   - In ra danh sách sinh viên có sắp xếp vị thứ theo điểm trung bình.
2. Viết chương trình để sắp xếp một mảng thực hai chiều theo thứ tự tăng dần, trong chương trình có có sử dụng toán tử new và delete.
3. Viết các hàm tải bội để tính diện tích tam giác, diện tích hình chữ nhật, diện tích hình tròn.
4. Viết chương trình nhân hai ma trận A_{mxn} và B_{nxp} , mỗi ma trân được cấp phát động và các giá trị của chúng phát sinh ngẫu nhiên.

CHươNG 3

LớP

Chương này trình bày những vấn đề sau đây:

- *Định nghĩa lớp*
- *Tạo lập đối tượng*
- *Truy nhập đến các thành phần của lớp*
- *Con trỏ đối tượng*
- *Con trỏ this*
- *Hàm bạn*
- *Dữ liệu thành phần tĩnh, hàm thành phần tĩnh*
- *Hàm tạo, hàm hủy*
- *Hàm tạo sao chép*

Lớp là khái niệm trung tâm của lập trình hướng đối tượng, nó là sự mở rộng của các khái niệm cấu trúc (struct) của C. Ngoài các thành phần dữ liệu, lớp còn chứa các thành phần hàm, còn gọi là phương thức (method) hoặc hàm thành viên (member function). Lớp có thể xem như một kiểu dữ liệu các biến, mảng đối tượng. Từ một lớp đã định nghĩa, có thể tạo ra nhiều đối tượng khác nhau, mỗi đối tượng có vùng nhớ riêng.

Chương này sẽ trình bày cách định nghĩa lớp, cách xây dựng phương thức, giải thích về phạm vi truy nhập, sử dụng các thành phần của lớp, cách khai báo biến, mảng cấu trúc, lời gọi tới các phương thức .

3.1. Định nghĩa lớp

Cú pháp: Lớp được định nghĩa theo mẫu :

```
class tên_lớp
 {
 private:    [Khai báo các thuộc tính]
        [Định nghĩa các hàm thành phần (phương thức)]
 public :    [Khai báo các thuộc tính]
        [Định nghĩa các hàm thành phần (phương thức)]
 } ;
```

Thuộc tính của lớp được gọi là dữ liệu thành phần và hàm được gọi là phương thức hoặc hàm thành viên. Thuộc tính và hàm được gọi chung là các thành phần của lớp. Các thành phần của lớp được tổ chức thành hai vùng: vùng sở hữu riêng (private) và vùng dùng chung (public) để quy định phạm vi sử dụng của các thành phần. Nếu không quy định cụ thể (không dùng các từ khóa private và public) thì C++ hiểu đó là private. Các thành phần private chỉ được sử dụng bên trong lớp (trong thân của các hàm thành phần).

Các thành phần public được phép sử dụng ở cả bên trong và bên ngoài lớp. Các hàm không phải là hàm thành phần của lớp thì không được phép sử dụng các thành phần này.

Khai báo các thuộc tính của lớp: được thực hiện y như việc khai báo biến. Thuộc tính của lớp không thể có kiểu chính của lớp đó, nhưng có thể là kiểu con trỏ của lớp này,

Ví dụ:

```
class A
{
   A   x;   //Không cho phép, vì x có kiểu lớp A
   A   *p ;    // Cho phép, vì p là con trỏ kiểu lớp A
} ;
```

Định nghĩa các hàm thành phần: Các hàm thành phần có thể được xây dựng bên ngoài hoặc bên trong định nghĩa lớp. Thông thường, các hàm thành phần đơn giản, có ít dòng lệnh sẽ được viết bên trong định nghĩa lớp, còn các hàm thành phần dài thì viết bên ngoài định nghĩa lớp. Các hàm thành phần viết bên trong định nghĩa lớp được viết như hàm thông thường. Khi định nghĩa hàm thành phần ở bên ngoài lớp, ta dùng cú pháp sau đây:

```
Kiểu_trả_về_của_hàm  Tên_lớp::Tên_hàm(khai báo các tham số)
                { [nội dung hàm]
                }
```

Toán tử :: được gọi là toán tử phân giải miền xác định, được dùng để chỉ ra lớp mà hàm đó thuộc vào.

Trong thân hàm thành phần, có thể sử dụng các thuộc tính của lớp, các hàm thành phần khác và các hàm tự do trong chương trình.

Chú ý :

- Các thành phần dữ liệu khai báo là private nhằm bảo đảm nguyên lý che dấu thông tin, bảo vệ an toàn dữ liệu của lớp, không cho phép các hàm bên ngoài xâm nhập vào dữ liệu của lớp .
- Các hàm thành phần khai báo là public có thể được gọi tới từ các hàm thành phần public khác trong chương trình .

Bài 34 : 3.2. Tạo lập đối tượng

Sau khi định nghĩa lớp, ta có thể khai báo các biến thuộc kiểu lớp. Các biến này được gọi là các đối tượng. Cú pháp khai báo biến đối tượng như sau:

```
Tên_lớp    Danh_sách_biến ;
```

Đối tượng cũng có thể khai báo khi định nghĩa lớp theo cú pháp sau:

```
class tên_lớp
   {
```

```
    ...
} <Danh_sách_biến>;
```

Mỗi đối tượng sau khi khai báo sẽ được cấp phát một vùng nhớ riêng để chứa các thuộc tính của chúng. Không có vùng nhớ riêng để chứa các hàm thành phần cho mỗi đối tượng. Các hàm thành phần sẽ được sử dụng chung cho tất cả các đối tượng cùng lớp.

Bài 35 : 3.3. Truy nhập tới các thành phần của lớp

- Để truy nhập đến dữ liệu thành phần của lớp, ta dùng cú pháp:

  Tên_đối_tượng. Tên_thuộc_tính

Cần chú ý rằng dữ liệu thành phần riêng chỉ có thể được truy nhập bởi những hàm thành phần của cùng một lớp, đối tượng của lớp cũng không thể truy nhập.

- Để sử dụng các hàm thành phần của lớp, ta dùng cú pháp:

  Tên_đối_tượng. Tên_hàm (Các_khai_báo_tham_số_thực_sự)

Ví dụ 3.1

```
#include <conio.h>
#include <iostream.h>
class DIEM
     {
  private :
       int x,y ;
  public :
   void nhapsl( )
   {
    cout << "\n Nhap hoanh do va tung do cua diem:";
    cin >>x>>y ;
   }
    void hienthi( )
    { cout<<"\n x = " <<x<<" y = "<<y<<endl;}
      } ;
void main()
  { clrscr();
    DIEM d1;
    d1.nhapsl();
    d1.hienthi();
    getch();
  }
```

Ví dụ 3.2

```
#include <conio.h>
#include <iostream.h>
class A
   { int m,n;
     public :
      void nhap( )
    {
     cout << "\n Nhap hai so nguyen : " ;
     cin>>m>>n ;
    }
       int max()
    {
          return m>n?m:n;
      }
      void hienthi( )
     { cout<<"\n Thanh phan du lieu lon nhat x = "
             <<max()<<endl;}
    };
void main ()
  { clrscr();
    A ob;
    ob.nhap();
    ob.hienthi();
    getch();
  }
```

Chú ý: Các hàm tự do có thể có các đối là đối tượng nhưng trong thân hàm không thể truy nhập đến các thuộc tính của lớp. Ví dụ giả sử đã định nghĩa lớp :

```
    class DIEM
    {
     private :
       double x,y ; // toa do cua diem
     public :
       void nhapsl()
         {
            cout << " Toa do x,y : " ;
```

```
            cin >> x>>y ;
          }
        void in()
          {
            cout << "x =""<<x<<"y="<<y ;
          }
          } ;
```

Dùng lớp DIEM, ta xây dựng hàm tự do tính độ đài của đoạn thẳng đi qua hai điểm như sau :

```
  double  do_dai ( DIEM d1, DIEM d2 )
   {
    return sqrt(pow(d1.x-d2.x,2) + pow(d1.y-d2.y,2));
  }
```

Chương trình dịch sẽ báo báo lỗi đối với hàm này. Bởi vì trong thân hàm không cho phép sử dụng các thuộc tính d1.x,d2.x,d1.y của các đối tượng d1 và d2 thuộc lớp DIEM .

Ví dụ 3.3 Ví dụ sau minh họa việc sử dụng hàm thành phần với tham số mặc định:

```
#include <iostream.h>
#include <conio.h>
class Box
{
  private:
    int dai;
    int rong;
    int cao;
public:
 int get_thetich(int lth,int wdth = 2,int ht = 3);
};
int Box::get_thetich(int l, int w, int h)
{
    dai = l;
    rong = w;
    cao = h;
    cout<< dai<<'\t'<< rong<<'\t'<<cao<<'\t';
    return dai * rong * cao;
}
```

```
void main()
{
    Box ob;
    int x = 10, y = 12, z = 15;
    cout <<"Dai Rong Cao Thetich\n";
    cout << ob.get_thetich(x, y, z) << "\n";
    cout << ob.get_thetich(x, y) << "\n";
    cout << ob.get_thetich(x) << "\n";
    cout << ob.get_thetich(x, 7) << "\n";
    cout << ob.get_thetich(5, 5, 5) << "\n";
   getch();
}
```

Kết quả chương trình như sau:

```
Dai Rong Cao Thetich
10 12 15 1800
10 12 3 360
10 2 3 60
10 7 3 210
5 5 5 125
```

Ví dụ 3.4 Ví dụ sau minh họa việc sử dụng hàm **inline** trong lớp:

```
#include <iostream.h>
#include <string.h>
#include <conio.h>
class phrase
{
private:
   char dongtu[10];
   char danhtu[10];
   char cumtu[25];
public:
   phrase();
   inline void set_danhtu(char* in_danhtu);
   inline void set_dongtu(char* in_dongtu);
   inline char* get_phrase(void);
};
```

```
void phrase::phrase()
{
   strcpy(danhtu,"");
   strcpy(dongtu,"");
   strcpy(cumtu,"");
}
inline void phrase::set_danhtu(char* in_danhtu)
{
   strcpy(danhtu, in_danhtu);
}
inline void phrase::set_dongtu(char* in_dongtu)
{
   strcpy(dongtu, in_dongtu);
}
inline char* phrase::get_phrase(void)
{
   strcpy(cumtu,dongtu);
   strcat(cumtu," the ");
   strcat(cumtu,danhtu);
   return cumtu;
}
void main()
{
  phrase text;
  cout << "Cum tu la : -> " << text.get_phrase()
       << "\n";
  text.set_danhtu("file");
  cout << "Cum tu la : -> " <<
       text.get_phrase()<<"\n";
  text.set_dongtu("Save");
  cout << "Cum tu la : -> " <<
       text.get_phrase()<<"\n";
  text.set_danhtu("program");
  cout << "Cum tu la : -> " <<
       text.get_phrase()<<"\n";
}
```

Kết quả chương trình như sau:

```
Cum tu la : -> the
Cum tu la : -> the file
Cum tu la : -> Save the file
Cum tu la : -> Save the program
```

Ví dụ 3.5 Ví dụ sau minh họa việc sử dụng từ khóa const trong lớp:

```
#include <iostream.h>
#include <conio.h>
class constants
{
  private:
    int number;
  public:
    void print_it(const int data_value);
};
void constants::print_it(const int data_value)
{
    number = data_value;
    cout << number << "\n";
}
void main()
{
    constants num;
    const int START = 3;
    const int STOP = 6;
    int index;
    for (index=START; index<=STOP; index++)
   {
       cout<< "index = " ;
       num.print_it(index);
       cout<< "START = " ;
       num.print_it(START);
   }
  getch();
}
```

Kết quả chương trình như sau:

```
index = 3
START = 3
index = 4
START = 3
index = 5
START = 3
index = 6
START = 3
```

Bài 36 : 3.4. Con trỏ đối tượng

Con trỏ đối tượng dùng để chứa địa chỉ của biến đối tượng, được khai báo như sau :

Tên_lớp * Tên_con_ trỏ ;

Ví dụ : Dùng lớp DIEM, ta có thể khai báo:

```
DIEM  *p1,  *p2,  *p3 ;  // Khai báo 3 con trỏ p1, p2, p3
DIEM  d1, d2 ;       //Khai báo hai đối tượng d1, d2
DIEM  d [20] ;       // Khai báo mảng đối tượng
```

Có thể thực hiện câu lệnh :

```
p1 = &d2 ;    //p1 chứa địa chỉ của d2, p1 trỏ tới d2
p2 =d ;      // p2 trỏ tới đầu mảng d
p3 =new DIEM //tạo một đối tượng và chứa địa chỉ của nó vào p3
```

Để truy xuất các thành phần của lớp từ con trỏ đối tượng, ta viết như sau :

Tên_con_trỏ -> Tên_thuộc_tính

Tên_con_trỏ -> Tên_hàm(các tham số thực sự)

Nếu con trỏ chứa đầu địa chỉ của mảng, có thể dùng con trỏ như tên mảng.

Ví dụ 3.6

```
#include <iostream.h>
#include <conio.h>
class mhang
 { int maso;
   float gia;
   public: void getdata(int a, float b)
     {maso= a; gia= b;}
     void show()
     { cout << "maso" << maso<< endl;
       cout << "gia" << gia<< endl;
     }
```

```
  };
const int k=5;
void main()
 { clrscr();
   mhang *p = new mhang[k];
   mhang *d = p;
   int x,i;
   float y;
   cout<<"\nNhap vao du lieu 5 mat hang :";
   for (i = 0; i <k; i++)
     { cout <<"\nNhap ma hang va don gia cho mat hang thu "
            <<i+1;
       cin>>x>>y;
       p -> getdata(x,y);
       p++;}
       for (i = 0; i <k; i++)
         { cout <<"\nMat hang  thu : " << i + 1 <<
                endl;
           d -> show();
           d++;
         }
   getch();
 }
```

Bài 37 : 3.5. Con trỏ this

Mỗi hàm thành phần của lớp có một tham số ẩn, đó là con trỏ **this**. Con trỏ this trỏ đến từng đối tượng cụ thể. Ta hãy xem lại hàm nhapsl() của lớp DIEM trong ví dụ ở trên:

```
  void nhapsl( )
  {
   cout << "\n Nhap hoanh do va tung do cua diem : ";
   cin >>x>>y;
  }
```

Trong hàm này ta sử dụng tên các thuộc tính x,y một cách đơn độc. Điều này dường như mâu thuẩn với quy tắc sử dụng thuộc tính. Tuy nhiên nó được lý giải như sau: C++ sử dụng một con trỏ đặc biệt trong các hàm thành phần. Các thuộc tính viết trong hàm thành phần được hiểu là thuộc một đối tượng do con trỏ **this** trỏ tới. Như vậy hàm nhapsl() có thể viết một cách tường minh như sau:

```
  void nhapsl( )
  {
   cout << "\n Nhap hoanh do va tung do cua diem:" ;
   cin >>this->x>>this->y ;
  }
```

Con trỏ **this** là đối thứ nhất của hàm thành phần. Khi một lời gọi hàm thành phần được phát ra bởi một đối tượng thì tham số truyền cho con trỏ **this** chính là địa chỉ của đối tượng đó.

Ví dụ: Xét một lời gọi tới hàm nhapsl() :

DIEM d1 ;
d1.nhapsl();

Trong trường hợp này của d1 thì this =&d1. Do đó this -> x chính là d1.x và this-> y chính là d1.y

Chú ý: Ngoài tham số đặc biệt **this** không xuất hiện một cách tường minh, hàm thành phần lớp có thể có các thamô1 khác được khai báo như trong các hàm thông thường.

Ví dụ 3.7

```
#include <iostream.h>
#include <conio.h>
class time
{ int h,m;
   public :
       void nhap(int h1, int m1)
        { h= h1; m = m1;}
       void hienthi(void)
       { cout <<h << " gio "<<m << " phut"<<endl;}
       void tong(time, time);
};
void time::tong(time t1, time t2)
{ m= t1.m+ t2.m;   //this->m = t1.m+ t2.m;
  h= m/60;      //this->h = this->m/60;
  m= m%60;      //this->m = this->m%60;
  h = h+t1.h+t2.h; //this->h = this->h + t1.h+t2.h;
}
void main()
{
 clrscr();
```

```
 time ob1, ob2,ob3;
 ob1.nhap(2,45);
 ob2.nhap(5,40);
 ob3.tong(ob1,ob2);
 cout <<"object 1 = "; ob1.hienthi();
 cout <<"object 2 = "; ob2. hienthi();
 cout <<"object 3 = "; ob3. hienthi();
 getch();
}
```

Chương trình cho kết quả như sau :

```
object 1 = 2 gio 45 phut
object 2 = 5 gio 40 phut
object 3 = 8 gio 25 phut
```

Bài 38 : 3.6. Hàm bạn

Trong thực tế thường xãy ra trường hợp có một số lớp cần sử dụng chung một hàm. C++ giải quyết vấn đề này bằng cách dùng hàm bạn. Để một hàm trở thành bạn của một lớp, có 2 cách viết:

Cách 1: Dùng từ khóa *friend* để khai báo hàm trong lớp và xây dựng hàm bên ngoài như các hàm thông thường (không dùng từ khóa friend). Mẫu viết như sau :

```
class  A
   {
      private :
      //  Khai báo các thuộc tính
      public :
      ...
      // Khai báo các hàm bạn của lớp A
      friend void f1 (...) ;
      friend double f2 (...) ;
      ...
   } ;
// Xây dựng các hàm f1,f2,f3
void f1 (...)
   {
      ...
   }
double f2 (...)
```

```
{
   ...
}
```

Cách 2: Dùng từ khóa friend để xây dựng hàm trong định nghĩa lớp . Mẫu viết như sau :

```
class  A
{
   private :
   //  Khai báo các thuộc tính
   public :
    ...
   // Khai báo các hàm bạn của lớp A
   void f1 (...)
      {
         ...
      }
   double f2 (...)
      {
         ...
      }

} ;
```

Hàm bạn có những tính chất sau:

- Hàm bạn không phải là hàm thành phần của lớp.
- Việc truy nhập tới hàm bạn được thực hiện như hàm thông thường.
- Trong thân hàm bạn của một lớp có thể truy nhập tới các thuộc tính của đối tượng thuộc lớp này. Đây là sự khác nhau duy nhất giữa hàm bạn và hàm thông thường.
- Một hàm có thể là bạn của nhiều lớp. Lúc đó nó có quyền truy nhập tới tất cả các thuộc tính của các đối tượng trong các lớp này. Để làm cho hàm f trở thành bạn của các lớp A, B và C ta sử dụng mẫu viết sau :

```
class B ; //Khai báo trước lớp A
class B ; // Khai báo trước lớp B
class C ; // Khai báo trước lớp C
// Định nghĩa  lớp A
class A
   {
      // Khai báo f là bạn của A
```

```
        friend void f(... )
      } ;
    // Định nghĩa lớp B
    class B
      {
        // Khai báo f là bạn của B
        friend void f(...)
      } ;
    // Định nghĩa lớp C
    class C
      {
        // Khai báo f là bạn của C
        friend void f(...)
      } ;
    // Xây dựng hàm f
    void f(...)
      {
        ...
      } ;
```

Ví dụ 3.8

```
#include <iostream.h>
#include <conio.h>
class sophuc
{float a,b;
 public : sophuc() {}
        sophuc(float x, float y)
        {a=x; b=y;}
    friend sophuc tong(sophuc,sophuc);
    friend void  hienthi(sophuc);
};
sophuc  tong(sophuc c1,sophuc c2)
 {sophuc c3;
  c3.a=c1.a + c2.a ;
  c3.b=c1.b + c2.b ;
  return (c3);
  }
```

```
void hienthi(sophuc c)
 {cout<<c.a<<" + "<<c.b<<"i"<<endl; }
void main()
{ clrscr();
  sophuc d1 (2.1,3.4);
  sophuc d2 (1.2,2.3) ;
  sophuc d3 ;
  d3 = tong(d1,d2);
  cout<<"d1= ";hienthi(d1);
  cout<<"d2= ";hienthi(d2);
  cout<<"d3= ";hienthi(d3);
  getch();
  }
```

Chương trình cho kết quả như sau :

```
d1= 2.1 + 3.4i
d2= 1.2 + 2.3i
d3= 3.3 + 5.7i
```

Ví dụ 3.9

```
#include <iostream.h>
#include <conio.h>
class LOP1;
class LOP2
  {
    int v2;
    public:
      void nhap(int a)
     { v2=a;}
      void hienthi(void)
     { cout<<v2<<"\n";}
      friend void traodoi(LOP1 &, LOP2 &);
  };
class LOP1
  {
    int v1;
    public:
```

```
      void nhap(int a)
   { v1=a;}
      void hienthi(void)
   { cout<<v1<<"\n";}
      friend void traodoi(LOP1 &, LOP2 &);
  };
void traodoi(LOP1 &x, LOP2 &y)
  {
    int t = x.v1;
    x.v1 = y.v2;
    y.v2 = t;
  }
void main()
  {
   clrscr();
   LOP1 ob1;
   LOP2 ob2;
   ob1.nhap(150);
   ob2.nhap(200);
   cout << "Gia tri ban dau :" << "\n";
   ob1.hienthi();
   ob2.hienthi();
   traodoi(ob1, ob2); //Thuc hien hoan doi
   cout << "Gia tri sau khi thay doi:" << "\n";
   ob1.hienthi();
   ob2.hienthi();
   getch();
  }
```

Chương trình cho kết quả như sau:

```
Gia tri ban dau :
150
200
Gia tri sau khi thay doi:
200
150
```

Bài 39 : 3.7. Dữ liệu thành phần tĩnh và hàm thành phần tĩnh

39.13.7.1. Dữ liệu thành phần tĩnh

Dữ liệu thành phần tĩnh được khai báo bằng từ khoá static và được cấp phát một vùng nhí cố định, nó tồn tại ngay cả khi lớp chưa có một đối tượng nào cả. Dữ liệu thành phần tĩnh là chung cho cả lớp, nó không phải là riêng của mỗi đối tượng, ví dụ:

```
class A
{
    private:
        static int ts; // Thành phần tĩnh
        int x;
        ...
};
A u, v; // Khai báo 2 đối tượng
```

Giữa các thành phần x và ts có sự khác nhau như sau: u.x và v.x có 2 vùng nhớ khác nhau, trong khi u.ts và v.ts chỉ là một, chúng cùng biểu thị một vùng nhớ, thành phần ts tồn tại ngay khi u và v chưa khai báo.

Để biểu thị thành phần tĩnh, ta có thể dùng tên lớp, ví dụ: A::ts

Khai báo và khởi gán giá trị cho thành phần tĩnh: Thành phần tĩnh sẽ được cấp phát bộ nhớ và khởi gán giá trị đầu bằng một câu lệnh khai báo đặt sau định nghĩa lớp theo mẫu như sau:

```
int A::ts;          // Khởi gán cho ts giá trị 0
int A::ts = 1234;  // Khởi gán cho ts giá trị 1234
```

Chú ý: Khi chưa khai báo thì thành phần tĩnh chưa tồn tại. Hãy xem chương trình sau:

Ví dụ 3.10

```
#include <conio.h>
#include <iostream.h>
class HDBH
{
    private:
        char *tenhang;
        double tienban;
        static int tshd;
        static double tstienban;
    public:
```

```
        static void in()
        {
            cout <<"\n" << tshd;
            cout <<"\n" << tstienban;
        }
};
void main ()
{
    HDBH::in();
    getch();
}
```

Các thành phần tĩnh tshd và tstienban chưa khai báo, nên chưa tồn tại. Vì vậy các câu lệnh in giá trị các thành phần này trong hàm in() là không thể được. Khi dịch chương trình, sẽ nhận được các thông báo lỗi. Có thể sửa chương trình trên bằng cách đưa vào các lệnh khai báo các thành phần tĩnh tshd và tstienban như sau:

Ví dụ 3.11

```
#include <conio.h>
#include <iostream.h>
class HDBH
{
    private:
        int shd;
        char *tenhang;
        double tienban;
        static int tshd;
        static double tstienban;
    public:
        static void in()
        {
            cout <<"\n" <<tshd;
            cout <<"\n" <<tstienban;
        }
};
int HDBH::tshd=5
double HDBH::tstienban=20000.0;
void main()
```

```
{
    HDBH::in();
    getch();
}
```

39.23.7.2. Hàm thành phần tĩnh

Hàm thành phần tĩnh được viết theo một trong hai cách:

- Dùng từ khoá static đặt trước định nghĩa hàm thành phần viết bên trong định nghĩa lớp.

- Nếu hàm thành phần xây dựng bên ngoài định nghĩa lớp, thì dùng từ khoá static đặt trước khai báo hàm thành phần bên trong định nghĩa lớp. Không cho phép dùng từ khoá static đặt trước định nghĩa hàm thành phần viết bên ngoài định nghĩa lớp.

Các đặc tính của hàm thành phần tĩnh:

- Hàm thành phần tĩnh là chung cho toàn bộ lớp và không lệ thuộc vào một đối tượng cụ thể, nó tồn tại ngay khi lớp chưa có đối tượng nào.
- Lời gọi hàm thành phần tĩnh như sau:

Tên lớp :: Tên hàm thành phần tĩnh(các tham số thực sự)

- Vì hàm thành phần tĩnh là độc lập với các đối tượng, nên không thể dùng hàm thành phần tĩnh để xử lý dữ liệu của các đối tượng trong lời gọi phương thức tĩnh. Nói cách khác không cho phép truy nhập các thuộc tính (trừ thuộc tính tĩnh) trong thân hàm thành phần tĩnh. Đoạn chương trình sau minh họa điều này:

```
class HDBH
{
 private:
  int shd;
  char *tenhang;
  double tienban;
  static int tshd;
  static double tstienban;
 public:
    static void in()
     {
     cout <<"\n" << tshd;
     cout << "\n" << tstienban;
     cout <<"\n"<< tenhang    //loi
     cout <<"\n" << tienban;  //loi
     }
```

```
};
```

Ví dụ 3.12

```
 #include <iostream.h>
 #include <conio.h>
 class A
  {
    int m;
    static int n; //n la bien tinh
    public: void set_m(void) { m= ++n;}
      void show_m(void)
      {
       cout << "\n Doi tuong thu:" << m << endl;
      }
      static void show_n(void)
      {
       cout << " m = " << n << endl;
      }
  };
  int A::n=1; //khoi gan gia tri ban dau 1 cho bien tinh n
void  main()
   {
     clrscr();
     A t1, t2;
     t1.set_m();
     t2.set_m();
     A::show_n();
     A t3;
     t3.set_m();
     A::show_n();
     t1.show_m();
     t2.show_m();
     t3.show_m();
     getch();
   }
```

Kết quả chương trình trên là:

```
m = 3
m = 4
Doi tuong thu : 2
Doi tuong thu : 3
Doi tuong thu : 4
```

Bài 40 : 3.8. Hàm tạo (constructor)

Hàm tạo là một hàm thành phần đặc biệt của lớp làm nhiệm vụ tạo lập một đối tượng mới. Chương trình dịch sẽ cấp phát bộ nhớ cho đối tượng, sau đó sẽ gọi đến hàm tạo. Hàm tạo sẽ khởi gán giá trị cho các thuộc tính của đối tượng và có thể thực hiện một số công việc khác nhằm chuẩn bị cho đối tượng mới. Khi xây dựng hàm tạo cần lưu ý những đặc tính sau của hàm tạo:

- Tên hàm tạo trùng với tên của lớp.
- Hàm tạo không có kiểu trả về.
- Hàm tạo phải được khai báo trong vùng public.
- Hàm tạo có thể được xây dựng bên trong hoặc bên ngoài định nghĩa lớp.
- Hàm tạo có thể có tham số hoặc không có tham số.
- Trong một lớp có thể có nhiều hàm tạo (cùng tên nhưng khác các tham số).

Ví dụ 3.13

```
class DIEM
    {
      private:
          int x,y;
      public:
          DIEM()      //Ham tao khong tham so
        {
            x = y = 0;
           }
         DIEM(int x1, int y1) //Ham tao co tham so
           {
            x = x1;y=y1;
           }
         //Cac thanh phan khac
    };
```

Chú ý 1: Nếu lớp không có hàm tạo, chương trình dịch sẽ cung cấp một hàm tạo mặc định không đối, hàm này thực chất không làm gì cả. Như vậy một đối tượng tạo ra chỉ được cấp phát bộ nhớ, còn các thuộc tính của nó chưa được xác định.

Ví dụ 3.14

```
#include <conio.h>
#inlcule <iostream.h>
class DIEM
 {
       private:
          int x,y;
       public:
         void in()
         {
            cout <<"\n" << y <<" " << m;
         }
  };
void main()
  {
  DIEM d;
  d.in();
  DIEM   *p;
  p= new DIEM [10];
  clrscr();
  d.in();
  for (int i=0;1<10;++i)
     (p+i)->in();
  getch();
  }
```

Chú ý 2:

- Khi một đối tượng được khai báo thì hàm tạo của lớp tương ứng sẽ tự động thực hiện và khởi gán giá trị cho các thuộc tính của đối tượng. Dựa vào các tham số trong khai báo mà chương trình dịch sẽ biết cần gọi đến hàm tạo nào.
- Khi khai báo mảng đối tượng không cho phép dùng các tham số để khởi gán cho các thuộc tính của các đối tượng mảng.
- Câu lệnh khai báo một biến đối tượng sẽ gọi tới hàm tạo một lần.
- Câu lệnh khai báo một mảng *n* đối tượng sẽ gọi tới hàm tạo mặc định *n* lần.

- Với các hàm có các tham số kiểu lớp, thì chúng chỉ xem là các tham số hình thức, vì vậy khai báo tham số trong dòng đầu của hàm sẽ không tạo ra đối tượng mới và do đó không gọi tới các hàm tạo.

Ví dụ 3.15

```
#include <conio.h>
#include <iostream.h>
#include <iomanip.h>
class DIEM
{
   private:
      int   x,y;
   public:
      DIEM()
     {
        x = y = 0;
      }
      DIEM(int x1, int y1)
    {
      x = x1; y = y1;
     }
   friend  void  in(DIEM  d)
    {
      cout <<"\n" << d.x <<" " << d.y;
     }
      void in()
          {
         cout <<"\n" << x <<" " << y ;
           }
         };
void main()
  {
    DIEM d1;
    DIEM d2(2,3);
    DIEM  *d;
    d = new DIEM (5,6);
    clrscr();
```

```
    in(d1); // Goi ham ban in()
    d2.in(); // Goi ham thanh phan in()
    in(*d); // Goi ham ban in()
    DIEM(2,2).in();// Goi ham thanh phan in()
    DIEM  t[3]; // 3 lan goi ham tao khong doi
    DIEM  *q; // Goi ham tao khong doi
    int n;
    cout << "\n N = ";
    cin >> n;
    q = new DIEM [n+1]; //n+1 lan goi ham tao khong doi
    for (int i=0;i<=n;++i)
    q[i]=DIEM (3+i,4+i);//n+1 lan goi ham tao co doi
    for (i=0;i<=n;++i)
      q[i].in();    // Goi ham thanh phan in()
    for (i=0;i<=n;++i)
      DIEM(5+i,6+i).in(); //Goi ham thanh phan in()
    getch();
   }
```

Chú ý 3: Nếu trong lớp đã có ít nhất một hàm tạo, thì hàm tạo mặc định sẽ không được phát sinh nữa. Khi đó mọi câu lệnh xây dựng đối tượng mới đều sẽ gọi đến một hàm tạo của lớp. Nếu không tìm thấy hàm tạo cần gọi thì chương trình dịch sẽ báo lỗi. Điều này thường xãy ra khi chúng ta không xây dựng hàm tạo không đối, nhưng lại sử dụng các khai báo không tham số như ví dụ sau:

Ví dụ 3.16

```
#include <conio.h>
#include <iostream.h>
class DIEM
 {
   private:
          int x,y;
   public:
           DIEM(int x1, int y1)
             {
               x=x1; y=y1;
             }
           void in()
```

```
            {
     cout << "\n" << x << " " << y <<" " << m;
       }
    };
void main()
{
    DIEM  d1(200,200); // Goi ham tao co doi
    DIEM  d2; // Loi, goi ham tao khong doi
    d2= DIEM_DH (3,5); // Goi ham tao co doi
    d1.in();
    d2.in();
    getch();
};
```

Trong ví dụ này, câu lệnh DIEM d2; trong hàm main() sẽ bị chương trình dịch báo lỗi. Bởi vì lệnh này sẽ gọi tới hàm tạo không đối, mà hàm tạo này chưa được xây dựng. Có thể khắc phục điều này bằng cách chọn một trong hai giải pháp sau:

- Xây dựng thêm hàm tạo không đối.
- Gán giá trị mặc định cho tất cả các đối x1, y1 của hàm tạo đã xây dựng ở trên.

Theo giải pháp thứ 2, chương trình trên có thể sửa lại như sau:

Ví dụ 3.17

```
#include <conio.h>
#include <iostream.h>
class DIEM
 {
    private:
           int x,y;
    public:
       DIEM(int x1=0, int y1=0)
        {
          x = x1; y = y1;
        }
       void in()
        {
         cout << "\n" << x << " " << y <<" " << m;
         }
 };
```

```
void main()
{
DIEM d1(2,3); //Goi ham tao,khong dung tham so mac dinh
    DIEM  d2; //Goi ham tao, dung tham so mac dinh
d2= DIEM(6,7);//Goi ham tao,khong dung tham so mac dinh
    d1.in();
    d2.in();
getch(); }
```

Ví dụ 3.18

```
#include <iostream.h>
#include <conio.h>
class rectangle
{
private:
     int dai;
     int rong;
     static int extra_data;
public:
     rectangle();
     void set(int new_dai, int new_rong);
    int get_area();
    int get_extra();
};
int rectangle::extra_data;
rectangle::rectangle()
{
    dai = 8;
    rong = 8;
    extra_data = 1;
}
void rectangle::set(int new_dai,int new_rong)
{
    dai = new_dai;
    rong = new_rong;
}
int rectangle::get_area()
```

```
{
    return (dai * rong);
}
int rectangle::get_extra()
{
   return extra_data++; }
void main()
{
   rectangle small, medium, large;
   small.set(5, 7);
   large.set(15, 20);
   cout<<"Dien tich la : "<<small.get_area()<<"\n";
   cout<<"Dien tich la : "<<
   medium.get_area()<<"\n";
   cout<<"Dien tich la : "<<large.get_area()<<"\n";
   cout <<"Gia tri du lieu tinh la : "<<
   small.get_extra()<<"\n";
   cout <<"Gia tri du lieu tinh la : "<<
   medium.get_extra()<<"\n";
   cout <<"Gia tri du lieu tinh la : "<<
   large.get_extra()<<"\n";
}
```

Chương trình cho kết quả như sau :

```
Dien tich la : 35
Dien tich la : 64
Dien tich la : 300
Gia tri du lieu tinh la : 1
Gia tri du lieu tinh la : 2
Gia tri du lieu tinh la : 3
```

Bài 41 : 3.9. Hàm tạo sao chép

41.13.9.1. Hàm tạo sao chép mặc định

Giả sử đã định nghĩa một lớp ABC nào đó. Khi đó:

- Ta có thể dùng câu lệnh khai báo hoặc cấp phát bộ nhớ để tạo các đối tượng mới, ví dụ:

```
ABC  p1, p2;
ABC *p = new ABC;
```

- Ta cũng có thể dùng lệnh khai báo để tạo một đối tượng mới từ một đối tượng đã tồn tại, ví dụ:

ABC u;

ABC v(u); // Tạo v theo u

Câu lệnh này có ý nghĩa như sau:

- Nếu trong lớp ABC chưa xây dựng hàm tạo sao chép, thì câu lệnh này sẽ gọi tới một hàm tạo sao chép mặc định của C++. Hàm này sẽ sao chép nội dung từng bit của u vào các bit tương ứng của v. Như vậy các vùng nhớ của u và v sẽ có nội dung như nhau.

- Nếu trong lớp ABC đã có hàm tạo sao chép thì câu lệnh:

PS v(u);

sẽ tạo ra đối tượng mới v, sau đó gọi tới hàm tạo sao chép để khởi gán v theo u.

Ví dụ sau minh họa cách dùng hàm tạo sao chép mặc định:

Trong chương trình đưa vào lớp PS (phân số):

+ Các thuộc tính gồm: t (tử số) và m (mẫu).

+ Trong lớp mà không có phương thức nào cả mà chỉ có hai hàm bạn là các hàm toán tử nhập (>>) và xuất (<<).

+ Nội dung chương trình là: Dùng lệnh khai báo để tạo một đối tượng u (kiểu PS) có nội dung như đối tượng đã có d.

Ví dụ 3.19

```
#include <conio.h>
#include <iostream.h>
class PS
{
 private:  int t,m;
 public:
 friend ostream& operator<< (ostream& os,const PS &p)
    {
    os<<" = "<<p.t<<"/"<<p.m;
    return os;
    }
   friend istream& operator>> (istream& is, PS &p)
    {
    cout <<" Nhap tu va mau : ";
    is>>p.t>>p.m;
    return is;
   }
```

```
};
void main()
 {
   PS d;
   cout <<"\n Nhap phan so d "; cin>>d;
   cout<<"\n PS d "<<d;
   PS u(d);
   cout<<"\n PS u "<<u;
   getch();
 }
```

41.23.9.2. Hàm tạo sao chép

Hàm tạo sao chép sử dụng một đối kiểu tham chiếu đối tượng để khởi gán cho đối tượng mới và được viết theo mẫu sau:

```
Tên_lớp (const Tên_lớp  &ob)
{
 // Các câu lệnh dùng các thuộc tính của đối tượng ob để khởi gán
      // cho các thuộc tính của đối tượng mới
}
```

Ví dụ:

```
class PS
  {   private:  int  t, m;
       public:
          PS(const PS  &p)
           {
             t= p.t;
             m= p.m;
          }
         …
     };
```

Hàm tạo sao chép trong ví dụ trên không khác gì hàm tạo sao chép mặc định.

Chú ý:

- Nếu lớp không có các thuộc tính kiểu con trỏ hoặc tham chiếu thì dùng hàm tạo sao chép mặc định là đủ.

- Nếu lớp có các thuộc tính con trỏ hoặc tham chiếu, thì hàm tạo sao chép mặc định chưa đáp ứng được yêu cầu.

class DT

```
{
  private:
     int n; // Bac da thuc
     double *a; // Tro toi vung nho chua cac he so da thuc a0, a1, …
  public:
     DT()
        {
           n = 0; a = NULL;
        }
      DT(int n1)
         {
           n = n1;
           a = new double[n1+1];
         }
      friend ostream& operator<< (ostream& os,const DT &d);
      friend istream& operator>> (istream& is,DT &d);
      …
};
```

Bây giờ chúng ta hãy theo dõi xem việc dùng hàm tạo mặc định trong đoạn chương trình sau sẽ dẫn đến sai lầm như thế nào:

```
DT d;
cin >> d;
/* Nhập đối tượng d gồm: nhập một số nguyên dương và gán cho d.n, cấp phát vùng nhớ cho d.n, nhập các hệ số của đa thức và chứa vào vùng nhớ được cấp phát
*/
DT u(d);
/* Dùng hàm tạo mặc định để xây dựng đối tượng u theo d. Kết quả: u.n = d.n và u.a = d.a. Như vậy hai con trỏ u.a và d.a cùng trỏ đến một vùng nhớ.
*/
```

Nhận xét: Mục đích của ta là tạo ra một đối tượng u giống như d, nhưng độc lập với d. Nghĩa là khi d thay đổi thì u không bị ảnh hưởng gì. Thế nhưng mục tiêu này không đạt được, vì u và d có chung một vùng nhớ chứa hệ số của đa thức, nên khi sửa đổi các hệ số của đa thức trong d thì các hệ số của đa thức trong u cũng thay đổi theo. Còn một trường hợp nữa cũng dẫn đến lỗi là khi một trong hai đối tượng u và d bị giải phóng (thu hồi vùng nhớ chứa đa thức) thì đối tượng còn lại cũng sẽ không còn vùng nhớ nữa.

Ví dụ sau sẽ minh họa nhận xét trên: Khi d thay đổi thì u cũng thay đổi và ngược lại khi u thay đổi thì d cũng thay đổi theo.

Ví dụ 3.20

```
#include <conio.h>
#include <iostream.h>
#include <math.h>
class DT
 {
   private:
       int n; // Bac da thuc
       double *a; // Tro toi vung nho chua cac he
                  //so da thuc a0,a1,...
   public:
       DT()
      {
        this->n=0; this->a=NULL;
      }
       DT(int n1)
      {
         this->n=n1;
         this->a= new double[n1+1];
      }
friend ostream& operator<< (ostream& os,const DT &d);
  friend istream& operator>> (istream& is,DT &d);
  };
  ostream& operator<< (ostream& os,const DT &d)
   {
     os <<" Cac he so ";
     for (int i=0; i<=d.n; ++i)
       os << d.a[i]<<" ";
     return os;
    }
  istream& operator>> (istream& is,DT &d)
   {
     if (d.a != NULL) delete d.a;
     cout << " \nBac da thuc:";
```

```
      cin >>d.n;
      d.a = new double[d.n+1];
      cout<<"Nhap cac he so da thuc:\n";
      for (int i=0 ; i<=d.n ; ++i)
       {
    cout<<"He so bac"<< i << "=";
    is >> d.a[i];
       }
      return is;
     }
void main()
{
  DT d;
  clrscr();
  cout<<"\nNhap da thuc d" ; cin >> d;
  DT u(d);
  cout <<"\nDa thuc d" << d ;
  cout <<"\nDa thuc u" << u ;
  cout <<"\nNhap da thuc d" << d ; cin >> d;
  cout <<"\nDa thuc d" << d ;
  cout <<"\nDa thuc u" << u ;
  cout <<"\nDa thuc u" << u ; cin >> u;
  cout <<"\nDa thuc d" << d ;
  cout <<"\nDa thuc u" << u ;
  getch();
}
```

Ví dụ 3.21 Ví dụ sau minh họa về hàm tạo sao chép:

```
#include <conio.h>
#include <iostream.h>
#include <math.h>
class DT
{
    private:
        int n; // Bac da thuc
    double *a; // Tro toi vung nho chua cac da thuc
             // a0, a1,...
```

```
  public:
      DT()
      {
        this->n=0; this->a=NULL;
      }
      DT(int n1)
      {
        this->n=n1;
        this->a= new double[n1+1];
      }
      DT(const DT &d);
friend ostream& operator<< (ostream& os,const DT &d);
   friend istream& operator>> (istream& is,DT &d);
 };
DT::DT(const DT &d)
{
 this->n = d.n;
 this->a = new double[d.n+1];
 for (int i=0;i<=d.n;++i)
 this->a[i] = d.a[i];
}
ostream& operator<< (ostream& os,const DT &d)
{
 os<<"-Cac he so (tu ao): ";
 for (int i=0 ; i<=d.n ; ++i)
 os << d.a[i] <<" ";
 return os;
}
istream& operator>> (istream& is,DT &d)
{
if (d.a != NULL) delete d.a;
cout << "\n Bac da thuc:";
cin >> d.n;
d.a = new double[d.n+1];
cout << "Nhap cac he so da thuc:\n";
for (int i=0 ; i<= d.n ; ++i)
```

```
{
  cout << "He so bac " << i << "=";
  is >> d.a[i];
}
return is;
}
void main()
{
  DT d;
  clrscr();
  cout <<"\nNhap da thuc d " ; cin >> d;
  DT u(d);
  cout << "\nDa thuc d " << d;
  cout << "\nDa thuc u " << u;
  cout << "\nNhap da thuc d "; cin >> d;
  cout << "\nDa thuc d " << d;
  cout << "\nDa thuc u " << u;
  cout << "\nNhap da thuc u " ; cin >> u;
  cout << "\nDa thuc d " << d;
  cout << "\nDa thuc u " << u;
  getch();
}
```

Bài 42 : 3.10. Hàm hủy (destructor)

Hàm huỷ là một hàm thành phần của lớp, có chức năng ngược với hàm tạo. Hàm huỷ được gọi trước khi giải phóng một đối tượng để thực hiện một số công việc có tính "dọn dẹp" trước khi đối tượng được hủy bỏ, ví dụ giải phóng một vùng nhớ mà đối tượng đang quản lý, xoá đối tượng khỏi màn hình nếu như nó đang hiển thị...Việc hủy bỏ đối tượng thường xảy ra trong 2 trường hợp sau:

- Trong các toán tử và hàm giải phóng bộ nhớ như delete, free...
- Giải phóng các biến, mảng cục bộ khi thoát khỏi hàm, phương thức.

Nếu trong lớp không định nghĩa hàm huỷ thì một hàm huỷ mặc định không làm gì cả được phát sinh. Đối với nhiều lớp thì hàm huỷ mặc định là đủ, không cần đưa vào một hàm huỷ mới. Trường hợp cần xây dựng hàm hủy thì tuân theo quy tắc sau:

- Mỗi lớp chỉ có một hàm hủy.

- Hàm hủy không có kiểu, không có giá trị trả về và không có tham số.

- Tên hàm hàm huỷ có một dấu ngã ngay trước tên lớp.

Ví dụ:

```
 class A
   {
    private:
          int n;
          double *a;
    public:
          ~A()
          {
             n = 0;
             delete a;
          }
        };
```

Ví dụ 3.22

```
#include <iostream.h>
class Count{
private:
    static int counter;
    int obj_id;
public:
   Count();
   static void display_total();
   void display();
   ~Count();
};
int Count::counter;
Count::Count()
{
    counter++;
    obj_id = counter;
}
Count::~Count()
{
   counter--;
   cout<<"Doi tuong "<<obj_id<<" duoc huy bo\n";
}
```

```
void Count::display_total()
{
cout <<"So cac doi tuong duoc tao ra la  = "<< counter <<
endl;
}
void Count::display()
{
   cout << "Object ID la  "<<obj_id<<endl;
}
void main()
{
   Count a1;
   Count::display_total();
   Count a2, a3;
   Count::display_total();
   a1.display();
   a2.display();
   a3.display();
}
```

Kết quả chương trình như sau:

```
So cac doi tuong duoc tao ra la  = 1
So cac doi tuong duoc tao ra la  = 3
Object ID la  1
Object ID la  2
Object ID la  3
Doi tuong 3 duoc huy bo
Doi tuong 2 duoc huy bo
Doi tuong 1 duoc huy bo
```

Bài tập

1. Xây dựng lớp thời gian **Time**. Dữ liệu thành phần bao gồm giờ, phút giây. Các hàm thành phần bao gồm: hàm tạo, hàm truy cập dữ liệu, hàm `normalize()` để chuẩn hóa dữ liệu nằm trong khoảng quy định của giờ ($0 \le$ giờ < 24) , phút ($0 \le$ phút <60), giây ($0 \le$ giây <60), hàm `advance(int h, int m, int s)` để tăng thời gian hiện hành của đối tượng đang tồn tại, hàm `reset(int h, int m, int s)` để chỉnh lại thời gian hiện hành của một đối tượng đang tồn tại và một hàm print() để hiển thị dữ liệu.
2. Xây dựng lớp **Date**. Dữ liệu thành phần bao gồm ngày, tháng, năm. Các hàm thành phần bao gồm: hàm tạo, hàm truy cập dữ liệu, hàm `normalize()` để chuẩn hóa dữ liệu nằm trong khoảng quy định của ngày ($1 \le$ ngày <daysIn(tháng)), tháng ($1 \le$ tháng $<$ 12), năm (năm ≥ 1), hàm daysIn(int) trả về số ngày trong tháng, hàm `advance(int y, int m, int d)` để tăng ngày hiện lên các năm y, tháng m, ngày d của đối tượng đang tồn tại, hàm `reset(int y, int m, int d)` để đặt lại ngày cho một đối tượng đang tồn tại và một hàm print() để hiển thị dữ liệu.
3. Thực hiện một lớp **String**. Mỗi đối tượng của lớp sẽ đại diện một chuỗi ký tự. Những thành phần dữ liệu là chiều dài chuỗi, và chuỗi ký tự. Các hàm thành phần bao gồm: hàm tạo, hàm truy cập, hàm hiển thị, hàm `character(int i)` trả về một ký tự trong chuỗi được chỉ định bằng tham số i.
4. Xây dựng lớp ma trận có tên là **Matrix** cho các ma trận, các hàm thành phần bao gồm: hàm tạo mặc định, hàm nhập xuất ma trận, cộng, trừ, nhân hai ma trận.
5. Xây dựng lớp ma trận có tên là **Matrix** cho các ma trận vuông, các hàm thành phần bao gồm: hàm tạo mặc định, hàm nhập xuất ma trận, tính định thức và tính ma trận nghịch đảo.
6. Xây dựng lớp **Stack** cho ngăn xếp kiểu int. Các hàm thành phần bao gồm: Hàm tạo mặc định, hàm hủy, hàm `isEmpty()` kiểm tra stack có rỗng không, hàm `isFull()` kiểm tra stack có đầy không, hàm `push()` , `pop()`, hàm in nội dung ngăn xếp. Sử dụng một mảng để thực hiện.
7. Xây dựng lớp hàng đợi **Queue** chứa phần tử kiểu int. Các hàm thành phần bao gồm: hàm tạo, hàm hủy và những toán tử hàng đợi thông thường: hàm `insert()` để thêm phần tử vào hàng đợi, hàm `remove()` để loại bỏ phần tử, hàm `isEmpty()` kiểm tra hàng đợi có rỗng không, hàm `isFull()` kiểm tra hàng đợi có đầy không. Sử dụng một mảng để thực hiện.
4. Xây dựng lớp đa thức và các phương thức cộng, trừ hai đa thức.

8. Xây dựng lớp **Sinhvien** để quản lý hộ tên sinh viên, năm sinh, điểm thi 9 môn học của các sinh viên. Cho biết sinh viên nào được làm khóa luận tốt nghiệp, bao nhiêu sinh viên thi tốt nghiệp, bao nhiêu sinh viên thi lại, tên môn thi lại> Tiêu chuẩn để xét như sau:
   - Sinh viên làm khóa luận phải có điểm trung bình từ 7 trở lên, trong đó không có môn nào dưới 5.
   - Sinh viên thi tố nghiệp khi điểm trung bình nhỏ hơn 7 và điểm các môn không dưới 5.
   - Sinh viên thi lại môn dưới 5.
9. Xây dựng lớp **vector** để lưu trữ các vectơ gồm các số thực. Các thành phần dữ liệu bao gồm:
   - Kích thước vectơ.
   - Một mảng động chứa các thành phần của vectơ.

Các hàm thành phần bao gồm hàm tạo, hàm hủy, hàm tính tích vô hướng hai vectơ, tính chuẩn của vectơ (theo chuẩn bất kỳ nào đó).

10. Xây dựng lớp **Phanso** với dữ liệu thành phần là tử và mẫu số. Các hàm thành phần bao gồm:
    - Cộng hai phân số, kết quả phải dược tối giản
    - Trừ hai phân số, kết quả phải dược tối giản
    - Nhân hai phân số, kết quả phải dược tối giản
    - Chia hai phân số, kết quả phải dược tối giản

CHươNG 4

TOáN Tử Tải Bội

Chương 4 trình bày các vấn đề sau:

- *Định nghĩa toán tử tải bội*
- *Một số lưu ý khi xây dựng toán tử tải bội*
- *Một số ví dụ minh họa*

Bài 43 : 4.1. Định nghĩa toán tử tải bội

Các toán tử cùng tên thực hiện nhiều chức năng khác nhau được gọi là toán tử tải bội. Dạng định nghĩa tổng quát của toán tử tải bội như sau:

```
Kiểu_trả_về   operator   op(danh sách tham số)
        {//thân toán tử}
```

Trong đó: Kiểu_trả_về là kiểu kết quả thực hiện của toán tử.

op là tên toán tử tải bội

operator op(danh sách tham số) gọi là hàm toán tử tải bội, nó có thể là hàm thành phần hoặc là hàm bạn, nhưng không thể là hàm tĩnh. Danh sách tham số được khai báo tương tự khai báo biến nhưng phải tuân theo những quy định sau:

- Nếu toán tử tải bội là hàm thành phần thì: không có tham số cho toán tử một ngôi và một tham số cho toán tử hai ngôi. Cũng giống như hàm thành phần thông thường, hàm thành phần toán tử có đối đầu tiên (không tường minh) là con trỏ this .
- Nếu toán tử tải bội là hàm bạn thì: có một tham số cho toán tử một ngôi và hai tham số cho toán tử hai ngôi.

Quá trình xây dựng toán tử tải bội được thực hiện như sau:

- Định nghĩa lớp để xác định kiểu dữ liệu sẽ được sử dụng trong các toán tử tải bội
- Khai báo hàm toán tử tải bội trong vùng public của lớp
- Định nghĩa nội dung cần thực hiện

Bài 44 : 4.2. Một số lưu ý khi xây dựng toán tử tải bội

1. Trong C++ ta có thể đưa ra nhiều định nghĩa mới cho hầu hết các toán tử trong C++, ngoại trừ những toán tử sau đây:
   - Toán tử xác định thành phần của lớp (‘.’)
   - Toán tử phân giải miền xác định (‘::’)
   - Toán tử xác định kích thước (‘sizeof’)
   - Toán tử điều kiện 3 ngôi (‘?:’)
2. Mặc dầu ngữ nghĩa của toán tử được mở rộng nhưng cú pháp, các quy tắc văn phạm như số toán hạng, quyền ưu tiên và thứ tự kết hợp thực hiện của các toán tử vẫn không có gì thay đổi.

3. Không thể thay đổi ý nghĩa cơ bản của các toán tử đã định nghĩa trước, ví dụ không thể định nghĩa lại các phép toán +, - đối với các số kiểu int, float.
4. Các toán tử = , ( ) , [ ] , -> yêu cầu hàm toán tử phải là hàm thành phần của lớp, không thể dùng hàm bạn để định nghĩa toán tử tải bội.

4.3. Một số ví dụ

Ví dụ 4.1 Toán tử tải bội một ngôi, dùng hàm bạn

```
#include <iostream.h>
#include <conio.h>
class Diem
   {
     private:
       float x,y,z;
     public:
       Diem() {}
       Diem(float x1,float y1,float z1)
    { x = x1; y = y1; z=z1;}
       friend Diem  operator -(Diem d)
     {
        Diem d1;
        d1.x = -d.x; d1.y = -d.y;d1.z=-d.z;
        return d1;
     }
       void hienthi()
     { cout<<"Toa do: "<<x<<" "<<y <<" "
            <<z<<endl;}
   };

void main()
   {
     clrscr();
     Diem p(2,3,-4),q;
     q = -p;
     p.hienthi();
     q.hienthi();
     getch();
   }
```

Ví dụ 4.2 Toán tử tải bội hai ngôi, dùng hàm bạn

```
#include <iostream.h>
#include <conio.h>
class Diem
   {
     private:
       float x,y,z;
     public:
       Diem() {}
       Diem(float x1,float y1,float z1)
    { x = x1; y = y1; z=z1;}
       friend Diem  operator +(Diem d1, Diem d2)
     {
        Diem tam;
        tam.x = d1.x + d2.x;
        tam.y = d1.y + d2.y;
        tam.z = d1.z + d2.z;
        return tam;
     }
       void hienthi()
     { cout<<x<<" "<<y <<" " <<z<<endl;}
   };

void main()
   {
     clrscr();
     Diem d1(3,-6,8),d2(4,3,7),d3;
     d3=d1+d2;
     d1.hienthi();
     d2.hienthi();
     cout<<"\n Tong hai diem co toa do la :";
     d3.hienthi();
     getch();
   }
```

Ví dụ 4.3 Toán tử tải bội hai ngôi, dùng hàm thành phần

```
#include <iostream.h>
```

```
#include <conio.h>
class Diem
   {
     private:
       float x,y;
     public:
       Diem() {}
       Diem(float x1,float y1)
       { x = x1; y = y1;}
       Diem operator -()
     { x = -x; y = -y; z = -z;
       return (*this); }
       void hienthi()
       { cout<<"Toa do: "<<x<<" "<<y <<" z = "<<z
          <<" \n";}
   };

void main()
   {
     clrscr();
     Diem p(2,3,-4),q;
     p.hienthi();
     q = -p;
     q.hienthi();
     getch();
   }
```

Ví dụ 4.4 Toán tử tải bội hai ngôi, dùng hàm thành phần

```
#include <iostream.h>
#include <conio.h>
class Diem
   {
     private:
       float x,y,z;
     public:
       Diem() {}
       Diem(float x1,float y1,float z1)
```

```
    { x = x1; y = y1; z=z1;}
        Diem  operator +(Diem d2)
      {
         x = x + d2.x;
         y = y + d2.y;
         z = z + d2.z;
         return (*this);
      }
        void hienthi()
      { cout<<"\n x="<<x<<" y= "<<y<<" z =  " << z
          <<endl;}
   };

void main()
   {
     clrscr();
     Diem d1(3,-6,8),d2(4,3,7),d3;
     d1.hienthi();
     d2.hienthi();
     d3=d1+d2;
     cout<<"\n Tong hai diem co toa do la :";
     d3.hienthi();
     getch();
   }
```

Ví dụ 4.5

```
#include <iostream.h>
#include <conio.h>
class Diem
   {
     private:
       int x,y;
     public:
       Diem() {}
       Diem(int x1,int y1)
       { x = x1; y = y1;}
       void operator -()
```

```
     {
        x = -x; y = -y;
       }
       void hienthi()
     { cout<<"Toa do: "<<x<<" "<<y <<" \n";}
   };
void main()
   {
     clrscr();
     Diem p(2,3),q;
     p.hienthi();
     -p;
     p.hienthi();
     getch();
   }
```

Ví dụ 4.6 Toán tử tải bội hai ngôi

```
#include <iostream.h>
#include <conio.h>
class sophuc
{float a,b;
 public : sophuc() {}
    sophuc(float x, float y)
       {a=x; b=y;}
    sophuc operator +(sophuc c2)
         {  sophuc c3;
       c3.a= a + c2.a ;
       c3.b= b + c2.b ;
       return (c3);
          }
    void  hienthi(sophuc c)
    { cout<<c.a<<" + "<<c.b<<"i"<<endl; }
};
void main()
{ clrscr();
  sophuc d1 (2.1,3.4);
  sophuc d2 (1.2,2.3) ;
```

```
  sophuc d3 ;
  d3 = d1+d2;   //d3=d1.operator +(d2);
  cout<<"d1= ";d1.hienthi(d1);
  cout<<"d2= ";d2.hienthi(d2);
  cout<<"d3= ";d3.hienthi(d3);
  getch();
  }
```

Chú ý: Trong các hàm toán tử thành phần hai ngôi (có hai toán hạng) thì con trỏ this ứng với toán hạng thứ nhất, vì vậy trong tham số của toán tử chỉ cần dùng một tham số tường minh để biểu thị toán hạng thứ hai .

Ví dụ 4.7 Phiên bản 2 của ví dụ 4.6

```
#include <iostream.h>
#include <conio.h>
class sophuc
{float a,b;
 public : sophuc() {}
    sophuc(float x, float y)
       {a=x; b=y;}
    sophuc operator +(sophuc c2)
         {
      a= a + c2.a ;
      b= b + c2.b ;
      return (*this);
         }
    void  hienthi(sophuc c)
    { cout<<c.a<<" + "<<c.b<<"i"<<endl; }
};
void main()
{ clrscr();
  sophuc d1 (2.1,3.4);
  sophuc d2 (1.2,2.3) ;
  sophuc d3 ;
  cout<<"d1= ";d1.hienthi(d1);
  cout<<"d2= ";d2.hienthi(d2);
  d3 = d1+d2;   //d3=d1.operator +(d2);
  cout<<"d3= ";d3.hienthi(d3);
```

```
  getch();
  }
```

Ví dụ 4.8 Phiên bản 3 của ví dụ 4.6

```
#include <iostream.h>
#include <conio.h>
class sophuc
{float a,b;
 public : sophuc() {}
    sophuc(float x, float y)
       {a=x; b=y;}
    friend sophuc operator +(sophuc c1,sophuc c2)
      { sophuc c;
     c.a= c1.a + c2.a ;
     c.b= c1.b + c2.b ;
     return (c);
         }
    void  hienthi(sophuc c)
    { cout<<c.a<<" + "<<c.b<<"i"<<endl; }
};
void main()
{ clrscr();
  sophuc d1 (2.1,3.4);
  sophuc d2 (1.2,2.3) ;
  sophuc d3 ;
  cout<<"d1= ";d1.hienthi(d1);
  cout<<"d2= ";d2.hienthi(d2);
  d3 = d1+d2;   //d3=operator +(d1,d2);
  cout<<"d3= ";d3.hienthi(d3);
  getch();
  }
```

Ví dụ 4.9 Toán tử tải bội trên lớp chuỗi ký tự

```
#include <iostream.h>
#include <string.h>
#include <conio.h>
class string
{ char s[80];
```

```
  public:
    string() { *s='\0'; }
    string(char *p) { strcpy(s,p); }
    char *get() { return s;}
    string operator + (string s2);
    string operator = (string s2);
    int operator < (string s2);
    int operator > (string s2);
    int operator == (string s2);
};
string string::operator +(string s2)
{
 strcat(s,s2.s);
 return *this ;
}

string string::operator =(string s2)
{
 strcpy(s,s2.s) ;
 return *this;
}
int string::operator <(string s2)
{
  return strcmp(s,s2.s)<0 ;
}
int string::operator >(string s2)
{
  return strcmp(s,s2.s)>0 ;
}
int string::operator ==(string s2)
{
  return strcmp(s,s2.s)==0 ;
}
void main()
{ clrscr();
  string o1 ("Trung Tam "), o2 (" Tin hoc"), o3;
```

```
  cout<<"o1 = "<<o1.get()<<'\n';
  cout<<"o2 = "<<o2.get()<<'\n';
  if (o1 > o2)
    cout << "o1 > o2 \n";
  if (o1 < o2)
    cout << "o1 < o2 \n";
  if (o1 == o2)
    cout << "o1 bang o3 \n";
  o3=o1+o2;
  cout<<"o3 ="<<o3.get()<<'\n';  //Trung tam tin hoc
  o3=o2;
  cout<<"o2 = "<<o2.get()<<'\n'; //Tin hoc
  cout<<"o3 = "<<o3.get()<<'\n'; //Tin hoc
  if (o2 == o3)
    cout << "o2 bang o3 \n";
  getch();
}
```

Bài 45 : 4.4. Định nghĩa chồng các toán tử ++ , --

Ta có thể định nghĩa chồng cho các toán tử ++/-- theo quy định sau:

- Toán tử ++/-- dạng tiền tố trả về một tham chiếu đến đối tượng thuộc lớp.
- Toán tử ++/-- dạng tiền tố trả về một đối tượng thuộc lớp.

Ví dụ 4.10

```
#include <iostream.h>
#include <conio.h>
class Diem
   {
     private:
       int x,y;
     public:
       Diem() {x = y = 0;}
       Diem(int x1, int y1)
     {x = x1;
      y = y1;}
  Diem & operator ++(); //qua tai toan tu ++ tien to
  Diem operator ++(int); //qua tai toan tu ++ hau to
  Diem & operator --(); //qua tai toan tu -- tien to
```

```
  Diem operator --(int); //qua tai toan tu -- hau to
  void hienthi()
   {
     cout<<" x = "<<x<<" y = "<<y;
     }
   };
Diem & Diem::operator ++()
 {
   x++;
   y++;
   return (*this);
  }
Diem Diem::operator ++(int)
 {
   Diem temp = *this;
   ++*this;
   return temp;
  }
Diem & Diem::operator --()
 {
   x--;
   y--;
   return (*this);
  }
Diem Diem::operator --(int)
 {
   Diem temp = *this;
   --*this;
   return temp;
   }
void main()
   {
     clrscr();
     Diem d1(5,10),d2(20,25),d3(30,40),d4(50,60);
     cout<<"\nd1 : ";d1.hienthi();
     ++d1;
```

```
    cout<<"\n Sau khi tac dong cac toan tu tang
    truoc :";
    cout<<"\nd1 : ";d1.hienthi();
    cout<<"\nd2 : ";d2.hienthi();
    d2++;
    cout<<" \n Sau khi tac dong cac toan tu tang
    sau";
    cout<<"\nd2 : ";d2.hienthi();
    cout<<"\nd3 : ";d3.hienthi();
    --d3;
    cout<<"\n Sau khi tac dong cac toan tu giam
    truoc :";
    cout<<"\nd3 : ";d3.hienthi();
    cout<<"\nd4 : ";d4.hienthi();
    d4--;
    cout<<"\n Sau khi tac dong cac toan tu giam
    sau : ";
    cout<<"\nd4 : ";d4.hienthi();
    getch();
  }
```

Chương trình cho kết quả như sau:

```
d1 :  x = 5 y = 10
 Sau khi tac dong cac toan tu tang truoc :
d1 :  x = 6 y = 11
d2 :  x = 20 y = 25
 Sau khi tac dong cac toan tu tang sau
d2 :  x = 21 y = 26
d3 :  x = 30 y = 40
 Sau khi tac dong cac toan tu giam truoc :
d3 :  x = 29 y = 39
d4 :  x = 50 y = 60
 Sau khi tac dong cac toan tu giam sau :
d4 :  x = 49 y = 59
```

Chú ý: Đối số int trong dạng hậu tố là bắt buộc, dùng để phân biệt với dạng tiền tố, thường nó mang trị mặc định là 0.

Bài 46 : 4.5. Định nghĩa chồng toán tử << và >>

Ta có thể định nghĩa chồng cho hai toán tử vào/ra << và >> kết hợp với **cout** và **cin** (cout<< và cin>>), cho phép các đối tượng đứng bên phải chúng. Lúc đó ta có thể thực hiện các thao tác vào ra như nhập dữ liệu từ bàn phím cho các đối tượng, hiển thị giá trị

thành phần dữ liệu của các đối tượng ra màn hình. Hai hàm toán tử << và >> phải là hàm tự do và khai báo là hàm bạn của lớp.

Ví dụ 4.11

```
#include <iostream.h>
#include <conio.h>
class SO
 {
  private:
     int giatri;
  public:
     SO(int x=0)
     {
       giatri = x;
      }
     SO (SO &tso)
     {
       giatri = tso.giatri;
      }
    friend istream& operator>>(istream&,SO&);
    friend ostream& operator<<(ostream&,SO&);
};
void main(){
   clrscr();
   SO so1,so2;
   cout<<"Nhap du lieu cho so1 va so2 " << endl;
   cin>>so1;
   cin>>so2;
   cout<<"Gia tri so1 la : " <<so1
       <<" so 2 la : "<<so2<<endl;
    getch();
}
istream& operator>>(istream& nhap,SO& so)
{
  cout << "Nhap gia tri so :";
  nhap>> so.giatri;
  return nhap; }
```

```
ostream& operator<<(ostream& xuat,SO& so)
{
  xuat<< so.giatri;
  return xuat;
}
```

Bài tập

1. Định nghĩa các phép toán tải bội =, ==, ++, --, +=, <<, >> trên lớp Time (bài tập 1 chương 3).
2. Định nghĩa các phép toán tải bội =, ==, ++, --, +=, <<, >> trên lớp Date (bài tập 2 chương 3).

1. Định nghĩa các phép toán tải bội +, -, *, =, ==, != trên lớp các ma trận vuông.
2. Định nghĩa các phép toán tải bội +, -, * trên lớp đa thức.
3. Định nghĩa các phép toán tải bội +, -, *, /, =, ==, +=, -=, *=, /= , <, >, <=, >=, != , ++, -- trên lớp Phanso (bài tập 10 chương 3).
4. Ma trận được xem là một vectơ mà mỗi thành phần của nó là một vectơ. Theo nghĩa đó, hãy định nghĩa lớp **Matran** dựa trên vectơ. Tìm cách để chương trình dịch hiểu được phép truy nhập m[i][j], trong đó m là một đối tượng thuộc lớp **Matran**.

Chương 5

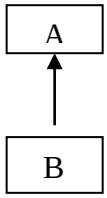
kế thừa

Chương 5 trình bày các vấn đề sau:

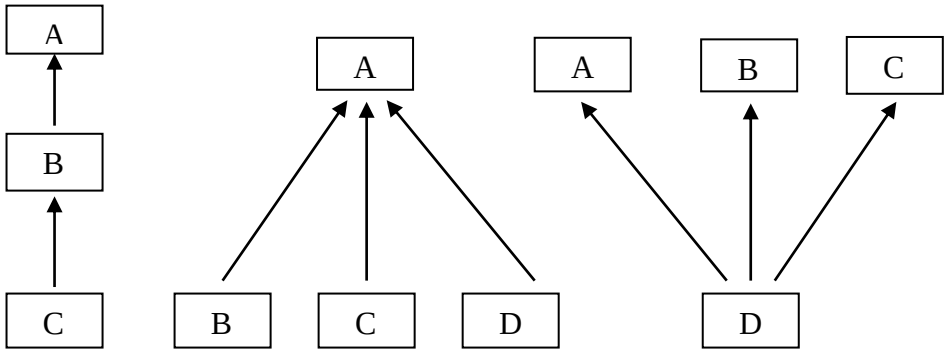
- *Đơn kế thừa, đa kế thừa*
- *Hàm tạo và hàm hủy đối với sự kế thừa*
- *Hàm ảo, lớp cơ sở ảo*

Bài 47 : 5.1. Giới thiệu

Kế thừa là một trong các khái niệm cơ sở của phương pháp lập trình hướng đối tượng. Tính kế thừa cho phép định nghĩa các lớp mới từ các lớp đã có. Một lớp có thể là lớp cơ sở cho nhiều lớp dẫn xuất khác nhau. Lớp dẫn xuất sẽ kế thừa một số thành phần (dữ liệu và hàm) của lớp cơ sở, đồng thời có thêm những thành phần mới. Có hai loại kế thừa là: đơn kế thừa và đa kế thừa, có thể minh họa qua các hình vẽ sau đây:



Hình 5.1. Đơn kế thừa, lớp A là lớp cơ sở của lớp B



Hình 5.2. Đa kế thừa

Hình (a): Lớp A là lớp cơ sở của lớp B, lớp B là lớp cơ sở của lớp C

Hình (b): Lớp A là lớp cơ sở của các lớp B, C, D

Hình (c): Lớp A, B, C là lớp cơ sở của lớp D

5.2. Đơn kế thừa

47.15.2.1. Định nghĩa lớp dẫn xuất từ một lớp cơ sở

Giả sử đã định nghĩa lớp A. Cú pháp để xây dựng lớp B dẫn xuất từ lớp A như sau:

```
class  B: mode A
{
```

```
    private:
            // Khai báo các thuộc tính của lớp B
    public:
          // Định nghĩa các hàm thành phần của lớp B
};
```

Trong đó mode có thể là **private** hoặc **public** với ý nghĩa như sau:

- Kế thừa theo kiểu public thì tất cả các thành phần public của lớp cơ sở cũng là thành phần public của lớp dẫn xuất.
- Kế thừa theo kiểu private thì tất cả các thành phần public của lớp cơ sở sẽ trở thành các thành phần private của lớp dẫn xuất.

Chú ý: Trong cả hai trường hợp ở trên thì thành phần private của lớp cơ sở là không được kế thừa. Như vậy trong lớp dẫn xuất không cho phép truy nhập đến các thành phần private của lớp cơ sở.

47.25.2.2. Truy nhập các thành phần trong lớp dẫn xuất

Thành phần của lớp dẫn xuất bao gồm: các thành phần khai báo trong lớp dẫn xuất và các thành phần mà lớp dẫn xuất thừa kế từ các lớp cơ sở. Quy tắc sử dụng các thành phần trong lớp dẫn xuất được thực hiện theo theo mẫu như sau:

Tên đối tượng.Tên_lớp::Tên_thành_phần

Khi đó chương trình dịch C++ dễ dàng phân biệt thành phần thuộc lớp nào.

Ví dụ 5.1 Giả sử có các lớp A và B như sau:

```
class A
{
public: int n;
        void nhap()
    {
      cout<<"\n Nhap n = ";
      cin>>n;
    }
};
  class B: public A
  {
  public: int m;
        void nhap()
    {
      cout<<"\n Nhap m = ";
      cin>>m;
```

```
        }
};
```

Xét khai báo: B ob;

Lúc đó: ob.B::m là thuộc tính n khai báo trong B

ob.A::n là thuộc tính n thừa kế từ lớp A

ob.D::nhap() là hàm nhap() định nghĩa trong lớp B

ob.A::nhap() là hàm nhap() định nghĩa trong lớp A

Chú ý: Để sử dụng các thành phần của lớp dẫn xuất, có thể không dùng tên lớp, chỉ dùng tên thành phần. Khi đó chương trình dịch phải tự phán đoán để biết thành phần đó thuộc lớp nào: trước tiên xem thành phần đang xét có trùng tên với các thành phần nào của lớp dẫn xuất không? Nếu trùng thì đó thành phần của lớp dẫn xuất. Nếu không trùng thì tiếp tục xét các lớp cơ sở theo thứ tự: các lớp có quan hệ gần với lớp dẫn xuất sẽ được xét trước, các lớp quan hệ xa hơn xét sau. Chú ý trường hợp thành phần đang xét có mặt đồng thời trong 2 lớp cơ sở có cùng một đẳng cấp quan hệ với lớp dẫn xuất. Trường hợp này chương trình dịch không thể quyết định được thành phần này thừa kế từ lớp nào và sẽ đưa ra một thông báo lỗi.

47.35.2.3. Định nghĩa lại các hàm thành phần của lớp cơ sở trong lớp dẫn xuất

Trong lớp dẫn xuất có thể định nghĩa lại hàm thành phần của lớp cơ sở. Như vậy có hai phiên bản khác nhau của hàm thành phần trong lớp dẫn xuất. Trong phạm vi lớp dẫn xuất, hàm định nghĩa lại "che khuất" hàm được định nghĩa. Việc sử dụng hàm nào cần tuân theo quy định ở trên.

Chú ý: Việc định nghĩa lại hàm thành phần khác với định nghĩa hàm quá tải. Hàm định nghĩa lại và hàm bị định nghĩa lại giống nhau về tên, tham số và giá trị trả về. Chúng chỉ khác nhau về vị trí: một hàm đặt trong lớp dẫn xuất và hàm kia thì ở trong lớp cơ sở. Trong khi đó, các hàm quá tải chỉ có cùng tên, nhưng khác nhau về danh sách tham số và tất cả chúng thuộc cùng một lớp. Định nghĩa lại hàm thành phần chính là cơ sở cho việc xây dựng tính đa hình của các hàm.

C++ cho phép đặt trùng tên thuộc tính trong các lớp cơ sở và lớp dẫn xuất. Các thành phần cùng tên này có thể cùng kiểu hay khác kiểu. Lúc này bên trong đối tượng của lớp dẫn xuất có tới hai thành phần khác nhau có cùng tên, nhưng trong phạm vi lớp dẫn xuất, tên chung đó nhằm chỉ định thành phần được khai báo lại trong lớp dẫn xuất. Khi muốn chỉ định thành phần trùng tên trong lớp cơ sở phải dùng tên lớp toán tử '::' đặt trước tên hàm thành phần.

Ví dụ 5.2 Xét các lớp A và B được xây dựng như sau:

```
class A
  {
  private:
       int a, b, c;
  public:
      void nhap()
       { cout<<"\na = "; cin>>a;
         cout<<"\nb = "; cin>>b;
         cout<<"\nc = "; cin>>c;
          }
       void hienthi()
      { cout<<"\na = "<<a<<" b = "<<b<<" c = "<<c; }
};
class B: private A
{
  private:
       double d;
    public:
      void nhap_d()
       { cout<<"\nd = "; cin>>d;
          }
    };
```

Lớp B kế thừa theo kiểu private từ lớp A, các thành phần public của lớp A là các hàm nhap() và hienthi() trở thành thành phần private của lớp B.

Xét khai báo : `B ob;` Lúc đó các câu lệnh sau đây là lỗi :

```
  ob.nhap();
  ob.d=10;
  ob.a = ob.b = ob.c = 5;
  ob.hienthi();
```

bởi vì đối tượng ob không thể truy nhập vào các thành phần private của lớp A và B.

Ví dụ 5.3 Chương trình minh họa đơn kế thừa theo kiểu public:

```
#include <iostream.h>
#include <conio.h>
class A
  {
    int a; //Bien private, khong duoc ke thua
```

```
    public:
       int b; //Bien public, se duoc ke thua
       void get_ab();
       int get_a(void);
       void show_a(void);
  };
class B: public A
  {
    int c;
    public:
       void mul(void);
       void display(void);
  };
void A::get_ab(void)
   { a = 5; b = 10;}
int A::get_a()
   { return a;}
void A::show_a()
   { cout << "a = " << a << " \n";}
void B::mul()
   { c = b*get_a();}
void B::display()
   {
     cout << "a = " << get_a() << "\n";
     cout << "b = " << b << "\n";
     cout << "c = " << c << "\n";
   }
void main()
   { clrscr();
     B d;
     d.get_ab(); //Ke thua tu A
     d.mul();
     d.show_a(); //Ke thua tu A
     d.display();
     d.b = 20;
     d.mul();
```

```
     d.display();
     getch();
   }
```

Chương trình cho kết quả:

```
a = 5
a = 5
b = 10
c = 50
a = 5
b = 20
c = 100
```

Ví dụ 5.4 Chương trình sau là minh họa khác cho trường hợp kế thừa đơn:

```
#include <conio.h>
#include <iostream.h>
class Diem
 {
  private:
     double x, y;
  public:
      void nhap()
        {
         cout<<"\n x = "; cin>>x;
         cout<<"\n y = "; cin>>y;
        }
      void hienthi()
       {
         cout<<"\n x = "<<x<<" y = "<<y;
       }
 };
class Hinhtron: public Diem
 {
   private:
   double r;
   public:
   void nhap_r()
      {
```

```
          cout<<"\n r = "; cin>>r;
        }
        double get_r()
    {
        return r;
      }
};
void main()
{
    Hinhtron  h;
  clrscr();
  cout<<"\n Nhap toa do tam va ban kinh hinh tron";
  h.nhap();
  h.nhap_r();
  cout<<"\n Hinh tron co tam:";h.hienthi();
  cout<<"\n Co ban kinh = " << h.get_r();
  getch();
    }
```

Chương trình cho kết quả:

```
Nhap toa do tam va ban kinh hinh tron
x = 2
y = 3
r = 10
Hinh tron co tam:
x = 2 y = 3
Co ban kinh = 10
```

47.45.2.4. Hàm tạo đối với tính kế thừa

Các hàm tạo của lớp cơ sở là không được kế thừa. Một đối tượng của lớp dẫn xuất về thực chất có thể xem là một đối tượng của lớp cơ sở, vì vậy việc gọi hàm tạo lớp dẫn xuất để tạo đối tượng của lớp dẫn xuất sẽ kéo theo việc gọi đến một hàm tạo của lớp cơ sở. Thứ tự thực hiện của các hàm tạo sẽ là: hàm tạo cho lớp cơ sở, rồi đến hàm tạo cho lớp dẫn xuất.

C++ thực hiện điều này bằng cách: trong định nghĩa của hàm tạo lớp dẫn xuất, ta mô tả một lời gọi tới hàm tạo trong lớp cơ sở. Cú pháp để truyền tham số từ lớp dẫn xuất đến lớp cơ sở như sau:

Tên_ lớp_dẫn_xuất(danh sách đối):Tên_ lớp_cơ_sở (danh sách đối)

```
        {
              //thân hàm tạo của lớp dẫn xuất
      } ;
```

Trong phần lớn các trường hợp, hàm tạo của lớp dẫn xuất và hàm tạo của lớp cơ sở sẽ không dùng tham số giống nhau. Trong trường hợp cần truyền một hay nhiều tham số cho mỗi lớp, ta phải truyền cho hàm tạo của lớp dẫn xuất **tất cả** các tham số mà cả hai lớp dẫn xuất và cơ sở cần đến. Sau đó, lớp dẫn xuất chỉ truyền cho lớp cơ sở những tham số nào mà lớp cơ sở cần.

Ví dụ 5.5 Chương trình sau minh họa cách truyền tham số cho hàm tạo của lớp cơ sở:

```
#include <iostream.h>
#include <conio.h>
class Diem
  {
    private:
      double x, y;
    public:
        Diem()
        {
            x=y=0.0;
               }
          Diem(double x1, double y1)
              {
                 x=x1; y=y1;
                 }
           void in()
           {
               cout<<"\nx="<<x<<"y="<<y;
                }
     };
class Hinhtron: public Diem
     {
          private:
         double r;
        public:
         Hinhtron()
          {
```

```
            r = 0.0;
                }
             Hinhtron(double x1,double y1,double r1): Diem
          (x1, y1)
       {
         r=r1;
        }
      double get_r()
       {
          return r;
        }
   };
void main()
 {
    Hinhtron  h(2.5, 3.5, 8);
    clrscr();
    cout<<"\n Hinh tron co tam:";
    h.in();
    cout<<"\n Co ban kinh =" << h.get_r();
    getch();
 }
```

Chú ý: Các tham số mà hàm tạo của lớp dẫn xuất truyền cho hàm tạo của lớp cơ sở không nhất thiết phải lấy hoàn toàn y như từ các tham số nó nhận được. Ví dụ:

```
Hinhtron(double x1,double y1,double r1):Diem (x1/2, y1/2)
```

47.55.2.5. Hàm hủy đối với tính kế thừa

Hàm hủy của lớp cơ sở cũng không được kế thừa. Khi cả lớp cơ sở và lớp dẫn xuất có các hàm hủy và hàm tạo, các hàm tạo thi hành theo thứ tự dẫn xuất. Các hàm hủy được thi hành theo thứ tự ngược lại. Nghĩa là, hàm tạo của lớp cơ sở thi hành trước hàm tạo của lớp dẫn xuất, hàm hủy của lớp dẫn xuất thi hành trước hàm hủy của lớp cơ sở.

Ví dụ 5.6

```
#include <iostream.h>
#include <conio.h>
class CS
{ public:
   CS()
     {cout<<"\nHam tao lop co so";}
```

```
    ~CS()
       {cout<<"\nHam huy lop co so";}
};
class DX:public CS
{ public:
    DX()
       {cout<<"\nHam tao lop dan xuat";}
    ~DX()
       {cout<<"\nHam huy lop dan xuat";}
};
void main()
 { clrscr();
   DX ob;
   getch();
 }
```

Chương trình này cho kết quả như sau :

```
Ham tao lop co so
Ham tao dan xuat
Ham huy lop dan xuat
Ham huy lop co so
```

47.65.2.6. Khai báo protected

Ta đã biết các thành phần khai báo private không được kế thừa trong lớp dẫn xuất. Có thể giải quyết vấn đề này bằng cách chuyển chúng sang vùng public. Tuy nhiên cách làm này lại phá vỡ nguyên lý che dấu thông tin của LTHĐT. C++ đưa ra cách giải quyết khác là sử dụng khai báo **protected.** Các thành phần **protected** có phạm vi truy nhập rộng hơn so với các thành phần private, nhưng hẹp hơn so với các thành phần public.

Các thành phần **protected** của lớp cơ sở hoàn toàn giống các thành phần private *ngoại trừ một điểm* là chúng có thể kế thừa từ lớp dẫn xuất trực tiếp từ lớp cơ sở. Cụ thể như sau:

- Nếu kế thừa theo kiểu public thì các thành phần proteted của lớp cơ sở sẽ trở thành các thành phần protected của lớp dẫn xuất.
- Nếu kế thừa theo kiểu private thì các thành phần proteted của lớp cơ sở sẽ trở thành các thành phần private của lớp dẫn xuất.

47.75.2.7. Dẫn xuất protected

Ngoài hai kiểu dẫn xuất đã biết là private và public, C++ còn đưa ra kiểu dẫn xuất **protected**. Trong dẫn xuất loại này thì các thành phần public, protected trong lớp cơ sở trở thành các thành phần **protected** trong lớp dẫn xuất.

Bài 48 : 5.3. Đa kế thừa

48.15.3.1. Định nghĩa lớp dẫn xuất từ nhiều lớp cơ sở

Giả sử đã định nghĩa các lớp A, B. Cú pháp để xây dựng lớp C dẫn xuất từ lớp A và B như sau:

```
class  C: mode A, mode B
   {
   private:
         // Khai báo các thuộc tính
   public:
      // Các hàm thành phần
  };
```

trong đó mode có thể là private, public hoặc protected. ý nghĩa của kiểu dẫn xuất này giống như trường hợp đơn kế thừa.

48.25.3.2. Một số ví dụ về đa kế thừa

Ví dụ 5.7 Chương trình sau minh họa một lớp kế thừa từ hai lớp cơ sở:

```
#include <iostream.h>
#include <string.h>
#include <conio.h>

class M
{ protected :
        int m;
  public :
        void getm(int x) {m=x;}
};
class N
{ protected :
        int n;
  public :
        void getn(int y) {n=y;}
};
```

```
class P : public M,public N
{
  public :
       void display(void)
      { cout<<"m= "<<m<<endl;
        cout<<"n= "<<n<<endl;
        cout<<"m * n = "<<m*n<<endl;
      }
};
void main()
{ P ob;
  clrscr();
  ob.getm(10);
  ob.getn(20);
  ob.display();
  getch();
}
```

Chương trình cho kết quả như sau:

```
m = 10
n = 20
m*n = 200
```

Ví dụ 5.8 Chương trình sau minh họa việc quản lý kết quả thi của một lớp không quá 100 sinh viên. Chương trình gồm 3 lớp: lớp cơ sở sinh viên (sinhvien) chỉ lưu họ tên và số báo danh, lớp điểm thi (diemthi) kế thừa lớp sinh viên và lưu kết quả môn thi 1 và môn thi 2. Lớp kết quả (ketqua) lưu tổng số điểm đạt được của sinh viên.

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

class sinhvien
{ char hoten[25];
     protected:
       int sbd;
     public:
       void nhap()
```

```
         {  cout<<"\nHo ten :";gets(hoten);
          cout<<"So bao danh :";cin>>sbd;
        }
      void hienthi()
      { cout<<"So bao danh : "<<sbd<<endl;
      cout<<"Ho va ten sinh vien : "<<hoten<<endl;
       }
};

class diemthi : public sinhvien
{ protected :
       float d1,d2;
  public :
       void nhap_diem()
      {
        cout<<"Nhap diem hai mon thi : \n";
        cin>>d1>>d2;
      }
       void hienthi_diem()
        {  cout<<"Diem mon 1 :"<<d1<<endl;
         cout<<"Diem mon 2 :"<<d2<<endl;
        }
};
class ketqua : public diemthi
{
  float tong;
  public :
       void display()
      { tong = d1+d2;
        hienthi();
        hienthi_diem();
        cout<<"Tong so diem :"<<tong<<endl;
      }
};
void main()
{ int i,n; ketqua sv[100];
```

```
  cout<<"\n Nhap so sinh vien : ";
  cin>>n;
  clrscr();
  for(i=0;i<n;++i)
  { sv[i].nhap();
    sv[i].nhap_diem();
  }
  for(i=0;i<n;++i)
    sv[i].display();
  getch();
}
```

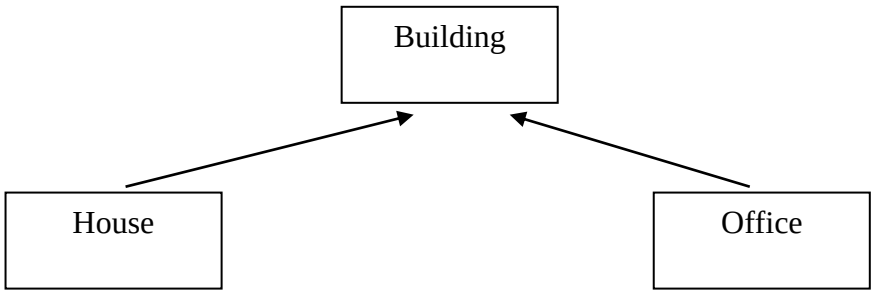
Ví dụ 5.9 Chương trình sau là sự mở rộng của chương trình ở trên, trong đó ngoài kết quả hai thi, mỗi sinh viên còn có thể có điểm thưởng. Chương trình mở rộng thêm một lớp ưu tiên (uutien).

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
class sinhvien
{ char hoten[25];
  protected : int sbd;
  public    :
       void nhap()
         {  cout<<"\nHo ten :";gets(hoten);
          cout<<"So bao danh :";cin>>sbd;
        }
      void hienthi()
    { cout<<"So bao danh : "<<sbd<<endl;
      cout<<"Ho va ten sinh vien : "<<hoten<<endl;
     }
};
class diemthi : public sinhvien
{ protected : float d1,d2;
  public :
       void nhap_diem()
      {
        cout<<"Nhap diem hai mon thi : \n";
```

```
        cin>>d1>>d2;
      }
       void hienthi_diem()
        {  cout<<"Diem mon 1 :"<<d1<<endl;
          cout<<"Diem mon 2 :"<<d2<<endl;
        }
};
class uutien
{ protected : float ut;
  public :
       void nhap_ut()
      {
        cout<<"\nNhap diem uu tien :";cin>>ut;
        }
     void hienthi_ut()
      {cout<<"Diem uu tien : "<<ut<<endl; }
};
class ketqua : public diemthi, public uutien
{ float tong;
  public :
       void display()
     { tong=d1+d2+ut;
       hienthi();
       hienthi_diem();
       hienthi_ut();
       cout<<"Tong so diem :"<<tong<<endl;
     }
};
void main()
{ int i,n; ketqua sv[100];
  cout<<"\n Nhap so sinh vien : ";
  cin>>n;
  clrscr();
  for(i=0;i<n;++i)
  { sv[i].nhap();
    sv[i].nhap_diem();
```

```
    sv[i].nhap_ut();
  }
  for(i=0;i<n;++i)
    sv[i].display();
  getch();
}
```

Ví dụ 5.10 Xem sơ đồ kế thừa các lớp như sau:



Hình 5.3.

Trong đó lớp cơ sở Building lưu trữ số tầng của một tòa nhà, tổng số phòng và tổng diện tích của tòa nhà. Lớp dẫn xuất House kế thừa lớp Building và lưu trữ số phòng ngủ, số phòng tắm. Lớp dẫn xuất Office từ lớp Building lưu trữ số máy điện thoại và số bình cứu hỏa. Chương trình sau minh họa việc tổ chức lưu trữ theo sơ đồ kế thừa này.

```
#include <iostream.h>
#include <conio.h>
class Building
    { protected :
       int floors; //tong so  tang
      int rooms; //tong so phong
      double footage; //tong so dien tich
    };
class house : public Building
    {   int bedrooms; //tong so phong ngu
      int bathrooms; //tong so phong tam
      public :
        house(int f, int r, int ft, int br, int bth)
      { floors=f; rooms=r; footage=ft;
        bedrooms=br; bathrooms=bth;
        }
      void show()
         { cout<<'\n';
```

```
          cout<<" So tang  : " <<floors <<'\n';
          cout<<" So phong : " <<rooms <<'\n';
        cout<<" So tong dien tich : "
            <<footage<<'\n';
        cout<<" So phong ngu : " <<bedrooms <<'\n';
        cout<<" So phong tam : " <<bathrooms<<'\n';
        }
    };
class office : public Building
    {   int phones; //tong so may dien thoai
      int extis; //tong so binh cuu hoa
      public :
       office(int f, int r, int ft, int p, int ext)
         { floors=f; rooms=r; footage=ft;
           phones=p; extis=ext;
         }
      void show()
        { cout<<'\n';
          cout<<" So tang  : " <<floors <<'\n';
          cout<<" So phong : " <<rooms <<'\n';
        cout<<" So tong dien tich : " <<footage
            <<"\n”;
        cout<<" So may dien thoai : " <<phones
            << "\n";
        cout<<" So binh cuu hoa : " <<extis<<'\n';
        }
    };
void main()
{
 house h_ob(2,12,5000,6,4);
 office o_ob(4,25,12000,30,8);
 cout<<"House  : ";
 h_ob.show();
 cout<<"Office : ";
 o_ob.show();
 getch();
```

```
}
```

Chương trình cho kết quả như sau:

```
House  :
 So tang  : 2
 So phong : 12
 So tong dien tich : 5000
 So phong ngu : 6
 So phong tam : 4
Office :
 So tang  : 4
 So phong : 25
 So tong dien tich : 12000
 So may dien thoai : 30
 So binh cuu hoa : 8
```

Bài 49 : 5.4. Hàm ảo

49.15.4.1 Đặt vấn đề

Trước khi đưa ra khái niệm về hàm ảo, ta hãy thảo luận ví dụ sau:

Giả sử có 3 lớp A, B và C được xây dựng như sau:

```
class A
                {
            public:
   void xuat()
    {
      cout <<"\n Lop A";
    }
     };
class B : public A
     {
        public:
    void xuat()
    {
        cout <<"\n Lop B";
       }
```

```
    };
class C : public B
    {
       public:
    void xuat()
      {
       cout <<"\n Lop C";
      }
    };
```

Cả 3 lớp này đều có hàm thành phần là xuat(). Lớp C có hai lớp cơ sở là A, B và C kế thừa các hàm thành phần của A và B. Do đó một đối tượng của C sẽ có 3 hàm xuat(). Xem các câu lệnh sau:

```
C  ob; //  ob là đối tượng kiểu C
ob.xuat();      // Gọi tới hàm thành phần xuat() của lớp D
ob.B::xuat() ;  // Gọi tới hàm thành phần xuat() của lớp B
ob.A::xuat() ;  // Gọi tới hàm thành phần xuat() của lớp A
```

Các lời gọi hàm thành phần trong ví dụ trên đều xuất phát từ đối tượng ob và mọi lời gọi đều *xác định rõ* hàm cần gọi.

Ta xét tiếp tình huống các lời gọi không phải từ một biến đối tượng mà từ một con trỏ đối tượng. Xét các câu lệnh:

```
A   *p,  *q, *r;   // p,q,r là các con trỏ kiểu A
A  a;       // a là đối tượng kiểu A
B  b;       // b là đối tượng kiểu B
C  c;       // c là đối tượng kiểu C
```

Bởi vì con trỏ của lớp cơ sở có thể dùng để chứa địa chỉ các đối tượng của lớp dẫn xuất, nên cả 3 phép gán sau đều hợp lệ:

```
p = &a;    q = &b;    r = &c;
```

Ta xét các lời gọi hàm thành phần từ các con trỏ p, q, r:

```
p->xuat();    q->xuat();    r->xuat();
```

Cả 3 câu lệnh trên đều gọi tới hàm thành phần xuat() của lớp A, bởi vì các con trỏ p, q, r đều có kiểu lớp A. Sở dĩ như vậy là vì một lời gọi (xuất phát từ một đối tượng hay con trỏ) tới hàm thành phần **luôn luôn liên kết với một hàm thành phần cố định** và sự liên kết này xác định trong quá trình biên dịch chương trình. Ta bảo đây là **sự liên kết tĩnh**.

Có thể tóm lược cách thức gọi các hàm thành phần như sau:

1. Nếu lời gọi xuất phát từ một đối tượng của lớp nào đó, thì hàm thành phần của lớp đó sẽ được gọi.

2. Nếu lời gọi xuất phát từ một con trỏ kiểu lớp, thì hàm thành phần của lớp đó sẽ được gọi bất kể con trỏ chứa địa chỉ của đối tượng nào.

Vấn đề đặt ra là: Ta muốn tại thời điểm con trỏ đang trỏ đến đối tượng nào đó thì lời gọi hàm phải liên kết đúng hàm thành phần của lớp mà đối tượng đó thuộc vào chứ không phụ thuộc vào kiểu lớp của con trỏ. C++ giải quyết vấn đề này bằng cách dùng khái niệm *hàm ảo*.

49.25.4.2. Định nghĩa hàm ảo

Hàm ảo là hàm thành phần của lớp, nó được *khai báo trong lớp cơ sở* và định nghĩa lại trong lớp dẫn xuất. Để định nghĩa hàm ảo thì phần khai báo hàm phải bắt đầu bằng từ khóa *virtual*. Khi một lớp có chứa hàm ảo được kế thừa, lớp dẫn xuất sẽ định nghĩa lại hàm ảo đó cho chính mình. Các hàm ảo triển khai tư tưởng chủ đạo của tính đa hình là " một giao diện cho nhiều hàm thành phần". Hàm ảo bên trong lớp cơ sở định nghĩa hình thức giao tiếp đối với hàm đó. Việc định nghĩa lại hàm ảo ở lớp dẫn xuất là thi hành các tác vụ của hàm liên quan đến chính lớp dẫn xuất đó. Nói cách khác, định nghĩa lại hàm ảo chính là tạo ra phương thức cụ thể. Trong phần định nghĩa lại hàm ảo ở lớp dẫn xuất, không cần phải sử dụng lại từ khóa virtual.

Khi xây dựng hàm ảo, cần tuân theo những quy tắc sau :

1. Hàm ảo phải là hàm thành phần của một lớp ;
2. Những thành phần tĩnh (static) không thể khai báo ảo;
3. Sử dụng con trỏ để truy nhập tới hàm ảo;
4. Hàm ảo được định nghĩa trong lớp cơ sở, ngay khi nó không được sử dụng;
5. Mẫu của các phiên bản (ở lớp cơ sở và lớp dẫn xuất) phải giống nhau. Nếu hai hàm cùng tên nhưng có mẫu khác nhau thì C++ sẽ xem như hàm tải bội;
6. Không được tạo ra hàm tạo ảo, nhưng có thể tạo ra hàm hủy ảo;
7. Con trỏ của lớp cơ sở có thể chứa địa chỉ của đối tượng thuộc lớp dẫn xuất, nhưng ngược lại thì không được;
8. Nếu dùng con trỏ của lớp cơ sở để trỏ đến đối tượng của lớp dẫn xuất thì phép toán tăng giảm con trỏ sẽ không tác dụng đối với lớp dẫn xuất, nghĩa là không phải con trỏ sẽ trỏ tới đối tượng trước hoặc tiếp theo trong lớp dẫn xuất. Phép toán tăng giảm chỉ liên quan đến lớp cơ sở.

Ví dụ:

```
class A
```

```
  {
      ...
    virtual void hienthi()
        {
        cout<<"\nDay la lop A";
        };
  };
class B : public A
  {
      ...
    void hienthi()
        {
        cout<<"\nDay la lop B";
        }
  };
class C : public B
  {
      ...
    void hienthi()
        {
       cout<<"\nDay la lop C";
        }
  };
class D : public A
  { ...
    void hienthi()
        {
        cout<<"\nDay la lop D";
        }
  };
```

Chú ý: Từ khoá virtual không được đặt bên ngoài định nghĩa lớp. Xem ví dụ :

```
class A
  {
    ...
    virtual void hienthi();
  };
```

```
virtual void hienthi()  // sai
  {
     cout<<”\nDay la lop A”;
   }
```

49.35.4.3. Quy tắc gọi hàm ảo

Hàm ảo chỉ khác hàm thành phần thông thường khi được gọi từ một con trỏ. Lời gọi tới hàm ảo từ một con trỏ chưa cho biết rõ hàm thành phần nào (trong số các hàm thành phần cùng tên của các lớp có quan hệ thừa kế) sẽ được gọi. Điều này sẽ *phụ thuộc vào đối tượng cụ thể* mà con trỏ đang trỏ tới: *con trỏ đang trỏ tới đối tượng của lớp nào thì hàm thành phần của lớp đó sẽ được gọi.*

49.45.4.5. Quy tắc gán địa chỉ đối tượng cho con trỏ lớp cơ sở

C++ cho phép gán địa chỉ đối tượng của một lớp dẫn xuất cho con trỏ của lớp cơ sở bằng cách sử dụng phép gán = và phép toán lấy địa chỉ &.

Ví dụ : Giả sử A là lớp cơ sở và B là lớp dẫn xuất từ A. Các phép gán sau là đúng:

```
A  *p;  // p là con trỏ kiểu A
A   a;  // a là biến đối tượng kiểu A
B   b;  // b là biến đối tượng kiểu B
p = &a;  // p và a cùng lớp A
p = &b;  // p là con trỏ lớp cơ sở, b là đối tượng lớp dẫn xuất
```

Chú ý: Không cho phép gán địa chỉ đối tượng của lớp cơ sở cho con trỏ của lớp dẫn xuất, chẳng hạn với khai báo B *q; A a; thì câu lệnh q = &a; là sai.

Ví dụ 5.11 Chương trình sau đây minh họa việc sử dụng hàm ảo:

```
#include <iostream.h>
#include <conio.h>
class A
 {
     public:
   virtual void hienthi()
    {
       cout <<"\n Lop A";
     }
 };
class B : public A
 {
     public:
```

```
    void hienthi()
     {
        cout <<"\n Lop B";
       }
  };
class C : public B
 {
     public:
    void hienthi()
     {
        cout <<"\n Lop C";
       }
  };
 void main()
 { clrscr();
   A *p;
   A a; B b; C c;
   a.hienthi();    //goi ham cua lop A
   p = &b;    //p tro to doi tuong b cua lop B
   p->hienthi();   //goi ham cua lop B
   p=&c;    //p tro to doi tuong c cua lop C
   p->hienthi();   //goi ham cua lop C
   getch();
  }
```

Chương trình này cho kết quả như sau:

```
 Lop A
 Lop B
 Lop C
```

Chú ý :

- Cùng một câu lệnh p->hienthi(); được tương ứng với nhiều hàm khác nhau khác nhau khi hienthi() là hàm ảo. Đây chính là sự *tương ứng bội*. Khả năng này cho phép xử lý nhiều đối tượng khác nhau theo cùng một cách thức.
- Cũng với lời gọi: p->hienthi(); (hienthi() là hàm ảo) thì lời gọi này không liên kết với một phương thức cố định, mà tùy thuộc và nội dung con trỏ. Đó là *sự liên kết*

động và phương thức được liên kết (được gọi) thay đổi mỗi khi có sự thay đổi nội dung con trỏ trong quá trình chạy chương trình.

Ví dụ 5.12 Chương trình sau tạo ra một lớp cơ sở có tên là ***num*** lưu trữ một số nguyên, và một hàm ảo của lớp có tên là *shownum*(). Lớp ***num*** có hai lớp dẫn xuất là ***outhex*** và ***outoct***. Trong hai lớp này sẽ định nghĩa lại hàm ảo *shownum*() để chúng in ra số nguyên dưới dạng số hệ 16 và số hệ 8.

```
#include <iostream.h>
#include <conio.h>
class num
{
    public :
      int i;
      num(int x) { i=x; }
      virtual void shownum()
     { cout<<"\n So he 10 : ";
       cout <<dec<<i<<'\n';
          }
};
class outhex : public num
{ public :
      outhex(int n) : num(n) {}
      void shownum()
        { cout <<"\n So he 10 : "<<dec<<i<<endl;
       cout <<"\n So he 16 : "<<hex << i <<'\n';
        }
};
class outoct : public num
{ public :
      outoct(int n) :  num(n) {}
      void shownum()
        { cout <<"\n So he 10 : "<<dec<<i<<endl;
        cout <<"\n So he 8  : "<< oct << i <<'\n';
         }
};
void main()
{ clrscr();
```

```
  num n (1234);
  outoct o (342);
  outhex h (747);
  num *p;
  p=&n;
  p->shownum(); //goi ham cua lop co so, 100
  p=&o;
  p->shownum(); //goi ham cua lop dan xuat, 12
  p=&h;
  p->shownum(); //goi ham cua lop dan xuat, f
  getch();
}
```

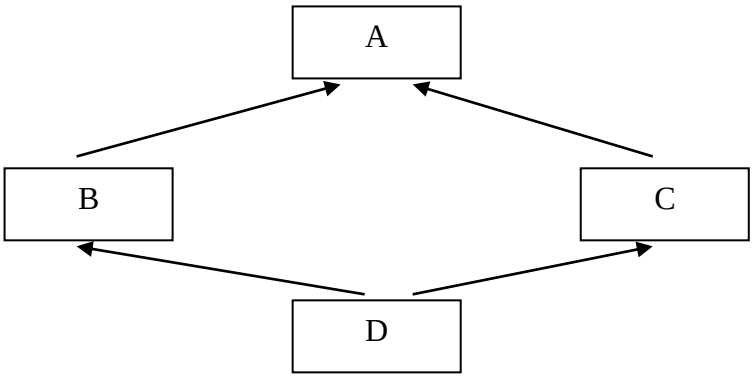
Chương trình trên cho kết quả:

```
So he 10 : 1234
So he 10 : 342
So he 8  : 526
So he 10 : 747
So he 16 : 2eb
```

Bài 50 : 5.5. Lớp cơ sở ảo

50.15.5.1. Khai báo lớp cơ sở ảo

Một vấn đề tồn tại là khi nhiều lớp cơ sở được kế thừa trực tiếp bởi một lớp dẫn xuất. Để hiểu rõ hơn vấn đề này, xét tình huống các lớp kế thừa theo sơ đồ như sau:



Hình 5.4.

ở đây, lớp A được kế thừa bởi hai lớp B và C. Lớp D kế thừa trực tiếp cả hai lớp B và C. Như vậy lớp A được kế thừa *hai lần* bởi lớp D: lần thứ nhất nó được kế thừa thông qua lớp B, và lần thứ hai được kế thừa thông qua lớp C. Bởi vì có hai bản sao của lớp A có trong lớp D nên một tham chiếu đến một thành phần của lớp A sẽ tham chiếu về lớp A được kế thừa gián tiếp thông qua lớp B hay tham chiếu về lớp A được kế thừa gián tiếp thông qua

lớp C? Để giải quyết tính không rõ ràng này, C++ có một cơ chế mà nhờ đó chỉ có một bản sao của lớp A ở trong lớp D: đó là sử dụng lớp *cơ sở ảo.*

Trong ví dụ trên, C++ sử dụng từ khóa vitual để khai báo lớp A là ảo trong các lớp B và C theo mẫu như sau:

clacc B : virtual public A
    {...};
clacc C : virtual public A
    {...};
clacc D : public B, public C
    {...};

Việc chỉ định A là ảo trong các lớp B và C nghĩa là A sẽ chỉ xuất hiện một lần trong lớp D. Khai báo này không ảnh hưởng đến các lớp B và C.

Chú ý: Từ khóa virtual có thể đặt trước hoặc sau từ khóa public, private, protected.

Ví dụ 5.13

```
#include <iostream.h>
#include <conio.h>
class A
 {   float x,y;
     public:
    void set(float x1, float y1)
     {
       x = x1; y = y1;
      }
    float getx()
     { return x;
        }
    float gety()
     { return y;
       }
 };
class B : virtual public A
 { };
class C : virtual public A
 { };
class D : public B, public C
 { };
```

```
void main()
{ clrscr();
  D d;
  cout<<"\nd.B::set(2,3)\n";
  d.B::set(2,3);
  cout<<"\nd.C::getx() = "; cout<<d.C::getx()<<endl;
  cout<<"\nd.B::getx() = "; cout<<d.B::getx()<<endl;
  cout<<"\nd.C::gety() = "; cout<<d.C::gety()<<endl;
  cout<<"\nd.B::gety() = "; cout<<d.B::gety()<<endl;
  cout<<"\nd.C::set(2,3)\n";
  d.C::set(2,3);
  cout<<"\nd.C::getx() = "; cout<<d.C::getx()<<endl;
  cout<<"\nd.B::getx() = "; cout<<d.B::getx()<<endl;
  cout<<"\nd.C::gety() = "; cout<<d.C::gety()<<endl;
  cout<<"\nd.B::gety() = "; cout<<d.B::gety()<<endl;
  getch();
 }
```

Chương trình trên sẽ cho kết quả:

```
d.B::set(2,3)
d.C::getx() = 2
d.B::getx() = 2
d.C::gety() = 3
d.B::gety() = 3

d.C::set(2,3)
d.C::getx() = 2
d.B::getx() = 2
d.C::gety() = 3
d.B::gety() = 3
```

50.25.5.2. Hàm tạo và hàm hủy đối với lớp cơ sở ảo

Ta đã biết, khi khởi tạo đối tượng lớp dẫn xuất thì các hàm tạo được gọi theo thứ tự xuất hiện trong danh sách các lớp cơ sở được khai báo, rồi đến hàm tạo của lớp dẫn xuất. Thông tin được chuyển từ hàm tạo của lớp dẫn xuất sang hàm tạo của lớp cơ sở. Trong tình huống có lớp cơ sở ảo, chẳng hạn như hình vẽ 5.4., cần phải tuân theo quy định sau:

Thứ tự gọi hàm tạo: Hàm tạo của một lớp ảo luôn luôn được gọi trước các hàm tạo khác.

Với sơ đồ kế thừa như hình vẽ 5.4., thứ tự gọi hàm tạo sẽ là A, B, C và cuối cùng là D. Chương trình sau minh họa điều này:

Ví dụ 5.14

```
#include <iostream.h>
#include <conio.h>
class A
 {   float x,y;
     public:
   A() {x = 0; y = 0;}
   A(float x1, float y1)
    {
      cout<<"A::A(float,float)\n";
      x = x1; y = y1;
    }
   float getx()
    {
        return x;
       }
   float gety()
      {
         return y;
        }

   };
class B : virtual public A
 {
     public:
   B(float x1, float y1):A(x1,y1)
     {
       cout<<"B::B(float,float)\n";
      }
  };
class C : virtual public A
  {
     public:
       C(float x1, float y1):A(x1,y1)
```

```
      {
        cout<<"C::C(float,float)\n";
       }
  };
class D : public B, public C
 {
     public:
  D(float x1, float y1):A(x1,y1), B(10,4), C(1,1)
   {
     cout<<"D::D(float,float)\n";
   }
  };
 void main()
 { clrscr();
   D d(2,3);
   cout<<"\nD d (2,3)\n";
   cout<<"\nd.C::getx() = "; cout<<d.C::getx()<<endl;
   cout<<"\nd.B::getx() = "; cout<<d.B::getx()<<endl;
   cout<<"\nd.C::gety() = "; cout<<d.C::gety()<<endl;
   cout<<"\nd.B::gety() = "; cout<<d.B::gety()<<endl;
   cout<<"\nd1 (10,20) \n";

   D d1 (10,20);   cout<<"\nd.C::getx() = ";
cout<<d.C::getx()<<endl;
   cout<<"\nd.B::getx() = "; cout<<d.B::getx()<<endl;
   cout<<"\nd.C::gety() = "; cout<<d.C::gety()<<endl;
   cout<<"\nd.B::gety() = "; cout<<d.B::gety()<<endl;
   getch();
  }
```

Kết quả chương trình trên như sau:

```
A::A(float,float)
B::B(float,float)
C::C(float,float)
D::D(float,float)
D d (2,3)
```

```
d.C::getx() = 2
d.B::getx() = 2
d.C::gety() = 3
d.B::gety() = 3
d1 (10,20)
A::A(float,float)
B::B(float,float)
C::C(float,float)
D::D(float,float)
d.C::getx() = 2
d.B::getx() = 2
d.C::gety() = 3
d.B::gety() = 3
```

Bài tập

1. Xây dựng lớp có tên là Stack với các thao tác cần thiết. Từ đó hãy dẫn xuất từ lớp stack để đổi một số nguyên dương sang hệ đếm bất kỳ.

2. Viết một phân cấp kế thừa cho các lớp hình tứ giác, hình thang, hình bình hành, hình chữ nhật, hình vuông.

3. Tạo một lớp cơ sở có tên là airship để lưu thông tin về số lượng hành khách tối đa và trọng lượng hàng hóa tối đa mà máy bay có thể chở được. Từ đó hãy tạo hai lớp dẫn xuất airplane và balloon, lớp airplane lưu thông tin về kiểu động cơ (gồm động cơ cánh quạt và động cơ phản lực), lớp balloon lưu thông tin về loại nhiên liệu sử dụng cho khí cầu (gồm hai loại là hydrogen và helium). Hãy viết chương trình minh họa.

4. Một nhà xuất bản nhận xuất bản sách. Sách có hai loại: loại có hình ảnh ở trang bìa và loại không có hình ảnh ở trang bìa. Loại có hình ảnh ở trang bìa thì phải thuê họa sĩ vẽ bìa. Viết chương trình thực hiện các yêu cầu :

- Tạo một lớp cơ sở có tên là SACH để lưu thông tin về tên sách, tác giả, số trang, giá bán và định nghĩa hàm thành phần cho phép nhập dữ liệu cho các đối tượng của lớp SACH.
- Tạo lớp BIA kế thừa từ lớp SACH để lưu các thông tin : Mã hình ảnh, tiền vẽ và định nghĩa hàm thành phần cho phép nhập dữ liệu cho các đối tượng của lớp BIA.
- Tạo lớp HOASY để lưu các thông tin họ tên, địa chỉ của họa sỹ và định nghĩa hàm thành phần cho phép nhập dữ liệu cho các đối tượng của lớp HOASY.
- Tạo lớp SACHVEBIA kế thừa từ lớp BIA và lớp HOASY và định nghĩa hàm thành phần cho phép nhập dữ liệu cho các đối tượng của lớp SACHVEBIA. Viết hàm main() cho phép nhập vào hai danh sách : danh sách các sách có vẽ bìa và danh sách các sách không có vẽ bìa (có thể dùng mảng tĩnh hoặc mảng con trỏ).

5. Xây dựng lớp hình vuông có tên là HVUONG với các dữ liệu thành phần như sau: độ dài cạnh. Các hàm thành phần để nhập dữ liệu, hiển thị dữ liệu, tính diện tích, chu vi hình vuông. Từ lớp HVUONG, xây dựng lớp dẫn xuất có tên là CHUNHAT, là lớp kế thừa của lớp HVUONG và được bổ sung thêm thuộc tính sau: độ dài cạnh thứ hai. Các hàm thành phần để nhập dữ liệu, hiển thị dữ liệu để tính diện tích và chu vi hình chữ nhật. Viết chương trình minh họa.

6. Xây dựng lớp cơ sở CANBO có dữ liệu thành phần là mã cán bộ, mã đơn vị, họ tên, ngày sinh. Các hàm thành phần bao gồm: nhập dữ liệu cán bộ, hiển thị dữ liệu. Lớp dẫn xuất LUONG kế thừa lớp CANBO và có thêm các thuộc tính: phụ cấp, hệ số lương, bảo hiểm. Hàm thành phần để tính lương cán bộ theo công thức:

lương = hệ số lương *290000 + phụ cấp - bảo hiểm

Hãy thiết kế chương trình để đáp ứng các yêu cầu:

- Nhập danh sách cán bộ
- In bảng lương các cán bộ theo từng đơn vị.

7. Nhân viên trong một cơ quan được lĩnh lương theo các dạng khác nhau. Dạng người lao động hưởng lương từ ngân sách Nhà nước gọi là cán bộ, công chức (dạng biên chế). Dạng người lao động lĩnh lương từ ngân sách của cơ quan gọi là người làm hợp đồng. Như vậy hệ thống có hai đối tượng: biên chế và hợp đồng.

- Hai loại đối tượng này có đặc tính chung là viên chức làm việc cho cơ quan. Từ đây có thể tạo nên lớp cơ sở để quản lý một viên chức (lớp **Nguoi**) bao gồm mã số, họ tên, lương.
- Hai lớp kế thừa từ lớp cơ sở trên:

+ Lớp Bienche gồm các thuộc tính: hệ số lương, tiền phụ cấp chức vụ.

+ Lớp Hopdong gồm các thuộc tính: tiền công lao động, số ngày làm việc trong tháng, hệ số vượt giờ.

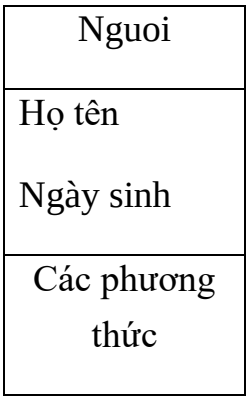
Hãy thiết kế các lớp trên và viết chương trình minh họa.

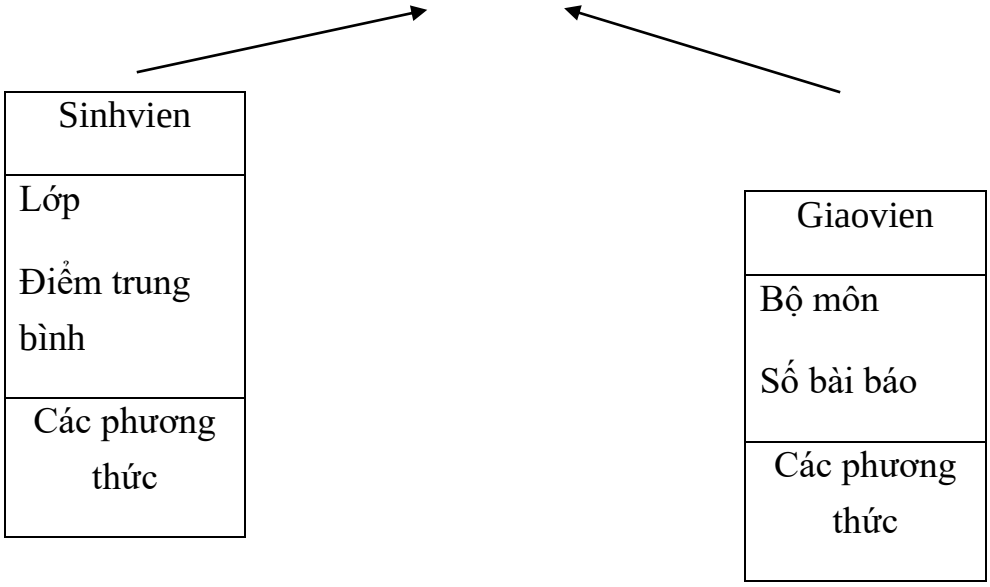
8. Viết chương trình quản lý sách báo, tạp chí của thư viện trong trường đại học, hằng tháng gởi về khoa họ tên của các giáo viên và sinh viên đã quá thời hạn mượn sách.

9. Viết chương trình tính và in bảng lương hàng tháng của giáo viên và người làm hợp đồng trong một trường đại học. Giả sử việc tính tóan tiền lương được căn cứ vào các yếu tố sau:

- Đối với giáo viên: số tiết dạy trong tháng, tiền dạy một tiết.
- Đối với người làm hợp đồng: tiền công ngày, số ngày làm việc

10. Giả sử cuối năm học cần trao phần thưởng cho các sinh viên giỏi, các giáo viên có tham gia nghiên cứu khoa học. Điều kiện khen thưởng của sinh viên là có điểm trung bình lớn hơn 8. Điều kiện khen thưởng của giáo viên là có ít nhất một bài báo nghiên cứu khoa học. Sơ đồ cấu trúc phân cấp lớp như sau:

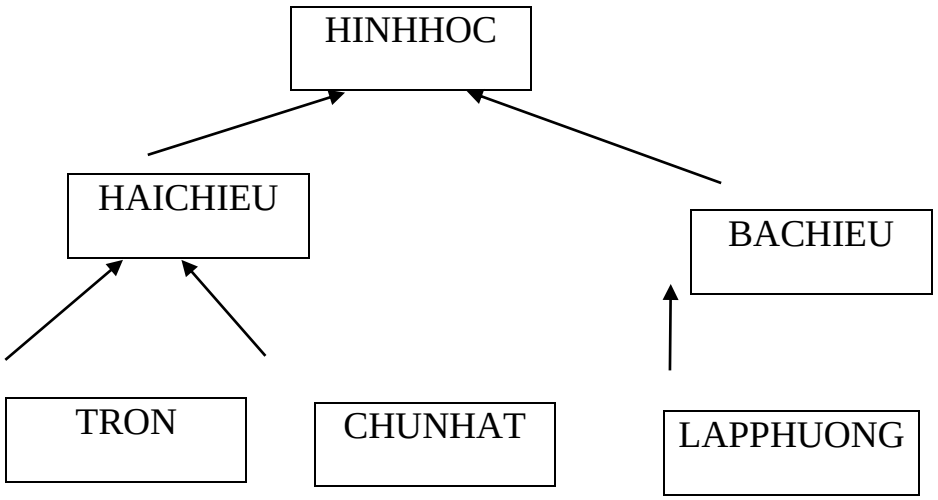




Yêu cầu:

- Xây dựng các lớp theo sơ đồ kề thừa ở trên, mỗi lớp có các hàm thành phần để nhập, xuất dữ liệu, hàm kiểm tra khen thưởng.
- Hãy viết hàm main() cho phép nhập vào dữ liệu của không quá 100 sinh viên và không quá 30 giáo viên. In ra danh sách sinh viên và giáo viên được khen thưởng.

11. Giả sử ta có sơ đồ kế thừa của các lớp như sau:



Yêu cầu:

- Thiết kế các lớp để có thể in ra các thông tin của các hình (tròn, chữ nhật, lập phương) bao gồm: diện tích, chu vi, thể tích.

- Viết chương trình minh họa.

12. Viết chương trình quản lý việc mượn và trả sách ở một thư viện theo phương pháp lập trình hướng đối tượng. Chương trình cho phép:

- Đăng ký bạn đọc mới với thông tin là mã và tên bạn đọc, số điện thoại
- Nhập sách mới với thông tin là mã sách, tên sách, số lượng, nhà xuất bản.
- Mượn và trả sách.
- Thống kê bạn đọc.
- Thống kê sách.

Chương 6

Khuôn hình

Bài 51 : 6.1. Khuôn hình hàm

51.16.1.1. Khái niệm

Ta đã biết hàm quá tải cho phép dùng một tên duy nhất cho nhiều hàm để thực hiện các công việc khác nhau. Khái niệm khuôn hình hàm cũng cho phép sử dụng cùng một tên duy nhất để thực hiện các công việc khác nhau, tuy nhiên so với định nghĩa hàm quá tải, nó có phần mạnh hơn và chặt chẽ hơn. Mạnh hơn vì chỉ cần viết định nghĩa khuôn hình hàm một lần, rồi sau đó chương trình biên dịch làm cho nó thích ứng với các kiểu dữ liệu khác nhau. Chặt chẽ hơn bởi vì dựa theo khuôn hình hàm, tất cả các hàm thể hiện được sinh ra bởi chương trình dịch sẽ tương ứng với cùng một định nghĩa và như vậy sẽ có cùng một giải thuật.

51.26.1.2. Tạo một khuôn hình hàm

Giả thiết rằng chúng ta cần viết một hàm ***min*** đưa ra giá trị nhỏ nhất trong hai giá trị có cùng kiểu. Ta có thể viết một định nghĩa như thế với kiểu **int** như sau:

```
int min (int a, int b)
{
 if (a<b)
          return a;
 else
          return b;
 }
```

Nếu ta muốn sử dụng hàm min cho kiểu double, float, char,.... ta lại phải viết lại định nghĩa hàm **min**, ví dụ:

```
float min  (float a, float b)
{
    if  (a < b)
        return a;
    else
        return b;
 }
```

Như vậy ta phải viết rất nhiều định nghĩa hàm hoàn toàn tương tự nhau, chỉ có kiểu dữ liệu là thay đổi. Chương trình dịch C++ cho phép giải quyết đơn giản vấn đề trên bằng cách định nghĩa một khuôn hình hàm duy nhất theo cú pháp:

template <danh sách tham số kiểu> <kiểu trả về> tên hàm(khai báo tham số)

{

```
    // định nghĩa hàm
}
```

trong đó <danh sách tham số kiểu> là các kiểu dữ liệu được khai báo với từ khoá *class*, cách nhau bởi dấu phẩy. Kiểu dữ liệu là một kiểu bất kỳ, kể cả kiểu *class*.

Ví dụ 6.1 Xây dựng khuôn hình cho hàm tìm giá trị nhỏ nhất của hai số:

```
template <class Kieuso> Kieuso min(Kieuso a, Kieuso b)
{
 if  (a<b)
        return a;
 else
       return b;
}
```

51.36.1.3. Sử dụng khuôn hình hàm

Để sử dụng khuôn hình hàm ***min*** vừa tạo ra, chỉ cần sử dụng hàm *min* trong những điều kiện phù hợp, trong trường hợp này là hai tham số của hàm phải cùng kiểu dữ liệu. Như vậy, nếu trong một chương trình có hai tham số nguyên n và m (kiểu int) với lời gọi min(n,m) thì chương trình dịch tự động sản sinh ra hàm min(), gọi là một hàm thể hiện, tương ứng với hai tham số kiểu int. Nếu chúng ta gọi min() với hai tham số kiểu float, chương trình biên dịch cũng tự động sản sinh một hàm thể hiện min khác tương ứng với các tham số kiểu float và cứ thế với các kiểu dữ liệu khác.

Chú ý:

- Các biến truyền cho danh sách tham số của hàm phải chính xác với kiểu khai báo.

- Muốn áp dụng được với kiểu lớp thì trong lớp phải định nghĩa các toán tử tải bội tương ứng.

51.46.1.4. Các tham số kiểu của khuôn hình hàm

Khuôn hình hàm có thể có một hay nhiều tham số kiểu, mỗi tham số đi liền sau từ khoá class. Các tham số này có thể ở bất kỳ đâu trong định nghĩa của khuôn hình hàm, nghĩa là :

- Trong dòng tiêu đề (ở dòng đầu khai báo template).

- Trong các khai báo biến cục bộ.

- Trong các chỉ thị thực hiện.

Trong mọi trường hợp, mỗi tham số kiểu phải xuất hiện ít nhất một lần trong khai báo danh sách tham số hình thức của khuôn hình hàm. Điều đó hoàn toàn logic, bởi vì nhờ các tham số này, chương trình dịch mới có thể sản sinh ra hàm thể hiện cần thiết.

ở khuôn hình hàm min trên, mới chỉ cho phép tìm min của hai số cùng kiểu, nếu muốn tìm min hai số khác kiểu thì khuôn hình hàm trên chưa đáp ứng được. Ví dụ sau sẽ khắc phục được điều này.

Ví dụ 6.2

```
#include <iostream.h>
template <class kieuso1,class kieuso2> kieuso1
         min(kieuso1 a,kieuso2 b)
   {
     return a<b ? a : b;
   }
   void main(){
       float a =2.5;
       int b = 8;
       cout << "so nho nhat la :" << min(a,b);
   }
```

Ví dụ 6.3 Giả sử trong lớp SO các số int đã xây dựng, ta có xây dựng các toán tử tải bội < , << cho các đối tượng của class SO (xem chương 4). Nội dung file ttclsso.h như sau:

```
class SO
{
  private:
     int giatri;
  public:
     SO(int x=0)
     {
       giatri = x;
     }
     SO (SO &tso)
     {
       giatri = tso.giatri;
     }
     SO (){}; //Giong nhu ham thiet lap ngam dinh
     ~SO()
     { }
```

```
    int operator<(SO & s)
    {
       return (giatri <s.giatri);
      }

    friend istream& operator>>(istream&,SO&);
    friend ostream& operator<<(ostream&,SO&);
};
```

Chương trình sau đây cho phép thử hàm min trên hai đối tượng kiểu class:

Ví dụ 6.4 Chương trình sau đây cho phép thử hàm min trên hai đối tượng kiểu class:

```
#include <iostream.h>
#include <ttclsso.h>
template <class kieuso1,class kieuso2> kieuso1
          min(kieuso1 a,kieuso2 b)
 {
  if (a<b)
     return a;
  else
     return b;
 }
void main(){
  float a =2.5;
  int b = 8;
  cout << "so nho nhat la :" << min(a,b)<<endl;
  SO so1(15),so2(20);
  cout << "so nho nhat la :" << min(so2,so1);
}
```

51.56.1.5. Định nghĩa chồng các khuôn hình hàm

Tương tự việc định nghĩa các hàm quá tải, C++ cho phép định nghĩa chồng các khuôn hình hàm, tức là có thể định nghĩa một hay nhiều khuôn hình hàm có cùng tên nhưng với các tham số khác nhau. Điều đó sẽ tạo ra nhiều họ các hàm (mỗi khuôn hình hàm tương ứng với họ các hàm).

Ví dụ có ba họ hàm min :

- Một họ gồm các hàm tìm giá trị nhỏ nhất trong hai giá trị

- Một họ gồm các hàm tìm giá trị nhỏ nhất trong ba giá trị
- Một họ gồm các hàm tìm giá trị nhỏ nhất trong một mảng giá trị.

Một cách tổng quát, ta có thể định nghĩa một hay nhiều khuôn hình cùng tên, mỗi khuôn hình có các tham số kiểu cũng như là các tham số biểu thức riêng. Hơn nữa, có thể cung cấp các hàm thông thường với cùng tên với cùng một khuôn hình hàm, trong trường hợp này ta nói đó là sự cụ thể hoá một hàm thể hiện.

Trong trường hợp tổng quát khi có đồng thời cả hàm quá tải và khuôn hình hàm, chương trình dịch lựa chọn hàm tương ứng với một lời gọi hàm dựa trên các nguyên tắc sau:

Đầu tiên, kiểm tra tất cả các hàm thông thường cùng tên và chú ý đến sự tương ứng chính xác; nếu chỉ có một hàm phù hợp, hàm đó được chọn; Còn nếu có nhiều hàm cùng thỏa mãn sẽ tạo ra một lỗi biên dịch và quá trình tìm kiếm bị gián đoạn.

Nếu không có hàm thông thường nào tương ứng chính xác với lời gọi, khi đó ta kiểm tra tất cả các khuôn hình hàm có trùng tên với lời gọi, khi đó ta kiểm tra tất cả các khuôn hình hàm có trùng tên với lời gọi; nếu chỉ có một tương ứng chính xác được tìm thấy, hàm thể hiện tương ứng được sản sinh và vấn đề được giải quyết; còn nếu có nhiều hơn một khuôn hình hàm điều đó sẽ gây ra lỗi biên dịch và quá trình dừng.

Cuối cùng, nếu không có khuôn hình hàm phù hợp, ta kiểm tra một lần nữa tất cả các hàm thông thường cùng tên với lời gọi. Trong trường hợp này chúng ta phải tìm kiếm sự tương ứng dựa vào cả các chuyển kiểu cho phép trong C/C++.

Bài 52 : 6.2. Khuôn hình lớp

52.16.2.1. Khái niệm

Bên cạnh khái niệm khuôn hình hàm, C++ còn cho phép định nghĩa khuôn hình lớp. Cũng giống như khuôn hình hàm, ở đây ta chỉ cần viết định nghĩa các khuôn hình lớp một lần rồi sau đó có thể áp dụng chúng với các kiểu dữ liệu khác nhau để được các lớp thể hiện khác nhau.

52.26.2.2. Tạo một khuôn hình lớp

Trong chương trước ta đã định nghĩa cho lớp SO, giá trị các số là kiểu **int**. Nếu ta muốn làm việc với các số kiểi **float**, **double**,... thì ta phải định nghĩa lại một lớp khác tương tự, trong đó kiểu dữ liệu **int** cho dữ liệu **giatri** sẽ được thay bằng **float,double,**...

Để tránh sự trùng lặp trong các tình huống như trên, chương trình dịch C++ cho phép định nghĩa một khuôn hình lớp và sau đó, áp dụng khuôn hình lớp này với các kiểu dữ liệu khác nhau để thu được các lớp thể hiện như mong muốn. Ví dụ :

```
template <class kieuso>  class SO
{
 kieuso giatri;
```

```
    public :
   SO (kieuso x =0);
   void Hienthi();
  ...
};
```

Cũng giống như các khuôn hình hàm, `template <class kieuso>` xác định rằng đó là một khuôn hình trong đó có một tham số kiểu `kieuso` . C++ sử dụng từ khoá **class** chỉ để nói rằng `kieuso` đại diện cho một kiểu dữ liệu nào đó.

Việc định nghĩa các hàm thành phần của khuôn hình lớp, người ta phân biệt hai trường hợp:

Khi hàm thành phần được định nghĩa bên trong định nghĩa lớp thì không có gì thay đổi.

Khi hàm thành phần được định nghĩa bên ngoài lớp, khi đó cần phải nhắc lại cho chương trình biết các tham số kiểu của khuôn hình lớp, có nghĩa là phải nhắc lại `template <class kieuso>` chẳng hạn, trước định nghĩa hàm. Ví dụ hàm Hienthi() được định nghĩa ngoài lớp:

```
template <class kieuso> void SO<kieuso>::Hienthi()
 {
   cout <<giatri;
  }
```

52.36.2.3. Sử dụng khuôn hình lớp

Sau khi một khuôn hình lớp đã được định nghĩa, nó sẽ được dùng để khai báo các đối tượng theo dạng sau :

Tên_lớp <Kiểu> Tên_đối_tượng;

Ví dụ câu lệnh khai báo SO <int> so1; sẽ khai báo một đối tượng so1 có thành phần dữ liệu giatri có kiểu nguyên int.

SO <int> có vai trò như một kiểu dữ liệu lớp; người ta gọi nó là một lớp thể hiện của khuôn hình lớp SO. Một cách tổng quát, khi áp dụng một kiểu dữ liệu nào đó với khuôn hình lớp SO ta sẽ có được một lớp thể hiện tương ứng với kiểu dữ liệu.

Tương tự với các khai báo SO <float> so2; cho phép khai báo một đối tượng so2 mà thành phần dữ liệu giatri có kiểu float.

Ví dụ 6.5

```
#include <iostream.h>
```

```
#include <conio.h>
template <class kieuso>  class SO
 {
  kieuso giatri;
     public :
  SO (kieuso x =0);
  void Hienthi(){
    cout<<"Gia tri cua so :"<<giatri<<endl;
    }
};
void main(){
  clrscr();
  SO <int> soint(10); soint.Hienthi();
  SO <float> sofl(25.4); sofl.Hienthi();
  getch();
}
```

Kết quả trên màn hình là:

```
Gia tri cua so : 10
Gia tri cua so : 25.4
```

52.46.2.4. Các tham số trong khuôn hình lớp

Hoàn toàn giống như khuôn hình hàm, các khuôn hình lớp có thể có các tham số kiểu và tham số biểu thức.

Ví dụ một lớp mà các thành phần có các kiểu dữ liệu khác nhau được khai báo theo dạng:

```
   template <class T, class U,.... class Z>
   class <ten lop>{
     T x;
     U y;
     .....
     Z fct1 (int);
       .....
```

```
};
```

Một lớp thể hiện được khai báo bằng cách liệt kê đằng sau tên khuôn hình lớp các tham số thực, là tên kiểu dữ liệu, với số lượng bằng các tham số trong danh sách của khuôn hình lớp (template<...>)

52.56.2.5. Tóm tắt

Khuôn hình lớp/hàm là phương tiện mô tả ý nghĩa của một lớp/hàm tổng quát, còn lớp/hàm thể hiện là một bản sao của khuôn hình tổng quát với các kiểu dữ liệu cụ thể.

Các khuôn hình lớp/hàm thường được tham số hoá. Tuy nhiên vẫn có thể sử dụng các kiểu dữ liệu cụ thể trong các khuôn hình lớp/hàm nếu cần.

Bài tập

1. Viết khuôn hình hàm để tìm số lớn nhất của hai số bất kỳ
2. Viết khuôn hình hàm để trả về giá trị trung bình của một mảng, các tham số hình thức của hàm này là tên mảng, kích thước mảng.
3. Cài đặt hàng đợi templete.
4. Viết khuôn hình hàm để sắp xếp kiểu dữ liệu bất kỳ.
5. Xây dựng khuôn hình lớp cho các tọa độ điểm trong mặt phẳng, các thành phần dữ liệu của lớp là toadox, toadoy.
6. Xây dựng khuôn hình lớp cho vector để quản lý các vector có thành phần có kiểu tùy ý.

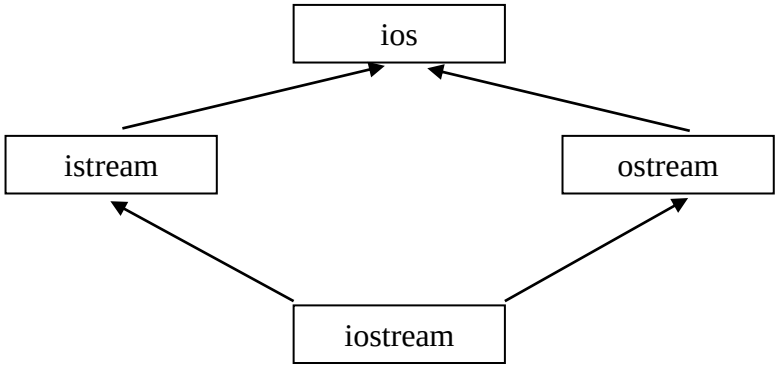
Bài 53 : Phụ lục

Bài 54 : Các dòng xuất nhập

C++ sử dụng khái niệm dòng (stream) và đưa ra các lớp dòng để tổ chức việc nhập xuất. Dòng có thể xem như một dãy tuần tự các byte. Thao tác nhập là đọc các byte từ dòng (gọi là dòng nhập – input) vào bộ nhớ. Thao tác xuất là đưa các byte từ bộ nhớ ra dòng (gọi là dòng xuất- output). Các thao tác này là độc lập thiết bị. để thực hiện việc nhập, xuất lên một thiết bị cụ thể, chúng ta chỉ cần gắn dòng tin với thiết bị này.

Bài 55 : 1.1. Các lớp stream

Có 4 lớp stream quan trọng là: lớp cơ sở ios, từ lớp ios dẫn xuất đến hai lớp istream và iostream. Hai lớp istream và ostream lại dẫn xuất với lớp iostream. Sơ đồ kế thừa giữa các lớp như sau;



- Lớp **ios:** định nghĩa các thuộc tính được sử dụng làm các cờ định dạng cho việc xuất nhập và các cờ kiểm tra lỗi, các phương thức của lớp ios phục vụ việc định dạng dữ liệu nhập xuất, kiểm tra lỗi.
- Lớp **iostream:** cung cấp toán tử nhập >> và nhiều phương thức nhập khác, chẳng hạn các phương thức: get, getline, read,ignore, peek, seekg, tellg, ...
- Lớp **ostream**: cung cấp toán tử xuất << và nhiều phương thức xuất khác, chẳng hạn các phương thức: put, write, flush, seekp, tellp, ...
- Lớp **iostream:** thừa kế các phương thức nhập của các lớp istream và ostream.

Bài 56 : 1.2. Dòng cin và toán tử nhập >>

56.11.2.1 Dòng cin

cin là đối tượng của lớp **istream**. Đó là dòng nhập và được nói là "bị ràng buộc tới" hoặc kết nối tới thiết bị nhập chuẩn, thông thường là bàn phím. Các thao tác nhập trên dòng **cin** đồng nghĩa với nhập dữ liệu từ bàn phím.

56.21.2.2. Toán tử trích >>

Toán tử trích >> được sử dụng như sau để đọc dữ liệu từ dòng **cin**:

```
cin>> biến;
```

Để nhập giá trị của nhiều biến trên một dòng lệnh ta dùng cú pháp sau:

cin>>biến 1>>biến 2>>...>>biến n;

Bài 57 : 1.3. Nhập ký tự và chuỗi ký tự

Có thể dùng các phương thức sau (định nghĩa trong lớp istream) để nhập ký tự và chuỗi: cin.get cin.getline cin.ignore

57.1 1.3.1. Phương thức get() có 3 dạng:

Dạng 1: int cin.get();

Dùng để đọc một ký tự (kể cả khoảng trắng).

Dạng 2: istream& cin.get(char &ch);

Dùng để đọc một ký tự (kể cả khoảng trắng) và đặt vào một biến kiểu char được tham chiếu bởi ch.

Dạng 3: istream& cin.get(char *str, int n, char d = '\n');

Dùng để đọc một dãy ký tự (kể cả khoảng trắng) và đưa vào vùng nhớ do str trỏ tới. Quá trình đọc kết thúc khi xảy ra một trong hai tình huống sau:

+ Gặp ký tự giới hạn (cho trong d). Ký tự giới hạn mặc định là '\n'.
+ Đã nhận đủ (n-1) ký tự.

Chú ý:

+ Ký tự kết thúc chuỗi '\0' được bổ sung vào cuối chuỗi nhận được.
+ Ký tự giới hạn vẫn còn lại trên dòng nhập để dành cho các lệnh nhận tiếp theo.
+ Ký tự <Enter> còn lại trên dòng nhập có thể làm trôi phương thức get() dạng 3. **Ví dụ**: Xét đoạn chương trình:

```
char hoten[25], diachi[50], quequan[30] ;
 cout<<"\nHọ tên:";
 cin.get(ht,25);
 cout<<"\nĐịa chỉ : ";
 cin.get(diachi,50);
 cout<<"\nQuê quán : ";
 cin.get(quequan,30);
 cout <<"\n" <<hoten<<" "<<diachi<<" "<<quequan;
```

Đoạn chương trình dùng để nhập họ tên, địa chỉ và quê quán. Nếu gõ vào Nguyen van X <Enter> thì câu lệnh get đầu tiên sẽ nhận được chuỗi "Nguyen van X" cất vào mảng hoten. Ký tự <Enter> còn lại sẽ làm trôi 2 câu lệnh get tiếp theo. Do đó câu lệnh cuối cùng sẽ chỉ in ra Nguyen van X.

Để khắc phục tình trạng trên, có thể dùng một trong các cách sau:

+ Dùng phương thức get() dạng 1 hoặc dạng 2 để lấy ra ký tự <Enter> trên dòng nhập trước khi dùng get (dạng 3).

+ Dùng phương thức ignore để lấy ra một số ký tự không cần thiết trên dòng nhập trước khi dùng get dạng 3. Phương thức này viết như sau:

cin.ignore(n) ; // Lấy ra (loại ra hay loại bỏ) n ký tự trên dòng nhập.

Như vậy để có thể nhập được cả quê quán và cơ quan, cần sửa lại đoạn chương trình trên như sau:

```
char hoten[25], diachi[50], quequan[30] ;
 cout<<"\nHọ tên : ";
 cin.get(hoten,25);
 cin.get(); // Nhấn < Enter>
 cout<<"\nĐịa chỉ : ";
 cin.get(diachi,50);
 ignore(1); // Bỏ qua <Enter>
 cout<<"\nQuê quán : ";
 cin.get(quequan,30);
 cout <<"\n" <<hoten<<" "<<diachi<<" "<<quequan;
```

57.21.3.2. Phương thức getline()

Phương thức getline để nhập một dãy ký tự từ bàn phím, được mô tả như sau:

istream& cin.getline(char *str, int n, char d = '\n');

Ví dụ:

```
char hoten[25], diachi[50];
 cout<<"\nHọ tên:";
 cin.getline(hoten,25);
 cout<<"\nĐịa chỉ";
 cin.getline(diachi,50);
 cout <<"\n" <<hoten<<" "<<diachi;
```

57.31.3.3. Phương thức ignore

Phương thức ignore dùng để bỏ qua (loại bỏ) một số ký tự trên dòng nhập. Trong nhiều trường hợp, đây là việc làm cần thiết để không làm ảnh hưởng đến các phép nhập tiếp theo. Phương thức ignore được mô tả như sau:

istream& cin.ignore(int n=1);

Phương thức sẽ bỏ qua (loại bỏ) n ký tự trên dòng nhập.

Bài 58 : 1.4. Dòng cout và toán tử xuất <<

58.11.4.1. Dòng cout

cout là một đối tượng kiểu ostream và được nói là "bị ràng buộc tới"thiết bị chuẩn, thông thường là màn hình. Các thao tác xuất trên dòng cout đồng nghĩa với xuất dữ liệu ra màn hình.

58.21.4.2. Toán tử xuất <<

C++ định nghĩa chồng toán tử dịch trái << để gửi các ký tự sang dòng xuất .

Cách dùng toán tử xuất để xuất dữ liệu từ bộ nhớ ra dòng cout như sau:

cout<<biểu thức;

Trong đó biểu thức biểu thị một giá trị cần xuất ra màn hình. Giá trị sẽ được biến đổi thành một dãy ký tự trước khi đưa ra dòng xuất.

Chú ý: Để xuất nhiều giá trị trên một dòng lệnh, có thể viết như sau:

cout<<biểu thức 1<<biểu thức 2 <<...<<biểu thức n;

58.31.4.3. Các phương thức định dạng

1. Phương thức int cout.width();

cho biết độ rộng quy định hiện tại.

2. Phương thức int cout.width(int n);

thiết lập độ rộng quy định mới là n và trả về độ rộng quy định trước đó.

Chú ý: Độ rộng quy định n chỉ có tác dụng cho một giá trị xuất. Sau đó C++ lại áp dụng độ rộng quy định bằng 0. Ví dụ với các câu lệnh:

```
int m=1234, n=56;
cout<<"\nAB";
cout.width(6);
cout<<m;
cout<<n;
```

thì kết quả là:

AB 123456

(giữa B và số 1 có 2 dấu cách).

3. Phương thức int cout.precision();

cho biết độ chính xác hiện tại (đang áp dụng để xuất các giá trị thức).

4. Phương thức int cout.precision(int n);

thiết lập dộ chính xác sẽ áp dụng là n và cho biết độ chính xác trước đó. Độ chính xác được thiết lập sẽ có hiệu lực cho tới khi gặp một câu lệnh thiết lập độ chính xác mới.

5. Phương thức char cout.fill();

cho biết ký tự độn hiện tại đang được áp dụng.

6. Phương thức char cout.fill(char ch);

quy định ký tự độn mới sẽ được dùng là ch và cho biết ký tự độn đang dùng trước đó. Ký tự độn được thiết lập sẽ có hiệu lực cho tới khi gặp một câu lệnh chọn ký tự độn mới.

Ví dụ:

```
#include <iostream.h>
#include <conio.h>
void main()
  {
     clrscr();
     float x=-3.1551, y=-23.45421;
     cout.precision(2);
     cout.fill('*');
     cout<<"\n";
     cout.width(8);
     cout<<x;
     cout<<"\n";
     cout.width(8);
     cout<<y;
     getch();
  }
```

Sau khi thực hiện, chương trình in ra màn hình 2 dòng sau:

***-3.16

**-23.45

58.41.4.4. Cờ định dạng

Mỗi cờ định dạng chứa trong một bit. Cờ có 2 trạng thái: Bật (on) – có giá trị 1, Tắt (off) – có giá trị 0. Các cờ có thể chứa trong một biến kiểu long. Trong tập tin iostream.h đã định nghĩa các cờ sau:

ios::left	ios::right	ios::internal
ios::dec	ios::oct	ios::hex
ios::fixed	ios::scientific	ios::showpos
ios::uppercase	ios::showpoint	ios::showbase

Có thể chia cờ định dạng thành các nhóm:

Nhóm 1 gồm các cờ căn lề:

ios::left	ios::right	ios::internal

Cờ ios::left: khi bật cờ ios::left thì giá trị in ra nằm bên trái vùng quy định, các ký tự độn nằm sau.

Cờ ios::right: khi bật cờ ios::right thì giá trị in ra nằm bên phải vùng quy định, các ký tự độn nằm trước.

Chú: ý mặc định cờ ios::right bật.

Cờ ios::internal: cờ ios::internal có tác dụng giống như cờ ios::right chỉ khác là dấu (nếu có) in đầu tiên.

Chương trình sau minh hoạ cách dùng các cờ căn lề:

Ví dụ

```
#include <iostream.h>
#include <conio.h>
void main()
 {
   clrscr();
   float x=-87.1551, y=23.45421;
   cout.precision(2);
   cout.fill('*');
   cout.setf(ios::left); //bat co ios::left
   cout<<"\n";
   cout.width(8);
   cout<<x;
   cout<<"\n";
   cout.width(8);
   cout<<y;
   cout.setf(ios::right); //bat co ios::right
   cout<<"\n";
   cout.width(8);
   cout<<x;
   cout<<"\n";
   cout.width(8);
   cout<<y;
   cout.setf(ios::internal); //bat co ios::internal
   cout<<"\n";
   cout.width(8);
   cout<<x;
   cout<<"\n";
```

```
    cout.width(8);
    cout<<y;
    getch();
}
```

Sau khi thực hiện chương trình in ra 6 dòng như sau:

```
      -87.16**
      23.45**
  **-87.16
  ***23.45
  -**87.16
  ***23.45
```

Nhóm 2 gồm các cờ định dạng số nguyên:

ios::dec ios::oct ios::hex

+ Khi ios::dec bật (mặc định): số nguyên được in dưới dạng cơ số 10

+ Khi ios::oct bật: số nguyên được in dưới dạng cơ số 8

+ khi ios::hex bật: số nguyên được in dưới dạng cơ số 16

Nhóm 3 gồm các cờ định dạng số thực:

ios::fixed ios::scientific ios::showpoint

Mặc định : cờ ios::fixed bật (on) và cờ ios::showpoint tắt (off).

+ Khi ios::fixed bật và cờ ios::showpoint tắt thì số thực in ra dưới dạng thập phân, số chữ số phần phân (sau dấu chấm) được tính bằng độ chính xác n nhưng khi in thì bỏ đi các chữ số 0 ở cuối.

Ví dụ nếu độ chính xác n = 4 thì

Số thực -87.1500 được in: -87.15

Số thực 23.45425 được in: 23.4543

Số thực 678.0 được in: 678

+ Khi ios::fixed bật và cờ ios::showpoint bật thì số thực in ra dưới dạng thập phân, số chữ số phần phân (sau dấu chấm) được in ra đúng bằng độ chính xác n.

Ví dụ nếu độ chính xác n = 4 thì

Số thực -87.1500 được in: -87.1500

Số thực 23.45425 được in: 23.4543

Số thực 678.0 được in: 6780000

+ Khi ios::scientific bật và cờ ios::showpoint tắt thì số thực in ra dưới dạng khoa học. Số chữ số phần phân (sau dấu chấm) được tính bằng độ chính xác n nhưng khi in thì bỏ đi các chữ số 0 ở cuối.

Ví dụ nếu độ chính xác n=4 thì

Số thực -87.1500 được in: -87.15e+01

Số thực 23.45425 được in: 23.4543e+01

Số thực 678.0 được in: 678e+02

+ Khi ios::scientific bật và cờ ios::showpoint bật thì số thực in ra dưới dạng mũ . Số chữ số phần phân (sau dấu chấm) của phần định trị được in đúng bằng độ chính xác n.

Ví dụ nếu độ chính xác n=4 thì

Số thực -87.1500 được in: -87.150e+01

Số thực 23.45425 được in: 23.4543e+01

Số thực 678.0 được in: 67800e+01

Nhóm 4 gồm các hiển thị:

ios::uppercase ios::showpos ios::showbase

Cờ ios::showpos

+ Nếu cờ ios::showpos tắt (mặc định) thì dấu cộng không được in trước số dương.

+ Nếu cờ ios::showpos tắt thì dấu cộng được in trước số dương.

Cờ ios::showbase bật thì số nguyên hệ 8 được in bắt đầu bằng ký tự 0 và số nguyên hệ 16 được bắt đầu bằng các ký tự 0x. Ví dụ nếu a = 40 thì:

Dạng in hệ 8 là: 050

Dạng in hệ 16 là 0x28

Cờ ios::showbase tắt (mặc định) thì không in 0 trước số nguyên hệ 8 và không 0x trước số nguyên hệ 16. Ví dụ nếu a = 40 thì:

Dạng in hệ 8 là: 50

Dạng in hệ 16 là 28

Cờ ios::uppercase

+ Nếu cờ ios::uppercase bật thì các chữ số hệ 16 (như A, B, C,...) được in dưới dạng chữ hoa.

+ Nếu cờ ios::uppercase tắt (mặc định) thì các chữ số hệ 16 (như A, B, C,...) được in dưới dạng chữ thường.

58.51.4.5. Các phương thức bật tắt cờ

Các phương thức này định nghĩa trong lớp ios.

1. Phương thức long cout.setf(long f) ;

sẽ bật các cờ liệt kê trong f và trả về một giá trị long biểu thị các cờ đang bật.

2. Phương thức long cout.unsetf(long f) ;

sẽ tắt các cờ liệt kê trong f và trả về một giá trị long biểu thị các cờ đang bật.

3. Phương thức long cout.flags(long f) ;

có tác dụng giống như cout.setf(long).

4. Phương thức long cout.flags() ;

sẽ trả về một giá trị long biểu thị các cờ đang bật.

58.61.4.6. Các bộ phận định dạng

Các bộ phận định dạng (định nghĩa trong tập tin iostream.h) bao gồm:

dec // như cờ ios::dec

oct // như cờ ios::oct

hex // như cờ ios::hex

endl // xuất ký tự '\n' (chuyển dòng)

flush // đẩy dữ liệu ra thiết bị xuất

Ví dụ Xét chương trình sau:

```
#include <iostream.h>
#include <iomanip.h>
#include <conio.h>
void main()
  {
   clrscr();
   cout.setf(ios::showbase);
   cout<<"ABC"<<endl<<hex<<40<<"  "<<41;
   getch();
  }
```

58.71.4.7. Các hàm định dạng

Các hàm định dạng (định nghĩa trong <iomanip.h>) bao gồm:

set(int n) // như cout.width(int n)

setpecision(int n) // như cout.setpecision(int n)

setfill(char ch) // như cout.setfill(char ch)

setiosflags(long l) // như cout. setiosflags(long f)

resetiosflags(long l) // như cout. setiosflags(long f)

Các hàm định dạng có tác dụng như các phương thức định dạng nhưng được viết nối đuôi trong toán tử xuất nên tiện sử dụng hơn.

Chú ý

- Các hàm định dạng (cũng như các bộ phận định dạng) cần viết trong toán tử xuất. Một hàm định dạng đứng một mình như một câu lệnh sẽ không có tác dụng dịnh dạng.
- Muốn sử dụng các hàm định dạng cần bổ sung vào đầu chương trình câu lệnh: #include <iomanip.h>

Chương trình trong ví dụ 1.2. có thể viết lại theo các phương án sau:

Phương án 1:

```
#include <iostream.h>
#include <conio.h>
void main()
  {
   clrscr();
   cout <<setiosflags(ios::showbase);
   cout<<"ABC"<<endl<<hex<<40<<" "<<41;
   getch();
  }
```

Phương án 2:

```
#include <iostream.h>
#include <conio.h>
void main()
  {
   clrscr();
   cout <<"ABC"<<endl<< setiosflags(ios::showbase)
   << hex<<40<<" "<<41;
   getch();
  }
```

Bài 59 : 1.5. Các dòng chuẩn

Có 4 dòng (đối tượng của các lớp stream) đã định nghĩa trước, được cài đặt khi chương trình khởi động là:

- cin dòng input chuẩn gắn với bàn phím, giống như stdin của C.
- cout dòng output chuẩn gắn với màn hình, giống như stdout của C.
- cerr dòng output lỗichuẩn gắn với màn hình, giống như stderr của C.
- clog giống cerr nhưng có thêm bộ đệm.

Chú ý

- Có thể dùng các dòng cerr và clog để xuất ra màn hình như đã dùng đối với cout.
- Vì clog có thêm bộ đệm, nên dữ liệu được đưa vào bộ đệm. Khi đầy bộ đệm thì đưa dữ liệu bộ đệm ra dòng clog. Vì vậy trước khi kết thúc xuất cần dùng phương thức: clog.flush(); để đẩy dữ liệu từ bộ đệm ra clog.

Chương trình sau minh họa cách dùng dòng clog. Chúng ta nhận thấy, nếu bỏ câu lệnh clog.flush() thì sẽ không nhìn thấy kết quả xuất ra màn hình khi chương trình tạm dừng bởi câu lệnh getch().

Ví dụ

```
#include  <iostream.h>
#include  <conio.h>
void main()
 {
   clrscr();
   float x=-87.1500, y=23.45425,z=678.0;
   clog.setf(ios::scientific);
   clog.precision(4);
   clog.fill('*');
   clog<<"\n";
   clog.width(10);
   clog<<x;
   clog<<"\n";
   clog.width(10);
   clog<<y;
   clog<<"\n";
   clog.width(10);
   clog<<z;
   clog.flush();
   getch();
 }
```

Bài 60 : 1.6. Xuất ra máy in

Bốn dòng chuẩn không gắn với máy in. Như vậy không thể dùng các dòng này để xuất dữ liệu ra máy in. Để xuất dữ liệu ra máy in (cũng như nhập, xuất trên tệp) cần tạo ra các dòng tin mới và cho nó gắn với thiết bị cụ thể. C++ cung cấp 3 lớp stream để làm điều này, đó là các lớp:

ofstream dùng để tạo các dòng xuất (ghi tệp)

ifstream dùng để tạo các dòng nhập (đọc tệp)

fstream dùng để tạo các dòng nhập, dòng xuất hoặc dòng nhập-xuất

Mỗi lớp có 4 hàm tạo dùng để khai báo các dòng (đối tượng dòng tin). Để tạo một dòng xuất và gắn nó với máy in ta có thể dùng một trong những hàm tạo sau đây:

ofstream Tên_dòng(int fd);

ofstream Tên_dòng(int fd, chả *buf, int n);

Trong đó:

- Tên_dòng là tên biến đối tượng kiểu ofstream chúng ta tự đặt.

- fd(file disciptor) là chỉ số tập tin. Chỉ số tập tin định sẵn đối với stdprn (máy in chuẩn) là 4.

- Các tham số buf và n xác định một vùng nhớ n byte do buff trỏ tới. Vùng nhớ sẽ được làm bộ đệm cho dòng xuất.

Ví dụ: Câu lệnh ofstream prn(4); sẽ tạo dòng tin xuất prn và gắn nó với máy in chuẩn. Dòng prn sẽ có bộ đệm mặc định. Dữ liệu trước hết chuyển vào bộ đệm, khi đầy bộ đêm thì dữ liệu sẽ được đẩy từ bộ đệm ra dòng prn và có thể sử dụng phương thức flush hoặc bộ phận định dạng flush. Cách viết như sau:

```
prn.flush ;// Phương thức
prn<<flush; //Bộ phận định dạng
```

Các câu lệnh sau sẽ xuất dữ liệu ra prn (máy in) và ý nghĩa của chúng như sau:

```
prn<<"\n Tong="<<(4+9); //Đưa một dòng vào bộ đệm
prn<<"\n Tich="<<(4*9); // Đưa dòng tiếp theo vào  bộ đệm
prn.flush(); //Đẩy dữ liệu từ bộ đệm ra máy in (in 2 dòng)
```

Các câu lệnh dưới đây sẽ xuất dữ liệu ra máy in nhưng xuất từng dòng một:

```
prn<<"\n Tong="<<(4+9)<<flush; // In một dòng
prn<<"\n Tích="<<(4*9)<<flush; //In dòng tiếp theo
```

Ví dụ: Các câu lệnh

```
char  buf [512];
ofstream prn(4,buf,512);
```

sẽ tạo dòng tin xuất prn và gắn nó với máy in chuẩn. Dòng xuất prn sử dụng 512 byte của mảng buf làm bộ đệm. Các câu lệnh dưới đây cũng xuất ra máy in:

```
prn<<"\n Tong="<<(4+9); //Đưa dữ liệu vào bộ đêm
prn<<"\n Tich="<<(4*9) ; //Đưa dữ liệu vào bộ đêm
prn.flush(); // Xuất 2 dòng (ở bộ đệm) ra máy in
```

Chú ý: Trước khi kết thúc chương trình, dữ liệu từ bộ đệm sẽ được tự động đẩy ra máy in.

tài liệu tham khảo

1. Ivar Jacobson, Object - Oriented Software Engineering, Addison-Wesley Publishing Company, 1992.
2. Michael Blaha, William Premerlani, Object - Oriented Modeling and Design for Database Applications, Prentice Hall, 1998.
2. Phạm Văn ất, C++ và Lập trình hướng đối tượng, NXB Khoa học và Kỹ thuật, 1999.
3. Đoàn Văn Ban, Phân tích và thiết kế hướng đối tượng, NXB Khoa học và Kỹ thuật, 1997.
4. Nguyễn Thanh Thủy, Lập trình hướng đối tượng với C++, NXB Khoa học và Kỹ thuật, 1999.

Mục lục

CHươNG 1

CáC KHáI NIệM Cơ Sở
của LậP TRìNH HướNG Đối TượNG

Chương 2

các mở rộng của ngôn ngữ C++

Bài 61 :

Bài 62 : Phụ lục

Bài 63 : Các dòng xuất nhập