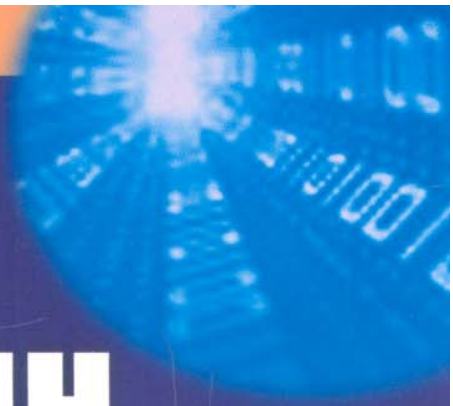
PHẠM CÔNG NGÔ



LẬP TRÌNH C#

TỪ CƠ BẢN ĐẾN NÂNG CAO

NHÀ XUẤT BẢN GIÁO DỤC

PHẠM CÔNG NGÔ

LẬP TRÌNH C#

từ cơ bản đến nâng cao

NHÀ XUẤT BẢN GIÁO DỤC

17 – 2007/CXB/67 – 2217/GD Mã số: 7B656M7 – DAI

Lời giới thiệu

Đầu thế kỷ XXI, Microsoft đưa ra một bộ Visual Studio .NET (VS 7.0). Có thể nói từ VS 2001 đến VS 2003 đã có một bước cải tiến đáng kể về nội dung. Bộ Visual Studio 6.0 (với các ngôn ngữ Visual Basic 6.0, Visual C++ 6.0, ASP 3.0,...) đã đem lại cho lập trình viên nhiều thuận tiện và lợi ích. Tuy nhiên khi bộ Visual Studio. NET được công bố thì mọi người đã chuyển sang học tập, nghiên cứu phiên bản mới này khá rầm rộ. Bộ Visual Studio 2003 đã chiếm lĩnh vị trí đáng kể trong mấy năm gần đây. Từ đầu năm 2006, bộ Visual Studio 2005 (VS 8.0) đã xuất hiện ở Việt Nam và mọi người đã thử nghiệm sử dụng bộ .NET 2005 này. Tính từ 2001 đến 2006, ba bộ .NET: 2001; 2003; 2005 lần lượt ra đời. Đội ngũ lập trình viên không còn lạ với chúng và hầu hết các sinh viên ngành Công nghệ thông tin, ngành Điện, Điện tử, Cơ khí, Kinh tế... đã bắt đầu học tập, nghiên cứu sử dụng chúng. Các trường và trung tâm hợp tác quốc tế như "Hà Nội Genetic – Singapore", các trung tâm của Australia, Pháp, Mỹ, Ấn Độ ở Hà Nội, thành phố Hồ Chí Minh hầu như giảng dạy cho sinh viên với tỷ lệ phần trăm rất cao về bộ Visual Studio .NET.

Để đáp ứng yêu cầu tìm hiểu, học tập của sinh viên các ngành kỹ thuật, kinh tế nói chung và của các sinh viên ngành Công nghệ thông tin nói riêng và đặc biệt cho những ai có nhu cầu tự học tập, dựa vào kinh nghiệm giảng dạy từ năm 2002 đến nay về bộ .NET, tác giả biên soạn tài liệu này nhằm đáp ứng phần nào cho các đối tượng trên.

Nếu các bạn đã có kiến thức về lập trình C/C++ hoặc Java thì việc học ngôn ngữ C# (C sharp) rất thuận lợi. Tuy nhiên đây là một cuốn sách dành cho sinh viên và các bạn tự học, do đó tài liệu đã cố gắng trình bày thật dễ hiểu và ngắn gọn nhất.

Lập trình C# từ cơ bản đến nâng cao bao gồm các lệnh chu trình, điều kiện, lựa chọn, về phương thức hay "hàm", lớp, thừa kế, nạp chồng hàm và toán tử trùng tên, uỷ nhiệm, quản lý sự kiện, đa luồng (multithreading), v.v...

Về Visual Studio .NET đã có khá nhiều sách ở nước ngoài cũng như

trong nước. Tuy nhiên để có một tài liệu tự học, tham khảo nhanh, ngắn gọn và dễ hiểu tác giả đã cố gắng dựa vào kinh nghiệm giảng dạy cũng như các bài giảng đã được soạn và chỉnh lý để cuốn sách được hoàn thiện hơn.

Cuốn sách này được xuất bản lần đầu cho nên không thể tránh khỏi khiếm khuyết. Tác giả rất mong các bạn đồng nghiệp gần xa, các sinh viên sử dụng bộ sách này đóng góp ý kiến để lần tái bản sau được hoàn chỉnh hơn. Cuối cùng tác giả cũng có một lời mách nhỏ là tất cả các chương trình trong sách đều đã chạy tốt trên máy PC có cài đặt bộ Visual Studio 2003 và các bạn hãy cố gắng tự mình đánh vào máy để kiểm tra kết quả, đây cũng là một phương pháp tự học rất hiệu quả mà trong bao năm qua tác giả và đồng nghiệp đã sử dụng.

Mọi ý kiến góp ý và thư từ xin gửi về Công ty CP Sách ĐH – DN, 25 Hàn Thuyên, Hà Nội, hoặc email: pcongngo@yahoo.com.vn

TÁC GIẢ

Hà Nội, 15 – 8 – 2007

Chương 1. Các nét cơ bản của C#

1.1. GIỚI THIỆU C#

C# (được phát âm là "C sharp") là một ngôn ngữ lập trình hướng đối tượng, hiện đại, tin cậy và được sử dụng nhiều cho Internet. C# được kế thừa từ các ngôn ngữ lập trình hướng đối tượng quen thuộc khác như CTT, Java.

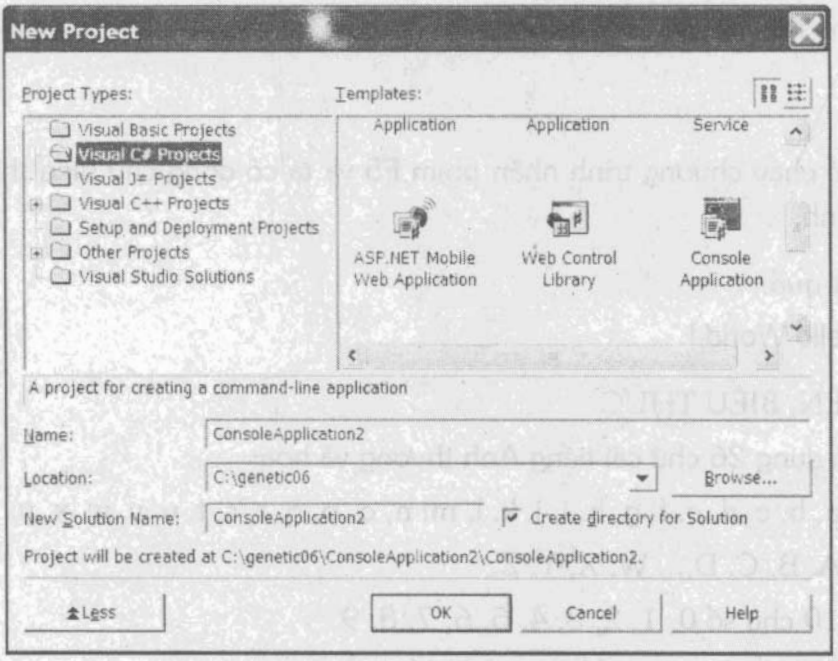
C# được xây dựng từ một nhóm các kỹ sư phần mềm của Microsoft do hai người lãnh đạo là Anders Hejlsberg và Scott Wiltamuth.

Các chương trình C# chủ yếu được viết trong môi trường "console" và môi trường "windows form". Để giới thiệu một chương trình đầu tiên làm ví dụ, các bạn phải cài đặt bộ Visual Studio .NET (2003) trong Windows 2000 hoặc Windows XP.

– Nhấn Start → Program trong Windows.

– Chọn Microsoft Studio .NET 2003.

– Từ menu File → New → Project.



Hình 1

Trong hình 1.1, phía trái Project Types chọn Visual C# Project. Ở phía phải Templates, chọn Console Application.

– Tại Textbox có nhãn Name: prg1.

– Tại Textbox có nhãn Location: C:\tu hoc csharp.

– Nhấn nút OK.

Sau đó, nhấn đồng thời 2 phím Ctrl + A để lựa chọn toàn bộ rồi bấm phím Delete.

Cuối cùng bạn gõ vào văn bản chương trình sau:

Ví dụ 1.1. Viết chương trình hiển thị trên màn hình dòng chữ: "Hello world !".

```
using System;
public class prog1
{
public static void Main ()
{
Console.WriteLine("Hello world !");
Console.ReadLine();
}
}
```

Để chạy chương trình nhấn phím F5 và ta có dòng chữ hiển thị trên màn hình:

Kết quả:

Hello World !

1.2. BIẾN, BIỂU THỨC

Sử dụng 26 chữ cái tiếng Anh thường và hoa:

– a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z.

– A, B, C, D,... W, X, Y, Z.

– 10 chữ số 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

– Dấu gạch dưới _ (không phải dấu trừ).

Tên biến, biểu thức hoặc phương thức (hay hàm) chỉ được sử dụng 63 kí tự trên mới hợp lệ. Ví dụ: Hanoi; HANOI; _Hanoi; Ha_noi.

Chú ý:

+ Chữ số 0,1,... 9 không được phép đặt vị trí đầu tiên của tên biến.

+ Tên biến không được phép đặt trùng với các từ khoá (keyword).

+ Tên biến phân biệt nhau bởi chữ thường và chữ hoa (điều này khác hoàn toàn với ngôn ngữ Pascal).

+ Dấu gạch dưới có thể đặt ở vị trí đầu hoặc giữa tên biến đều hợp lệ.

1.3. TỪ KHOÁ (KEYWORD)

Bảng 1.1. Từ khoá của ngôn ngữ C#

abstract	as	base	bool
break	byte	ase	catch
char	checked	class	const
continue	decimal	default	delegate
do	double	else	enum
event	explicit	extern	false
finally	fixed	float	for
foreach	goto	if	implicit
in	int	Interface	internal
is	loca	long	namespace
new	null	object	operator
out	override	params	private
protected	public	readonly	res
return	sbyte	sealed	short
sifeof	stackalloc	static	string
struct	switch	this	throw
true	try	typeof	unit
ulong	unchecked	unsafe	ushort
using	virtual	void	while

1.4. KIỂU (TYPE)

Bảng 1.2. Mô tả các kiểu dữ liệu xây dựng sẵn

Kiểu C#	Số byte	Kiểu .NET	Mô tả
byte	1	Byte	Số nguyên dương không dấu: 0 ÷ 255
char	2	Char	Ký tự Unicode
bool	1	Boolean	Giá trị logic true/ false
sbyte	1	Sbyte	Số nguyên có dấu: -128 ÷ 127
short	2	Int16	Số nguyên có dấu giá trị: -32768 ÷ 32767.
ushort	2	Uint16	Số nguyên không dấu: 0 ÷ 65.535
int	4	Int32	Số nguyên có dấu: -2.147.483.647 ÷ 2.147.483.647
uint	4	Uint32	Số nguyên không dấu: 0 ÷ 4.294.967.295
float	4	Single	Kiểu dấu chấm động giá trị xấp xỉ từ 3,4E-38 đến 3,4E+38, với 7 chữ số có nghĩa
double	8	Double	Kiểu dấu chấm động có độ chính xác gấp đôi, giá trị xấp xỉ từ 1,7E-308 đến 1,7E+308, với 1516 chữ số có nghĩa.
decimal	8	Decimal	Có độ chính xác đến 28 con số và giá trị thập phân, được dùng trong tính toán tài chính, kiểu này đòi hỏi phải có hậu tố "m" hay "M" theo sau giá trị.
long	8	Int64	Số nguyên có dấu: -9.223.370.036.854.775.808 ÷ 9.223.372.036.854.775.807
ulong	8	Uint64	Số nguyên không dấu: 0 ÷ 0xffffffffffffffff

1.5. KHAI BÁO BIẾN (DECLARATION)

kiểu biến 1, biến 2 = giá trị;

```
int a, b = 2;
double x = 1.456378, y, &.
string s = "Hanoi";
```

```
long m = 356L;
byte vb = 67;
sbyte sb = -205;
ushort us1 = 7653;
ulong ul1 = 568ul;
float f1 = 1.367F;
double d1 = 1.25439;
bool b1 = true;
char ch = 'M';
decimal vd1 = 1.28659M;
string s1 = "Ha noi";
string s2 = "Viet nam";
string s3 = s1 + s2;
```

Chú ý:

– Với kiểu float sau giá trị gán cho biến luôn phải ghi thêm chữ F (viết hoa). Ví dụ: float f1 = 1.2589F;

– Các kiểu biến khác không nhất thiết phải ghi thêm. Có thể bỏ chữ L vẫn hợp lệ. Ví dụ: long l1 = 125987654L.

1.6. CÁC TOÁN TỬ SỐ HỌC

+	cộng
–	trừ
*	nhân
/	chia
%	chia lấy số dư
++i; i++	tăng giá trị i lên 1 đơn vị
--i; i--	giảm giá trị i xuống 1 đơn vị

Ví dụ:

```
int a = 16, b = 3;
int q1 = a + b; kết quả q1 = 19;
```

int q2 = a / b; kết quả q2 = 5;

int q3 = a % b; kết quả q3 = 1;

double q4 = (double) a/b; kết quả q4 = 3.2.

kiểu viết lệnh gán ngắn gọn:

i+ = 5; tương đương i = i + 5;

i– = 4; tương đương i = i – 4;

i* = 6; tương đương i = i * 6;

i/ = 2; tương đương i = i / 2;

1.7. TOÁN TỬ QUAN HỆ VÀ LOGIC

>	lớn hơn
>=	lớn hơn hoặc bằng
<	nhỏ hơn
<=	nhỏ hơn hoặc bằng
==	bằng
!=	không bằng
&&	và
||	hoặc
!	phủ định

Ví dụ:

```
if ((a > b) && (a > c)) lệnh 1;
if (a != b) lệnh 2;
```

1.8. TOÁN TỬ XỬ LÝ BIT

&	và bit
|	hoặc bit
!	phủ định bit
>>	dịch phải (right shift)
<<	dịch trái (left shift)
^	xor bit

Ví dụ 1.2.

```
using System;
class xulybit
{
static void Main(string [] args)
{
int a = 126, b = 86;
int q1, q2, q3, q4, q5;
q1 = a & b
q2 = a | b;
q3 = a ^ b;
q4 = a >> 2;
q5 = b << 3;
Console.WriteLine("a = "+a+"; b = "+b);
Console.WriteLine("q1 = "+q1+"; q2 = "+q2);
Console.WriteLine("q3 = "+q3+"; q4 = "+q4);
Console.WriteLine("q5 = "+q5);
Console.ReadLine();
}
}
```

Kết quả:

a = 126; b = 86;

q1 = 86; q2 = 126;

q3 = 40; q4 = 31;

q5 = 688.

Để hiểu được kết quả này ta nhớ lại bảng các phép toán logic với bit:

a	b	a & b	a | b	a^b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

+ Đổi các số a, b từ hệ 10 về hệ 16.

a = 126 → a = $(7E)_{16}$.

b = 86 → b = $(56)_{16}$.

+ Đổi a từ hệ 16 về hệ 2.

a = $(7E)_{16}$ → 0 1 1 1 1 1 1 0

q4 = a >> 2 0 0 0 1 1 1 1 1

q4 = $(1F)_{16}$ = $(16 + 15)_{10}$ = $(31)_{10}$.

+ Đổi b từ hệ 16 về hệ 2.

b = $(56)_{16}$ → 0 1 0 1 0 1 1 0

q5 = b << 3; 0 1 0 1 0 1 1 0 0 0 0

2 B 0

q5 = $(2B0)_{16}$ = $(2.256 + 11.16)_{10}$ = $(688)_{10}$.

1.9. CHÚ THÍCH

Cách chú thích các đầu dòng lệnh và đoạn lệnh chương trình trong C# cũng giống như trong C++ hay Java.

- Chú thích cho một dòng: // dòng được chú thích đến hết dòng
- Chú thích cho nhiều dòng:

```
/* dòng 1;
 dòng 2;
 dòng n;
*/
```

1.10. KIỂU LIỆT KÊ (ENUM)

Cú pháp kiểu liệt kê phải đặt từ khoá enum lên đầu:

```
enum E 1 { a, b, c = 15, d, e} ;
```

Trong đó E1 là tên biến có kiểu enum (liệt kê);

bằng số a lấy giá trị nguyên bằng 0

b lấy giá trị 1

c lấy giá trị 15

d lấy giá trị 16

e lấy giá trị 17

Ví dụ 1.3.

```
using System;
class Enumeration
{
enum E1{ a, b, c = 15, d, e} ;
static void Main()
{
Console.WriteLine("Gia tri b = "+(int)E1.b);
Console.WriteLine("Gia tri d = "+(int)E1.d);
Console.ReadLine();
}
}
```

Kết quả:

Giá trị b = 1

Giá trị d = 16

1.11. KIỂU CẤU TRÚC – STRUCT

Kiểu struct giống như trong ngôn ngữ C. Struct được định nghĩa với tất cả các biến có kiểu public đặt trên class và dưới using System.

Trong hàm Main () ta khai báo các biến có kiểu struct và gán trực tiếp dữ liệu cho các biến struct qua toán tử chấm (dot operator).

```
struct St1
{
public string ten;
public int maso;
}
```

+ Khai báo biến struct trong hàm Main ()

```
St1 p;
```

+ Truy cập vào các biến trong struct

```
p.ten = "Hoa";
p.maso = 555;
```

Ví dụ 1.4.

```
using System;
public struct St1
{
public string ten;
public int maso;
}
class ViduStruct
{
static void Main()
{
St1 p; //khai bao bien p co kieu St1
p.ten = "Hoa";
p.maso = 555;
Console.WriteLine("Ten: "+p.ten+"\nMa so: "+p.maso);
Console.ReadLine ()
}
}
```

Kết quả:

Ten: Hoa

Ma so: 555

1.12. KIỂU MẢNG (ARRAY)

+ Mảng một chiều:

```
int [ ] a = new int [ 10] ;
```

+ Mảng hai chiều:

```
double [ ,] mat = new double [ 3,3] ;
```

Mục này ta chỉ nêu kiểu khai báo mảng. Để hiểu kỹ về mảng các bạn xem ở chương 5.

Chương 2. Xuất nhập dữ liệu

2.1. XUẤT DỮ LIỆU RA MÀN HÌNH

2.1.1. Xuất dữ liệu không định dạng

```
int a = 15;
double b = 123,45789;
char ch = 'M';
string str = "Ha Noi";
Console.Writeline("a = "+a+"; b = "+b);
Console.Writeline("ch = "+ch+"; str = "+str);
```

2.1.2. Xuất dữ liệu có định dạng với số dấu phảy động

```
float f1 = 123.7658437;
double d1 = 325.45468;
Console.Writeline("f1 = {0:0.000}; d1 = {1:0.0000}", f1,d1);
```

Kết quả:

f1 = 123.766; d1 = 325.4345

2.2. NHẬP DỮ LIỆU TỪ BÀN PHÍM

```
string s;//khai báo chuỗi ký tự s
int n;//khai báo biến nguyên n
s = Console.Readline();//các ký tự từ bàn phím gán cho s
n = int 32.Parse(s);//chuyển đổi các ký tự số về giá trị số
```

Hoặc ta có thể viết gộp:

```
int n;
n = Int32.Parse(Console.ReadLine());
```

Hay: `n = int.Parse(Console.ReadLine());`

Ứng với một kiểu khai báo phải có một lệnh tương ứng. Sau đây ta nêu lên bảng liệt kê của kiểu khai báo với lệnh chuyển đổi chuỗi nhập từ bàn phím.

short a	a = Int16.Parse (Console.ReadLine());
ushort a	a = UInt16.Parse (Console.ReadLine());
unit a	a = UInt32.Parse(Console.ReadLine());
long a	a = Int64.Parse (Console.ReadLine());
ulong a	a = UInt64.Parse(Console.ReadLine());
string s	s = Console.ReadLine();
char ch	ch = Char.Parse(Console.ReadLine());
	ch = char.Parse(Console.ReadLine());
float a	a = Single.Parse(Console.ReadLine());
	a = float.Parse(Console.ReadLine());
double a	a = Double.Parse(Console.ReadLine());
	a = double.Parse(Console.ReadLine());

Ví dụ 2.1.

```
using System;
class InOut
{
static void Main()
{
int ID; //ma so
string name; //ten
float salary; //luong
ulong tel; //dien thoai
Console.Write("Nhap ma so: ")
ID = Int32.Parse(Console.ReadLine());
Console.Write("Nhap ten: ")
name = Console.ReadLine();
Console.Write("Nhap luong: ");
salary = Single.Parse(Console.ReadLine());
Console.Write("Nhap so dien thoai: ")
tel = UInt64.Parse(Console.ReadLine());
Console.Write("Ma so: "+ID);
```

```
Console.Write("; Ten: "+name);
Console.Write("; Luong: "+salary);
Console.WriteLine("; So dien thoai: "+tel);
Console.ReadLine();// doi go mot phim
}
}
```

 *Kết quả:*

Nhap ma so: 111

Nhap ten: An

Nhap luong: 2000

Nhap so dien thoai: 8676432

Ma so: 111; Ten: An; Luong: 2000; So dien thoai: 8676432

Chương 3. Các lệnh điều khiển

Các lệnh điều khiển bao gồm ba nhóm:

– Lệnh lặp hay lệnh chu trình.

– Lệnh điều kiện.

– Lệnh lựa chọn.

3.1. CÁC LỆNH LẶP (LOOP)

Trong C# có bốn lệnh lặp.

3.1.1. Lệnh lặp for

```
for (bt khởi tao; bt điều kiên; bt thay đổi)
{
lệnh 1;
lệnh 2;
}
```

Trong đó biểu thức khởi tạo, biểu thức điều kiện và biểu thức thay đổi đứng tách nhau bởi dấu chấm phẩy.

Ví dụ: `for (int i = 0; i < 5; i++)`

Ví dụ 3.1. Sử dụng vòng lặp for để hiển thị bình phương của số nguyên i từ 0 đến 10.

```
using System
class LoopFor1
{
static void Main()
{
for (int i = 0; i <= 10; i++)
Console.WriteLine ("i = "+i+"; i*i = "+i*i);
Console.ReadLine();
}
}
```

Vòng lặp for còn thực hiện với các biểu thức phức tạp. Trong ba biểu thức tách nhau bởi dấu chấm phẩy thì ở biểu thức khởi tạo và biểu thức thay đổi có thể chứa nhiều biểu thức con và chúng tách nhau bởi dấu phẩy còn biểu thức điều kiện thì có các phép toán quan hệ như && (và); !! (hoặc) tạo thành.

Ví dụ 3.2. Hiển thị các giá trị i và j đồng thời.

```
using System;
class LoopFor2
{
static void Main ()
{
int i, j;
for (i = 0, j = 5; (i < 4) && (j > 2); i++, j--)
Console.WriteLine("i = "+i+"; j = "+j);
Console.ReadLine();
}
}
```

Kết quả:

i = 0; j = 5

i = 1; j = 4

i = 2; j = 3

Nếu thay đổi toán tử && bằng toán tử | | ta sẽ có kết quả khác một chút. Các bạn hãy tự thay đổi.

Vòng lặp for còn thực hiện với biểu thức có kiểu float, double.

Trong vòng for, lệnh thay đổi được viết:

```
for (double i = 0; i <= 2; i = i+0.1)
```

Ở đây bước thay đổi (step) sẽ bằng 0.1.

3.1.2. Vòng lặp do... while

Vòng lặp do... while cũng có 3 biểu thức:

– Biểu thức khởi tạo.

– Biểu thức thay đổi (sau khi thực hiện lệnh).

– Biểu thức kiểm tra điều kiện. Nếu điều kiện thoả mãn (đúng) thì vòng lặp thực hiện còn nếu điều kiện không thoả mãn (sai) thì vòng lặp kết thúc.

Ví dụ 3.3. Tính bình phương giá trị i với bước thay đổi bằng 0.2 từ 0 đến 2.

```
using System;
class Do1
{
static void Main()
{
double i = 0; //lenh khoi tao
do
{
Console.WriteLine("i = "+i+"; i*i = "+i*i);
i+ = 0.2; //lenh thay doi
}
while (i < 2); //lenh kiem tra dieu kien
Console.ReadLine();
}
}
```

Vòng lặp do... while cũng được thực hiện đồng thời nhiều biểu thức (tương tự vòng lặp for).

Ví dụ 3.4. Hiển thị đồng thời hai biến i, j với giá trị i giảm từ 2 đến 1 với bước lùi bằng 0.2 và j tăng từ 0 đến 2 với bước tăng bằng 0.1.

```
using System;
class Do2
{
static void Main()
{
double i = 2, j = 0;
```

```
do
{
Console.WriteLine("i = "+i+"; j = "+j);
i- = 0.2;
j+ = 0.1;
}
while((i > 1) && (j < 2));
Console.ReadLine();
}
}
```

Kết quả:

i = 2; j = 0

i = 1.8; j = 0.1

i = 1.6; j = 0.2

i = 1.4; j = 0.3

i = 1.2; j = 0.4

3.1.3. Vòng lặp while

Tương tự vòng lặp do... while, vòng lặp while cũng có 3 biểu thức:

– Biểu thức khởi tạo.

– Biểu thức kiểm tra điều kiện.

– Thực hiện lệnh và biểu thức thay đổi.

Điểm khác biệt của 2 vòng lặp này là lệnh do... while sẽ kiểm tra điều kiện sau còn lệnh while kiểm tra điều kiện trước rồi mới đến thực hiện lệnh.

Ví dụ 3.5. Ta viết lại chương trình cho ví dụ 3.4 bằng lệnh while.

```
using System;
class While
{
static void Main()
{
```

```
int i = Console.WriteLine("i = "+i+"; j = "+j);
i- = 0.2;
j+ = 0.1;
}
Console.ReadLine();
}
}
```

Kết quả tương tự như ở ví dụ 3.4.

3.1.4. Vòng lặp foreach

Lệnh foreach thường sử dụng cho mảng. Cú pháp lệnh foreach:

```
foreach (kiểu tên in biểu thức lệnh)
{
// lenh;
}
```

Trong đó in là từ khoá.

Ví dụ 3.6. Khai báo mảng một chiều các số nguyên được gán trước bao gồm cả số lẻ và số chẵn. Hãy hiển thị số lượng các số lẻ và số lượng các số chẵn có trong mảng đó.

```
using System;
class ForEach
{
static void Main()
{
int [ ] arr = new int[ 5]{ 1, 4, 3, 6, 5} ;
int odd = 0, even = 0;
foreach (int m in arr)
{
if (m % 2 == 0)
even++;
else
odd++;
```

```
}
Console.WriteLine("So phan tu le odd = "+odd);
Console.WriteLine("So phan tu chan even = "+even);
Console.ReadLine();
}
}
```

Kết quả:

So phan tu le odd = 3

So phan tu chan even = 2

3.1.5. Lệnh break

Khi một vòng lặp đang thực hiện, nếu thoả mãn một điều kiện nào đó thì ngắt vòng lặp, ta dùng lệnh break.

Ví dụ 3.7.

```
using System
class Break
{
static void Main()
{
for (int i = 0; i < 100; i++)
{
if(i*i > 150)
break;
if(i%2 == 0)
Console.Write(" "+i);
}
Console.ReadLine()
}
}
```

Kết quả:

0 2 4 6 8 10 12

3.2. LỆNH ĐIỀU KIỆN

3.2.1. Cú pháp lệnh điều kiện

+ if (điều kiện) lệnh 1;

+ if (điều kiện) {lệnh 1; lệnh 2;}

+ if (điều kiện) lệnh 1;
  else lệnh 2;

+ Lệnh điều kiện lồng nhau:

```
if (điều kiện 1)
        {
        if (điều kiện 2) lệnh 1;
        else lệnh 2;
        }
else
        {
        if (điều kiện 3) lệnh 3;
        else lệnh 4;
        }
```

Ví dụ 3.8. Nhập ba số a, b, c từ bàn phím. Tìm số lớn nhất của chúng.

```
using System;
class IfElse
{
static void Main()
{
float a, b, c, m1, max;
Console.Write("Enter a, b, c: ");
a = Single.Parse(Console.ReadLine());
b = Single.Parse(Console.ReadLine());
c = Single.Parse(Console.ReadLine());
```

```
if (a > b) ml = a;
else ml = b;
if (ml > c) max = ml;
else max = c;
Console.WriteLine("a = {0}, b = {1}, c = {2}", a, b, c);
Console.WriteLine("Gia tri lon nhat = "+max);
}
}
```

Kết quả:

Enter a, b, c: 8.5

15.2

2.8

a = 8.5, b = 15.2, c = 2.8

Giá trị lớn nhất = 15.2

Ví dụ 3.9. Hãy nhập vào điểm số của các môn học. Thang để phân loại điểm được biểu diễn trên trục số.

0 ----- 5 ----- 8 ----- 10 (diem)

loại C loại B loại A

Cách 1: Sử dụng câu lệnh if với lệnh logic.

```
using System;
class If
{
static void Main()
{
int diem; //diem
Console.Write("Nhap diem: ");
diem = Int32.Parse(Console.ReadLine());
if((diem >= 5) && (diem < 8))
Console.WriteLine("Xep loai B.");
if((diem >= 0) && (diem < 5))
Console.WriteLine("Xep loai C.");
```

```
if((diem > = 8) && (diem < = 10))
Console.WriteLine("Xep loai A.");
if((diem < 0) !! (diem > 10))
Console.WriteLine("Diem khong hop le.");
Console.ReadLine();
}
}
```

Cách 2. Sử dụng câu lệnh if... else lồng nhau.

```
using System;
class IfNested //if long nhau.
{
static void Main ()
{
int diem;
Console.Write("Nhap diem: ")
diem = Int32.Parse(Console.ReadLine());
if (diem >= 8)
{
if (diem > 10)
Console.Write("Diem khong hop le.");
else
Console.WriteLine("Xep loai A.")
}
else
{
if(diem >= 5)
Console.WriteLine("Xep loai B.");
else
{
if(diem >= 0)
Console.WriteLine("Xep loai C.")
else
```

```
Console.WriteLine("Diem khong hop le.");
}
}
Console.ReadLine();
}
}
```

3.2.2. Sử dụng lệnh rút gọn cho if... else

Khi viết câu lệnh:

```
if (x > y) max = x
else max = y;
```

ta có thể viết bằng câu lệnh đơn giản tương đương sau:

```
max = (x > y ? x : y);
```

3.3. LỆNH LỰA CHỌN (SWITCH... CASE)

Lệnh switch... case trong C# thường sử dụng biến lựa chọn chủ yếu là kiểu nguyên.

Ví dụ 3.10. Nhập 2 số a, b. Tính, tổng, hiệu, tích, thương.

```
using System;
class SwitchCase
{
static void Main ()
{
double a, b;
int pt;
Console.Write("Nhap so a,b: ");
a = Double.Parse(Console.ReadLine());
b = Double.Parse(Console.ReadLine());
do
{
Console.WriteLine("MENU");
Console.WriteLine("1. Cong.");
Console.WriteLine("2. Tru.");
```

```
Console.WriteLine("3. Nhan.");
Console.WriteLine("4. Chia.");
Console.WriteLine("5. Thoat.");
pt = Int32.Parse(Console.ReadLine());
switch(pt)
{
case 1: Console.WriteLine("a + b = "+(a+b));
break;
case 2: Console.WriteLine("a -b = "+(a-b));
break;
case 3: Console.WriteLine("a * b = "+(a*b));
break;
case 4: Console.WriteLine("a / b = "+(a/b));
break;
case 5: continue;
}
}
while(pt != 5);
}
}
```

Chương 4. Phương thức (Method)

Phương thức cũng giống như hàm trong ngôn ngữ C/C++ hoặc ngôn ngữ Java. Phương thức có hai loại chính: Phương thức trả về giá trị và phương thức kiểu void.

Trong chương 4, tất cả các hàm đều có từ khoá static trước kiểu vì tất cả các hàm sử dụng ở đây đều thuộc các chương trình "không hướng đối tượng".

Trong chương này và cả các chương sau, thuật ngữ "phương thức" (method), có khi để cho dễ hiểu, như trong ngôn ngữ C/C++, ta có thể gọi là "hàm" vẫn không làm lệch nội dung.

4.1. PHƯƠNG THỨC TRẢ VỀ MỘT GIÁ TRỊ

Phương thức này có kiểu float, double, int, string... Bên trong phương thức luôn có lệnh:

```
return(giá_trị);
```

Định nghĩa phương thức trả về giá trị, ví dụ phương thức tính tổng bình phương của a, b:

```
static double Tinh(double a, double b)
{
return a*a + b*b;
}
```

Khi gọi phương thức, cần phải khai báo biến và gán tên phương thức cho biến này.

```
int m;
m = Tinh(a,b);
```

Ví dụ 4.1.

```
using System;
class Method1
{
static double Nhap()
```

```
{
double a;
Console.Write("Nhap mot so: ");
a = double.Parse(Console.ReadLine());
return a;
}
//-------------------------------------------------
static double Tinh(double a, double b)
{
return(a*a + b*b);
}
//-------------------------------------------------
static void Hienthi(double a, double b, double m)
//Ham kieu void chi co doi so "vao"
{
Console.WriteLine("a = "+a+"; b= "+b);
Console.WriteLine("m = a*a + b*b = "+m);
}
//-------------------------------------------------
static void Main(string[] args)
{
double a, b, m;
a = Nhap();//goi ham va gan cho bien a
b = Nhap();//goi ham va gan cho bien b
m = Tinh(a, b);//goi ham va gan cho bien moi
Hienthi(a, b, m);//ham kieu void khong co lenh gan
Console.ReadLine();
}
}
```

Kết quả:

Nhap mot so: 2

Nhap mot so: 4

a = 2; b = 4

m = a*a + b*b = 20

4.2. PHƯƠNG THỨC KIỂU VOID

Có người gọi phương thức kiểu void là phương thức không trả về giá trị. Tuy nhiên nếu phương thức có đối số là tham chiếu thì nó có thể trả về nhiều giá trị đồng thời. Vì vậy, ở đây ta vẫn để nguyên kiểu "void" mà không cần dịch ra tiếng Việt.

Có thể chia phương thức kiểu void thành 4 loại:

1 – Kiểu void không có đối số. Nó được dùng để hiển thị các thông báo.

```
void Method1() { }
```

2 – Kiểu void có đối số đưa vào nhưng không có đối số lấy ra. Nó thường dùng để hiển thị các thông báo, tính toán một biểu thức nào đó rồi hiển thị kết quả ra màn hình.

3 – Kiểu void không có đối số đưa vào nhưng có đối số lấy ra khỏi hàm. Nó thường dùng để nhập dữ liệu từ bàn phím và đưa ra khỏi hàm. Các đối số lấy ra phải là đối số kiểu tham chiếu như kiểu có từ khoá ref hay từ khoá out.

4 – Kiểu void có cả đối số vào và đối số ra. Các đối số lấy ra phải là đối số kiểu tham chiếu như kiểu có từ khoá ref hay từ khoá out.

Ví dụ 4.2. Các hàm sau đây không cần phải có đối số bởi vì ngay đầu class, trên tất cả các hàm ta đặt từ khoá static cho tất cả các biến a, b, m. Ba biến này gọi là biến toàn cục – global.

```
using System;
class Method2
{
//global variable,bien toan cuc
static int a, b, m;
static int Input()
{
```

```
int a;
a = Int32.Parse(Console.ReadLine());
return(a);
}
//-----------------------------------------------------
static int Cal()
{
m = a + b;
return m;
}
//-----------------------------------------------------
static void Show()
{
Console.WriteLine("a = "+a+"; b = "+b+");
Console.WriteLine("m = a + b = "+m);
}
//-----------------------------------------------------
static void Main()
{
Console.Write("Enter a = ");
a = Input();
Console.Write("Enter b = ");
b = Input();
m = Cal();
Show();
Console.ReadLine();
}
}
```

Kết quả:

Enter a = 4

Enter b = 5

a = 4; b = 5

a + b = 9

4.3. TRUYỀN ĐỐI SỐ KIỂU THAM CHIẾU CÓ TỪ KHOÁ REF HOẶC OUT

Khi muốn đưa ra khỏi hàm một tham số nào đó ta phải sử dụng từ khoá ref hoặc từ khoá out đặt trước kiểu. Để thấy được ý nghĩa của từ khoá ref hoặc out ta xét một ví dụ sau đây không sử dụng tham chiếu với từ khoá ref hoặc out.

Ví dụ 4.3.

```
using System;
class PassValue //truyen theo gia tri - Passing by value
{
static void Hoandoi(int a, int b) //a va b la gia tri
{
int temp;
temp = a;
a = b;
b = temp;
Console.Write("\nTrong ham Hoandoi: a = "+a+";b = "+b);
}
//-----------------------------------------------------
public static void Main()
{
int a = 10, b = 25;
Console.Write("\nTruoc khi hoan doi: a = "+a+"; b = "+b);
Hoandoi(a,b);
Console.Write("\nSau khi hoan doi: a = "+a+"; b = "+b);
Console.ReadLine();
}
}
```

Kết quả:

Truoc khi hoan doi: a = 10; b = 25

Trong ham Hoan doi: a = 25; b = 10

Sau khi hoan doi: a = 10; b = 25

Nhận xét: Nhìn vào kết quả ta thấy bên trong hàm hoán đổi, các giá trị của a và b được hoán đổi cho nhau. Tuy nhiên sau khi hoán đổi, giá trị của a và b lại như trước khi hoán đổi, nghĩa là a = 10 và b = 25. Điều này có nghĩa là hàm "Hoandoi" không có tác dụng hoán đổi.

Các đối số của hàm "Hoandoi" không sử dụng "tham chiếu" mà chỉ là các giá trị thông thường.

4.3.1. Từ khoá ref

Bất kỳ một đối số nào của hàm khi cần lấy ra khỏi hàm đều phải sử dụng kiểu tham chiếu có từ khoá ref đặt trước kiểu. Ví dụ định nghĩa hàm Input để nhập các số từ bàn phím:

```
static void Input(ref double a, ref double b)
{
Console.Write("Nhap a, b: ");
a = double.Parse(Console.ReadLine());
b = double.Parse(Console.ReadLine());
}
```

Điều cần nhớ là khi gọi hàm ta khai báo các biến và gán giá trị 0 hoặc giá trị cụ thể cho a, b. Đồng thời các đối số a, b phải có từ khoá ref đặt trước tên biến.

```
double a = 0;
double b = 0;
Input(ref a, ref b);
```

Ví dụ 4.4.

```
using System;
class PassRef //truyen theo tham chieu - Passing by reference
{
static void Hoandoi(ref int a,ref int b)
{
int temp;
temp = a;
a = b;
```

```
b = temp;
Console.Write("\nTrong ham Hoandoi a = "+a+"; b = "+b);
}
//-----------------------------------------------
public static void Main()
{
int a = 10, b = 25;
Console.WriteLine("\nTruoc khi hoan doi: = "+a+"; b = "+b);
Hoandoi(ref a, ref b);
Console.WriteLine("\nSau khi hoan doi: a = "+a+"; b = "+b);
Console.ReadLine();
}
}
```

Kết quả:

Truoc khi hoan doi: a = 10; b = 25

Trong ham Hoan doi: a = 25; b = 10

Sau khi hoan doi: a = 25; b = 10

Nhận xét: Nhìn vào kết quả ta thấy bên trong hàm hoán đổi, các giá trị của a và b được hoán đổi cho nhau. Mặt khác sau khi hoán đổi, giá trị của a và b cũng được hoán đổi, nghĩa là a = 25 và b = 10. Điều này có nghĩa là hàm "Hoandoi" đã thực sự có tác dụng hoán đổi.

Khi gọi hàm Hoandoi, các biến a, b có thể gán giá trị 0, cũng có thể gán giá trị khác 0 như ở trên gán a = 10 và b = 25.

Ví dụ sau đây sử dụng từ khoá ref cho hàm nhập số liệu từ bàn phím, khi gọi hàm ta phải gán giá trị bằng 0.

Ví dụ 4.5.

```
using System;
class RefVar
{
static void Input(ref double a,ref double b)
{
```

```
Console.Write("Enter a = ");
a = double.Parse(Console.ReadLine());
Console.Write("Enter b = ");
b = double.Parse(Console.ReadLine());
}
//-----------------------------------------------------
static void Calculate(double a, double b, ref double m)
{
m = a*a + b*b;
}
//-----------------------------------------------------
static void Show(double a, double b, double m)
{
Console.WriteLine("a = "+a+"; b = "+b+"\n m = a*a + b*b = "+m);
}
//-----------------------------------------------------
public static void Main()
{
double a = 0;
double b = 0;
double m = 0;
Input(ref a, ref b);
Calculate(a, b, ref m);
Show(a, b, m);
Console.ReadLine();
}
}
```

Kết quả:

Enter a = 2

Enter b = 5

a = 2; b = 5

m = a*a + b*b = 29

4.3.2. Từ khoá out

Từ khoá out được sử dụng như từ khoá ref chỉ có một điểm khác biệt là khi sử dụng từ khoá out ta chỉ cần khai báo biến tham chiếu mà không cần gán giá trị. Ta viết lại chương trình của ví dụ 4.5 với từ khoá out.

Ví dụ 4.6.

```
using System;
class OutVar
{
static void Input(out double a, out double b)
{
Console.Write("Enter a = ");
a = double.Parse(Console.ReadLine());
Console.Write("Enter b = ");
b = double.Parse(Console.ReadLine());
}
//-----------------------------------------------------
static void Calculate(double a, double b, out double m)
{
m = a*a + b*b;
}
//-----------------------------------------------------
static void Show(double a, double b, double m)
{
Console.WriteLine("a = "+a+"; b = "+b);
Console.WriteLine(" m = a*a + b*b = "+m);
}
//-----------------------------------------------------
public static void Main()
{
double a, b, m; //Khong can gan gia tri
Input(out a, out b);
Calculate(a, b, out m);
```

```
    Show(a, b, m);
    Console.ReadLine();
    }
    }
```

 *Kết quả:*

Enter a = 2

Enter b = 5

a = 2; b = 5

m = a*a + b*b = 29

Chương 5. Lớp trong lập trình hướng đối tượng C# (class)

Từ chương này trở đi, các chương trình được viết theo phương pháp lập trình hướng đối tượng. Lập trình hướng đối tượng (OOP: Object Oriented Programming) là một phương pháp dựa vào khái niệm về lớp và đối tượng. Lớp là một kiểu mới. Lớp chứa nhiều đối tượng. Ví dụ một lớp ôtô có nhiều đối tượng là các loại ôtô khác nhau. Loại ôtô đó là một đối tượng. Giống như một sinh viên là một đối tượng của một lớp học. Một lớp bao gồm: thuộc tính (hay dữ liệu), các thao tác (hay phương thức). Một đối tượng là ôtô sẽ có các thuộc tính như số cánh cửa, số bánh xe, số ghế ngồi,... Mặt khác ôtô phải có các thao tác như khởi động, chạy, tăng tốc, giảm tốc, dừng,... Đó chính là các phương thức của lớp ôtô.

Thuộc tính của lớp có thể khai báo theo các kiểu:

– private: các thuộc tính khai báo kiểu private chỉ có hiệu lực ngay trong một lớp.

– public: có hiệu lực cho cả các lớp khác. Các phương thức đa phần được khai báo kiểu public.

– protected: được sử dụng cho các biến thuộc lớp dẫn xuất (lớp kế thừa).

1. Định nghĩa một lớp:

```
class Student
{
private int ID;
private string name;
public Student()
{ }
public void Input();
public void Show();
```

```
}
```

2. Khai báo đối tượng (s1) thuộc lớp Student:

```
Student s1 = new Student();
```

3. Truy cập đối tượng s1 vào các hàm thành viên:

```
s1.Input();
s1.Show();
```

Các hàm định nghĩa bên trong lớp còn được gọi là hàm thành viên. Các hàm này không có từ khoá static mà chỉ có từ khoá public, ví dụ:

```
public void show()
{ }
```

Ví dụ 5.1.

```
using System;
class person
{
private string name;
private int age;
private double salary;
//-----------------------------------------------------
public person() //ham tao -constructor
{
name = "";
age = 0;
salary = 0;
}
//-----------------------------------------------------
public void input()
{
Console.Write("Enter name: ");
name = Console.ReadLine();
```

```
    Console.Write("Enter age: ");
    age = Int32.Parse(Console.ReadLine());
    Console.Write("Enter salary: ");
    salary = Double.Parse(Console.ReadLine());
    }
    //-------------------------------------------------
    public void show()
    {
    Console.WriteLine("name: "+name+"; age: "+age+"; salary:
"+salary);
    }
    //-------------------------------------------------
    public static void Main()
    {
    person p = new person();
    p.input();
    p.show();
    Console.ReadLine();
    }
    }
```

 *Kết quả:*

Enter name: Mai

Enter age: 25

Enter salary: 1000

name: Mai; age: 25; salary: 1000

5.1. HÀM TẠO (CONSTRUCTOR METHOD)

Khi khai báo một biến, ta có thể gán giá trị cho biến. Ví dụ:

```
int a, b = 5;
```

Hàm tạo là một hàm đặc biệt được dùng để khởi tạo dữ liệu cho đối

tượng. Đặc điểm của hàm tạo:

Hàm tạo có tên trùng với tên lớp. Ví dụ 5.1 ở trên ta có hàm tạo:

```
public person()
{
name = " ";
age = 0;
salary = 0;
}
```

Hàm tạo không có lệnh return(giatri);

Khi khai báo hàm tạo trong hàm Main() phải sử dụng toán tử new.

Hàm tạo có hai loại: Hàm tạo không có đối số và hàm tạo có đối số.

Dùng hàm tạo có đối số, khi khai báo đối tượng ta gán dữ liệu cho đối tượng đó.

Ví dụ 5.2.

```
using System;
class person
{
private string name;
private int age;
private double salary;
//-----------------------------------------------------
public person()
{
name = "";
age = 0;
salary = 0;
}
//-----------------------------------------------------
public Person(string name, int age, double salary)
```

```
    {
    this.name = name;
    this.age = age;
    this.salary = salary;
    }
    //-------------------------------------------------
    public void show()
    {
    Console.WriteLine("name: "+name+"; age: "+age+"; salary:
"+salary);
    }
    //-------------------------------------------------
    public static void Main()
    {
    person p = new person("Mai",25,1000);
    p.show();
    Console.ReadLine();
    }
    }
```

Kết quả:

name: Mai; age = 25; salary = 1000

Trong ví dụ này ở hàm Main() đối tượng p được khai báo kiểu lớp person với các dữ liệu gán lúc khai báo name: "Mai", age: 25, salary: 1000.

Hiển thị dữ liệu chỉ cần lấy đối tượng p truy cập hàm show() qua toán tử chấm: p.show();

Khi định nghĩa hàm tạo có đối số, ta dùng toán tử "this" nếu tên các đối số trùng tương ứng với tên các biến private ở bên trong lớp. Nếu tên các đối số này khác đi, ta không cần dùng toán tử "this". Hàm tạo sau đây có đối số, không dùng toán tử this hoàn toàn tương đương với hàm tạo đã

có ở trên ví dụ 5.2.

```
public person(string name1, int age1, double sal1)
{
name = name1;
age = age1;
salary = sal1;
}
```

5.2. KIỂU PROPERTY – THUỘC TÍNH

Đây là một kiểu đặc biệt chỉ có trong C#. Thuộc tính (properties) là thành viên của lớp. Nó có tên và kiểu private static. Khi sử dụng thuộc tính ta có thể gán giá trị dữ liệu qua việc truy cập tên đối tượng vào tên thuộc tính nhờ toán tử chấm.

Ví dụ 5.3.

```
using System;
class Property1
{
private int ID; //ma so
private string name;
public int ID1
{
get{ return ID;}
set{ ID = value;}
}
public string name1
{
get{ return name;}
set{ name = value;}
}
public void show()
```

```
{
Console.WriteLine("ID = "+ID+"; Name: "+name);
}
static void Main(string[] args)
{
Property1 p1 = new Property1();
  p1.ID1 = 555;
  p1.name1 = "Hoa";
Console.Write("ID: "+p1.ID1);
Console.WriteLine("; Name: "+p1.name1); //p1.show();
Console.ReadLine();
}
}
```

Kết quả:

```
ID: 555; Name: Hoa
```

Trong ví dụ trên ta định nghĩa hai thuộc tính: ID1 và name1. Ở đây ta lấy tên thuộc tính ID1 và name1 chỉ khác các biến của lớp bởi con số 1, các bạn cũng có thể sử dụng tên bất kỳ không có ảnh hưởng gì.

```
public int ID1 //luôn có từ khoá public,kiểu trùng với
               // kiểu của ID, đó là kiểu int
{
get{return ID;} //chỉ một lệnh return ID
set{ID = value;} //chỉ một lệnh ID = value
}
```

Trong hàm Main(), ta có phép gán:

```
p1.ID1 = 555; //đối tượng p1 qua toán tử chấm
              //đến thuộc tính ID1 được gán bởi dấu =
```

Mặt khác ta có thể sử dụng lệnh in trực tiếp Console.WriteLine như ở trên hoặc qua hàm show().

5.3. MẢNG

5.3.1. Khai báo mảng

Mảng là một kiểu dữ liệu gồm các phần tử có cùng một kiểu như int. double, string, char, kiểu của một lớp được định nghĩa.

+ Mảng một chiều:

```
int [] a = new int[ 10] ;
double [] da = new da[ 4]{ 1.5,2.4,3.7,4.1} ; //Gán giá trị
```

+ Mảng hai chiều:

```
long [ ,] = new long[ 4,4] ;
```

Khi xử lý với mảng ta sử dụng vòng lặp for.

5.3.2. Mảng một chiều

Mảng một chiều thường dùng để tính toán các vector và mảng các ký tự.

Ví dụ 5.4. Nhập vào 2 mảng một chiều (2 vector) a[n], b[n] với kích thước n nhập từ bàn phím. Tính tổng 2 mảng c = a + b. Hiển thị a, b, c.

```
using System;
class Arr1d
{
public void Input(int [] a, int n)
{
for (int i = 0; i < n; i++)
a[ i] = Int32.Parse(Console.ReadLine());
}
//-------------------------------------------------
public void Sum(int [] a, int [] b, int [] c, int n)
{
for(int i = 0; i < n; i++)
c[ i] = a[ i] + b[ i] ;
}
```

```
//-----------------------------------------------------
public void Show(int [] a, int n)
{
for(int i = 0; i < n; i++)
Console.Write("  "+a[ i] );
}
//-----------------------------------------------------
static void Main(string[] args)
{
Arr1d ob = new Arr1d(); //khai bao doi tuong ob
int n;
Console.Write("Nhap n = ");
n = Int32.Parse(Console.ReadLine());
int [] a = new int[ n] ;
int [] b = new int[ n] ;
int [] c = new int[ n] ;
Console.WriteLine("Nhap cac phan tu mang a.");
ob.Input(a, n);
Console.WriteLine("Nhap cac phan tu mang b.");
ob.Input(b, n);
ob.Sum(a, b, c, n);
Console.WriteLine("\nHien thi cac mang a, b, c.");
Console.WriteLine("\nMang a.");
ob.Show(a, n);
Console.WriteLine("\nMang b.");
ob.Show(b, n);
Console.WriteLine("\nMang c = a + b.");
ob.Show(c, n);
Console.ReadLine();
}
}
```

Kết quả:

```
Nhap n = 3
Nhap cac phan tu mang a.
1
2
3
Nhap cac phan tu mang b.
4
5
6
Hien thi cac mang a, b, c.
Mang a
1 2 3
Mang b
4 5 6
Mang c = a + b.
5 7 9
```

Trong chương trình trên, các hàm thành viên Input, Sum, Show có đối số là mảng, kích thước n. Chúng đều có từ khoá public và không có từ khoá static.

5.3.3. Mảng đối tượng

Mảng đối tượng cũng giống như mảng một chiều thông thường.

```
Tên_lớp đối_tượng = new Tên_lớp();
```

Các thao tác với mảng đối tượng thực hiện bởi vòng lặp for. Điều cần lưu ý là trong vòng lặp for ta phải khởi gán 0 cho đối tượng trước khi truy cập vào hàm input() trong ví dụ sau. Nếu không có phép gán `p[ i] = new person();` thì mảng đối tượng sẽ không hoạt động.

```
for(int i = 0; i < n; i++)
```

```
{
p[ i]  = new person();
p[ i] .input();
}
```

Ví dụ 5.5. Viết chương trình hướng đối tượng để nhập dữ liệu cho một mảng đối tượng gồm n người (n nhập từ bàn phím).

```
using System;
class person
{
private string name;
private int age;
private double salary;
//-----------------------------------------------------
public person()
{
name = "";
age = 0;
salary = 0;
}
//-----------------------------------------------------
public void input()
{
Console.Write("Nhap ten: ");
name = Console.ReadLine();
Console.Write("Nhap tuoi: ");
age = Int32.Parse(Console.ReadLine());
Console.Write("Nhap luong: ");
salary = Double.Parse(Console.ReadLine());
}
//-----------------------------------------------------
```

```
    public void show()
    {
    Console.WriteLine("Ten: "+ name+"; Tuoi:"+age+"; Luong:
"+salary);
    }
    //-----------------------------------------------------
    public static void Main()
    {
    int n;
    Console.Write("Nhap so nguoi n = ");
    n = Int32.Parse(Console.ReadLine());
    person [ ]p = new person[ n] ;
    {
    for(int i = 0; i < n; i++)
    p[ i] = new person();//khoi gan doi tuong
    p[ i] .input();//mang doi tuong
    }
    for(int i = 0; i < n; i++)
    p[ i] .show();
    Console.ReadLine();
    }
   }
```

Kết quả:

Nhap so nguoi n = 2

Nhap ten: Anh

Nhap tuoi: 25

Nhap luong: 555

Nhap ten: Hung

Nhap tuoi: 30

Nhap luong: 999

Ten: Anh; Tuoi: 25; Luong: 555

Ten: Hung; Tuoi: 30; Luong: 999

5.3.4. Mảng hai chiều

Mảng hai chiều thường dùng để tính toán với ma trận. Với mảng hai chiều ta sử dụng hai vòng lặp for.

Ví dụ 5.6. Tính tổng và tích hai ma trận a, b nhập từ bàn phím.

```
using System;
class Matran1
{
public void Input(int [ ,] a, int m, int n)
{
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
a[ i,j] = Int32.Parse(Console.ReadLine());
}
//-----------------------------------------------------
public void Add(int [,] a, int [ ,] b, int [ ,] c, int m, int n)
{
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
c[ i,j] = a[ i,j] + b[ i,j];
}
//-----------------------------------------------------
public void Mul(int[ ,] a, int[ ,] b, int[ ,] d, int m, int n)
{
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
{
d[ i,j] = 0;
```

```
for(int k = 0; k < m; k++)
d[ i,j] + = a[ i,k] * b[ k,j] ;
}
}
//-----------------------------------------------------
public void Show(int [ ,] a, int m, int n)
{
for (int i = 0; i < m; i++)
{
for(int j = 0; j < n; j++)
Console.Write("  "+a[ i,j] );
Console.WriteLine();
}
}
static void Main(string[ ] args)
{
Matran1 mt = new Matran1();
int m, n;
Console.Write("Nhap kich thuoc ma tran m, n: ");
m = Int32.Parse(Console.ReadLine());
n = Int32.Parse(Console.ReadLine());
int [ ,] a = new int[ m, n] ;
int [ ,] b = new int[ m, n] ;
int [ ,] c = new int[ m, n] ;
int [ ,] d = new int[ m, n] ;
Console.WriteLine("Nhap ma tran A.");
mt.Input(a, m, n);
Console.WriteLine("Nhap ma tran B.");
mt.Input(b, m, n);
mt.Add(a, b, c, m, n);
```

```
mt.Mul(a, b, d, m, n);
Console.WriteLine("Ma tran A.");
mt.Show(a, m, n);
Console.WriteLine("Ma tran B.");
mt.Show(b, m, n);
Console.WriteLine("Ma tran C = A + B.");
mt.Show(c, m, n);
Console.WriteLine("Ma tran D = A * B.");
mt.Show(d, m, n);
Console.ReadLine();
}
}
```

 *Kết quả:*

```
Nhap kich thuoc ma tran m, n: 2
2
Nhap ma tran A.
1
2
3
4
Nhap ma tran B.
5
6
7
8
Ma tran A.
1 2
3 4
```

Ma tran B.

5 6

7 8

Ma tran C = A + B.

6 8

10 12

Ma tran D = A * B.

19 22

43 50

Trong ví dụ 5.7 sau đây, ta khai báo các ma trận có kiểu private với chiều 5×5. Các biến m, n cũng có kiểu private và được nhập từ bàn phím ở trong hàm constructor. Các hàm Input (int [,]a, int m, int n) và hàm Input () có tên như nhau nhưng có đối số khác nhau. Hai hàm này ta gọi là hàm trùng tên (hoặc còn gọi là chồng hàm – overloading method). Cũng tương tự hai hàm Show (int [,]a, int m, int n) và hàm Show().

Ví dụ 5.7.

```
using System;
class Matran2
{
private int m, n;
private int [ ,] a = new int[ 5,5] ;
private int [ ,] b = new int[ 5,5] ;
private int [ ,] c = new int[ 5,5] ;
private int [ ,] d = new int[ 5,5] ;
//-------------------------------------------------
public Matran2()
{
Console.Write("Nhap kich thuoc ma tran m, n: ");
m = Int32.Parse(Console.ReadLine());
```

```
n = Int32.Parse(Console.ReadLine());
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
a[ i,j] = 0;
}
//-----------------------------------------------------
public void Input(int [ ,] a, int m, int n)
{
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
a[ i,j] = Int32.Parse(Console.ReadLine());
}
//-----------------------------------------------------
public void Input()
{
Console.WriteLine("Nhap ma tran A.");
Input(a, m, n);
Console.WriteLine("Nhap ma tran B.");
Input(b, m, n);
}
//-----------------------------------------------------
public void Add(int[ ,] a, int[ ,] b, int[ ,] c, int m, int n)
{
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
c[ i,j] = a[ i,j] + b[ i,j];
}
//-----------------------------------------------------
public void Mul(int[ ,] a, int[ ,] b, int[ ,] d, int m, int n)
{
```

```
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
{
d[ i,j] = 0;
for(int k = 0; k < m; k++)
d[ i,j] + = a[ i,k] * b[ k,j] ;
}
}
//-----------------------------------------------------
public void Show(int[ ,] a, int m, int n)
{
for(int i = 0; i < m; i++)
{
for(int j = 0; j < n; j++)
Console.Write("  "+a[ i,j] );
Console.WriteLine();
}
}
//-----------------------------------------------------
public void Show()
{
Add(a, b, c, m, n);
Mul(a, b, d, m, n);
Console.WriteLine("Ma tran A.");
Show(a, m, n);
Console.WriteLine("Ma tran B.");
Show(b, m, n);
Console.WriteLine("Ma tran C = A + B.");
Show(c, m, n);
Console.WriteLine("Ma tran D = A * B.");
```

```
    Show(d, m, n);
    }
    //-----------------------------------------------------
    static void Main(string[] args)
    {
    Matran2 mt = new Matran2();
    mt.Input();
    mt.Show();
    Console.ReadLine();
    }
    }
```

Kết quả:

```
Nhap kich thuoc ma tran m, n: 2
2
Nhap ma tran A.
1
2
3
4
Nhap ma tran B.
5
6
7
8
Ma tran A.
1 2
3 4
Ma tran B.
```

```
5 6
7 8
Ma tran C = A + B.
6 8
10 12
Ma tran D = A * B.
19 22
43 50
```

5.3.5. Từ khoá params

Sử dụng từ khoá params trước khai báo đối số mảng sẽ cho phép ta không cần xác định số lượng đối số mảng (kích thước mảng cụ thể).

Khi xem kết quả của chương trình đã chạy, việc gán giá trị mảng với kiểu int là bất kỳ. Còn khi khai báo kiểu mảng là object thì trong một mảng, số lượng các phần tử không cần xác định trước, còn các phần tử trong mảng đó cũng có kiểu bất kỳ (có thể là int, string, char, double,...). Đây là một kiểu khai báo khá lý thú của C# (trong C++ không thể có!).

Ví dụ 5.8.

```
using System;
class Params1
{
public void Method1(params int [] a)
{
for(int i = 0; i < a.Length; i++)
Console.Write(" "+a[ i] );
Console.WriteLine();
}
//-----------------------------------------------------
public void Method2(params object [] a)
{
```

```
for(int i = 0; i < a.Length; i++)
Console.Write(" "+(object)a[ i] );
Console.WriteLine();
}
//-----------------------------------------------------
static void Main(string[ ] args)
{
Params1 arr = new Params1();
arr.Method1(5,7,9);
arr.Method2(100, "Ha noi", 'M', 1.56);
int [ ] a = new int[ 4]{ 2, 4, 6, 8} ;
arr.Method1(a);
Console.ReadLine();
}
}
```

 *Kết quả:*

5 7 9

100 Ha noi M 1.56

2 4 6 8

5.3.6 Chỉ mục Indexer

Indexer cho phép các thực thể của lớp được đánh số chỉ mục giống như kiểu mảng một chiều. Indexer cũng giống property ngoại trừ một điểm là các accessor (phần tử truy cập) có tham số.

Cú pháp của Indexer:

```
class Myclass
{
public object this[ int idx]
{
get
```

```
{
// return du lieu
}
set
{
// thiet lap du lieu
}
}
//...
}
```

Ví dụ 5.9.

```
using System;
class Indexer1
{
private int x, y, z;
public int this[ int idx]
{
get
{
switch(idx)
{
case 0: return x;
case 1: return y;
case 2: return z;
default:
throw new IndexOutOfRangeException();
}
}
set
{
```

```
switch(idx)
{
case 0: x = value; break;
case 1: y = value; break;
case 2: z = value; break;
}
}
}
//-----------------------------------------------------
public override string ToString()
{
return "P("+x+", "+y+", "+z+")";
}
//-----------------------------------------------------
static void Main(string[] args)
{
Indexer1 p = new Indexer1();
p[0] = 5;
p[1] = 8;
p[2] = 9;
Console.WriteLine("Toa do diem P(x, y, z).");
Console.WriteLine("P({0}, {1}, {2})",p[0],p[1],p[2]);
Indexer1 p2 = new Indexer1();
for(int i = 0; i < 3; i++)
p2[i] = i * i;
Console.WriteLine(p2.ToString());
Console.ReadLine();
}
}
```

Kết quả:

Toa do diem P(x, y, z).

P (7, 8, 9)

P (0, 1, 4)

Sau đây là ví dụ về Indexer với chức năng mảng chuỗi.

Ví dụ 5.10.

```
using System;
class Indexer2
{
public class ListTest
{
private string[ ] s1;
.rivate int i;
public ListTest(params string[ ] str)
{
s1 = new String[ 256] ;
foreach(string s in str)
{
s1[ i++] = s;
}
}
//-----------------------------------------------------
public void Add(string str)
{
if(i >= s1.Length)
{}
else s1[ i++] = str;
}
//-----------------------------------------------------
public string this[ int idx]
{
```

```
get
{
if(idx < 0 || idx > s1.Length)
{ }
return s1[ idx] ;
}
set
{
if(idx >= i)
{ }
else
s1[ idx]  = value;
}
}
public int GetnumEntries()
{
return i;
}
}
//-----------------------------------------------------
public class Tester
{
static void Main(string[ ] args)
{
ListTest lob = new
ListTest("Hello","World");//chi so 0, chi so 1
lob.Add("Ha"); //chi so 2
lob.Add("noi"); //chi so 3
lob.Add("Viet"); //chi so 4
lob.Add("nam"); //chi so 5 lob.Add("==");
```

```
string s3 = "Thu do";
lob[1] = s3; //Se the chi so 1 bang "thu do" cho "world"
for(int j = 0; j < lob.GetnumEntries(); j++)
{
Console.WriteLine("lob[{0}]:{1}",j,lob[j]);
}
Console.ReadLine();
}
}
}
```

 *Kết quả:*

lob[0] = Hello

lob[1] = Thu do

lob[2] = Ha

lob[3] = noi

lob[4] = Viet

lob[5] = nam

 *Ví dụ 5.11.* Sử dụng mảng thư viện ArrayList (cần phải khai báo using System.Collections).

```
using System;
using System.Collections; //ArrayList
class Indexer3
{
private string name;
private object data;
public string name1
  {
        get{ return name;}
        set{ name = value;}
```

```
    }
  public object data1
    {
          get{ return data;}
          set{ data = value;}
    }
  public Indexer3(string name, object data)
    {
          this.name = name;
          this.data = data;
    }
  }
  //-----------------------------------------------------
  class DataRow
  {
    ArrayList row;
    public DataRow()
    {
          row = new ArrayList();
    }
    public void Load()
    {
          row.Add(new Indexer3("ID",555));
          row.Add(new Indexer3("Name","Fred"));
          row.Add(new Indexer3("Salary",777));
   }
    public object this[ int column]
    {
          get{ return(row[ column] );}
          set{ row[ column]  = value;}
```

```
    }
}
class MainIndexer
{
static void Main(string[] args)
{
DataRow row = new DataRow();
row.Load();
Indexer3 val0 = (Indexer3)row[0];
Indexer3 val1 = (Indexer3)row[1];
Indexer3 val2 = (Indexer3)row[2];
Console.WriteLine("Column 0:{0}", val0.data1);
Console.WriteLine("Column 1:{0}", val1.data1);
Console.WriteLine("Column 2:{0}", val2.data1);
// Thay doi du lieu
val0.data1 = 6666;
val1.data1 = "Alice";
val2.data1 = 8888;
Console.WriteLine("\nColumn 0:{0}", val0.data1);
Console.WriteLine("Column 1:{0}", val1.data1);
Console.WriteLine("Column 2:{0}", val2.data1);
Console.ReadLine();
}
}
```

Kết quả:

column 0 = 555

column 1 = Fred

column 2 = 777

column 0 = 6666

column 1 = Fred

column 2 = 8888

Ví dụ 5.12.

```
using System;
using System.Collections; //ArrayList
class Employee
{
private string name;
private string ID;
public Employee()
{
name = "";
ID = "";
}
public Employee(string name1, string ID1)
{
name = name1;
ID = ID1;
}
public string name1
{
get{ return name;}
set{ name = value;}
}
public string ID1
{
get{ return ID;}
set{ ID = value;}
}
}
```

```
//-----------------------------------------------------
class Employ1
{
private ArrayList row;
private int m_MaxEmps;
public Employ1(int n)
{
m_MaxEmps = n;
row = new ArrayList(n);
}
//indexer
public Employee this[ int idx]
{
get
{
if(idx < 0 || idx > row.Count - 1)
return(null);
return (Employee)row[ idx] ;
}
set
{
if(idx < 0 || idx > m_MaxEmps - 1)
return;
row.Insert(idx, value);
}
}
//-----------------------------------------------------
public Employee this[ string ID1]
{
get
```

```
{
Employee em_return = null;
foreach(Employee emp in row)
{
if(emp.ID1 == ID1)
{
em_return = emp;
break;
}
}
return em_return;
}
}
//--------------------------------------------------
public int Length
{
get{ return row.Count;}
}
}
//--------------------------------------------------
class MainIndexer
{
static void Main(string[] args)
{
try
{
Employ1 empl = new Employ1(4);
empl[ 0] = new Employee("An","111");
empl[ 1] = new Employee("Hoa","222");
empl[ 2] = new Employee("Binh","333");
```

```
empl[ 3] = new Employee("Mai","444");
for(int i = 0; i < empl.Length; i++)
{
string name = empl[ i] .name1;
string ID = empl[ i] .ID1;
Console.WriteLine("Name:{ 0} ,ID:{ 1}",name,ID);
}
// can tim employee voi ID = 333
Employee e11 = empl["333"];
if(e11 != null)
{
string name = e11.name1;
string ID = e11.ID1;
Console.WriteLine("Da tim thay Ten :{ 0} , ID:{ 1}", name, ID);
}
else
{
Console.WriteLine("Khong tim thay ID = 333")
}
}
catch(Exception e)
{
Console.WriteLine(e.Message);
}
Console.ReadLine();
}
}
```

Kết quả:

Name An; ID: 111

Name Hoa; ID: 222

Name Binh; ID: 333

Name Mai; ID: 444

Da tim thay Ten: Binh, ID: 333

5.4. KHÔNG GIAN TÊN (NAMESPACE)

Không gian tên là một kiểu lớp tổng quát. Nó cho phép ta định nghĩa các không gian tên khác nhau nhưng lại có các lớp giống nhau, các dữ liệu giống nhau và các phương thức giống nhau.

Tuy nhiên vì các đối tượng khác nhau nên từng đối tượng lại truy cập với dữ liệu riêng của mình.

Ví dụ 5.13. Chương trình sau đây sẽ định nghĩa hai không gian tên khác nhau đó là: Student và Teacher, nhưng đều có chung một lớp là Manage. Trong lớp Manage này có các biến thành viên giống nhau, các hàm thành viên giống nhau, nhưng có các dữ liệu cụ thể khác nhau.

```
using System;
namespace Student
{
class Manage
{
private string name;
private int ID;
private double salary;
public Manage()
{
Init("", 0, 0)
}
public Manage(string name1, int ID1, double sal1)
{
Init(name1, ID1, sal1);
}
```

```
    public void Init(string name1, int ID1, double sal1)
    {
    name = name1;
    ID = ID1;
    salary = sal1;
    }
    public void Show()
    {
    Console.WriteLine("Student namespace.");
    Console.WriteLine("Name: "+name+"; ID: "+ID+"; Salary:
"+salary);
    }
    }
    }
    //-------------------------------------------------
    namespace Teacher
    {
    class Manage
    {
    private string name;
    private int ID;
    private double salary;
    public Manage()
    {
    Init("", 0, 0);
    }
    public Manage(string name1, int ID1, double sal1)
    {
    Init(name1, ID1, sal1);
    }
    public void Init(string name1, int ID1, double sal1)
```

```
    {
    name = name1;
    ID = ID1;
    salary = sal1;
    }
    public void Show()
    {
    Console.WriteLine("Teacher namespace.");
    Console.WriteLine("Name: "+name+"; ID: "+ID+"; Salary:
"+salary);
    }
    }
    }
    //-----------------------------------------------------
    class NSMain
    {
    static void Main(string[] args)
    {
    Student.Manage std = new Student.Manage("An", 22, 1000);
    std.Show();
    Teacher.Manage tch = new Teacher.Manage("Mai", 20, 800);
    tch.Show();
    Console.ReadLine();
    }
    }
```

Kết quả:

Student namespace.

Name: An; ID: 22; salary: 1000

Teacher namespace.

Name: Mai; ID: 20; salary: 800

Chương 6. Tính thừa kế trong C# (inheritance)

Là một ngôn ngữ lập trình hướng đối tượng, C# có đặc điểm về thừa kế. Thừa kế là một trong các đặc điểm của hướng đối tượng. Nhờ sử dụng khái niệm thừa kế mà ta có thể tạo ra các lớp mới từ lớp cơ sở đã có. Khi sử dụng thừa kế, ta có thể tái sử dụng mã chương trình, tạo ra một giao diện chung mà các đối tượng thừa kế giao diện này có thể truy cập. Lớp mới thừa kế từ lớp cơ sở có thể thừa kế các thuộc tính (dữ liệu), thừa kế các phương thức của lớp cơ sở (base class).

Lớp thừa kế còn gọi là lớp dẫn xuất (derived class).

Có hai cách để viết chương trình có thừa kế.

6.1. SỬ DỤNG TỪ KHOÁ NEW

Cú pháp của các lớp thừa kế có sử dụng từ khoá new:

```
class Base1 //lop co so
{
private type data1;
private type data2;
public void Method1()
{ }
public void Method2()
{ }
}
//-----------------------------------------------------
class Derived1: Base1 //lop dan xuat 1
{
private type data3; //du lieu moi cua lop Derived1
public new void Method1()
{ }
public new void Method2()
{ }
```

```
}
//-------------------------------------------------
class Derived2: Base1 //lop dan xuat 2
{
private type data4; //du lieu moi cua lop Derived2
public new void Method1()
{ }
public new void Method2()
{ }
}
```

Ví dụ 6.1. Hãy viết chương trình có thừa kế để quản lý các lớp thuộc lớp cơ sở Employee (nhân viên) là: lớp các nhà khoa học (Scientist), lớp các nhà quản lý (Manager) và lớp người công nhân (Worker).

```
using System;
class Employee
{
private int ID;
private string name;
public Employee()
{
ID = 0;
name = "";
}
public void Input()
{
Console.Write("Nhap ID: ");
ID = Int32.Parse(Console.ReadLine());
Console.Write("Nhap Ten: ");
name = Console.ReadLine();
}
public void Show()
{
```

```
Console.Write("ID: "+ID);
Console.WriteLine("; Ten: "+name);
}
}
//-----------------------------------------------------
class Scientist: Employee
{
private int pub;
public Scientist(): base()
{
pub = 0;
}
public new void Input()
{
base.Input();
Console.Write("Nhap so cong trinh khoa hoc: ");
pub = Int32.Parse(Console.ReadLine());
}
public new void Show()
{
base.Show();
Console.WriteLine("; Cong trinh khoa hoc: "+pub);
}
}
//-----------------------------------------------------
class Manager: Employee
{
private string contract;
public Manager(): base()
{
contract = "";
}
```

```
public new void Input()
{
base.Input();
Console.Write("Nhap noi dung hop dong: ");
contract = Console.ReadLine();
}
public new void Show()
{
base.Show();
Console.WriteLine("; Hop dong: "+contract);
}
}
//-----------------------------------------------------
class Worker: Employee //Lop nay de trong
{ }
//-----------------------------------------------------
public class InherMain
{
static void Main(string[] args)
{
Scientist s1 = new Scientist();
Console.WriteLine("Nha khoa hoc.");
s1.Input();
s1.Show();
Console.WriteLine("Nha quan ly.");
Manager m1 = new Manager();
m1.Input();
m1.Show();
Worker w1 = new Worker();
Console.WriteLine("Nguoi cong nhan.");
w1.Input();
w1.Show();
```

```
Console.ReadLine();
}
}
```

Kết quả:

Nha khoa hoc.
Nhap ID: 111
Nhap ten: AAA
ID: 111; Ten: AAA

Nha quan ly.
Nhap ID: 222
Nhap ten: BBB
Cong trinh khoa hoc: 5
ID: 222; Ten: BBB; Cong trinh khoa hoc: 5

Nguoi cong nhan.
Nhap ID: 333
Nhap ten: CCC
ID: 333; Ten: CCC

– Trong ví dụ trên, khi định nghĩa lớp thừa kế ta dùng dấu hai chấm: để tách giữa hai lớp dẫn xuất và lớp cơ sở.

```
class Manager: Employee
{ }
```

– Đối với hàm tạo (constructor) ở lớp thừa kế ta phải nhớ có hàm với từ khoá base() nằm sau dấu hai chấm:

```
public Manager(): base()
{
Contract = "";
}
```

– Các hàm thành viên, chẳng hạn, hàm Input(), hàm Show() trong cả 3 lớp đều có tên giống nhau. Các hàm này được gọi là hàm trùng tên hay

nạp chồng hàm (overloading function). Việc gọi hàm thành viên đã có ở lớp cơ sở, ta phải dùng từ khoá base truy cập qua toán tử chấm.

```
base.Input();
```

– Khi khai báo đối tượng trong hàm Main(), ta không hề quan tâm tới lớp cơ sở. Đối tượng nào thuộc lớp dẫn xuất nào sẽ được khai báo kiểu của lớp dẫn xuất đó.

```
Scientist s1 = new Scientist();
Manager m1 = new Manager();
Worker w1 = new Worker();
```

Khi định nghĩa các hàm thành viên trong lớp cơ sở, phải luôn nhớ đặt từ khoá new sau từ khoá public và trước từ khoá void.

6.2. SỬ DỤNG TỪ KHOÁ VIRTUAL VÀ OVERRIDE

– Các biến thuộc lớp cơ sở được khai báo theo kiểu private (hoặc protected). Các biến thuộc lớp dẫn xuất được khai báo theo kiểu private.

– Các hàm thành viên thuộc lớp cơ sở phải có từ khoá virtual trước từ khoá void.

– Các hàm thành viên thuộc lớp dẫn xuất phải có từ khoá override trước từ khoá void.

Cú pháp của nó như sau:

```
class Base1
{
private int a, b;
public virtual void Method1()
{ }
}
class Derived1: Base1
{
private int c;
public override void Show()
{ }
}
```

Ví dụ 6.2. Viết lại chương trình 6.1 theo cách sử dụng từ khoá virtual và override.

```
using System;
class Employee
{
private int ID;
private string name;
public Employee()
{
ID = 0;
name = "";
}
public virtual void Input()
{
Console.Write("Nhap ID: ");
ID = Int32.Parse(Console.ReadLine());
Console.Write("Nhap Ten: ");
name = Console.ReadLine();
}
public virtual void Show()
{
Console.Write("ID: "+ID);
Console.WriteLine("; Ten: "+name);
}
}
//-------------------------------------------------
class Scientist: Employee
{
private int pub;
public Scientist(): base()
{
pub = 0;
```

```
}
public override void Input()
{
base.Input();
Console.Write("Nhap so cong trinh khoa hoc: ");
pub = Int32.Parse(Console.ReadLine());
}
public override void Show()
{
base.Show();
Console.WriteLine("; Cong trinh khoa hoc: "+pub);
}
}
//-------------------------------------------------------
class Manager: Employee
{
private string contract;
public Manager(): base()
{
contract = "";
}
public override void Input()
{
base.Input();
Console.Write("Nhap noi dung hop dong: ");
contract = Console.ReadLine();
}
public override void Show()
{
base.Show();
Console.WriteLine("Hop dong: "+contract);
}
```

```
}
class Worker: Employee
{ }
//-------------------------------------------------------
public class InherMain
{
static void Main(string[] args)
{
Scientist s1 = new Scientist();
s1.Input();
s1.Show();
Manager m1 = new Manager();
m1.Input();
m1.Show();
Worker w1 = new Worker();
w1.Input();
w1.Show();
Console.ReadLine();
}
}
```

Kết quả của chương trình 6.2 hoàn toàn giống với kết quả của chương trình 6.1.

Ví dụ 6.3. Viết chương trình có thừa kế để tạo các cửa sổ khác nhau khi ta đã định nghĩa cửa sổ Window chính. Có hai lớp Listbox và Button thừa kế lớp cơ sở Window này. Ở đây ta sử dụng hai từ khoá virtual và override để viết thừa kế. Trong hàm Main() ta sử dụng mảng đối tượng để gán giá trị và hiển thị.

```
using System;
class Window
{
protected int top;
protected int left;
```

```
public Window(int top, int left)
{
this.top = top;
this.left = left;
}
public virtual void DrawWindow()
{
Console.WriteLine("Window: at {0} , {1}", top, left);
}
}
//---------------------------------------------------
class Listbox: Window
{
private string lboxContent;
public Listbox(int top, int left, string content): base(top, left)
{
lboxContent = content;
}
public override void DrawWindow()
{
base.DrawWindow();
Console.WriteLine("Ghi noi dung vao cua so Listbox: "+lboxContent);
}
}
class Button: Window
{
public Button(int top, int left): base(top, left)
{ }
public override void DrawWindow()
{
Console.WriteLine("Drawing a button at {0} , {1}", top, left);
}
```

```
}
class ExamInherit
{
static void Main(string[] args)
{
Window w = new Window(4, 5);
ListBox lbox = new Listbox(7, 8, "Day la mot Listbox!");
Button b = new Button(10, 11);
w.DrawWindow();
lbox.DrawWindow();
b.DrawWindow();
Window [] winarr = new Window[ 3];//mang cac doi tuong
winarr[ 0] = new Window(11, 22);
winarr[ 1] = new ListBox(33, 44, "Mot Listbox khac!");
winarr[ 2] = new Button(44, 55);
for(int i = 0; i < 3; i++)
winarr[ i].DrawWindow();
Console.ReadLine();
}
}
```

Kết quả:

```
Window: at 4, 5
Window: at 7, 8
Ghi noi dung vao cua so Listbox: Day la mot Listbox!
Drawing a button at 11, 12
Window: at 11, 22
Window: at 33, 44
Ghi noi dung vao cua so Listbox: Mot Listbox khac!
Drawing a button at 44, 55.
```

Chương 7. Tính đa dạng (polymorphism)

Polymorphism là từ ghép bởi "poly" nghĩa là nhiều (đa) còn "morph" nghĩa là dạng (hình). Như vậy polymorphism có nghĩa là sử dụng nhiều dạng của một kiểu (type) mà không cần quan tâm đến các chi tiết cụ thể. Tính đa dạng (đa hình) là một trong các đặc tính rất ưu việt của lập trình hướng đối tượng.

Tính đa dạng được thể hiện trong C# bởi một từ khác là "nạp chồng" hay "trùng tên" (overloading). Ở đây tôi sử dụng từ "trùng tên" hợp với nội dung hơn.

Có hai loại đa dạng (từ nay về sau ta chỉ gọi là "trùng tên"): hàm trùng tên (overloading function) và toán tử trùng tên (overloading operator).

Ta thấy câu lệnh có hàm:

```
Console.WriteLine();
Console.WriteLine("Ha noi");
Console.WriteLine("Age" +25);
Console.WriteLine("Ky tu: " +'A');
```

Đây là một hàm thư viện được xây dựng sẵn trong C#. Hàm này có các kiểu đối số khác nhau: kiểu string, kiểu int, kiểu ký tự... nhưng chỉ có một tên: Console.WriteLine(). Hàm này mặc nhiên được gọi là hàm trùng tên.

Trong chương 5 và chương 6, nhiều ví dụ ta đều thấy trong một chương trình có thể có hai hàm tạo trùng tên nhau nhưng một hàm tạo không có đối số còn một hàm tạo lại có đối số.

```
class Person
{
private int ID;
private string name;
public Person()
{
ID = 0;
```

```
Name = " ";
}
//-----------------------------------------------------
public Person(int ID, string name)
{
This.ID = ID;
This.name = name;
}
}
```

Trong thừa kế, các hàm ở lớp dẫn xuất đều có thể trùng tên với các hàm ở lớp cơ sở. Các toán tử trùng tên ta xét sau đây sẽ thể hiện sức mạnh dồi dào của lập trình hướng đối tượng.

7.1. HÀM TRÙNG TÊN

Hàm trùng tên được định nghĩa trong một lớp nếu chúng có tên giống nhau nhưng khác nhau về:

– Số lượng các đối số.

– Kiểu của các đối số (nếu có số lượng đối số giống nhau).

Ví dụ 7.1.

```
using System;
class Vector
{
  private int [] a = new int[ 10];
  private int [] b = new int[ 10];
  private int [] c = new int[ 10];
  public int n;
//-----------------------------------------------------
public Vector()
  {
   for(int i = 0; i < 10; i++)
        {
          a[ i] = b[ i] = c[ i] = 0;
```

```
        }
    }
//-------------------------------------------------
    public void Input(int []a, int n)
    {
     for(int i = 0; i < n; i++)
          a[ i] = Int32.Parse(Console.ReadLine());
    }
//-------------------------------------------------
    public void Input()
{
Console.Write("Nhap n = ");
n = Int32.Parse(Console.ReadLine());
Console.WriteLine("Nhap vector a.");
Input(a, n);
Console.WriteLine("Nhap vector b.");
Input(b, n);
}
//-------------------------------------------------
public void Add()
{
for(int i = 0; i < n; i++)
c[ i] = a[ i] + b[ i];
}
//-------------------------------------------------
public void Show(int []a, int n)
{
for(int i = 0; i < n; i++)
Console.Write(" "+a[ i]);
}
//-------------------------------------------------
public void Show()
```

```
{
Console.WriteLine("\nVector a.");
Show(a, n);
Console.WriteLine("\nVector b.");
Show(b, n);
Add();
Console.WriteLine("\nVector c = a + b.");
Show(c, n);
}
//-------------------------------------------------------
static void Main(string[] args)
{
Vector ob = new Vector();
ob.Input();
ob.Show();
Console.ReadLine();
}
}
```

 *Kết quả:*

```
Nhap n = 3
Nhap vector a.
1 2 3
Nhap vector b.
4 5 6
Vector a.
1 2 3
Vector b.
4 5 6
Vector c = a + b.
5 7 9
```

Trong ví dụ trên ta thấy có hai hàm trùng tên Input() và hai hàm trùng tên Show() nhưng có đối số khác nhau.

Nếu trong một chương trình có hai hàm trùng tên, có số lượng đối số như nhau và kiểu của từng đối số giống hệt nhau thì hai hàm đó chỉ là một và định nghĩa như vậy là sai cú pháp.

7.2. TOÁN TỬ TRÙNG TÊN

Trong tính toán số học các biến vừa khai báo vừa gán giá trị:

```
double a = 5, b = 6;
double m = a + b; // Kết quả m = 11
```

Thông thường ta định nghĩa hàm:

```
public double Tinh(double a, double b)
{
return(a + b);
}
```

Để có kết quả ta viết lời gọi hàm:

```
double m = Tinh(a, b);
```

Rõ ràng là phép cộng số học bình thường m = a + b dễ hiểu hơn nhiều so với cách định nghĩa hàm và gọi hàm.

Toán tử trùng tên cho phép ta thực hiện phép cộng của các đối tượng với nhau.

Ví dụ, khi khai báo đối tượng có 3 toạ độ x, y, z:

```
Toado M1(x1, y1, z1);
Toado M2(x2, y2, z2);
Toado M3.
```

Nếu ta định nghĩa phép cộng: M3 = M1 + M2; thì việc cộng hai đối tượng M1 và M2 hoàn toàn giống với phép cộng trong số học.

Cú pháp của toán tử trùng tên:

```
public static Toado operator+(Toado ob1, Toado ob2)
{
Toado m;
m.x = ob1.x + ob2.x;
m.y = ob1.y + ob2.y;
m.z = ob1.z + ob2.z;
```

```
return(m);
}
```

Chú ý: Trong định nghĩa trên ta thấy có từ khoá static, từ khoá operator và bên trong hàm toán tử phải có lệnh:

```
return(đối_tượng);
```

Sau từ khoá operator có thể đặt ký hiệu các phép toán trong số học như +, –, *, /, ++, -- và cả các phép toán logic.

Ví dụ 7.2. Tính tổng, hiệu, tích, thương toạ độ các vector a,b.

```
using System;
class Vector
{
private double [] a = new double[ 5];
public vector(double [] a)
{
for(int i = 0; i < 5; i++)
this.a[ i] = a[ i];
}
//-------------------------------------------------------
public vector()
{
for(int i = 0; i < 5; i++)
a[ i] = 0;
}
//-------------------------------------------------------
public void Show()
{
for(int i = 0; i < 5; i++)
Console.Write("{ 0:0.00}",a[ i] );
}
//-------------------------------------------------------
public static vector operator+(Vector ob1, Vector ob2)
{
```

```
vector v = new Vector();
for(int i = 0; i < 5; i++)
v.a[ i] = ob1.a[ i] + ob2.a[ i] ;
return v;
}
//-----------------------------------------------------
public static vector operator-(Vector ob1, Vector ob2)
{
vector v = new vector();
for(int i = 0; i < 5; i++)
v.a[ i] = ob1.a[ i] - ob2.a[ i] ;
return v;
}
//-----------------------------------------------------
public static vector operator*(vector ob1, vector ob2)
{
vector v = new vector();
for(int i = 0; i < 5; i++)
v.a[ i] = ob1.a[ i] * ob2.a[ i] ;
return v;
}
//-----------------------------------------------------
public static vector operator/(vector ob1, vector ob2)
{
vector v = new vector();
for(int i = 0; i < 5; i++)
v.a[ i] = ob1.a[ i] / ob2.a[ i] ;
return v;
}
//-----------------------------------------------------
static void Main(string[ ] args)
{
```

```
double [ ] a = { 1, 2, 3, 4, 5} ;
double [ ] b = { 6, 7, 8, 9, 10} ;
vector ob1 = new vector(a);
vector ob2 = new vector(b);
vector ob3, ob4, ob5, ob6;
ob3 = ob1 + ob2;
ob4 = ob1 - ob2;
ob5 = ob1 * ob2;
ob6 = ob1 / ob2;
Console.WriteLine("\nVector a.");
ob1.Show();
Console.WriteLine("\nVector b.");
ob2.Show();
Console.WriteLine("\nVector c = a + b.");
ob3.Show();
Console.WriteLine("\nVector d = a - b.");
ob4.Show();
Console.WriteLine("\nVector e = a * b.");
ob5.Show();
Console.WriteLine("\nVector f = a / b.");
ob6.Show();
Console.ReadLine();
}
}
```

Kết quả:

Vector a.

1 2 3 4 5

Vector b.

6 7 8 9 10

Vector c = a + b.

7.00 9.00 11.00 13.00 15.00

Vector d = a - b.

-5.00 -5.00 -5.00 -5.00 -5.00

Vector e = a * b.

6.00 14.00 24.00 36.00 50.00

Vector f = a / b.

0.17 0.29 0.38 0.44 0.50

Ví dụ 7.3. Tính tổng, hiệu và tích hai ma trận A, B có các phần tử được gán trước.

```
using System;
class Mat
{
private int[ ,] a = new int[ 3,3] ;
public Mat(int [ ,] a)
{
for(int i = 0; i < 3; i++)
for(int j = 0; j < 3; j++)
this.a[ i,j] = a[ i,j] ;
}
//-----------------------------------------------------
public Mat()
{
for(int i = 0; i < 3; i++)
for(int j = 0; j < 3; j++)
a[ i,j] = 0;
}
//-----------------------------------------------------
public void Show()
{
for(int i = 0; i < 3; i++)
{
for(int j = 0; j < 3; j++)
Console.Write("{ 0}", a[ i,j] );
Console.WriteLine();
```

```
}
}
//-----------------------------------------------------
public static Mat operator+(Mat ob1, Mat ob2)
{
Mat v = new Mat();
for(int i = 0; i < 3; i++)
for(int j = 0; j < 3; j++)
v.a[ i,j] = ob1.a[ i,j] + ob2.a[ i,j] ;
return v;
}
//-----------------------------------------------------
public static Mat operator-(Mat ob1, Mat ob2)
{
Mat v = new Mat();
for(int i = 0; i < 3; i++)
for(int j = 0; j < 3; j++)
v.a[ i,j] = ob1.a[ i,j] - ob2.a[ i,j] ;
return v;
}
//-----------------------------------------------------
public static Mat operator*(Mat ob1, Mat ob2)
{
Mat v = new Mat();
for(int i = 0; i < 3; i++)
for(int j = 0; j < 3; j++)
{
v.a[ i,j] = 0;
for(int k = 0; k < 3; k++)
v.a[ i,j] += ob1.a[ i,k] * ob2.a[ k,j] ;
}
return v;
```

```
}
//-------------------------------------------------------
static void Main(string[] args)
{
int [,] a = new int[ 3, 3]{{ 1, 2, 3} , { 4, 5, 6} , { 1, 0, 0}};
int [,] b = new int[ 3, 3]{{ 0, 0, 1} , { 6, 5, 4} , { 3, 2, 1}};
Mat ob1 = new Mat(a);
Mat ob2 = new Mat(b);
Mat ob3, ob4, ob5;
ob3 = ob1 + ob2;
ob4 = ob1 - ob2;
ob5 = ob1 * ob2;
Console.WriteLine("\nMatran A. ");
ob1.Show();
Console.WriteLine("\nMatran B. ");
ob2.Show();
Console.WriteLine("\nMatran C = A + B. ");
ob3.Show();
Console.WriteLine("\nMatran D = A - B. ");
ob4.Show();
Console.WriteLine("\nMatran E = A * B. ");
ob5.Show();
Console.ReadLine();
}
}
```

Kết quả:

```
Matran A.
1 2 3
4 5 6
1 0 0
Matran B.
0 0 1
```

```
6 5 4
3 2 1
Matran C = A + B.
1 2 4
10 10 10
4 2 1
Ma tran D = A – B
1 2 1
–2 0 2
–2 –2 –1
Matran E = A * B.
21 16 12
48 37 30
0 0 1
```

Ví dụ 7.4. Trong ví dụ 7.3 ta gán sẵn giá trị các phần tử của hai ma trận A, B. Trong ví dụ này ta viết chương trình để nhập vào kích thước m, n của ma trận theo cách định nghĩa property (nếu tính đồng thời cả tổng, hiệu và tích hai ma trận thì thêm điều kiện m = n). Các phần tử của hai ma trận cũng được nhập từ bàn phím nhờ hàm Input().

```
using System;
class Mat
{
private static int m, n;
private int[ ,] a = new int[ 5, 5] ;
//-----------------------------------------------------
public int m1
{
get{ return m;}
set{ m = value;}
}
//-----------------------------------------------------
public int n1
```

```
{
get{ return n;}
set{ n = value;}
}
public Mat(int [ ,] a  int m, int n)
{
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
this.a[ i,j] = a[ i,j];
}
//-----------------------------------------------------
public Mat()
{
for(int i = 0; i < 5; i++)
for(int j = 0; j < 5; j++)
a[ i,j] = 0;
}
//-----------------------------------------------------
public void Input()
{
Console.WriteLine("Nhap cac phan tu ma tran: ");
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
a[ i,j] = Int32.Parse(Console.ReadLine());
}
//-----------------------------------------------------
public void Show()
{
for(int i = 0; i < m; i++)
{
for(int j = 0; j < n; j++)
Console.Write("{ 0}", a[ i,j] );
Console.WriteLine();
}
```

```
}
//-------------------------------------------------------
public static Mat operator+(Mat ob1, Mat ob2)
{
Mat v = new Mat();
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
v.a[ i,j] = ob1.a[ i,j] + ob2.a[ i,j];
return v;
}
//-------------------------------------------------------
public static Mat operator-(Mat ob1, Mat ob2)
{
Mat v = new Mat();
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
v.a[ i,j] = ob1.a[ i,j] - ob2.a[ i,j];
return v;
}
//-------------------------------------------------------
public static Mat operator*(Mat ob1, Mat ob2)
{
Mat v = new Mat();
for(int i = 0; i < m; i++)
for(int j = 0; j < n; j++)
{
v.a[ i,j] = 0;
for(int k = 0; k < n; k++)
v.a[ i,j] += ob1.a[ i,k] * ob2.a[ k,j];
}
return v;
}
//-------------------------------------------------------
```

```
static void Main(string[] args)
{
int mr, nc;
Console.Write("Nhap kich thuoc m, n:");
mr = Int32.Parse(Console.ReadLine());
nc = Int32.Parse(Console.ReadLine());
Mat ob = new Mat();
ob.m1 = mr;
ob.n1 = nc;
Mat ob1 = new Mat();
Mat ob2 = new Mat();
Mat ob3, ob4, ob5;
Console.WriteLine("Nhap ma tran A.");
ob1.Input();
Console.WriteLine("Nhap ma tran B.");
ob2.Input();
ob3 = ob1 + ob2;
ob4 = ob1 - ob2;
ob5 = ob1 * ob2;
Console.WriteLine("\nMa tran A.");
ob1.Show();
Console.WriteLine("\nMa tran B.");
ob2.Show();
Console.WriteLine("\nMa tran C = A + B.");
ob3.Show();
Console.WriteLine("\nMa tran D = A - B.");
ob4.Show();
Console.WriteLine("\nMa tran E = A * B.");
ob5.Show();
Console.ReadLine();
}
}
```

Kết quả:

```
Nhap kich thuoc m, n: 2
2
Nhap ma tran A.
Nhap cac phan tu ma tran:
1
2
3
4
Nhap ma tran B.
Nhap cac phan tu ma tran:
5
6
7
8
Ma tran A.
  1     2
  3     4
Ma tran B.
  5     6
  7     8
Ma tran C = A + B.
  6     8
  10    12
Ma tran D = A – B.
  –4    –4
  –4    –4
Ma tran E = A * B.
  19    22
  43    50
```

Chương 8. Files

Khi cần lưu dữ liệu vào đĩa, C# cung cấp một cơ chế để thực hiện giống như mọi ngôn ngữ khác. Có hai cách ghi và đọc dữ liệu vào files: kiểu file text và kiểu file nhị phân.

8.1. GHI/ĐỌC FILE TEXT

Khi viết chương trình để lưu dữ liệu vào file, ta phải thêm dòng tiêu đề:

```
using System.IO;
```

8.1.1. Ghi dữ liệu vào file

Cú pháp để ghi dữ liệu vào file:

```
string filename = "c:\\CS\\test1.txt ";
StreamWriter sw = new StreamWriter(filename)
sw.Write("This is the text.");
```

Sau khi ghi xong cần phải đóng file lại.

```
sw.Close();
```

8.1.2. Đọc dữ liệu từ file

```
string filename = "c:\\cs\\test1.txt";
StreamReader sr1 = new StreamReader(filename);
sr1.ReadLine();
sr1.Close();
```

Ví dụ 8.1.

```
using System;
using System.IO;
using System.Text;
class FileText
{
static string file1 = "c:\\testfile.txt";
static double a = 123.456;
static int b = 789;
static char kt = 'M';
```

```
static void Main(string[] args)
{
using(StreamWriter sw = new StreamWriter(file1))
{
sw.Write("This is a text; a = ");
sw.WriteLine(a);
sw.WriteLine(b);
sw.WriteLine(kt);
sw.Close();
}
StreamReader sr1 = new StreamReader(file1);
string input;
while((input = sr1.ReadLine())!= null)
{
Console.WriteLine(input);
}
Console.WriteLine("Ket thuc!");
sr1.Close();
Console.ReadLine();
}
}
```

 *Kết quả:*

This is a text; a = 123.456

789

M

Ket thuc.

Chú ý: Một cách khác để ghi dữ liệu vào file. Trước hết phải khai báo biến file:

```
StreamWriter sw;
```

Tạo file text nhờ hàm CreateText():

```
static string file1 = "c:\\testfile.txt";
sw = File.CreateText(file1);
```

Ghi dữ liệu vào file:

```
sw.WriteLine("This is a text.");
```

Sau khi ghi xong dữ liệu cần phải đóng file lại.

```
sw.Close();
```

Đọc dữ liệu từ file:

Khai báo đối tượng file:

```
StreamReader sr1;
```

Mở file:

```
string input;
sr1 = File.OpenText(file1);
```

Đọc file:

```
input=sr1.ReadToEnd();
```

Hiển thị dữ liệu sau khi đọc:

```
Console.WriteLine(input);
```

Ví dụ 8.2.

```
using System;
using System.IO;
class FileText
{
static string file1 = "c:\\testfile.txt";
static void Main(string[] args)
{
StreamWriter sw; // ghi du lieu vao file
sw = File.CreateText(file1);
sw.WriteLine("This is a text.");
sw.WriteLine("Dai hoc Bach Khoa Ha Noi.");
sw.WriteLine("Trung tam Genetic Singapore.");
sw.WriteLine(123.456);
sw.Close();
StreamReader sr1; // doc du lieu tu file
// = new StreamReader(file1);
```

```
string input;
sr1 = File.OpenText(file1);
input = sr1.ReadToEnd();
Console.WriteLine(input);
Console.WriteLine("The end of the stream!");
sr1.Close();
Console.ReadLine();
}
}
```

Kết quả:

```
This is a text
Dai hoc Bach Khoa Ha Noi
Trung tam Genetic Singapore
123.456
```

Ta cũng có thể đọc file theo các dòng kế tiếp như đoạn mã dưới đây:

```
StreamReader sr;
string input;
sr = File.OpenText(file1);
while(true)
{
if(input == null) break;
Console.WriteLine(input);
}
sr.Close();
```

Ví dụ 8.3.

```
using System;
using System.IO;
using System.Text;
class FileText
{
static string file1 = "c:\\testfile.txt";
```

```
static double a = 123.456;
static int b = 789;
static char kt = 'M';
static void Main(string[] args)
{
using(StreamWriter sw = new StreamWriter(file1))
{
sw.Write("This is a text; a = ");
sw.WriteLine(a);
sw.WriteLine(b);
sw.WriteLine(kt);
sw.Close();
}
StreamReader sr;
string input;
sr = File.OpenText(file1);
while(true)
{
input = sr.ReadLine();
if(input == null) break;
Console.WriteLine(input);
}
Console.WriteLine("Ket thuc!");
sr.Close();
Console.ReadLine();
}
}
```

Kết quả:

This is a text; a = 123.456
789
M
Ket thuc.

8. 2. GHI/ĐỌC FILE NHỊ PHÂN

Việc ghi/đọc dữ liệu của file nhị phân có nhiều ưu điểm hơn file text vì bảo mật file cao hơn. Ở đây sẽ trình bày cách ghi/đọc dữ liệu cho một byte và cách ghi/đọc dữ liệu cho một đối tượng chứa nhiều trường dữ liệu.

– Ghi dữ liệu vào file kiểu byte:

+ Giả sử ta khai báo một biến mảng kiểu byte:

```
byte [] b1 = { 1, 2, 3, 4, 255};
```

+ Khai báo đối tượng để ghi dữ liệu vào file "testbin.dat" ở ổ đĩa C:

```
FileStream("c:\\testbin.dat",FileMode.OpenOrCreate,
FileAccess.Write, FileShare.None);
```

+ Thực hiện quá trình ghi các phần tử của mảng số b1 có kiểu byte:

```
foreach(byte bnext in b1)
{
sw.WriteByte(bnext);
}
```

+ Sau khi ghi xong, ta phải đóng file lại.

```
sw.Close();
```

– Đọc dữ liệu từ file kiểu byte:

+ Mở file để đọc:

```
FileInfo fr = new FileInfo("c:\\testbin.dat");
Stream strm = fr.OpenRead();
```

+ Sử dụng vòng lặp để đọc:

```
int i;
while(true)
{
i = strm.ReadByte();
if(i == -1) break;
Console.Write("{ 0,4}", i);
}
```

+ Đóng file sau khi đọc xong:

```
strm.Close();
```

Ví dụ 8.4. Ví dụ này cho phép ta ghi và đọc các số có kiểu byte (giá trị 0 ÷ 255).

```
using System;
using System.IO;
class Filebin //cac so kieu byte
{
static void Main(string[] args)
{
//Ghi du lieu tu file voi so kieu byte(0 ÷ 255)
byte [] b1 = { 255,33,44,55,66} ;
FileStream sw = new FileStream ("c:\\testbin.dat",
FileMode.OpenOrCreate, FileAccess.Write, FileShare.None);
foreach(byte bnext in b1)
{
sw.WriteByte(bnext);
}
sw.Close();
// Doc du lieu tu file
FileInfo f = new FileInfo("c:\\testbin.dat");
Stream strm = f.OpenRead();
int i;
while(true)
{
i = strm.ReadByte();
if(i == -1) break;
Console.Write("{0,4}", i);
}
strm.Close();
Console.ReadLine();
}
}
```

Kết quả:

255 33 44 55 66

Ta có thể thực hiện việc ghi/đọc dữ liệu của nhiều kiểu dữ liệu khác nhau bằng file nhị phân.

– Ghi dữ liệu từ file nhị phân:

```
string file1 = "c:\\FileBin1.dat";
FileStream file1;
string name; //dữ liệu thành viên
double salary; //dữ liệu thành viên
// khai báo đối tượng wf cho việc ghi vào file nhị phân
BinaryWriter wf = new BinaryWriter(file1);
// Các hàm Write () để ghi từng thành viên
wf.Write(name);
wf.Write(salary);
```

– Đọc dữ liệu từ file nhị phân:

Khai báo file cần đọc:

```
FileStream file1;
```

Khai báo đối tượng rf cho việc đọc:

```
BinaryReader rf = new BinaryReader(file1);
```

Các hàm để đọc dữ liệu nhị phân:

```
name = rf.ReadString();
salary = rf.ReadDouble();
```

Phụ thuộc vào kiểu khai báo từng dữ liệu mà ta có các hàm đọc dữ liệu khác nhau.

```
byte b1; b1 = rf.ReadByte();
char ch1; ch1 = rf.ReadChar();
decimal d1; d1 = rf.ReadDecimal();
double d2; d2 = rf.ReadDouble();
float f1; f1 = rf.ReadSingle();
int a; a = rf.ReadInt32();
long b; b = rf.ReadInt64();
sbyte sb1; sb1 = rf.ReadSByte();
uint ui1; ui1 = rf.ReadUInt32();
ulong ul1; ul1 = rf.ReadUInt64();
```

Ví dụ 8.5.

```
using System;
using System.IO;
class FileBin2
{
private string name;
private int age;
private double salary;
public FileBin2()
{
name = "";
age = 0;
salary = 0;
}
public FileBin2(string name, int age, double salary)
{
this.name = name;
this.age = age;
this.salary = salary;
}
public void Input()
{
Console.Write("\nNhap ten: ");
name = Console.ReadLine();
Console.Write("Nhap tuoi: ");
age = Int32.Parse(Console.ReadLine());
Console.Write("Nhap luong: ");
salary = Double.Parse(Console.ReadLine());
}
public void Show()
{
Console.WriteLine("Ten: "+name+"; Tuoi: "+age);
```

```
Console.WriteLine("Luong: "+salary);
}
public void SaveData(FileStream file1)
{
BinaryWriter wf = new BinaryWriter(file1);
wf.Write(name);
wf.Write(age);
wf.Write(salary);
}
public void ReadData(FileStream file1)
{
BinaryReader rf = new BinaryReader(file1);
name = rf.ReadString();
age = rf.ReadInt32();
salary = rf.ReadDouble();
}
public void Menu()
{
Console.WriteLine("      MENU     ");
Console.WriteLine("===============");
Console.WriteLine("1. Nhap du lieu");
Console.WriteLine("2. Doc va xem ");
Console.WriteLine("3. Thoat");
Console.WriteLine("===============");
Console.Write("Nhap so 1 -3 de lua chon: ");
}
}
class TestFile
{
static void Main(string[] args)
{
FileBin2 p = new FileBin2();
```

```
    string filename = "c:\\Data.dat";
    string choice = " ";
    while(choice != "3")
    {
    p.Menu();
    choice = Console.ReadLine();
    switch(choice)
    {
    case "1": Console.WriteLine("Save data to file");
    FileStream fwr = new FileStream(filename,
FileMode.Create, FileAccess.Write, FileShare.Write);
    p = new FileBin2();
    p.Input();
    p.SaveData(fwr);
    fwr.Close();
    break;
    case "2": Console.WriteLine("Read data from file");
    FileStream fr = new FileStream(filename, FileMode.Open,
FileAccess.Read, FileShare.Read);
    p = new FileBin2();
    p.ReadData(fr);
    p.Show();
    break;
    case "3": continue;
    }
    }
    }
    }
```

Kết quả:

```
        MENU
    ================
    1. Nhap du lieu
    2. Doc va xem
```

```
3. Thoat
==================
Nhap so 1 – 3 de lua chon: 1
Save data to file
Nhap ten: Hoa
Nhap tuoi: 22
Nhap luong: 1000
        MENU
==================
1. Nhap du lieu
2. Doc va xem
3. Thoat
==================
Nhap so 1 – 3 de lua chon: 2
Read data from file
Ten: Hoa; Tuoi: 22
Luong: 1000
```

Việc ghi/đọc bằng file nhị phân có thể thực hiện cho một mảng đối tượng. Mỗi đối tượng lại bao gồm nhiều kiểu dữ liệu thành phần khác nhau.

Ví dụ 8.6. Viết chương trình để lưu dữ liệu vào file.

```
using System;
using System.IO;
class FileBin3
{
private string name;
private int age;
private double salary;
public FileBin3()
{
name = "";
age = 0;
```

```
salary = 0;
}
public void Input()
{
Console.Write("\nNhap ten: ");
name = Console.ReadLine();
Console.Write("Tuoi: ");
age = Int32.Parse(Console.ReadLine());
Console.Write("Luong: ");
salary = Double.Parse(Console.ReadLine());
}
public void Show()
{
Console.WriteLine("Doc du lieu tu file");
Console.WriteLine("Ten "+name+"; Tuoi:"+age);
Console.WriteLine("Luong: "+salary);
}
public void SaveData(FileStream file1)
{
BinaryWriter wf = new BinaryWriter(file1);
wf.Write(name);
wf.Write(age);
wf.Write(salary);
}
public void ReadData(FileStream file1)
{
BinaryReader rf = new BinaryReader(file1);
name = rf.ReadString();
age = rf.ReadInt32();
salary = rf.ReadDouble();
}
}
```

```
class TestFile
{
static void Main(string[] args)
{
int n;
Console.Write("Nhap n = ");
n = Int32.Parse(Console.ReadLine());
FileBin3 [] p = new FileBin3[n];
string filename = "c:\\Data.dat";
Console.WriteLine("Ghi du lieu vao file");
FileStream fwr = new FileStream(filename, FileMode.Create,
FileAccess.Write, FileShare.Write);
for(int i = 0; i < n; i++)
{
p[i] = new FileBin3();
p[i].Input();
p[i].SaveData(fwr);
}
fwr.Close();
//-----------------------------------------------------
Console.WriteLine("Doc du lieu tu file");
FileBin3 [] p1 = new FileBin3[n];
FileStream fr = new FileStream(filename, FileMode.Open,
FileAccess.Read, FileShare.Read);
for(int i = 0; i < n; i++)
{
p1[i] = new FileBin3();
p1[i].ReadData(fr);
p1[i].Show();
}
Console.ReadLine();
}
}
```

Kết quả:

```
Nhap n = 2
Ghi du lieu vao file
Nhap ten: Mai
Tuoi: 22
Luong: 1000
Nhap ten: Ha
Tuoi: 35
Luong: 2000
Doc du lieu tu file
Ten: Mai; Tuoi: 22
Luong: 1000
Doc du lieu tu file
Ten: Ha; Tuoi: 32
Luong: 2000
```

Ví dụ 8.7. Xây dựng một menu để thực hiện các thao tác: Nhập, liệt kê, xoá, sửa, tìm kiếm, bổ sung.

```
using System; //Console.Application
class person
{
private string name;
private long ID;
private int age;
//-----------------------------------------------------
public person()
{
name = "";
ID = 0;
age = 0;
}
//-----------------------------------------------------
```

```
    public static int n;
    public person [] p = new person[ 100];
    //-----------------------------------------------------
    public void Input()
    {
    for(int i = 0; i < n; i++)
    {
    p[ i] = new person();
    Console.WriteLine("person { 0}",i);
    Console.WriteLine("Enter name: ");
    p[ i] .name = Console.ReadLine();
    Console.WriteLine("Enter ID: ");
    p[ i] .ID = Int64.Parse(Console.ReadLine());
    Console.WriteLine("Enter age: ");
    p[ i] .age = Int32.Parse(Console.ReadLine());
    }
    }
    //-----------------------------------------------------
    public void Show()
    {
    for(int i = 0; i < n; i++)
    Console.WriteLine("Name:{ 0} ; ID:{ 1} ; age: { 2}", p[ i] .name,
p[ i] .ID, p[ i] .age);
    }
    //-----------------------------------------------------
    public void SearchName()
    {
    string name1;
    Console.Write("Enter name to search: ");
    name1 = Console.ReadLine();
    for(int i = 0; i < n; i++)
    {
    if(String.Compare(name1, p[ i] .name) == 0)
```

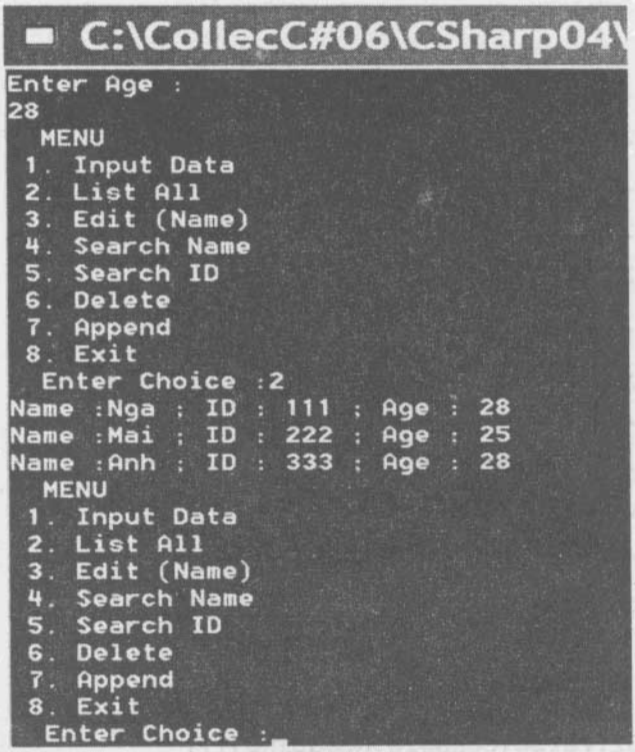
```
    Console.WriteLine("Name: {0} ; ID: {1} ;age: {2}",
p[i].name, p[i].ID, p[i].age);
    }
    }
    //-----------------------------------------------------
    public void SearchID()
    {
    long id1;
    Console.Write("Enter ID to search: ");
    id1 = Int64.Parse(Console.ReadLine());
    for(int i = 0; i < n; i++)
    {
    if(id1 == p[i].ID)
    Console.WriteLine("Name: {0} ; ID: {1} ; age : {2}",
p[i].name, p[i].ID, p[i].age);
    }
    }
    //-----------------------------------------------------
    public void Delete()
    {
    long id1;
    Console.Write("Enter ID to search: ");
    id1 = Int64.Parse(Console.ReadLine());
    for(int i = 0; i < n; i++)
    {
    if(id1 == p[i].ID)
    {
    Console.WriteLine("Name:{0} ;ID:{1} ; age: {2}", p[i].name,
p[i].ID, p[i].age);
    while(i < n)
    {
    p[i] = p[i+1];
    i++;
```

```
    }
    }
    n = n -1;
    }
    }
    //-------------------------------------------------
    public void Edit()
    {
    string name1;
    Console.Write("Enter name to search: ");
    name1 = Console.ReadLine();
    for(int i = 0; i < n; i++)
    {
    if(String.Compare(name1,p[ i] .name) == 0)
    {
    Console.WriteLine("Name: { 0} ; ID: { 1} ; age: { 2}",
p[ i] .name, p[ i] .ID, p[ i] .age);
    Console.WriteLine("person { 0}",i);
    Console.WriteLine("Enter name: ");
    p[ i] .name = Console.ReadLine();
    Console.WriteLine("Enter ID: ");
    p[ i] .ID = Int64.Parse(Console.ReadLine());
    Console.WriteLine("Enter age: ");
    p[ i] .age = Int32.Parse(Console.ReadLine());
    }
    }
    }
    //-------------------------------------------------
    public void Append()
    {
    p[ n] = new person();
    Console.WriteLine("Enter name: ");
    p[ n] .name = Console.ReadLine();
```

```
Console.WriteLine("Enter ID: ");
p[ n] .ID = Int64.Parse(Console.ReadLine());
Console.WriteLine("Enter age: ");
p[ n] .age = Int32.Parse(Console.ReadLine());
n = n + 1;
}
//-------------------------------------------------
public static void Main()
{
person per = new person();
int choice = 0;
do
{
Console.WriteLine(" MENU ");
Console.WriteLine(" 1. Input Data ");
Console.WriteLine(" 2. List All ");
Console.WriteLine(" 3. Edit (Name) ");
Console.WriteLine(" 4. Search Name ");
Console.WriteLine(" 5. Search ID ");
Console.WriteLine(" 6. Delete ");
Console.WriteLine(" 7. Append ");
Console.WriteLine(" 8. Exit ");
Console.Write(" Enter Choice:");
choice = Int32.Parse(Console.ReadLine());
switch(choice)
{
case 1: Console.Write("Enter number of persons: ");
n = Int32.Parse(Console.ReadLine());
per.Input();break;
case 2: per.Show();break;
case 3: per.Edit();break;
case 4: per.SearchName();break;
```

```
case 5: per.SearchID();break;
case 6: per.Delete();break;
case 7: per.Append();break;
case 8: continue;
}
}
while(choice != 8);
}
}
//-------------------------------------------------
```

Kết quả sau khi chạy chương trình được biểu diễn trên hình 8.3.



Hình 8.3

Ví dụ 8.8. Ví dụ này được coi là một bài tập dài trong cuốn Lập trình C# từ cơ bản đến nâng cao. Yêu cầu của bài tập này như sau:

Công ty truyền thông ABC giữ vai trò quan trọng trong sự phát triển

về công nghiệp điện thoại của khu vực. Yêu cầu bạn phải phát triển một hệ thống thông tin để quản lý bằng máy tính về điện thoại nhằm cung cấp cho khách hàng sự hỗ trợ những thông tin cần thiết.

Các yêu cầu đặt ra như sau:

1. Tạo một lớp phone để lưu các trường dữ liệu sau:

Phone number

Fullname

Address

No Of Line

Type

Organization

2. Dữ liệu này cần phải được:

a) Khởi tạo nhờ hàm tạo (constructor)

b) Truy cập dữ liệu nhờ sử dụng thuộc tính (properties)

3. Lớp phone phải có phương thức để lưu đối tượng nhờ lớp FileStream.

4. Tạo một lớp phone book để lưu toàn bộ lớp phone vào trong một mảng.

5. Trong lớp phone book phải sử dụng Indexer để sao cho ta có thể truy cập vào từng phần tử của mảng phone.

6. Người sử dụng phải thực hiện được các chức năng sau:

a) Add New (thêm bản ghi mới)

b) Edit (sửa đổi một bản ghi)

c) Delete (xoá một bản ghi)

d) Tìm một bản ghi trong mảng phone (tìm theo tên và tìm theo số điện thoại)

e) Hiển thị các số điện thoại khẩn cấp

Dưới đây liệt kê toàn bộ chương trình của bài tập dài nêu trên.

```
/* Programming using C# - Examples
 * Project 1: Telephone inquiry system
```

```
    * In class PhoneBook: an array of phones is implemented
by an ArrayList instance
    * File I/O: binary file, FileStream is processed with
BinaryWriter/BinaryReader
    */
    using System;
    using System.IO;
    using System.Collections;
    class Phone
    {
    private ulong phoneNumber;
    private string fullName;
    private string address;
    private int noOfLine;
    private string type;
    private string organization;
    public Phone(ulong phoneNumber, string fullName, string
address, int noOfLine, string type, string organization)
    {
    this.phoneNumber = phoneNumber;
    this.fullName = fullName;
    this.address  = address;
    this.noOfLine = noOfLine;
    this.type = type;
    this.organization = organization;
    }
    public Phone(): this(0, "", "", 0, "", "") {}
    public ulong PhoneNumber
    {
    get{ return phoneNumber;}
    set{ phoneNumber = value;}
    }
    public string FullName
    {
```

```
get{ return fullName;}
set{ fullName = value;}
}
public string Address
{
get{ return this.address;}
set{ address = value;}
}
public int noOfLine
{
get{ return noOfLine;}
set{ noOfLine = value;}
}
public string Type
{
get{ return type;}
set{ type = value;}
}
public string Organization
{
get{ return organization;}
set{ organization = value;}
}
public void Input()
{
Console.Write("Phone number: ");
phoneNumber = UInt64.Parse(Console.ReadLine());
Console.Write("Full name: ");
fullName = Console.ReadLine();
Console.Write("Address: ");
address = Console.ReadLine();
Console.Write("No of line: ");
noOfLine = Int32.Parse(Console.ReadLine());
Console.Write("Type: ");
```

```
    type = Console.ReadLine();
    Console.Write("Organization: ");
    organization = Console.ReadLine();
    }
    public void Display()
    {
    //Console.WriteLine(ToString());
    Console.WriteLine("Telephone Number:\t{ 0, -40} Lines:
{ 1,5}", phoneNumber, noOfLine);
    Console.WriteLine("Name/Organization:\t" +fullName+ " / "
+organization);
    Console.WriteLine("Address:\t\t" + address);
    Console.WriteLine("Type:\t\t\t" + type);
    }
    public override string ToString()
    {
    Return(String.Format("Phone: { 0}\nName: { 1}\nAddress:
{ 2}\nLine: { 3}\nType: { 4}\nOrg: { 5}\n", phoneNumber, fullName,
address, noOfLine, type, organization));
    }
    //public void Save(FileStream fs)
    public void Save(BinaryWriter bw)
    {
    //BinaryWriter bw = new BinaryWriter(fs);
    bw.Write(phoneNumber);
    bw.Write(fullName);
    bw.Write(address);
    bw.Write(noOfLine);
    bw.Write(type);
    bw.Write(organization);
    //bw.Close();
    }
    public void Load(BinaryReader br)
    {
```

```
phoneNumber = br.ReadUInt64();
fullName = br.ReadString();
address = br.ReadString();
noOfLine = br.ReadInt32();
type = br.ReadString();
organization = br.ReadString();
}
}
class PhoneBook
{
ArrayList data = new ArrayList();
//dynamic array of objects
public Phone this[ int i]  // indexer
{
get
{
if ((-1 < i) && (i < data.Count))
return (Phone)data[ i];
else
throw new IndexOutOfRangeException("Index out of range");
}
set
{
if ((-1 < i) && (i < data.Count))
// change a phone
data[ i]  = value;
else if (i == data.Count)
// add new phone
data.Add(value);
else
throw new IndexOutOfRangeException("Index out of range");
}
}
public void Clear()
```

```
{
data.Clear();
}
public void Delete(int index)
{
if ( (-1 < index) && (index < data.Count))
{
data.RemoveAt(index);
Console.WriteLine("Done");
}
else
throw new IndexOutOfRangeException("Nothing to delete");
}
public void Edit(int index)
{
if ( (-1 < index) && (index < data.Count))
{
this[ index] .Display();
this[ index] .Input();
Console.WriteLine("Done");
}
else
throw new IndexOutOfRangeException("Nothing to edit");
}
public void Search(string name)
{
int found = 0;
int[ ] indexArray = new int[ data.Count] ;
// store indexes of phones that
// matches the search condition
int j = 0;
for (int i = 0; i < data.Count; ++i)
{
if (this[ i] .FullName.ToUpper() == name.ToUpper())
```

```
    {
    ++found;
    indexArray[ j++] = i;
    }
    }
    if (found > 0)
    {
    Console.WriteLine("Found phone(s): {0}", found);
    for (int k = 0; k < found; ++k)
    {
    Console.WriteLine("Index of the phone info: {0}",
indexArray[ k] );
    Console.WriteLine(this[ indexArray[ k]].ToString());
    }
    }
    else
    Console.WriteLine("Not found!");
    }
    public void Search(ulong phoneNumber)
    {
    for (int i = 0; i < data.Count; ++i)
    {
    if (this[ i].PhoneNumber == phoneNumber)
    {
    Console.WriteLine("Index of the phone info: {0}", i);
    Console.WriteLine(this[ i].ToString());
    return;
    }
    }
    Console.WriteLine("Not found!");
    }
    public int Count
    {
    get { return data.Count; }
```

```
}
}
//-----------------------------------------------------
class App
{
static PhoneBook list = new PhoneBook();
public static int Main()
{
while(true)
{
int choice = Menu();
switch (choice)
{
case 1: // Add new Phone
Add();
break;
case 2: // Edit a Phone
Edit();
Console.ReadLine();
break;
case 3: // Delete a Phone
Delete();
Console.ReadLine();
break;
case 4: // Search a Phone
Search();
Console.ReadLine();
break;
case 5: // Display PhoneBook
Display();
break;
case 6: // Save to file
Save();
break;
```

```
case 7: // Load from file
Load();
break;
case 8: // Display Emergency Numbers
EmergencyNumbers();
break;
case 9: // Exit program
return 0;
}
}
}
static int Menu()
{
Console.WriteLine("\n\n============ MENU ============");
Console.WriteLine("1. Add a Phone");
Console.WriteLine("2. Edit a Phone");
Console.WriteLine("3. Delete a Phone");
Console.WriteLine("4. Search a Phone");
Console.WriteLine("5. Display PhoneBook");
Console.WriteLine("6. Save to file");
Console.WriteLine("7. Load from file");
Console.WriteLine("8. Display Emergency Numbers");
Console.WriteLine("9. Exit");
int ch = 0;
while ( (ch < 1) || (ch > 9))
{
try
{
Console.Write("\nEnter your choice: ");
ch = Int32.Parse(Console.ReadLine());
}
catch
{
continue;
```

```
}
}
return ch;
}
static int index = -1;
static void Add()
{
bool next = true;
while (next)
{
Console.WriteLine("Enter Phone info  ");
Phone phone = new Phone();
phone.Input();
list[ ++index] = phone;
Console.Write("Do you want to continue? (y/n): ");
string choice = Console.ReadLine();
if (choice.ToUpper() == "Y")
next = true;
else next = false;
}
}
static void Display()
{
for (int i = 0; i < list.Count; ++i)
{
list[ i] .Display();
Console.ReadLine();
}
}
static void Search()
{
Console.Write("Search by Name or PhoneNumber(N/P)?: ");
string select = Console.ReadLine();
if (select.ToUpper() == "N")
```

```
    {
    Console.Write("Enter Name: ");
    string name = Console.ReadLine();
    list.Search(name);
    }
    else
    {
    Console.Write("Enter Phone Number: ");
    ulong tel = UInt64.Parse(Console.ReadLine());
    list.Search(tel);
    }
    }
    static void Edit()
    {
    Console.Write("\nEnter index of Phone to edit: ");
    int i = Int32.Parse(Console.ReadLine());
    list.Edit(i);
    }
    static void Delete()
    {
    Console.Write("\nEnter index of Phone to delete: ");
    int i = Int32.Parse(Console.ReadLine());
    list.Delete(i);
    }
    static void Save()
    {
    Console.Write("\nEnter file name: ");
    string fileName = Console.ReadLine();
    if (File.Exists(fileName))
    File.Delete(fileName);
    //remove pre-stored file if any
    FileStream fs = new FileStream(fileName, FileMode.Append,
FileAccess.Write, FileShare.Write);
    BinaryWriter bw = new BinaryWriter(fs);
```

```
    for (int i = 0; i < list.Count; ++i)
    list[ i] .Save(bw);
    bw.Close();
    fs.Close();
    }
    static void Load()
    {
    Console.Write("\nEnter file name: ");
    string fileName = Console.ReadLine();
    if (File.Exists(fileName))
    {
    list.Clear();
    FileStream fs = new FileStream(fileName, FileMode.Open,
FileAccess.Read, FileShare.Read);
    BinaryReader br = new BinaryReader(fs);
    int i = -1;
    while(br.PeekChar() > -1) // not end of file
    {
    Phone phone = new Phone();
    phone.Load(br);
    list[ ++i] = phone;                              }
    br.Close();
    fs.Close();
    }
    else
    throw new FileNotFoundException("there is no file: ",
fileName);
    }
    static void EmergencyNumbers()
    {
    Console.WriteLine("Police: 113");
    Console.WriteLine("Police: 114");
    Console.WriteLine("Police: 115");
    }
    }
```

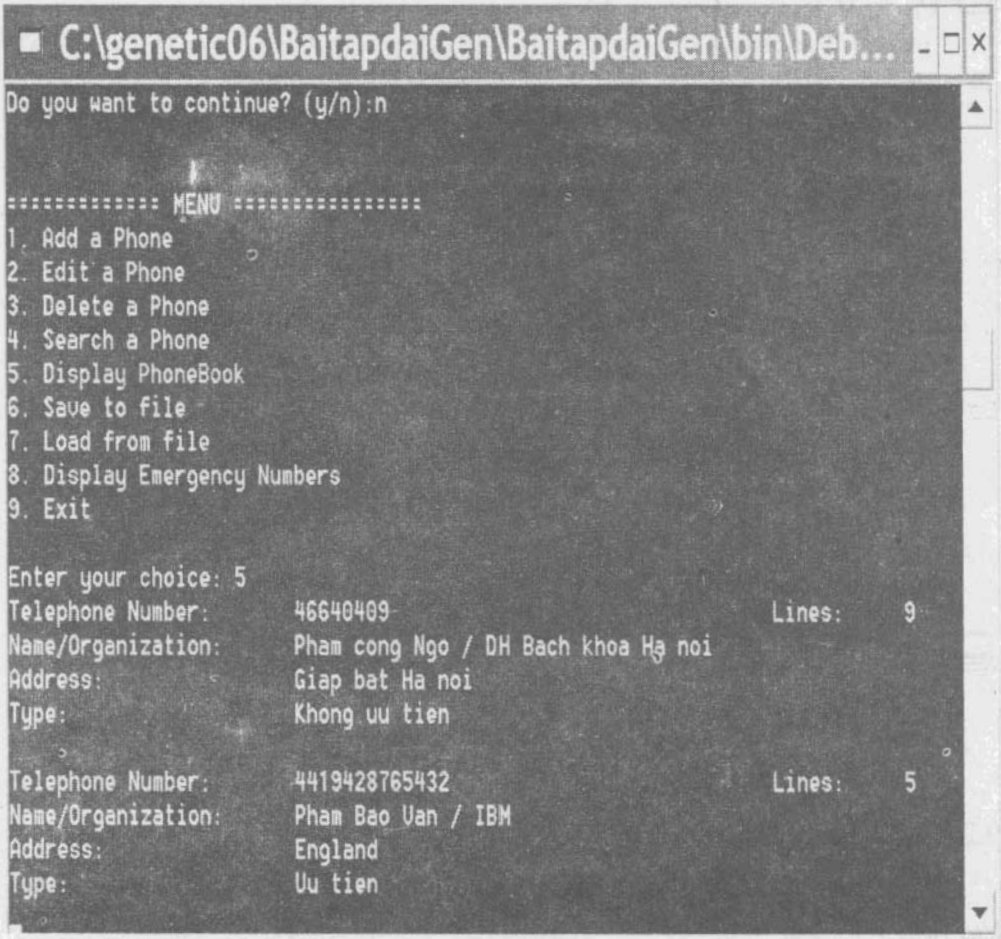
Sau khi biên dịch và chạy chương trình, ta có kết quả như trên hình 8.4. Khi đó, chọn một yêu cầu tương ứng với một phím số để cung cấp dịch vụ cần thiết. Ví dụ:

– Chọn số 1 để nhập số liệu.

– Chọn số 5 để hiển thị dữ liệu.

– Để lưu số liệu vào ổ đĩa cứng, gõ số 6 với File name: nn1.dat.

– Khi chạy lại chương trình, gõ số 7 và gọi tập tin nằm mặc định trong thư mục chứa chương trình. Các bạn có thể gõ đường dẫn cho tập tin, c:\nn1.dat và khi đọc tập tin này cũng phải có đường dẫn.



Hình 8.4

Chương 9. Chuỗi ký tự (string)

Xử lý chuỗi trong các ngôn ngữ lập trình là hết sức cần thiết. Trong C#, có nhiều hàm (phương thức) thư viện để thực hiện việc xử lý chuỗi ký tự. Sau đây ta sẽ xét một số phương thức cơ bản.

9.1. PHƯƠNG THỨC STRING.FORMAT()

Phương thức này cho phép ta định dạng một chuỗi kết hợp nhiều dạng khác nhau.

Ví dụ 9.1.

```
using System;
class STR1
{
static void Main(string[] args)
{
int x = 15;
string s1 = String.Format(" {0} x {1} = {2}", x, 3, x*3);
string s2 = String.Format("Ngay gio hien hanh: {0}",
DateTime.Now);
Console.WriteLine("{0}\r\n {1}", s1, s2);
Console.ReadLine();
}
}
```

Kết quả:

15 x 3 = 45

Ngay gio hien hanh: 8/15/2007 2:35:45 PM

9.2. PHƯƠNG THỨC CONCAT()

Phương thức này cho phép nối các chuỗi với nhau thành một chuỗi lớn hơn.

Ví dụ 9.2.

```
using System;
```

```
class STR2
{
static void Main(string[] args)
{
string s1 = "Hello";
string s2 = "world !";
Console.WriteLine(String.Concat(s1," ",s2));
Console.ReadLine();
}
}
```

Kết quả: Hello world !

9.3. PHƯƠNG THỨC JOIN()

Phương thức này cho phép chèn một chuỗi con vào giữa các chuỗi nhỏ.

Ví dụ 9.3.

```
using System;
class STR3
{
static void Main(string[] args)
{
string [] s1 = {"Hello", "and", "welcome", "to", "you"};
string s2= "---";
for(int i = 0; i < s1.Length; i++)
Console.Write(" "+s1[ i] );
Console.WriteLine();
Console.WriteLine(String.Join(s2,s1));
Console.ReadLine();
}
}
```

Kết quả:

Hello and welcome to you

Hello---and---welcome---to---you

9.4. PHƯƠNG THỨC INSERT()

Ta có thể chèn một chuỗi con vào vị trí n của một chuỗi đã cho.

Ví dụ 9.4.

```
using System;
class STR4
{
static void Main(string[] args)
{
string s1 = "Once a time";
Console.WriteLine("s1: "+s1);
Console.WriteLine(s1.Insert(4, " upon "));
Console.ReadLine();
}
}
```

Kết quả:

s1: Once a time

Once upon a time

9.5. PHƯƠNG THỨC COPYTO()

Nếu có một chuỗi s1 = "Hello beautiful world!", s2 = "Viet nam", khi thực hiện câu lệnh: s1.CopyTo(6, s2, 0, 4) trong đó 6 là vị trí bắt đầu trích các phần tử của chuỗi s1, 0 là vị trí đầu tiên để copy chuỗi mới vào, 4 là độ dài chuỗi cần copy vào.

Ví dụ 9.5.

```
using System;
class STR5
{
static void Main(string[] args)
{
string s1 = "Hello beautiful world!";
char [] s2 = { 'V','i','e','t',' ','n','a','m'};
Console.WriteLine("s1: "+s1);
```

```
Console.WriteLine(mchar);
s1.CopyTo(6, s2, 0, 4);
//6 là vị trí đầu của chuỗi cần trích ra
//0 là vị trí đầu của chuỗi mới
//4 là độ dài "beau"
Console.WriteLine("s2: "+s2);
Console.WriteLine("s1: "+s1);
Console.ReadLine();
}
}
```

Kết quả:

s1: Hello beautiful world !

s2: Viet nam

beau nam

s1: Hello beautiful world !

9.6. PHƯƠNG THỨC TRIMSTART(), TRIMEND(), TRIM()

Phương thức TrimStart() cho phép cắt bỏ ký tự trắng ở đầu chuỗi.

Phương thức TrimEnd() cho phép cắt bỏ ký tự trắng ở cuối chuỗi.

Phương thức Trim() cho phép cắt bỏ ký tự trắng ở hai đầu chuỗi.

Ví dụ 9.6.

```
using System;
class STR6
{
static void Main(string[] args)
{
string s1 = " Big "; //co hai dau cach hai ben
Console.WriteLine("Hello{0}World", s1);
string s2 = s1.TrimStart(); //cat bo dau cach dau chuoi
Console.WriteLine("Hello{0}World: ", s2);
s2 = s1.TrimEnd(); //cat bo dau cach cuoi chuoi
Console.WriteLine("Hello{0}World: ", s2);
```

```
s2 = s1.Trim();//cat bo cac dau cach ca hai phia cua chuoi
Console.WriteLine("Hello{ 0} World: ", s2);
Console.ReadLine();
}
}
```

Kết quả:

Hello Big World // có cả 2 dấu cách hai bên

HelloBig World //cắt bỏ các dấu cách đầu (bên trái)

Hello BigWorld // cắt bỏ các dấu cách cuối (bên phải)

HelloBigWorld // cát bỏ cả 2 dấu cách đầu và cuối (hai bên)

9.7. PHƯƠNG THỨC REMOVE(), PADLEFT(), PADRIGHT()

Phương thức Remove() cho phép cắt bỏ các phần tử của chuỗi từ vị trí m và có độ dài n.

Phương thức PadLeft() cho phép điền một ký tự mới vào phía trái một chuỗi đã cho để lấp đầy n ký tự của chuỗi (bao gồm cả ký tự mới).

Phương thức PadRight() cho phép điền một ký tự mới vào phía phải một chuỗi đã cho để lấp đầy n ký tự của chuỗi (bao gồm cả ký tự mới).

Ví dụ 9.7.

```
using System;
class STR7
{
static void Main(string[] args)
{
string s1 = "Hello Beautiful World";
Console.WriteLine(s1);
string s2 = s1.Remove(5,10);
Console.WriteLine(s2);
s2 = s1.PadLeft(30,'*');
Console.WriteLine(s2);
s2 = s1.PadRight(30,'*');
Console.WriteLine(s2);
```

```
Console.ReadLine();
}
}
```

Kết quả:

Hello Beautiful World
Hello World
*********Hello Beautiful World
Hello Beautiful World*********

9.8. PHƯƠNG THỨC SO SÁNH CÁC CHUỖI KÝ TỰ

Phương thức String.Compare(s1,s2); cho kết quả trả về là một số nguyên n. Nếu s1 > s2 thì n > 0. Nếu s1 > s2 thì n < 0. Nếu s1 > s2 thì n = 0.

Phương thức s1.CompareTo(s2) cũng trả về kết quả giống như phương thức String.Compare(s1, s2).

Phương thức String.CompareOrdinal(s1, s2); cũng giống hai phương thức trên.

Phương thức String.Equals(s1, s2); trả về giá trị true nếu hai chuỗi s1 = s2, trả về giá trị false nếu hai chuỗi s1 và s2 khác nhau.

Ví dụ 9.8.

```
using System;
class STR8
{
static void Main(string[] args)
{
string s1 = "Hello World!";
string s2 = "Hello World!";
string s3 = "HELLO WORLD!";
string s4 = "hello world!";
int n1, n2, n3, n4;
n1 = String.Compare(s1, s2); //s1 = s2; n1 = 0
n2 = String.Compare(s1, s3); //s1 < s3; n2 = -1
```

```
    n3 = String.Compare(s1, s4); //s1 > s4; n3 = 1
    n4 = String.CompareOrdinal(s1, "hello world?");
    //s1 < s4; n4 = -32
    Console.WriteLine("n1 = "+n1+"; n2 = "+n2);
    Console.WriteLine("n3 = "+n3+"; n4 = "+n4);
    Console.WriteLine(String.CompareOrdinal(s1, s3));
    Console.WriteLine(String.Equals(s1, s3));
    Console.WriteLine(String.Equals(s1, s2));
    Console.WriteLine(s1.CompareTo(s2));
    Console.ReadLine();
    }
    }
```

 *Kết quả:*

n1 = 0; n2 = -1

n3 = 1; n4 = -32

32

False

True

0

9.9. PHƯƠNG THỨC XÁC ĐỊNH CHUỖI CON ĐẦU VÀ CUỐI

Đổi thành chữ hoa và chữ thường.

Cho các chuỗi s1 = "Hello World";

s2 = "Hello";

s3 = "World";

Phương thức s1.StartsWith(s2); cho phép ta so sánh chuỗi s2 với từ đầu tiên của chuỗi s1. Nếu chúng bằng nhau sẽ trả về giá trị true. Nếu chúng khác nhau sẽ trả về giá trị false.

Phương thức s1.ToUpper(); sẽ đổi toàn bộ ký tự của chuỗi s1 thành ký tự hoa.

Phương thức s1.ToLower(); sẽ đổi toàn bộ ký tự của chuỗi s1 thành

ký tự thường.

Ví dụ 9.9.

```
using System;
class STR9
{
static void Main(string[] args)
{
string s1 = "Hello World";
string s2 = "Hello";
string s3 = "World";
Console.WriteLine(s1.StartsWith(s2));
Console.WriteLine(s1.EndsWith(s2));
Console.WriteLine(s1.StartsWith(s3));
Console.WriteLine(s1.EndsWith(s3));
Console.WriteLine(s1.ToUpper());
Console.WriteLine(s1.ToLower());
Console.ReadLine();
}
}
```

Kết quả:

```
True
False
False
True
HELLO WORLD
hello world
```

9.10. PHƯƠNG THỨC XÁC ĐỊNH VỊ TRÍ KÝ TỰ TRONG CHUỖI

`IndexOf(), LastIndexOf().`

Cho chuỗi s1 = "Hello World";

Phương thức s1.IndexOf('o'); cho ta biết ký tự 'o' đầu tiên trong chuỗi s1 có vị trí n (tính vị trí đầu tiên là 0).

Phương thức s1.LastIndexOf('l'); cho ta biết ký tự 'l' cuối cùng của chuỗi s1 có vị trí m.

Ví dụ 9.10.

```
using System;
class STR10
{
static void Main(string[] args)
{
string s1 = "Hello World";
Console.WriteLine("Vi tri ky tu o: "+s1.IndexOf('o'));
Console.WriteLine("Vi tri l: "s1.LastIndexOf('l'));
Console.ReadLine();
}
}
```

Kết quả:

Vi tri ky tu o: 4

Vi tri l: 9

Sau đây là một ví dụ kết hợp nhiều hàm xử lý chuỗi để tách một chuỗi có dấu \ và dấu chấm (.) thành các mẩu có chức năng khác nhau.

Ví dụ 9.11.

```
using System;
class STR11
{
static void Main(string[] args)
{
Console.WriteLine("Enter string to parse");
string s = Console.ReadLine();
string Drive = s.Substring(0,s.IndexOf("\\"));
string Path = s.Remove(s.LastIndexOf(@"\"),
s.Lengths.LastIndexOf(@"\"));
string filename = s.Substring(s.LastIndexOf(@"\")+1);
string extension = filename.Substring(filename.IndexOf("."));
```

```
        Console.WriteLine("Drive: {0,15}  Path: {1}\nFilename:
{2,12}  Extension: {3}",Drive,Path,filename,extension);
        Console.ReadLine();
        }
        }
```

 *Kết quả:*

Enter string to parse
c:\CS0807\Data\file1.dat
Drive: c Path: c:\CS0807\Data
Filename: file1.dat Extension: dat

Chương 10. MULTITHREADING (Đa luồng)

Luồng (thread) là trình tự thực hiện trong chương trình. Tất cả các chương trình C# cho đến nay chỉ có một điểm vào (entry point) – hàm Main(). Việc thực hiện chương trình được bắt đầu với câu lệnh đầu tiên của hàm Main() và tiếp tục cho tới lúc phương thức được trả về. Cấu trúc chương trình kiểu như thế này rất hợp lý cho những chương trình trong đó chỉ có một trình tự thực hiện xác định sẵn.

Khi một chương trình cần đợi một cái gì đó xảy ra, chẳng hạn nhập thông tin vào, download tập tin từ Internet thì toàn bộ chương trình phải chờ cho tới khi công việc đó xong xuôi rồi mới tiếp tục chạy được. Chương trình được phân nhánh thành các chương trình con, chúng được gọi là luồng (thread).

Tuy nhiên thông thường một chương trình cần thực hiện nhiều việc đồng thời, ví dụ, khi khởi động trình duyệt Internet Explorer (IE) cần phải thực hiện ít nhất ba nhiệm vụ sau:

– Lấy dữ liệu cho trang đưa về từ IE.

– Trình bày trang.

– Xem xét đối với bất kỳ đầu vào nào của người sử dụng và có thể chỉ ra rằng người sử dụng muốn IE làm thứ gì khác nữa.

Tình huống tương tự có thể áp dụng cho bất cứ trường hợp nào mà chương trình thực hiện một vài việc, đồng thời lúc này vẫn hiển thị hộp thoại cho phép bạn cơ hội để huỷ bỏ nhiệm vụ hiện hành.

Một chương trình khi chạy có thể phải làm nhiều việc đồng thời. Hiểu thuật ngữ "đồng thời" không phải là ngay cùng một thời điểm (one moment) mà là một khoảng thời gian hữu hạn. Ý tưởng về luồng (thread) có nghĩa là thực hiện nhiều việc cùng khoảng thời gian có sự phân định từng mẩu thời gian nhỏ cho từng việc ("time slice" hay là "for a short of period of time"). Thông thường một máy tính chỉ có một vi xử lý và nó cũng chỉ làm được một việc ở một thời điểm nhất định. Việc một máy tính – một vi xử lý làm nhiều việc ngay lập tức, đồng thời là điều vô lý, không tưởng. Chức năng "đa nhiệm" (multitasking) cũng là một cách mà luồng

thực thi. Hiện nay do công nghệ phát triển của các vi xử lý nhiều lõi (ví dụ chip vi xử lý "core 2 dua" thì tốc độ thực hiện chương trình tăng lên đáng kể và ý tưởng về "multithread" càng phát huy mạnh mẽ. Thread đã được hệ điều hành Windows áp dụng từ phiên bản Windows 9X.

Thread được quản lý bởi lớp Thread khi thực hiện khai báo:

```
using System.Threading;
```

Sau đây ta sẽ cho một ví dụ về thread.

Ví dụ 10.1.

```
using System;
using System.Threading;
class ThreadApp
{
public static void ThreadMethod1()
{
try
{
Console.WriteLine("Thread1 bat dau khoi dong");
Console.WriteLine("Thread1 dem tu 0 den 5.");
for(int i = 0; i < 6; i++)
{
Console.WriteLine("I = {0,2}",i);
Thread.Sleep(20);
}
}
catch(Exception e)
{}
Console.WriteLine("Thread1 da hoan thanh.");
}
//-----------------------------------------------------
public static void ThreadMethod2()
{
try
```

```
    {
    Console.WriteLine("Thread2 bat dau khoi dong");
    Console.WriteLine("Thread2 dem tu 6 den 11.");
    for(int j = 6; j < 12; j++)
    {
    Console.WriteLine("\t J = {0,6}",j);
    Thread.Sleep(20);
    }
    Console.WriteLine("Thread2 da hoan thanh.");
    }
    catch(Exception e)
    {}
    }
    //-----------------------------------------------------
    public static void Main(string[] args)
    {
    ThreadStart w1 = new ThreadStart(ThreadMethod1);
    ThreadStart w2 = new ThreadStart(ThreadMethod2);
    Console.WriteLine("Phuong thuc Main tao ra cac Thread");
    Thread t1 = new Thread(w1);
    Thread t2 = new Thread(w2);
    t1.Priority = ThreadPriority.Highest;
    t2.Priority = ThreadPriority.Lowest;
    t1.Start();
    t2.Start();
    Console.ReadLine();
    }
    }
```

 *Kết quả:*

Phuong thuc Main tao cac Thread

Thread1 bat dau khoi dong

Thread1 dem tu 0 den 5.

I = 0

Thread2 bat dau khoi dong

Thread2 dem tu 6 den 11.

J = 6

I = 1

J = 7

I = 2

J = 8

I = 3

J = 9

I = 4

J = 10

I = 5

J = 11

Thread1 da hoan thanh.

Thread2 da hoan thanh.

```
//-------------------------------------------------
```

Trong ví dụ trên, các đối tượng w1 và w2 được khai báo là các đối tượng của lớp ThreadStart. Các đối tượng t1 và t2 được khai báo là các đối tượng của lớp Thread.

Phương thức Start() dùng để khởi động các đối tượng.

Phương thức Thread.Sleep(20); để tạo thời gian trễ giữa các Thread1 và Thread2. Nếu tăng thời gian từ 20 lên 1000 thì quá trình diễn ra khá chậm (1sec) và ta có thể nhìn thấy kết quả biến đổi chậm trên màn hình. Nếu ta giảm từ 20 xuống 1 thì quá trình thay đổi rất nhanh và phương thức Thread2 có thể bắt đầu sau vài bước tính toán của Thread1.

Bây giờ nếu ta không sử dụng phương thức Thread.Sleep(20); và bỏ qua câu lệnh ưu tiên:

```
t1.Priority = ThreadPriority.Highest;
t2.Priority = ThreadPriority.Lowest;
```

Khi đó ta sẽ có kết quả khác với kết quả của ví dụ 1.

Ví dụ 10.2.

```
using System;
using System.Threading;
class ThreadApp
{
public static void ThreadMethod1()
{
try
{
Console.WriteLine("Thread1 bat dau khoi dong.");
Console.WriteLine("Thread1 dem tu 0 den 5.");
for(int i = 0; i < 6; i++)
{
Console.WriteLine("I = {0,2}",i);
}
}
catch(Exception e)
{}
Console.WriteLine("Thread1 da hoan thanh.");
}
//-----------------------------------------------------
public static void ThreadMethod2()
{
try
{
Console.WriteLine("Thread2 bat dau khoi dong.");
Console.WriteLine("Thread2 dem tu 6 den 11.");
for(int j = 6; j < 12; j++)
{
Console.WriteLine("\t J = {0,6} ",j);
}
Console.WriteLine("Thread2 da hoan thanh.");
```

```
}
catch(Exception e)
{}
}
//-----------------------------------------------------
public static void Main(string[] args)
{
ThreadStart w1 = new ThreadStart(ThreadMethod1);
ThreadStart w2 = new ThreadStart(ThreadMethod2);
Console.WriteLine("Phuong thuc Main tao ra cac Thread");
Thread t1 = new Thread(w1);
Thread t2 = new Thread(w2);
//t1.Priority = ThreadPriority.Highest;
//t2.Priority = ThreadPriority.Lowest;
t1.Start();
t2.Start();
Console.ReadLine();
}
}
```

 *Kết quả:*

Phuong thuc Main tao cac Thread
Thread1 bat dau khoi dong.
Thread1 dem tu 0 den 5.
I = 0
I = 1
I = 2
I = 3
Thread2 bat dau khoi dong.
Thread2 dem tu 6 den 11.
J = 6
J = 7

```
J = 8
J = 9
J = 10
J = 11
Thread2 da hoan thanh.
I = 4
I = 5
Thread1 da hoan thanh.
//-------------------------------------------------
```

Ở hai ví dụ trên, hai phương thức được sử dụng đều là các phương thức static. Chương trình trên chưa phải là chương trình theo phong cách hướng đối tượng. Sau đây ta sẽ nêu lên chương trình sử dụng Thread viết hoàn toàn theo phong cách hướng đối tượng.

Ví dụ 10.3.

```
using System;
using System.Threading;
class ThreadOOP
{
public void ThreadMethod1()
{
try
{
Console.WriteLine("Thread1 bat dau khoi dong.");
Console.WriteLine("Thread1 dem tu 0 den 5.");
for(int i = 0; i < 6; i++)
{
Console.WriteLine("i = {0,2}", i);
Thread.Sleep(20);
}
}
catch(Exception e)
{}
```

```
Console.WriteLine("Thread1 da hoan thanh.");
}
//-------------------------------------------------
public void ThreadMethod2()
{
try
{
Console.WriteLine("Thread2 bat dau khoi dong.");
Console.WriteLine("Thread2 dem tu 6 den 11.");
for(int j = 6; j < 12; j++)
{
Console.WriteLine("\t j= {0,6} ",j);
Thread.Sleep(20);
}
Console.WriteLine("Thread2 da hoan thanh.");
}
catch(Exception e)
{}
}
//-------------------------------------------------
public static void Main(string [] args)
{
ThreadOOP ob = new ThreadOOP();
ThreadStart w1 = new ThreadStart(ob.ThreadMethod1);
ThreadStart w2 = new ThreadStart(ob.ThreadMethod2);
Console.WriteLine("Phuong thuc Main tao ra cac Thread");
Thread t1 = new Thread(w1);
Thread t2 = new Thread(w2);
t1.Priority = ThreadPriority.Highest;
t2.Priority = ThreadPriority.Lowest;
t1.Start();
t2.Start();
```

```
    Console.ReadLine();
    }
    }
```

 *Kết quả:*

```
Phuong thuc Main tao cac Thread
Thread1 bat dau khoi dong.
Theard1 dem tu 0 den 5.
I = 0
Thread2 bat dau khoi dong.
Theard2 dem tu 6 den 11.
J = 6
I = 1
J = 7
I = 2
J = 8
I = 3
J = 9
I = 4
J = 10
I = 5
J = 11
Thread1 da hoan thanh.
Thread2 da hoan thanh.
```

 *Ví dụ 10.4.* Sau đây ta sẽ trình bày một ví dụ về việc sử dụng Thread để tính toán các biểu thức toán học.

```
Fx = 2.x.cos(15.x)
Gx = 7.x^3.sin(15.x)
Hx = 7.x^3.tan(15.x)
using System;
using System.Threading;
```

```
    class ThreadCal
    {
    public void Fx()
    {
    for(int i = -10; i < 16; i++)
    {
    Console.WriteLine("Fx = {0:0.00,2}",
2*i*Math.Cos(15*i*Math.PI/180));
    Thread.Sleep(20);
    }
    Console.WriteLine("Thread1 da hoan thanh.");
    }
    //-------------------------------------------------
    public void Gx()
    {
    for(int i = -10; i < 16; i++)
    {
    Console.WriteLine("Gx = {0:0.00,6}",
7*i*Math.Pow(i,3)*Math.Sin(15*i*Math.PI/180));
    Thread.Sleep(20);
    }
    Console.WriteLine("Thread2 da hoan thanh.");
    }
    //-------------------------------------------------
    public void Hx()
    {
    for(int i = -10; i < 16; i++)
    {
    Console.WriteLine("Hx = {0:0.00,2}",
2*i*Math.Tan(15*i*Math.PI/180));
    Thread.Sleep(20);
    }
    Console.WriteLine("Thread3 da hoan thanh.");
```

```
}
//-------------------------------------------------------
public static void Main()
{
ThreadCal ob = new ThreadCal();
ThreadStart f1 = new ThreadStart(ob.Fx);
ThreadStart g1 = new ThreadStart(ob.Gx);
ThreadStart h1 = new ThreadStart(ob.Hx);
Console.WriteLine("Ham Main tao ra cac Thread");
Thread t1 = new Thread(f1);
Thread t2 = new Thread(g1);
Thread t3 = new Thread(h1);
t1.Start();
t2.Start();
t3.Start();
Console.ReadLine();
}
}
```

 *Kết quả:*

```
Ham Main tao ra cac Thread
Fx = 17.322
Gx = -35000.026
Hx = -11.552
Fx = 12.732
Gx = -32475.296
Hx = -18.002
Thread2 da hoan thanh.
Thread1 da hoan thanh.
Thread3 da hoan thanh.
```

Chương 11. Lập trình WINDOWS với C#

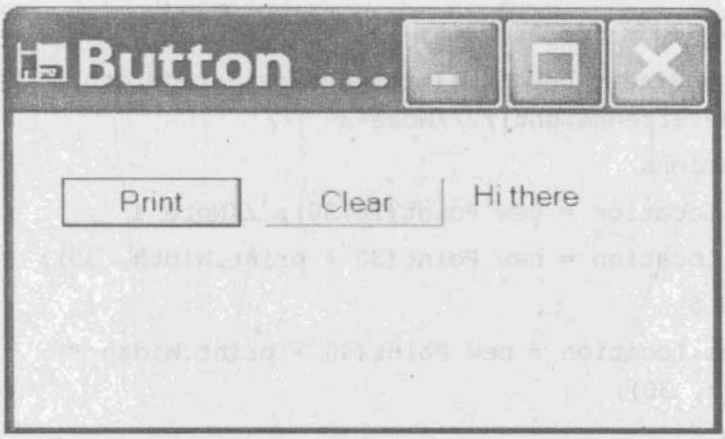
Cũng như bộ MS Visual Studio 6.0, đối với bộ MS Visual Studio 7.0 hoặc 8.0 (chính là bộ Visual Studio.NET) đều lấy lập trình Windows làm trung tâm. Có hai phương pháp chính cho việc lập trình Windows:

– Sử dụng các câu lệnh trong khai báo.

– Sử dụng các biểu tượng theo kiểu kéo – thả (controls).

11.1. LẬP TRÌNH WINDOWS THEO KIỂU KHAI BÁO

Ví dụ 11.1. Tạo một cửa sổ có hai nút nhấn với nhãn Print và Clear. Khi chương trình chạy, ta chưa nhấn chuột vào nút nào cả, bên phải sẽ hiển thị dòng chữ: "Message goes here". Nếu ta nhấn chuột vào nút Print, bên phải sẽ hiển thị "Hi There", còn nếu nhấn vào nút Clear sẽ xoá dòng: "Hi There". Cửa sổ khi chạy chương trình như sau đây.



Hình 11.1

Đây là mã chương trình.

```
/* A print button prints a message.
 * A clear button erases the message.
 * We use a Label to hold the message.
 */
using System;
using System.Drawing;
```

```
    using System.Windows.Forms;
    public class ButtonPress: Form
    {
    // Create the controls
    private Button print = new Button(); //Note 1
    private Button clear = new Button();
    private Label message = new Label();
    public ButtonPress()
    {
    // Captions
    Text = "Button Press";
    print.Text = "Print"; //Note 2
    clear.Text = "Clear";
    message.Text = "Message goes here";
    // Sizes
    Size = new Size(300,200);
    message.Size = new Size(message.PreferredWidth,
message.PreferredHeight); //Note 3
    //Locations
    print.Location = new Point(20,30); //Note 4
    clear.Location = new Point(30 + print.Width, 30);
    //Note 5
    message.Location = new Point(40 + print.Width +
clear.Width, 30);
    //Note 6
    //Add them to the form
    Controls.Add(print); //Note 7
    Controls.Add(clear);
    Controls.Add(message);
    //Tell the Click events which methods to call
    print.Click+ = new EventHandler(Print_Click);
    clear.Click+ = new EventHandler(Clear_Click);
    }
```

```
//method to make the Print button work
protected void Print_Click(Object sender, EventArgs e)
{
//Note 9
message.Text = "Hi there";
}
// method to make the Clear button work
protected void Clear_Click(Object sender, EventArgs e)
{
message.Text = "";
}
//-------------------------------------------------
static void Main()
{
Application.Run(new ButtonPress());
}
}
```

Phân tích chương trình.

Đầu chương trình ta phải khai báo các tệp tiêu đề:

```
using System;
using System.Drawing;
using System.Windows.Forms;
```

Sau đó đến việc định nghĩa lớp ButtonPress, thừa kế lớp Form.

Trong lớp này định nghĩa 3 biến: nút nhấn Print và nút nhấn Clear thuộc lớp Button; nhãn message thuộc lớp Label.

```
private Button print = new Button();
private Button clear = new Button();
private Label message = new Label();
```

Hàm tạo của lớp (constructor) khởi tạo tên và gán giá trị cho các đối tượng này.

```
public ButtonPress()
{ }
```

Trong hàm này cũng xác định kích thước cho nhãn và vị trí các nút.

Để cho phép nhấn nút Print rồi hiển thị thông điệp, ta phải định nghĩa phương thức:

```
protected void Print_Click(Object sender, EventArgs e)
{
message.Text = "Hi there";
}
```

Để cho phép nhấn nút Print rồi hiển thị thông điệp, ta phải định nghĩa phương thức:

```
protected void Clear_Click(Object sender, EventArgs e)
{
message.Text = "";
}
```

Trong hàm constructor còn khởi tạo cho sự kiện nhấn nút:

```
print.Click+ = new EventHandler(Print_Click);
clear.Click+ = new EventHandler(Clear_Click);
```

Cuối cùng ta phải định nghĩa hàm Main() như bất kỳ chương trình C# nào nhưng với câu lệnh chỉ riêng cho chương trình Windows.

```
static void Main()
{
Application.Run(new ButtonPress());
}
```

Qua ví dụ vừa trình bày, để thực hiện một vài chức năng đơn giản như vậy mà ta cũng phải viết mã chương trình tương đối phức tạp.

Bây giờ ta sẽ trình bày phương pháp lập trình Windows dựa vào các biểu tượng kéo – thả (sử dụng Wizard).

11.2. LẬP TRÌNH WINDOWS THEO ĐIỀU KHIỂN THƯ VIỆN

Để tạo giao diện Windows, ta thực hiện các bước sau:

Start MS Visual Studio .NET

Chọn File → Chọn New → Project

Trên hình 11.3, chọn Visual C# Project trong Project Types, chọn

Windows Application trong Templates Name, ta đặt tên: WinCalculate. Trong Location, chọn thư mục tùy ý, ví dụ CS_toshiba. Nhấn phím OK. Sau đó ta có một cửa sổ trống.

Ví dụ 11.2. Để tạo ra một giao diện theo yêu cầu trong cửa sổ trông này, ví dụ thực hiện việc tính toán của hai số theo các phép tính số học: cộng, trừ, nhân, chia và chia lấy phần dư.

Xây dựng giao diện như hình 11.4, ta kéo các điều khiển tương ứng ở hình 11.2 vào và sắp xếp theo yêu cầu người sử dụng.

Hình 11.2

Với ví dụ trên ta có 4 Textbox; 4 Label; 2 Button; 5 RadioButton.

Khi nhập vào 2 Textbox trái và phải các số của các toán hạng. Sau đó chọn một một nút RadioButton và nhấn vào nút Calculate. Kết quả trong Textbox Expression sẽ hiện lên biểu thức, chẳng hạn 5/4 và trong Textbox Result sẽ hiển thị 1.25.

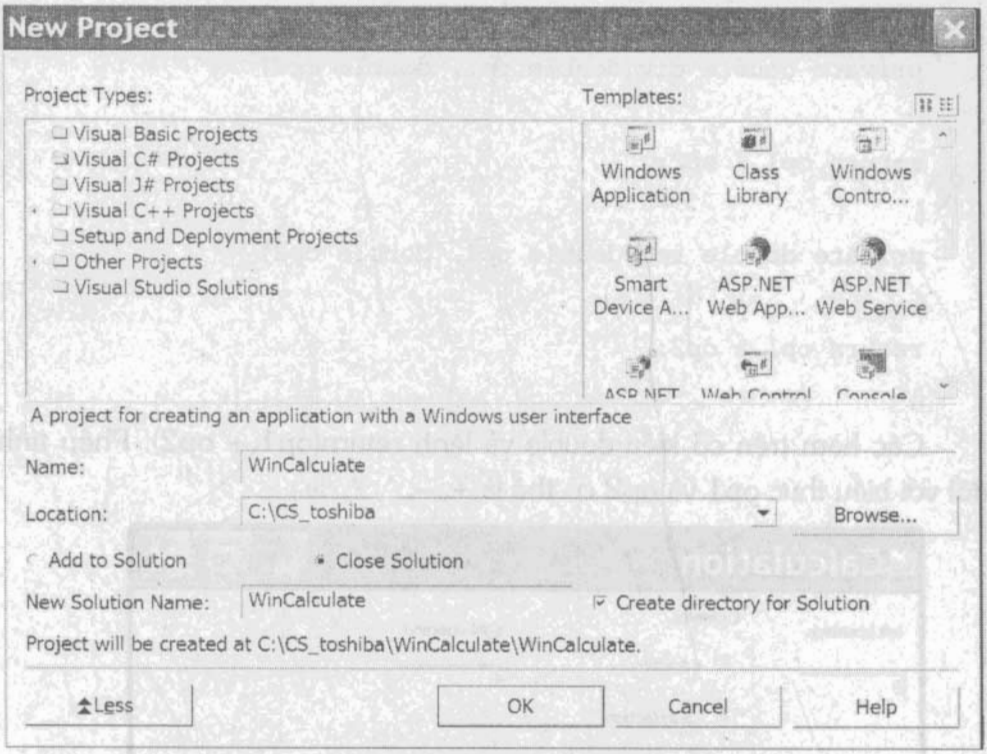
Để viết mã lệnh chương trình ta nhấn đúp vào nút Calculate và sẽ có phương thức.

```
    private void butCal_Click(object sender,
System.EventArgs e)
    {
    double lo = 0, ro = 0, res = 0;
    lo = double.Parse(txtlo.Text);
    ro = double.Parse(txtro.Text);
    if(radAdd.Checked)
    {
    res = add(lo, ro);
    txtexp.Text = lo.ToString() + " + " + ro.ToString();
    }
    else if(radSub.Checked)
```

```
{
res = sub(lo, ro);
txtexp.Text = lo.ToString() + " - " +
ro.ToString();
}
else if(radMul.Checked)
{
res = mul(lo,ro);
txtexp.Text = lo.ToString() + " * " + ro.ToString();
}
else if(radDiv.Checked)
{
res = div(lo, ro);
txtexp.Text = lo.ToString() + " / " +
ro.ToString();
}
else
{
res = rem(lo,ro);
txtexp.Text = lo.ToString() + " % " +
ro.ToString();
}
txtresult.Text = res.ToString();
}
```

Muốn thoát khỏi chương trình, ta nhấn đúp vào nút Quit và viết mã chương trình rất đơn giản:

```
private void butQuit_Click(object sender, System.EventArgs e)
{
this.Close();
}
```



Hình 11.3

Trong phương thức:

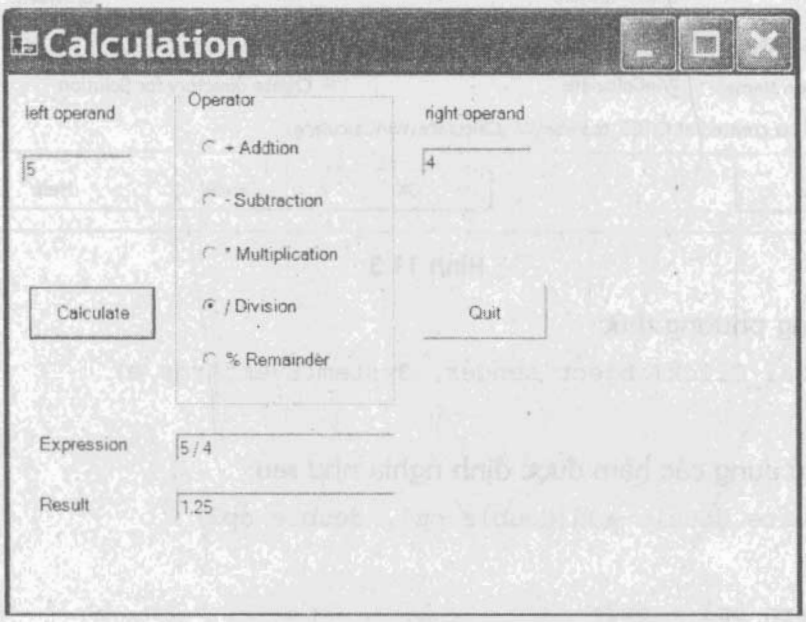
```
butCal_Click(object sender, System.EventArgs e)
{ }
```

có sử dụng các hàm được định nghĩa như sau:

```
private double add(double op1, double op2)
{
return op1 + op2;
}
private double sub(double op1, double op2)
{
return op1 -op2;
}
private double mul(double op1, double op2)
{
return op1 * op2;
```

```
}
private double div(double op1, double op2)
{
return op1 / op2;
}
private double rem(double op1, double op2)
{
return op1 % op2;
}
```

Các hàm trên có kiểu double và lệnh return(op1 + op2). Phép tính đối với biểu thức op1 và op2 có thể là +, –, *, /, %.



Hình 11.4

Trong bảng Tool Control có khá nhiều công cụ điều khiển.

Ta lần lượt xét một số control điển hình trong các ví dụ sau đây.

Ví dụ 11.3. Sử dụng các Textbox, Listbox và Button để tạo giao diện tính biểu thức a.x^3 + b.x^2 + c với các hệ số a, b, c dương.

Nhấn nút Calculate để tính giá trị biểu thức và hiển thị các hệ số cũng như kết quả ở các Textbox và trong Listbox. Xem hình 11.5.

Các thuộc tính của các điều khiển:

```
Textbox1: Name: tbA
Textbox2: tbB
Textbox3: tbC
Textbox4: tbX
Textbox 5: tbResult
Đặt thuộc tính ReadOnly:True)
listBox1: lbDisplay
button1: btCal
```

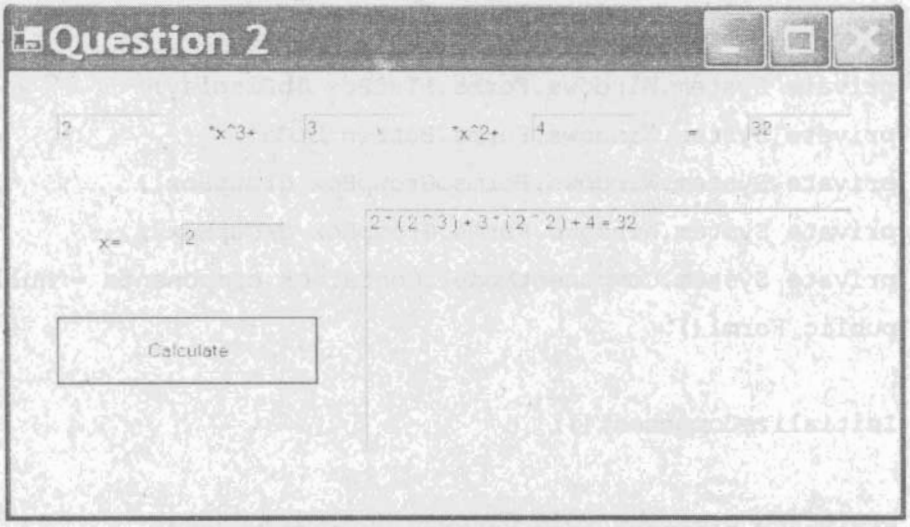
Để viết mã lệnh cho chương trình, từ giao diện ta nhấn đúp vào nút Calculate và có mã lệnh cho phương thức:

```
private void btCal_Click(object sender, System.EventArgs e)
{ }
```

như đã liệt kê dưới đây.

Ngoài ra, ta còn định nghĩa một phương thức:

```
private Double Cal(double a, double b, double c, double x)
{
return a*Math.Pow(x,3)+b*Math.Pow(x,2)+c;
}
```



Hình 11.5

Mã chương trình của ví dụ 11.3 như sau:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
namespace Exam_Question_2
{
public class Form1: System.Windows.Forms.Form
{
private System.Windows.Forms.Textbox tbA;
private System.Windows.Forms.Textbox tbB;
private System.Windows.Forms.Textbox tbC;
private System.Windows.Forms.Textbox tbResult;
private System.Windows.Forms.Textbox tbX;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.ListBox lbDisplay;
private System.Windows.Forms.Button btCal;
private System.Windows.Forms.GroupBox groupBox1;
private System.Windows.Forms.GroupBox groupBox2;
private System.ComponentModel.Container components = null;
public Form1()
{
InitializeComponent();
}
protected override void Dispose(bool disposing)
{
```

```
    if(disposing)
    {
    if (components != null)
    {
    components.Dispose();
    }
    }
    base.Dispose(disposing);
    }
    [STAThread]
    static void Main()
    {
    Application.Run(new Form1());
    }
    private Double Cal(double a, double b, double c, double x)
    {
    return a*Math.Pow(x,3) + b*Math.Pow(x,2) + c;
    }
    private void btCal_Click(object sender, System.EventArgs e)
    {
    try
    {
    double a = Double.Parse(tbA.Text);
    double b = Double.Parse(tbB.Text);
    double c = Double.Parse(tbC.Text);
    double x = Double.Parse(tbX.Text);
    if(a > 0 && b > 0 && c > 0)
    {
    tbResult.Text = Cal(a, b, c, x).ToString();
    string Display = tbA.Text+" * ("+tbX.Text+"^3) +
"+tbB.Text+" * ("+tbX.Text+"^2) + "+tbC.Text+" = "+Cal(a, b,
c, x).ToString();
```

```
lbDisplay.Items.Add(Display);
}
else
MessageBox.Show("a, b, c must be larger than 0");
}
catch(Exception ex)
{
MessageBox.Show("Errors: "+ex.Message);
}
}
}
}
```

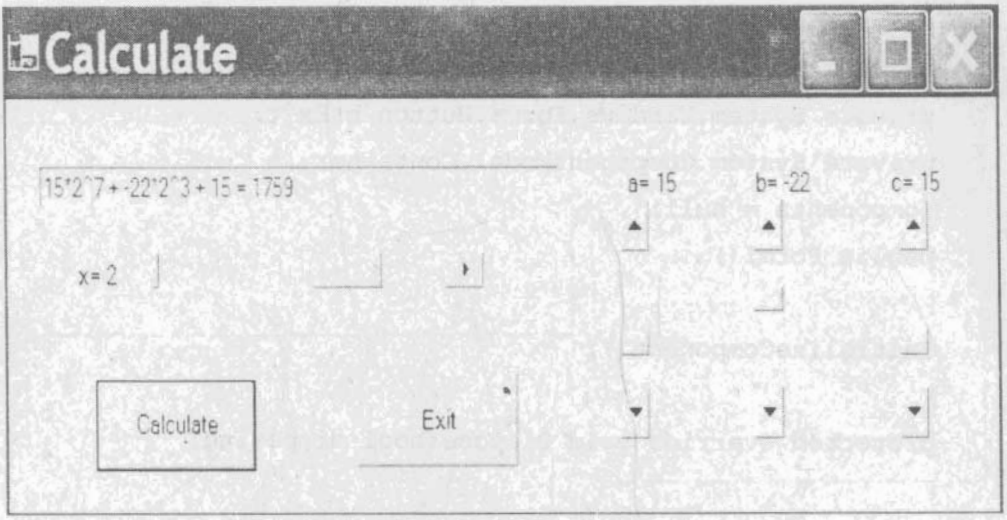
Sau khi chạy chương trình và nhập số liệu vào các Textbox rồi nhấn nút Calculate ta có kết quả như trên hình 11.5.

Ví dụ 11.4. Xây dựng một giao diện có các control: 1 HScollBar, 3 VScrollBar, 1 Textbox, 2 Button, 4 Label như trên hình 11.6.

Khi xác định thuộc tính ta cần đặt giá trị max = 50 và giá trị min = –50.

Các thuộc tính của các điều khiển:

1. Textbox1: name: tbDisplay.
2. Button1: btCalculate
3. Button2: btExit
4. Label1: lblA
5. Label2: lblB
6. Label3: lblC
7. Label4: lblX
8. HScollBar1: sbX
9. VScrollBar1: sbA
10. VScrollBar2: sbB
11. VScrollBar3: sbC



Hình 11.6.

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
namespace Exam_Question_3
{
public class Form1: System.Windows.Forms.Form
{
private System.Windows.Forms.GroupBox groupBox1;
private System.Windows.Forms.GroupBox groupBox2;
private System.Windows.Forms.Button btCalculate;
private System.Windows.Forms.VScrollBar sbA;
private System.Windows.Forms.VScrollBar sbB;
private System.Windows.Forms.VScrollBar sbC;
private System.Windows.Forms.HScrollBar sbX;
private System.Windows.Forms.Label lblA;
private System.Windows.Forms.Label lblB;
private System.Windows.Forms.Label lblC;
```

```
private System.Windows.Forms.Label lblX;
private System.Windows.Forms.Textbox tbDisplay;
private System.Windows.Forms.Button btExit;
private System.ComponentModel.Container
components = null;
public Form1()
{
InitializeComponent();
}
protected override void Dispose(bool disposing)
{
if(disposing)
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}
#region Windows Form Designer generated code
#endregion
[ STAThread]
static void Main()
{
Application.Run(new Form1());
}
private double cal(int a, int b, int c, int x)
{
return (a*Math.Pow(x,7)+b*Math.Pow(x,3)+c);
}
private void btCalculate_Click(object sender, System.EventArgs e)
```

```
    {
    int a = sbA.Value;
    int b = sbB.Value;
    int c = sbC.Value;
    int x = sbX.Value;
    tbDisplay.Text = a.ToString()+" * "+x.ToString()+"^7"+" +
"+b.ToString()+"*"+x.ToString()+"^3"+"+"+c.ToString()+" =
"+cal(a, b, c, x).ToString();
    }
    private void Form1_Load(object sender, System.EventArgs e)
    {
    lblA.Text = "a = "+sbA.Value.ToString();
    lblB.Text = "b = "+sbB.Value.ToString();
    lblC.Text = "c = "+sbC.Value.ToString();
    lblX.Text = "x = "+sbX.Value.ToString();
    }
    private void sbA_Scroll(object sender,
System.Windows.Forms.ScrollEventArgs e)
    {
    lblA.Text = "a = "+sbA.Value.ToString();
    }
    private void sbB_Scroll(object sender,
System.Windows.Forms.ScrollEventArgs e)
    {
    lblB.Text = "b = "+sbB.Value.ToString();
    }
    private void sbC_Scroll(object sender,
System.Windows.Forms.ScrollEventArgs e)
    {
    lblC.Text = "c = "+sbC.Value.ToString();
    }
    private void sbX_Scroll(object sender,
System.Windows.Forms.ScrollEventArgs e)
```

```
{
lblX.Text = "x = "+sbX.Value.ToString();
}
private void btExit_Click(object sender, System.EventArgs e)
{
Close();
}
}
}
```

Ví dụ 11.5. Sau đây ta sẽ trình bày một ví dụ về giao diện nhập dữ liệu từ các Textbox và thực hiện các nhiệm vụ:

– Nhấn vào nút Add, dữ liệu từ các Textbox sẽ được chuyển sang Listbox.

– Nhấn nút Clear để xoá thông tin trong các Textbox.

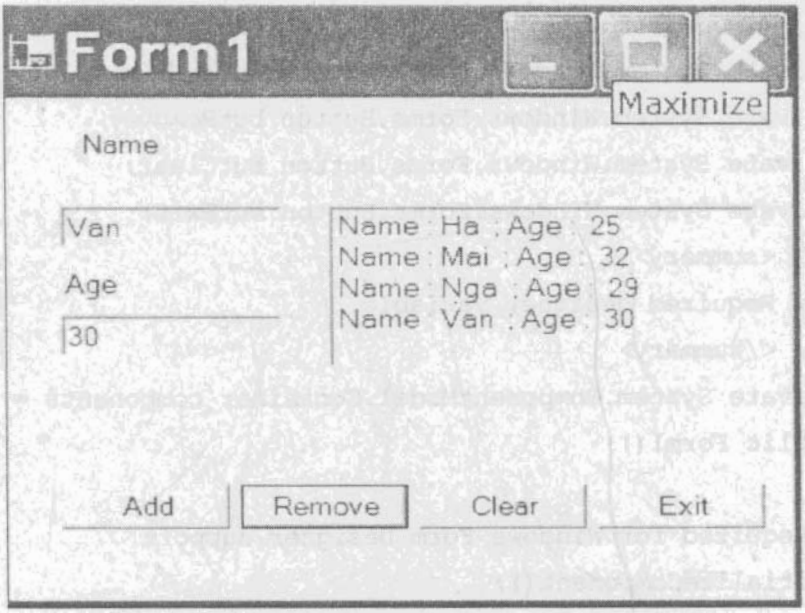
Trước hết ta phải chọn một hàng (bản ghi) bất kỳ trong Listbox, sau đó nhấn nút Remove, bản ghi vừa chọn sẽ bị xoá khỏi Listbox.

– Thoát khỏi chương trình, nhấn nút Exit.

Giao diện được trình bày như hình 11.7.

Các thuộc tính điều khiển như sau:

1. Textbox1: Name tbName
2. Textbox2 tbAge
3. Label1 lbName
4. Label2 lbAge
5. listBox1 giữ nguyên
6. button1 butAdd
7. button2 butRemove
8. button3 butClear
9. button4 butExit



Hình 11.7

Mã chương trình của ví dụ trên được trình bày như sau.

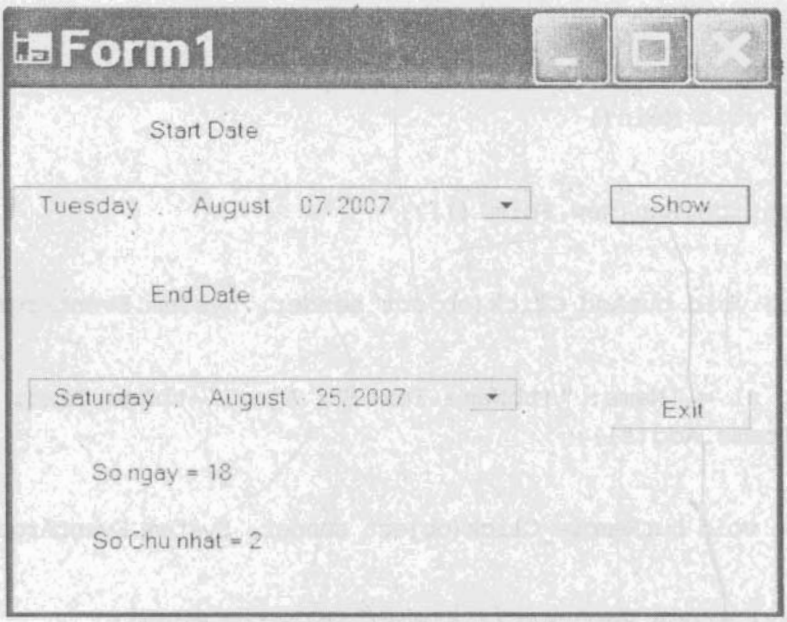
```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
namespace WinAppListBox
{
/// <summary>
/// Summary description for Form1.
/// </summary>
public class Form1: System.Windows.Forms.Form
{
private System.Windows.Forms.Label lbName;
private System.Windows.Forms.Listbox listBox1;
private System.Windows.Forms.Textbox tbName;
private System.Windows.Forms.Label lbAge;
```

```
private System.Windows.Forms.Textbox tbAge;
private System.Windows.Forms.Button butAdd;
private System.Windows.Forms.Button butRemove;
private System.Windows.Forms.Button butClear;
private System.Windows.Forms.Button butExit;
/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.Container components = null;
public Form1()
{
//Required for Windows Form Designer support
InitializeComponent();
//TODO: Add any constructor code after
//InitializeComponent call
}
// <summary>
// Clean up any resources being used.
// </summary>
protected override void Dispose(bool disposing)
{
if(disposing)
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}
/// <summary>
/// The main entry point for the application
```

```
    /// </summary>
    [STAThread]
    static void Main()
    {
    Application.Run(new Form1());
    }
    private void butAdd_Click(object sender, System.EventArgs e)
    {
    string s1 = "Name: "+tbName.Text+"; Age: "+tbAge.Text;
listBox1.Items.Add(s1);
    }
    private void butRemove_Click(object sender, System.EventArgs e)
    {
    listBox1.Items.RemoveAt(listBox1.SelectedIndex);
    }
    private void butExit_Click(object sender, System.EventArgs e)
    {
    Close();
    }
    private void butClear_Click(object sender, System.EventArgs e)
    {
    tbNmae.Text = "";
    tbAge.Clear();
    }
    }
    }
```

Ví dụ 11.6. Hãy xây dựng một giao diện Windows để tính số ngày và số chủ nhật nếu nhập ngày/tháng/năm từ điều khiển dateTimePicker1 (ngày bắt đầu) đến ngày/tháng/năm trong điều khiển dateTimePicker2 (ngày kết thúc).

Giao diện biểu diễn trên hình 11.8.



Hình 11.8

Các thuộc tính điều khiển như sau:

1. Label1:Name Start Date
2. Label2 End Date
3. Label3 Số ngày lbnumday
4. Label4 số ngày chủ nhật lbnumsunday
5. button1 Show(Text); butShow (name)
6. button2 Exit (Text); butExit (name)
7. dateTimePicker1
8. dateTimePicker2

Trong phương thức:

```
public void butShow_Click(object sender, System.EventArgs e)
{ }
```

Có sử dụng lớp TimeSpan để tính số ngày giữa hai mốc thời gian:

```
TimeSpan t = DateTime.Parse(dateTimePicker2.
Value.ToShortDateString()) -DateTime.Parse(dateTimePicker1.
Value.ToShortDateString());
using System;
using System.Drawing;
```

```
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
namespace CalendarPicker
{
public class Form1: System.Windows.Forms.Form
{
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.DateTimePicker dateTimePicker1;
private System.Windows.Forms.DateTimePicker dateTimePicker2;
private System.Windows.Forms.Label lbnumday;
private System.Windows.Forms.Label lbnumsunday;
private System.Windows.Forms.Button butExit;
private System.Windows.Forms.Button butShow;
private System.ComponentModel.Container components = null;
public Form1()
{
InitializeComponent();
}
protected override void Dispose(bool disposing)
{
if(disposing)
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}
```

```
    #region Windows Form Designer generated code
    [STAThread]
    static void Main()
    {
    Application.Run(new Form1());
    }
    private void butExit_Click(object sender, System.EventArgs e)
    {
    this.Close();
    }
    private void butShow_Click(object sender, System.EventArgs e)
    {
    TimeSpan t = DateTime.Parse(dateTimePicker2.Value.
ToShortDateString()) -DateTime.Parse(dateTimePicker1.Value.
ToShortDateString());
    DateTime ts = DateTime.Parse(dateTimePicker1.Text);
    int numSunday = 0;
    if(ts.DayOfWeek.ToString() == "Sunday")
    numSunday++;
    for(int i = 0; i < t.Days; i++)
    {
    ts = ts.AddDays(1.0);
    if(ts.DayOfWeek.ToString() == "Sunday")
    numSunday++;
    }
    lbnumday.Text = "So ngay = "+t.Days.ToString();
    lbnumsunday.Text = "So Chu nhat = "+numSunday.ToString();
    }
    }
    }
```

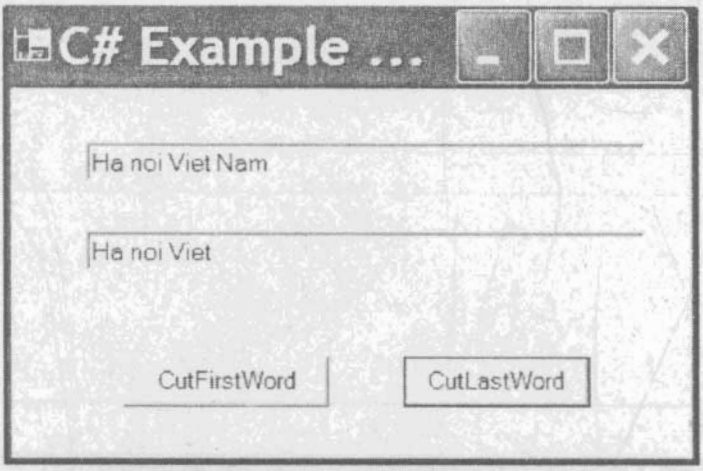
Ví dụ 11.7. Đây là một ví dụ đặc biệt có nhiệm vụ phải tạo một class, ví dụ có tên là "cStrX" trong đó định nghĩa hai phương thức:

Phương thức thứ nhất dùng để cắt từ đầu tiên của một chuỗi. Ví dụ có chuỗi nhập vào: s1 = "Ha noi Viet Nam" thì sau khi phương thức này hoạt động, ta chỉ thu được chuỗi con: "noi Viet Nam".

Phương thức thứ hai dùng để cắt từ cuối cùng của chuỗi. Ví dụ chuỗi s1, sau khi phương thức này hoạt động, ta thu được chuỗi nhỏ: "Ha Noi Viet".

Trên hình 11.9, sau khi nhập Textbox1: "Ha Noi Viet Nam",

Sau đó nhấn nút CutLastWord, ở Textbox2 hiển thị: "Ha Noi Viet". Từ cuối cùng "Nam" đã bị cắt đi.



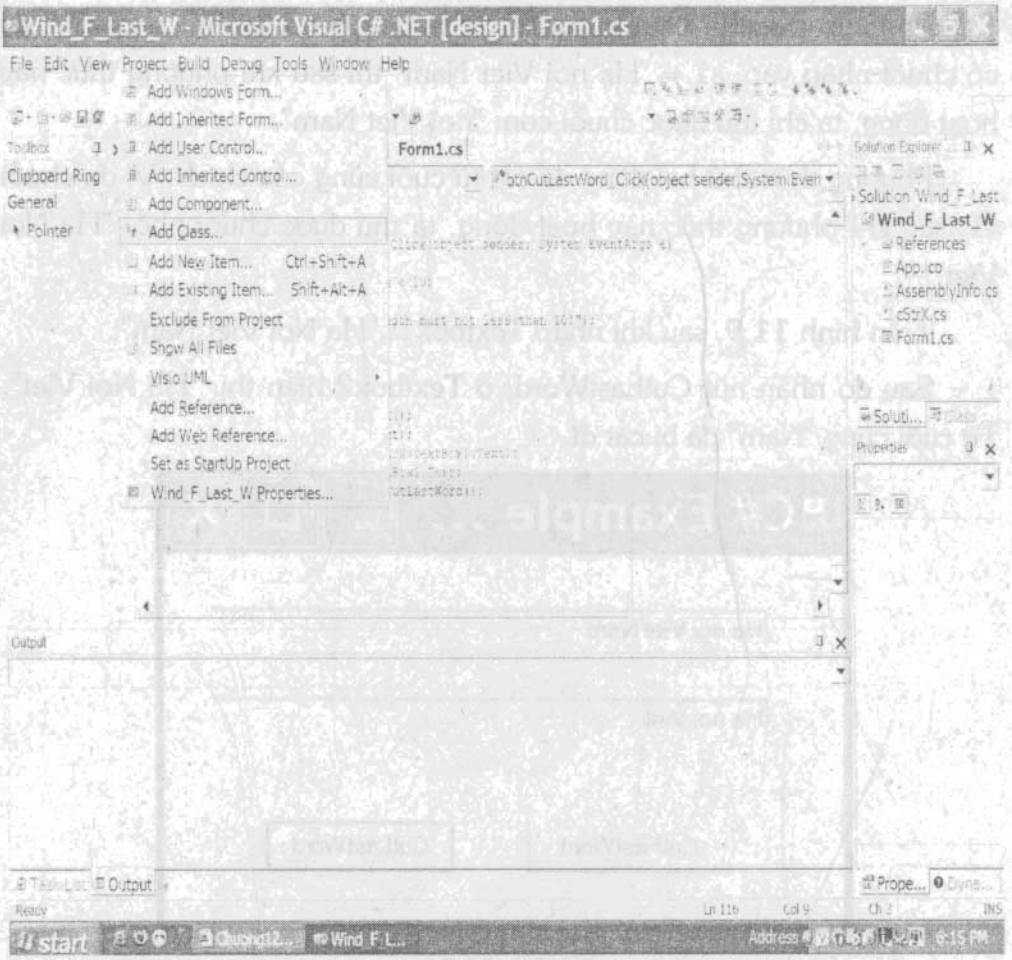
Hình 11.9

Độ dài chuỗi bắt buộc phải lớn hơn 10.

Các bước tạo giao diện và thuộc tính các điều khiển như sau.

1. Textbox1
2. Textbox2
3. Button1: CutFirstWord (Text); btnCutFirstWord (Name)
4. Button2: CutLastWord (Text); btnCutLastWord (Name)

Sau khi thiết lập xong giao diện, ta mở cửa sổ code và chọn submenu: Project → Add Class (hình 11.10).



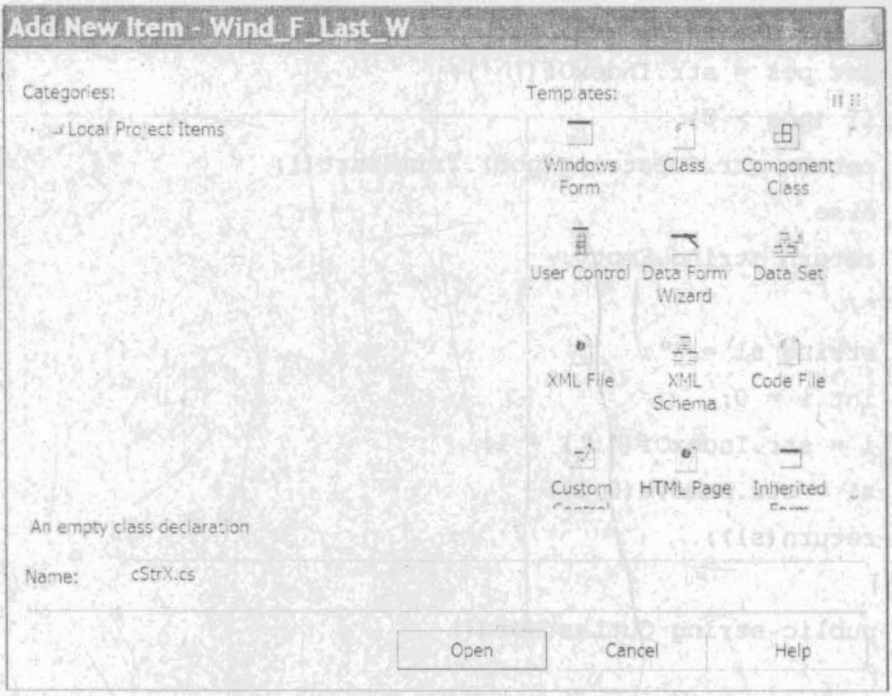
Hình 11.10

Sau đó ta có cửa sổ mới (hình 11.11), trong Textbox có nhãn Name, ta đặt tên: cStrX rồi nhấn nút Open và gõ các dòng lệnh như mã lệnh sau.

Trong mã lệnh chương trình này, ta định nghĩa hai phương thức:

```
public string CutFirstWord() { }
public string CutLastWord() { }
```

Chương trình chi tiết được trình bày sau đây:



Hình 11.11

```
using System;
public class cStrX
{
private string str;
public cStrX()
{
str = "";
}
//-------------------------------------------------------
public void Init(string ss)
{
str = ss;
}
//-------------------------------------------------------
public string CutFirstWord()
{
```

```
str = str.TrimStart();
int pos = str.IndexOf(' ');
if (pos > 0)
return str.Substring(pos).TrimStart();
else
return string.Empty;
*/
string s1 = "";
int i = 0;
i = str.IndexOf(" ") + 1;
s1 = str.Remove(0,i);
return(s1);
}
public string CutLastWord()
{
/*str = str.TrimEnd();
int pos = str.LastIndexOf(' ');
if (pos > 0)
return str.Substring(0, pos).TrimEnd();
else
return str;
*/
string s1;
int i = 0;
i = str.LastIndexOf(" ");
s1 = str.Remove(i, str.Length - i);
return s1;
}
}
```

Khi viết xong mã lệnh như đã liệt kê ở trên, ta quay trở về Windows Form như hình 11.9. Bây giờ ta viết tiếp mã lệnh cho cửa sổ này nhờ việc nhấn đúp vào nút CutFirstWord và nút CutLastWord như ở dưới đây.

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
public class Form1: System.Windows.Forms.Form
{
private System.Windows.Forms.Textbox Textbox1;
private System.Windows.Forms.Textbox Textbox2;
private System.Windows.Forms.Button btnCutFirstWord;
private System.Windows.Forms.Button btnCutLastWord;
private System.ComponentModel.Container components = null;
public Form1()
{
InitializeComponent();
}
protected override void Dispose(bool disposing)
{
if(disposing)
{
if(components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}
#region Windows Form Designer generated code
[STAThread]
static void Main()
{
```

```
    Application.Run(new Form1());
    }
    //-------------------------------------------------
    private void btnCutFirstWord_Click(object sender,
System.EventArgs e)
    {
    if (Textbox1.Text.Length < 10)
    {
    MessageBox.Show("Do dai chuoi bat buoc lon hon 10!");
    return;
    }
    else
    {
    cStrX ms = new cStrX();
    ms.Init(Textbox1.Text);
    Textbox2.Text = ms.CutFirstWord();
    }
    }
    //-------------------------------------------------
    private void btnCutLastWord_Click(object sender,
System.EventArgs e)
    {
    if (Textbox1.Text.Length < 10)
    {
    MessageBox.Show("Length must not less than 10!");
    return;
    }
    else
    {
    cStrX ms = new cStrX();
    ms.Init(Textbox1.Text);
    Textbox2.Text = ms.CutLastWord();
    }
```

}
}

*Ví dụ 11.8*Trong Windows Form, ta có thể xây dựng chương trình để xử lý chuỗi. Hãy nhập một chuỗi vào Textbox. Sau đó nhấn vào nút Process, trong các Textbox sẽ hiển thị:

– Số lượng các dấu cách (space bar).

– Số lượng các nguyên âm a.

– Số lượng các nguyên âm e.

– Số lượng các nguyên âm i.

– Số lượng các nguyên âm o.

– Số lượng các nguyên âm u.

– Hiển thị các ký tự đã lọc bỏ tất cả các nguyên âm.

– Hiển thị chuỗi ký tự vừa nhập theo chữ in hoa.

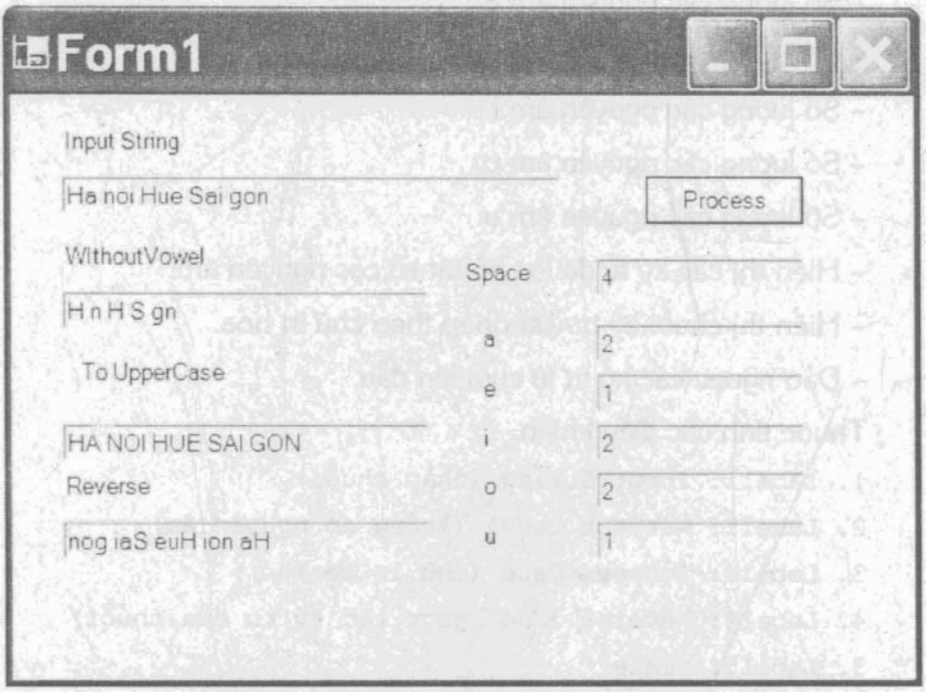
– Đảo ngược các ký tự từ cuối lên đầu.

Thuộc tính các điều khiển:

```
1. Label1: Input String (nhập chuỗi)
2. Label1: Without vowel (không có nguyên âm)
3. Label3: ToUpper Case (chữ in hoa)
4. Label4: Reverse (đảo ngược các ký tư của chuỗi)
5. Label5: Space
6. Label6: a
7. Label7: e
8. Label8: i
9. Label9: o
10. Label10: u
11. Textbox1: tBinput
12. Textbox2: tBWithout
13. Textbox3: tBupper
14. Textbox4: tBRev
15. Textbox5: tBsp
```

16. Textbox6: tBa
17. Textbox7: tBe
18. Textbox8: tBi
19. Textbox9: tBo
20. Textbox10: tBu
21. Button: Process(text); butProcess (Name)

Giao diện được tạo ra như hình 11.12.



Hình 11.12

Khi viết mã lệnh chương trình, ta cần phải khai báo tệp tiêu đề:

```
using System.Text.RegularExpressions;
```

Có như vậy trong chương trình ta mới sử dụng được lớp Regex.

Sau đây là toàn bộ mã chương trình của ví dụ trên.

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
```

```
using System.Text.RegularExpressions;
using System.Data;
namespace WindowsApplication15
{
public class Form1: System.Windows.Forms.Form
{
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.Label label8;
private System.Windows.Forms.Label label9;
private System.Windows.Forms.Label label10;
private System.Windows.Forms.Button butProcess;
private System.Windows.Forms.Textbox tBinput;
private System.Windows.Forms.Textbox tBRev;
private System.Windows.Forms.Textbox tBWithout;
private System.Windows.Forms.Textbox tBupper;
private System.Windows.Forms.Textbox tBsp;
private System.Windows.Forms.Textbox tBa;
private System.Windows.Forms.Textbox tBe;
private System.Windows.Forms.Textbox tBi;
private System.Windows.Forms.Textbox tBo;
private System.Windows.Forms.Textbox tBu;
private System.ComponentModel.Container components = null;
public Form1()
{
InitializeComponent();
}
```

```
protected override void Dispose(bool disposing)
{
if(disposing)
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}
[ STAThread]
static void Main()
{
Application.Run(new Form1());
}
//-----------------------------------------------------
private string s;
private int myNumberOfAEIOU(string a)
{
Regex lett = new Regex(a,RegexOptions.IgnoreCase);
return lett.Matches(s).Count;
}
//-----------------------------------------------------
private int myNumberOfSpace()
{
Regex lett = new Regex(" ");
return lett.Matches(s).Count;
}
//-----------------------------------------------------
private string myToupper()
{
```

```
string s1 = s;
s1 = s1.ToUpper();
return s1;
}
//-----------------------------------------------------
private string myFilterAEIOU()
{
string s1 = s;
s1 = s1.Replace("a","");
s1 = s1.Replace("e","");
s1 = s1.Replace("i","");
s1 = s1.Replace("o","");
s1 = s1.Replace("u","");
s1 = s1.Replace("A","");
s1 = s1.Replace("E","");
s1 = s1.Replace("I","");
s1 = s1.Replace("O","");
s1 = s1.Replace("U","");
return s1;
}
//-----------------------------------------------------
private string Reverse()
{
string s1 = "";
for(int i = s.Length -1; i >= 0; i--)
s1 = s1 + s[i];
return s1;
}
//-----------------------------------------------------
private void butProcess_Click(object sender, System.EventArgs e)
{
```

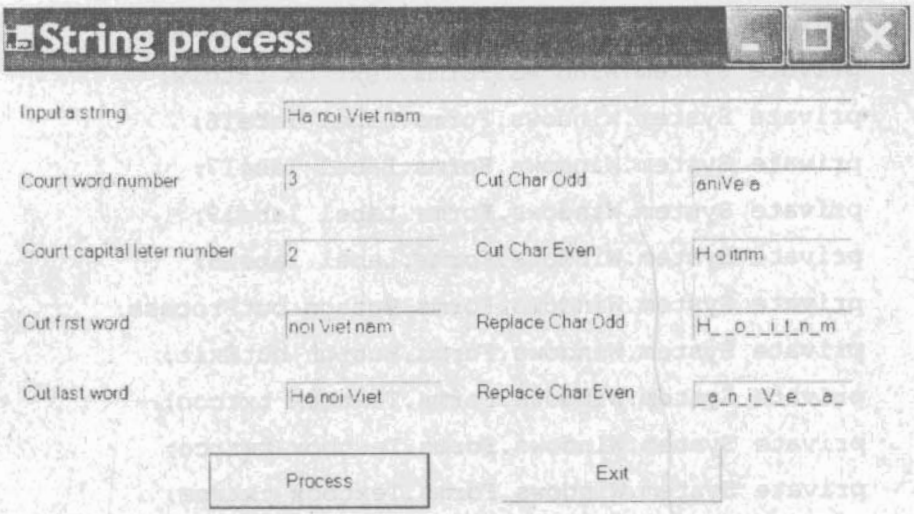
```
s = tBinput.Text;
tBa.Text = myNumberOfAEIOU("a").ToString();
tBe.Text = myNumberOfAEIOU("e").ToString();
tBi.Text = myNumberOfAEIOU("i").ToString();
tBo.Text = myNumberOfAEIOU("o").ToString();
tBu.Text = myNumberOfAEIOU("u").ToString();
tBsp.Text = myNumberOfSpace().ToString();
tBWithout.Text = myFilterAEIOU();
tBRev.Text = Reverse();
tBupper.Text = myToupper();
}
}
}
```

Ví dụ 11.9. Hãy tạo giao diện như hình 11.13. Nhập một chuỗi ký tự vào Textbox. Sau đó nhấn nút Process ta có kết quả hiển thị trong các Textbox còn lại, với các yêu cầu như sau:

- Đếm số từ trong chuỗi vừa nhập.
- Đếm số ký tự viết hoa (in hoa).
- Cắt từ đầu chuỗi.
- Cắt từ cuối chuỗi.
- Cắt các ký tự ở vị trí lẻ.
- Cắt các ký tự ở vị trí chẵn.
- Thay thế các ký tự lẻ bằng dấu cách.
- Thay thế các ký tự ở vị trí chẵn bằng dấu cách.

Mã lệnh chương trình được liệt kê dưới đây.

Ví dụ 11.9 cũng có thể xem là một bài tập dài. Các bạn có thể xây dựng giao diện như hình 11.13. Sau đó tự tìm hiểu mã lệnh đã liệt kê để có thể làm lại từ đầu.



Hình 11.13

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace process_string
{
public class Form1: System.Windows.Forms.Form
{
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Textbox txtinput;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.Textbox txtcwn;
private System.Windows.Forms.Textbox txtccln;
```

```
private System.Windows.Forms.Textbox txtcfw;
private System.Windows.Forms.Text ox txtclw;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.Label label9;
private System.Windows.Forms.Label label8;
private System.Windows.Forms.Button butProcess;
private System.Windows.Forms.Button butExit;
private System.Windows.Forms.Textbox txtcco;
private System.Windows.Forms.Textbox txtrco;
private System.Windows.Forms.Textbox txtrce;
private System.Windows.Forms.Textbox txtcce;
private System.ComponentModel.Container components =  null;
public Form1()
{
InitializeComponent();
}
static void Main()
{
Application.Run(new Form1());
}
int i = 0;
private void butExit_Click(object sender, System.EventArgs e)
{
this.Close();
}
private void butProcess_Click(object sender, System.EventArgs e)
{
int cwn = 0;
int ccln = 0;
string sinput = txtinput.Text.Trim();
sinput = standard(sinput);
```

```
for (i = 0; i < sinput.Length; i++)
{
if (sinput[ i] == ' ')
{
cwn++;
}
if (char.IsUpper(sinput[ i] ))
{
ccln++;
}
}
txtcwn.Text = cwn.ToString();
tx*~cln.Text = ccln.ToString();
txtcfw.Text = cut_first_word(sinput);
txtclw.Text = cut_last_word(sinput);
txtcco.Text = cut_char_odd(sinput);
txtcce.Text = cut_char_even(sinput);
txtrco.Text = replace_char_odd(sinput);
txtrce.Text = replace_char_even(sinput);
}
private string standard(string sinput)
{
for (i = 0; i < sinput.Length; i++)
{
if (sinput[ i] == ' ')
if (sinput[ i+1] == ' ')
{
sinput = sinput.Remove(i + 1, 1);
i--;
}
}
return sinput;
```

```
}
private string cut_first_word(string s)
{
i = s.IndexOf(" ") + 1;
scfw = s.Remove(0,i);
return scfw;
}
private string cut_last_word(string s)
{
string sclw = "";
i = s.LastIndexOf(" ");
sclw = s.Remove(i, s.Length - i);
return sclw;
}
private string cut_char_odd(string s)
{
string scoc = "";//save all even char
i = 0;
for (i = i + 1; i < s.Length; i+ = 2)
scoc+ = s[ i] ;
return scoc;
}
private string cut_char_even(string s)
{
string scec = "";
for(i = 0; i < s.Length; i+ = 2)
{
scec+ = s[ i] ;
}
return scec;
}
private string replace_char_odd(string s)
```

```
{
string srco = "";
char[] s1 = s.ToCharArray();
for(i = 0; i < s1.Length; i++)
{
if (i % 2 != 0)
{
s1[i] = '_';
}
srco+ = s1[i];
}
return srco;
}
private string replace_char_even(string s)
{
char[] s1 = s.ToCharArray();
string srce = "";
for(i = 0; i < s1.Length; i++)
{
if(i % 2 == 0)
s1[i] = '_';
srce+ = s1[i];
}
return srce;
}
}
}
```

Chương 12. Đồ hoạ trong Windows

Đồ hoạ trong Windows khá phong phú. Ở đây ta chỉ xét một vấn đề đơn giản nhất là vẽ đồ thị các đường cong, các hình, tô màu các hình và viết chữ lồng vào trong hình vẽ.

Khi thực hiện vẽ đồ thị trong môi trường Windows của C#, đầu mỗi chương trình ta phải thêm vào các tệp tiêu đề:

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Text;
```

Ngoài ra còn các tệp tiêu đề có sẵn khi ta tạo Windows Form.

12.1. VẼ ĐỒ THỊ CÁC HÌNH VÀ CÁC ĐƯỜNG CONG

Các đồ thị có thể vẽ trong Panel đặt ở trong Form.

Nhấn đúp vào Panel này ta có phương thức sau:

```
private void panel1_Paint(object sender,
System.Windows.Forms.PaintEventArgs e)
{ }
```

Công cụ chủ yếu để vẽ hình trong Windows là sử dụng các phương thức thuộc lớp Pen (bút vẽ).

Khai báo bút vẽ:

```
Pen pen1 = new Pen(Color.DarkGreen,3);
```

Trong đó màu đường cong được chọn tùy ý theo mã màu thư viện khi ta gõ dấu chấm sau từ Color; số 3 thể hiện độ dày của nét vẽ.

Trong một chương trình muốn thay đổi màu ta chỉ cần viết:

```
pen1.Color = Color.Red;
```

Các toạ độ điểm cần vẽ luôn luôn phải là số nguyên. Nếu khi tính toán ta có số float hoặc double, ta cần phải chuyển kiểu về số nguyên.

Các phương thức thư viện để vẽ các đường cong và hình khác nhau được trình bày điển hình sau đây.

1. Vẽ đường thẳng từ điểm có toạ độ (x1, y1) đến điểm (x2, y2):

```
e.Graphics.DrawLine(pen1, x1, y1, x2, y2);
```

2. Vẽ hình chữ nhật với toạ độ điểm góc trên trái (x1, y1) và toạ độ điểm góc dưới phải (x2, y2):

```
e.Graphics.DrawRectangle(pen1, x1, y1, x2, y2);
```

3. Vẽ hình Ellipse có tâm (x1, y1), bán trục đứng h, bán trục ngang w:

```
e.Graphics.DrawEllipse(pen1, x1, y1, w, h);
```

4. Vẽ đường tròn giống như vẽ đường Ellipse nhưng chỉ cần cho giá trị bán trục đứng bằng giá trị bán trục ngang w = h:

5. Vẽ cung cong theo hình Ellipse (hình tròn)

```
e.Graphics.DrawArc(pen1, x1, y1, x2, y2, startAng, endAng);
```

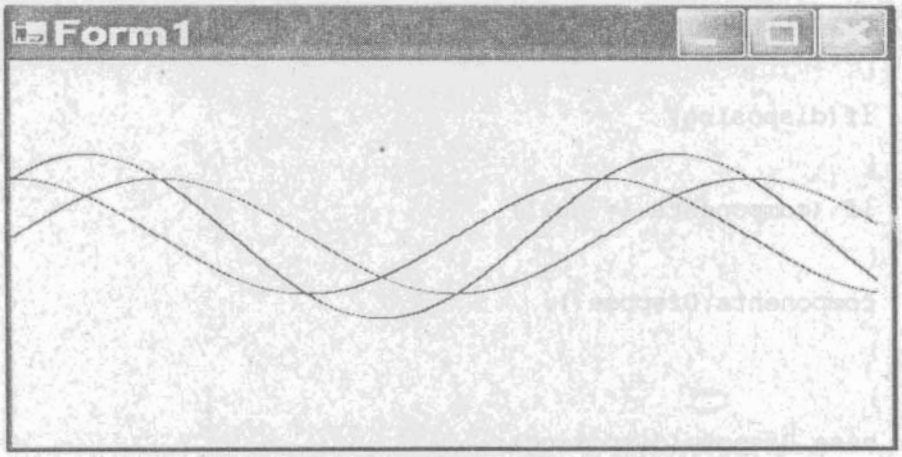
Trong đó (x1, y1) là điểm góc trên trái, (x2, y2) là điểm góc dưới phải của hình chữ nhật để cung tròn nội tiếp bên trong hình chữ nhật này.

Khi viết câu lệnh:

```
e.Graphics.DrawArc(pen1, 0, 0, 100, 150, 0, 360);
```

Trong đó hình chữ nhật có toạ độ (x1 = 0, y1 = 0, x2 = 100, y2 = 150), góc đầu startAngle = 0, góc cuối endAngle = 360. Cung lúc này đã trở thành một đường Ellipse dẹt.

Ví dụ 12.1. Vẽ đồ thị các đường cong hình y1 = 50.Sin(x), hình y2 = 50.Cos(x) và tổng của chúng y3 = y1 + y2.



Hình 12.1

Sau khi chạy chương trình, ta có giao diện như hình 12.1. Đồ thị có ba đường cong dạng sin, cos và tổng của hai đường.

Mã lệnh chương trình được liệt kê như sau:

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Text;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
namespace Graphic1
{
public class Form1: System.Windows.Forms.Form
{
private System.Windows.Forms.Panel panel1;
private System.ComponentModel.Container components = null;
public Form1()
{
InitializeComponent();
}
protected override void Dispose(bool disposing)
{
if(disposing)
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}
```

```
        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
        this.panel1 = new System.Windows.Forms.Panel();
        this.SuspendLayout();
        this.panel1.Location = new System.Drawing.Point(0, 0);
        this.panel1.Name = "panel1";
        this.panel1.Size = new System.Drawing.Size(528, 320);
        this.panel1.TabIndex = 0;
        this.panel1.Paint += new
    System.Windows.Forms.PaintEventHandler(this.panel1_Paint);
        this.AutoScaleBaseSize = new System.Drawing.Size(6, 15);
        this.ClientSize = new System.Drawing.Size(536, 330);
        this.Controls.Add(this.panel1);
        this.Name = "Form1";
        this.Text = "Form1";
        this.ResumeLayout(false);
        }
        #endregion
        /// <summary>
        /// The main entry point for the application
        /// </summary>
        [STAThread]
        static void Main()
        {
        Application.Run(new Form1());
        }
        //-----------------------------------------------------
        private void panel1_Paint(object sender,
    System.Windows.Forms.PaintEventArgs e)
        {
        Pen pen1 = new Pen(Color.Red,2);
        double [] y1 = new double[ 1001];
```

```
    for (int i = 0; i < 1001; i++)
    y1[ i] = 50*Math.Sin(Math.PI*i/180);
    for (int i = 0; i < 1000; i++)
    e.Graphics.DrawLine(pen1, i, 150 - (int)y1[ i], (i+1), 150
- (int)y1[ i+1] );
    pen1.Color = Color.DarkGreen;
    double [ ] y2 = new double[ 1001] ;
    for(int i = 0; i < 1001; i++)
    y2[ i] = 50*Math.Cos(Math.PI*i/180);
    for(int i = 0; i < 1000; i++)
    e.Graphics.DrawLine(pen1, i, 150 - (int)y2[ i], (i+1), 150
- (int)y2[ i+1] );
    pen1.Color = Color.DarkMagenta;
    double [ ] y3 = new double[ 1001] ;
    for (int i = 0; i < 1001; i++)
    y3[ i] = y1[ i] + y2[ i] ;
    for(int i = 0; i < 1000; i++)
    e.Graphics.DrawLine(pen1, i, 150 - (int)y3[ i], (i+1), 150
- (int)y3[ i+1] );
    }
    }
    }
```

12.2. VẼ CÁC HÌNH CÓ MÀU TÔ ĐẬM VÀ HIỂN THỊ STRING

Khác với bút vẽ Pen đã xét ở trên, để tô màu các hình và hiển thị chuỗi ký tự lên màn hình, ta phải sử dụng lớp Brush.

– Khai báo chổi vẽ nhờ câu lệnh:

```
Brush br = new SolidBrush(Color.Blue);
```

Đối số trong hàm là biến chỉ màu sắc.

– Muốn thay đổi màu ta viết như sau:

```
br = new SolidBrush(Color.LightGreen);
```

– Để chọn được font chữ, kích thước chữ, kiểu (đậm, nghiêng....) ta

có câu lệnh:

```
Font f1 = new Font(kiểu font chữ, size, FontStyle);
```

Trong đó kiểu font chữ được chọn khá phong phú:

FontStyle là Bold, Italic, Underline.

Ví dụ:

```
Font f1 = new Font("Courrier new", 15, FontStyle.Bold);
```

– Viết chữ lên màn hình ta sử dụng câu lệnh khai báo định dạng:

```
StringFormat format1 = new StringFormat();
```

– Chữ viết theo hướng nằm ngang:

```
format1.FormatFlags = StringFormatFlags.DirectionRightToLeft;
```

– Chữ viết theo hướng thẳng đứng:

```
format1.FormatFlags = StringFormatFlags.DirectionVertical;
```

– Lệnh thực hiện việc hiển thị lên màn hình dòng chữ (String):

```
e.Graphics.DrawString("string len man hinh", f1, br, x1,
y1, format1);
```

Trong đó:

f1: biến font;

br: biến chổi vẽ;

x1, y1: toạ độ điểm đặt chữ đầu tiên của string;

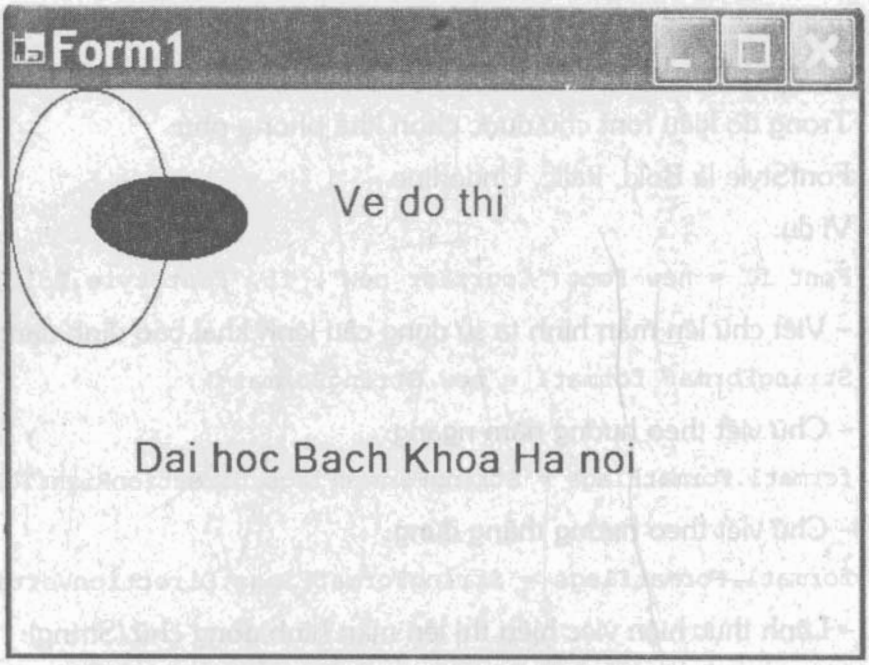
format1: hướng chữ (nằm ngang, hoặc thẳng đứng).

Ví dụ 12.2. Xây dựng giao diện để vẽ lên màn hình một hình Ellipse tô đậm, một hình chữ nhật tô đậm, ba dòng chữ (hai nằm ngang và một thẳng đứng ở các vị trí khác nhau).

Để thiết kế giao diện, ta kéo điều khiển "Panel" vào cửa sổ Form và dãn rộng đến một mức phù hợp. Tiếp đến, ta nhấn đúp chuột vào Panel và viết chương trình, với nội dung phương thức:

```
private void panel1_Paint(object sender,
System.Windows.Forms.PaintEventArgs e)
{ }
```

Sau khi chạy chương trình, ta có giao diện như trên hình 12.2.



Hình 12.2

Toàn bộ mã lệnh chương trình được liệt kê dưới đây.

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Text;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
namespace Graphic3
{
public class Form1: System.Windows.Forms.Form
{
private System.Windows.Forms.Panel panel1;
/// <summary>
/// Required designer variable
/// </summary>
```

```
    private System.ComponentModel.Container components = null;
    public Form1()
    {
    InitializeComponent();
    }
    protected override void Dispose(bool disposing)
    {
    if(disposing)
    {
    if (components != null)
    {
    components.Dispose();
    }
    }
    base.Dispose(disposing);
    }
    #region Windows Form Designer generated code
    private void InitializeComponent()
    {
    this.panel1 = new System.Windows.Forms.Panel();
    this.SuspendLayout();
    this.panel1.Location = new System.Drawing.Point(0, 0);
    this.panel1.Name = "panel1";
    this.panel1.Size = new System.Drawing.Size(528, 320);
    this.panel1.TabIndex = 0;
    this.panel1.Paint += new
System.Windows.Forms.PaintEventHandler(this.panel1_Paint);
    this.AutoScaleBaseSize = new System.Drawing.Size(6, 15);
    this.ClientSize = new System.Drawing.Size(536, 330);
    this.Controls.Add(this.panel1);
    this.Name = "Form1";
    this.Text = "Form1";
    this.ResumeLayout(false);
```

```
    }
    #endregion
    [STAThread]
    static void Main()
    {
    Application.Run(new Form1());
    }
    //-----------------------------------------------------
    private void panel1_Paint(object sender,
System.Windows.Forms.PaintEventArgs e)
    {
    Pen pen1 = new Pen(Color.Red,2);
    e.Graphics.DrawArc(pen1,0,0,100,150,0,360);
    Brush br = new SolidBrush(Color.Blue);
    Font f1 = new Font("Courrier new",15,FontStyle.Bold);
    e.Graphics.FillEllipse(br,50,50,100,50);
    br = new SolidBrush(Color.LightGreen);
    e.Graphics.FillRectangle(br,150,150,50,40);
    br= new SolidBrush(Color.DarkGreen);
    e.Graphics.DrawString("Ve do thi",f1,br,200,50);
    br = new SolidBrush(Color.Blue);
    StringFormat format1 = new StringFormat();
    format1.FormatFlags =
StringFormatFlags.DirectionRightToLeft;
    e.Graphics.DrawString("D H Bach Khoa Ha noi", f1, br, 400,
200, format1);
    br = new SolidBrush(Color.Violet);
    format1.FormatFlags = StringFormatFlags.DirectionVertical;
    e.Graphics.DrawString("Genetic -Singapore ",f1, br, 400,
20, format1);
    }
    }
    }
```

Cũng giống như các phương thức mẫu (thư viện) để vẽ cho các hình khác nhau, ở đây ta sẽ liệt kê một số phương thức chủ yếu cho việc tô màu các hình đậm.

1.Tô màu hình Ellipse:

```
e.Graphics.FillElipse (br, x1, y1, w, h);
```

2. Tô màu hình chữ nhật:

```
e.Graphics.FillRectangle (br, x1, y1, x2, y2);
```

3. Tô màu một cung hình quạt:

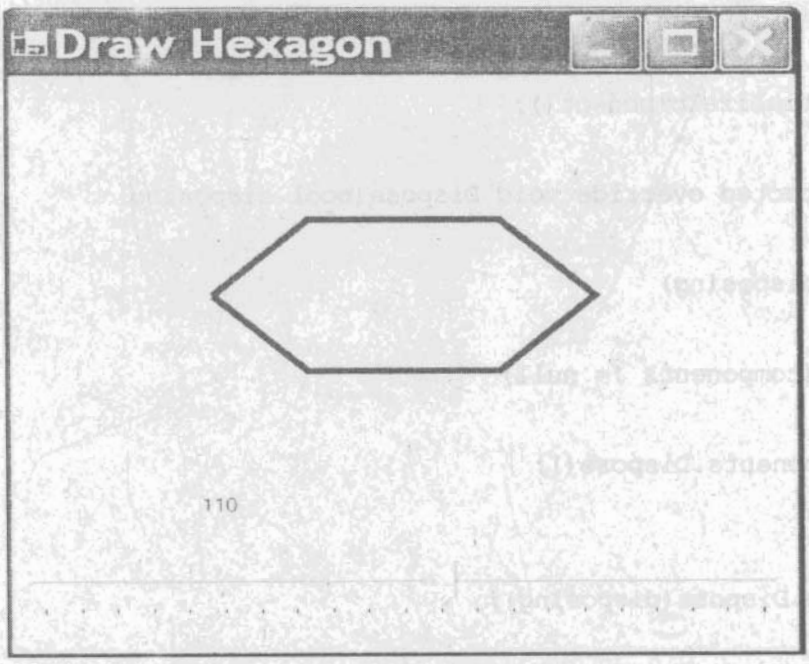
```
e.Graphics.FillPie (br, x1, y1, x2, y2, startAng, endAng);
```

Các phương thức khác các bạn có thể nhìn thấy khi gõ vào câu lệnh

```
e.Graphics.Fill
```

Sau dấu chấm, ta sẽ thấy hàng loạt các hàm có liệt kê các đối số.

Ví dụ 12.3. Tạo một Windows Form để vẽ một hình lục giác có các cạnh là giá trị thay đổi lấy từ Trackbar. Khi dịch chuyển con trượt Trackbar, hình dạng của lục giác sẽ thay đổi theo giá trị của Trackbar.



Hình 12.3

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
namespace draw_Hexagon
{
/// <summary>
/// Summary description for Form1
/// </summary>
public class Form1: System.Windows.Forms.Form
{
private System.Windows.Forms.Panel panel1;
private System.Windows.Forms.Trackbar Trackbar1;
private System.Windows.Forms.Label lblvalue;
private System.ComponentModel.Container components = null;
public Form1()
{
InitializeComponent();
}
protected override void Dispose(bool disposing)
{
if(disposing)
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}
#region Windows Form Designer generated code
/// <summary>
```

```
        /// Required method for Designer support - do not modify
        /// the contents of this method with
        // the code editor.
        /// </summary>
        private void InitializeComponent()
        {
        this.panel1 = new System.Windows.Forms.Panel();
        this.Trackbar1 = new System.Windows.Forms.Trackbar();
        this.lblvalue = new System.Windows.Forms.Label();
        ((System.ComponentModel.ISupportInitialize)
    (this.Trackbar1)).BeginInit();
        this.SuspendLayout();
        //
        // panel1
        //
        this.panel1.Location = new System.Drawing.Point(19, 9);
        this.panel1.Name = "panel1";
        this.panel1.Size = new System.Drawing.Size(413, 303);
        this.panel1.TabIndex = 0;
        //
        // Trackbar1
        //
        this.Trackbar1.Dock = System.Windows.Forms.DockStyle.Bottom;
        this.Trackbar1.Location = new System.Drawing.Point(0, 352);
        this.Trackbar1.Maximum = 20;
        this.Trackbar1.Name = "Trackbar1";
        this.Trackbar1.Size = new System.Drawing.Size(451, 69);
        this.Trackbar1.TabIndex = 1;
        this.Trackbar1.Scroll += new
    System.EventHandler(this.Trackbar1_Scroll);
        // lblvalue
        this.lblvalue.Location = new System.Drawing.Point(106, 305);
        this.lblvalue.Name = "lblvalue";
        this.lblvalue.Size = new System.Drawing.Size(96, 27);
```

```
    this.lblvalue.TabIndex = 2;
    // Form1
    this.AutoScaleBaseSize = new System.Drawing.Size(6, 15);
    this.ClientSize = new System.Drawing.Size(451, 421);
    this.Controls.Add(this.lblvalue);
    this.Controls.Add(this.Trackbar1);
    this.Controls.Add(this.panel1);
    this.Name = "Form1";
    this.Text = "Draw Hexagon";
    ((System.ComponentModel.ISupportInitialize)
(this.Trackbar1)).EndInit();
    this.ResumeLayout(false);
    ;
    #endregion
    /// <summary>
    /// The main entry point for the application
    /// </summary>
    [STAThread]
    static void Main()
    {
    Application.Run(new Form1());
    }
    private void Trackbar1_Scroll(object sender,
System.EventArgs e)
    {
    Graphics g = panel1.CreateGraphics();
    panel1.Refresh();
    int n = Trackbar1.Value*10;
    g.TranslateTransform(panel1.Width/2,panel1.Height/2);
    Pen p = new Pen(Color.Red,5);
    Point[] po = {new Point(-n, 0), new Point(-n/2, n/2), new
Point(n/2, n/2), new Point(n, 0), new Point(n/2, -n/2), new
Point(-n/2, -n/2), new Point(-n, 0)};
    g.DrawCurve(p, po, 0.00F);
```

```
lblvalue.Text = n.ToString();
}
}
}
```

Ví dụ 12.4. Sử dụng Trackbar để thay đổi giá trị.

```
int a = Trackbar1.Value;
```

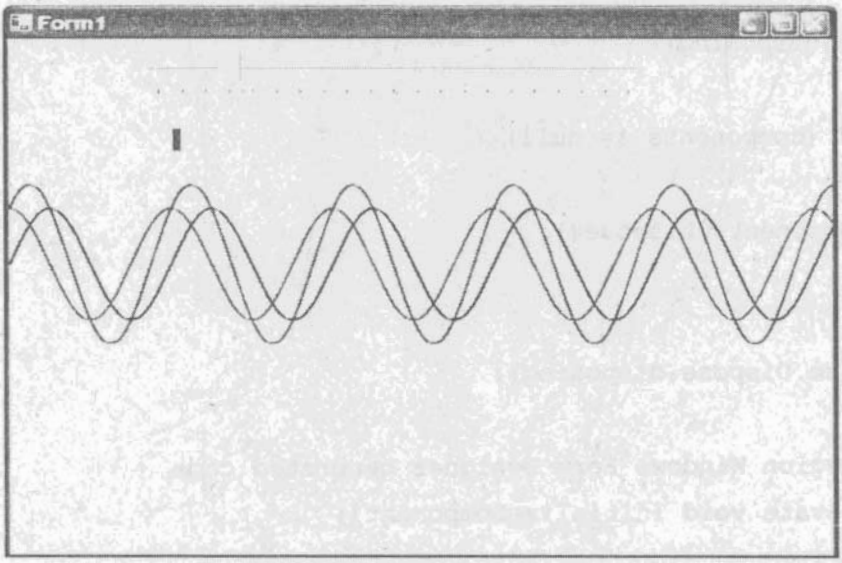
Đại lượng a được đưa vào làm tham số cho một vài hàm tính toán:

```
y1[ i] =(50)*Math.Sin(i*3.14159/180*a);
```

Khi con trượt thay đổi thì giá trị a thay đổi và đồng thời đồ thị cũng thay đổi theo. Các bạn có thể so sánh hai cách viết chương trình gần giống nhau của ví dụ 12.3 và ví dụ 12.4.

Trong ví dụ 12.3, chúng ta viết hàm trực tiếp cho điều khiển Trackbar, còn trong ví dụ 12.4 ta lại viết hàm:

```
private void Form1_Paint(object sender,
System.Windows.Forms.PaintEventArgs e)
{ }
```



Hình 12.4

Đồ thị của ví dụ này được hiển thị ở hình 12.4.

Toàn bộ mã lệnh chương trình được liệt kê sau đây.

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
namespace WindowsAppDrawing
{
public class Form1: System.Windows.Forms.Form
{
private System.Windows.Forms.Trackbar Trackbar1;
private System.ComponentModel.Container components = null;
public Form1()
{
InitializeComponent();
protected override void Dispose(bool disposing)
{
if(disposing)
{
if (components != null)
{
components.Dispose();
}
}
base.Dispose(disposing);
}
#region Windows Form Designer generated code
private void InitializeComponent()
{
this.Trackbar1 = new System.Windows.Forms.Trackbar();
((System.ComponentModel.ISupportInitialize)(this.Trackbar1
)).BeginInit();
this.SuspendLayout();
```

```
    // Trackbar1
    this.Trackbar1.Location = new System.Drawing.Point(72, 8);
    this.Trackbar1.Maximum = 20;
    this.Trackbar1.Name = "Trackbar1";
    this.Trackbar1.Size = new System.Drawing.Size(304, 56);
    this.Trackbar1.TabIndex = 0;
    this.Trackbar1.Scroll+ = new
System.EventHandler(this.Trackbar1_Scroll);
    // Form1
    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
    this.ClientSize = new this.Controls.Add(this.Trackbar1);
    System.Drawing.Size(480, 372);
    this.Name = "Form1";
    this.Text = "Form1";
    this.Paint + = new
System.Window.Forms.PaintEventHandler(this.Form1_Paint);
    ((System.ComponentModel.ISupportInitialize)
(this.Trackbar1)).EndInit();
    this.ResumeLayout(false);
    }
    #endregion
    [ STAThread]
    static void Main()
    {
    Application.Run(new Form1());
    }
    private void Form1_Paint(object sender,
System.Windows.Forms.PaintEventArgs e)
    {
    int a = Trackbar1.Value;
    Pen pen1 = new Pen(Color.Red,6);
    e.Graphics.DrawLine(pen1, 80 + a*15, 80, 80 + a*15, 100);
    double [ ] y1 = new double[ 1000];
```

```
    double [] y2 = new double[1000];
    double [] y3 = new double[1000];
    for(int i = 0; i < 1000; i++)
    y1[ i] = (50)*Math.Sin(i*3.14159/180*a);
    pen1 = new Pen(Color.Blue, 2);
    for(int i = 0; i < 999; i++)
    e.Graphics.DrawLine(pen1, i, 200 - (int)y1[ i], i+1, 200 -
(int)y1[ i+1] );
    for(int i = 0; i < 1000; i++)
    {
    y2[ i] = (50)*Math.Cos(i*3.14159/180*a);
    y3[ i] = y1[ i] + y2[ i];
    }
    pen1 = new Pen(Color.Red, 2);
    for(int i = 0; i < 999; i++)
    e.Graphics.DrawLine(pen1, i, 200 - (int)y2[ i], i + 1, 200
- (int)y2[ i+1] );
    pen1 = new Pen(Color.DarkMagenta, 2);
    for(int i = 0; i < 999; i++)
    e.Graphics.DrawLine(pen1, i, 200 - (int)y3[ i], i + 1, 200
- (int)y3[ i+1] );
    //Brush br1 = new SolidBrush(Color.LightSeaGreen);
    //e.Graphics.FillEllipse(br1, 100, 100, 50 + a, 50 + a);
    }
    private void Trackbar1_Scroll(object sender,
System.EventArgs e)
    {
    Invalidate();
    }
    }
    }
```

Chương 13. Lập trình cơ sở dữ liệu

Để quản trị cơ sở dữ liệu (CSDL), ở đây ta phải hiểu là CSDL được xây dựng trong hai môi trường thông dụng. Đó là môi trường Access và môi trường SQL (môi trường Oracle cũng rất hiệu quả nhưng trong khuôn khổ cuốn sách này không thể trình bày được, vì chương trình để quản trị CSDL bằng Oracle rất đồ sộ).

Trong chương này ta sẽ tìm hiểu về cách kết nối CSDL trong môi trường Access 2000 với ngôn ngữ lập trình C#.

Có hai cách để chương trình C# truy cập vào CSDL Access:

– Viết chương trình trong Console Application.

– Viết chương trình trong Windows Application.

Trước hết, ta cần tạo bảng dữ liệu trong Access 2000 (hoặc có thể Access 97, 2003, 2005).

13.1. TẠO CƠ SỞ DỮ LIỆU TRONG ACCESS 2000

Giả sử có bảng dữ liệu như hình 13.3.

1. Khởi động Access 2000. Ta có cửa sổ như hình 13.1: Chọn tên CSDL là "data.mdb".

2. Chọn Create Table in Design View → ta có cửa sổ như trên hình 13.2.

3. Lần lượt nhập các trường vào các hàng của hình 13.2:

– CustomerID

– CompanyName

– ContactName

– Phone

Kiểu biến (Data Type) có thể chọn dựa vào thực tế, nhưng ở đây ta chọn kiểu Text.

4. Từ hình 13.2, ta chọn khoá chính (Primary Key) cho trường CustomerID.

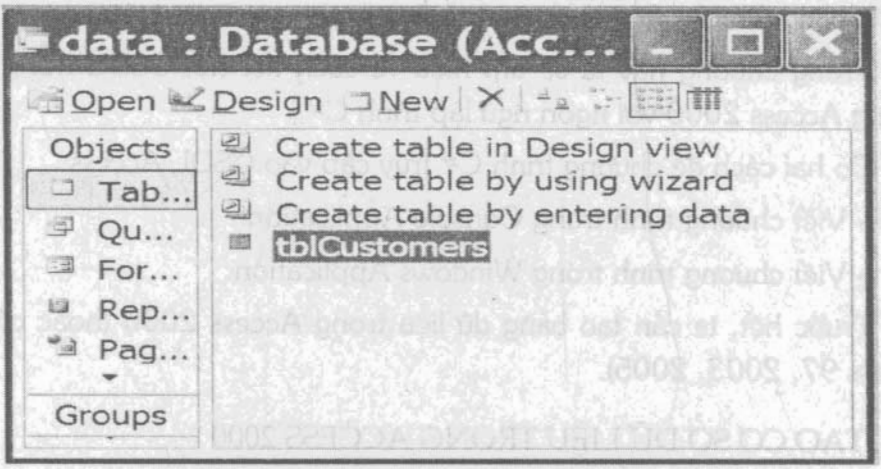
5. Đóng (Close) hình 13.2 và ta cần phải lưu tên bảng dữ liệu (data

table) với tên: " tblCustomer ".

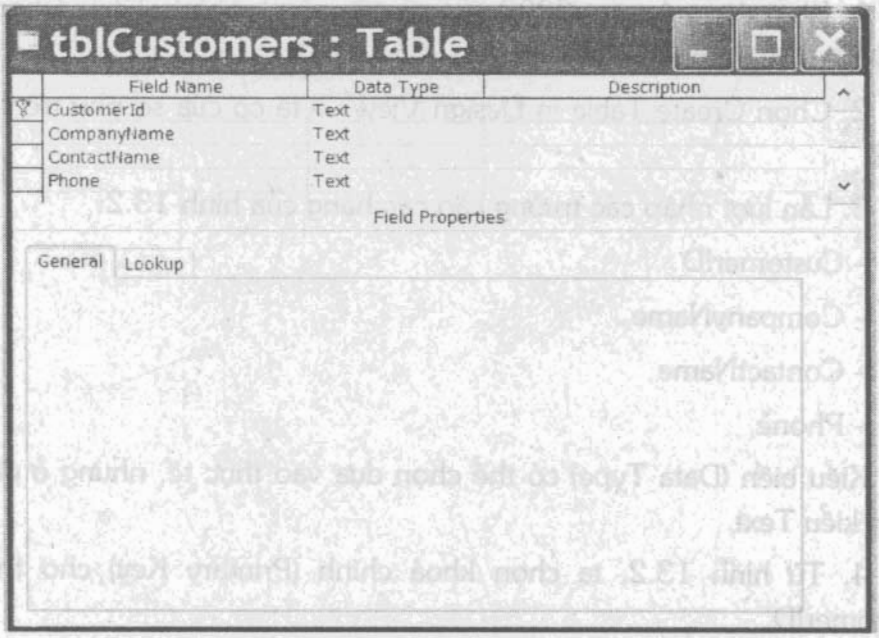
6. Từ hình 13.1, nhấn đúp vào tên bảng tblCustomer, ta sẽ có bảng như hình 13.3. Sau đó trên từng hàng ta nhập dữ liệu vào các cột.

Hoàn tất công việc, ta đóng chương trình Access lại.

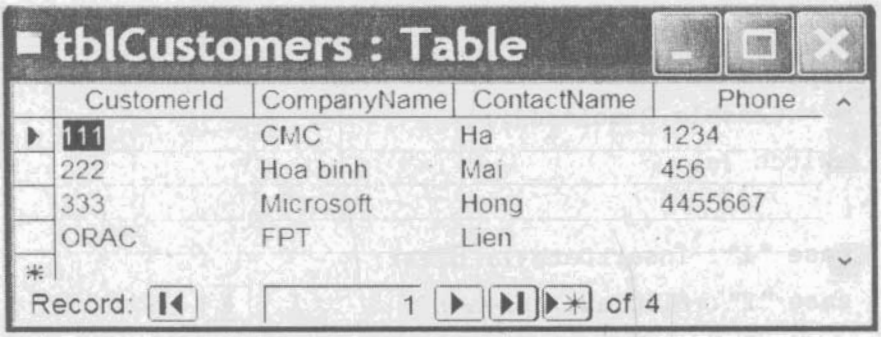
Chú ý: Sau khi tạo tệp dữ liệu data.mdb, ta sẽ chép sang thư mục của chương trình C# (không nên để trong My Document).



Hình 13.1



Hình 13.2



Hình 13.3

13.2. XÂY DỰNG CHƯƠNG TRÌNH TRONG CONSOLE APPLICATION

Ví dụ 13.1. Trước hết ta thêm tệp tiêu đề

```
using System.Data.OleDb;
```

vào đầu chương trình.

```
using System;
using System.Data;
using System.Data.OleDb;
namespace CSDL
{
class Class1
{
static OleDbConnection con;
static void Main(string[] args)
{
string s = "";
do
{
Console.WriteLine("---Customer Management---");
Console.WriteLine("1. Insert new customer");
Console.WriteLine("2. Display customer");
Console.WriteLine("3. Display customers dataset");
Console.WriteLine("4. Delete with Command");
```

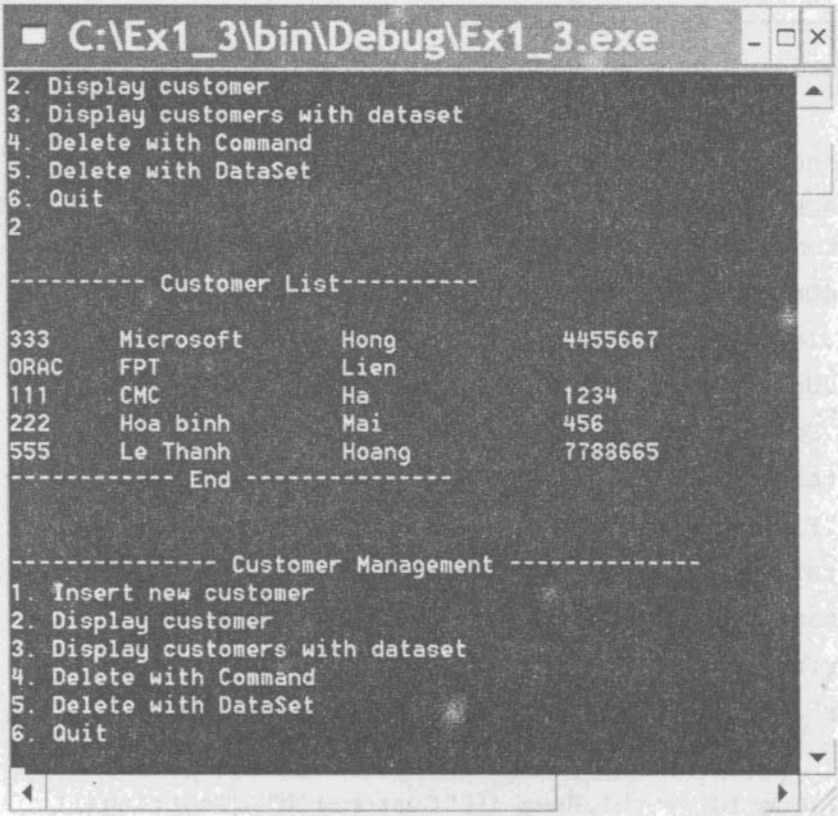
```
        Console.WriteLine("5. Delete with DataSet");
        Console.WriteLine("6. Quit");
        s = Console.ReadLine();
        switch (s)
        {
        case "1": InsertData(); break;
        case "2": Display();break;
        case "3": DisplayWithDataSet();break;
        case "4": DeleteWithCommand();break;
        case "5": DeleteWithDataSet();break;
        }
        }
        while (s != "6");
        }
        //-------------------------------------------------
        static void Connect()
        {
        string strConString = @"Provider =
Microsoft.Jet.OleDb.4.0; Data Source=..\..\data.mdb";
        con = new OleDbConnection(strConString);
        con.Open();
        }
        //-------------------------------------------------
        static void InsertData ()
        {
        Console.WriteLine("Enter information: ");
        Console.Write("Customer ID: ");
        string CustID = Console.ReadLine();
        Console.Write("Company Name: ");
        string CompName = Console.ReadLine();
        Console.Write("Contact Name: ");
        string ContName = Console.ReadLine();
        Console.Write("Phone: ");
```

```
    string Phone = Console.ReadLine();
    Connect();
    OleDbCommand cmd = new OleDbCommand("INSERT INTO
tblCustomers (CustomerId, CompanyName, ContactName, Phone)
VALUES ('" + CustID + "','" + CompName + "', '" + ContName +
"', '" + Phone + "')",con);
    OleDbCommand cmd = new OleDbCommand(string.Format("INSERT
INTO tblCustomers (CustomerId,CompanyName,ContactName,Phone)
VALUES ('{0}', '{1}', '{2}', '{3}')", CustID, CompName,
ContName, Phone), con);
    OleDbCommand cmd = new OleDbCommand("INSERT INTO
tblCustomers (CustomerId,CompanyName,ContactName,Phone) VALUES
(?,?,?,?)",con);
      cmd.Parameters.Add("@CustId", CustID);
      cmd.Parameters.Add("@Compname", CompName);
      cmd.Parameters.Add("@ContName", ContName);
      cmd.Parameters.Add("@phone", Phone);
      int n = cmd.ExecuteNonQuery();
      Console.WriteLine("{0} record inserted", n);
      con.Close();
      }
    //-------------------------------------------------
    static void Display()
    {
    Connect();
    OleDbCommand cmd = new OleDbCommand("SELECT * FROM
tblCustomers", con);
    OleDbDataReader dtr = cmd.ExecuteReader();
    Console.WriteLine("\n--- Customer List ---\n");
    while (dtr.Read())
    {
    Console.WriteLine("{0, -5}\t{1, -12}\t{2, -15}\t{3, -10}",
dtr["CustomerId"], dtr["CompanyName"], dtr["ContactName"],
dtr["Phone"]);
```

```
    }
    Console.WriteLine("--- End ---\n\n");
    dtr.Close();
    con.Close();
    }
    //-----------------------------------------------------
    static void DisplayWithDataSet()
    {
    Connect();
    OleDbDataAdapter da = new OleDbDataAdapter("SELECT * FROM
tblCustomers", con);
    DataSet ds = new DataSet();
    da.Fill(ds, "tblCustomers");
    DataTable tbl = ds.Tables["tblCustomers"];
    Console.WriteLine("\n--- Customer List ---\n");
    for (int i = 0; i < tbl.Rows.Count; i++)
    {
    Console.WriteLine("{0, -5}\t{1, -12}\t{2, -15}\t{3, -10}",
tbl.Rows[i]["CustomerId"], tbl.Rows[i]["CompanyName"],
tbl.Rows[i]["ContactName"], tbl.Rows[i]["Phone"]);
    }
    Console.WriteLine("--- End ---\n\n");
    con.Close();
    }
    //-----------------------------------------------------
    static void DeleteWithCommand()
    {
    Console.Write("Enter customer Id to delete: ");
    string CustId = Console.ReadLine();
    Connect();
    OleDbCommand cmd = new OleDbCommand("DELETE FROM
tblCustomers WHERE CustomerId = '" + CustId + "'", con);
    int n = cmd.ExecuteNonQuery();
```

```
    Console.WriteLine("{0} records deleted successfully", n);
    con.Close();
    }
    //-----------------------------------------------------
    static void DeleteWithDataSet()
    {
    Connect();
    OleDbDataAdapter da = new OleDbDataAdapter("SELECT * FROM
tblCustomers", con);
    OleDbCommandBuilder cb = new OleDbCommandBuilder(da);
    da.InsertCommand = cb.GetInsertCommand();
    da.UpdateCommand = cb.GetUpdateCommand();
    da.DeleteCommand = cb.GetDeleteCommand();
    DataSet ds = new DataSet();
    da.Fill(ds, "tblCustomers");
    DataTable tbl = ds.Tables["tblCustomers"];
    Console.Write("Enter customer Id to delete: ");
    string CustId = Console.ReadLine();
    for (int i = 0; i < tbl.Rows.Count; i++)
    {
    if (CustId == tbl.Rows[ i]["CustomerId"] .ToString())
    {
    tbl.Rows[ i] .Delete();
    break;
    }
    }
    da.Update(ds, "tblCustomers");
    con.Close();
    Console.WriteLine("Delete succesfully");
    }
    }
    }
```

Sau khi biên dịch và chạy chương trình ta có kết quả như ở hình 13.4. Từ Menu, vì CSDL đã có sẵn trong Access nên chọn số 2, ta đã có kết quả như trên hình 13.4.

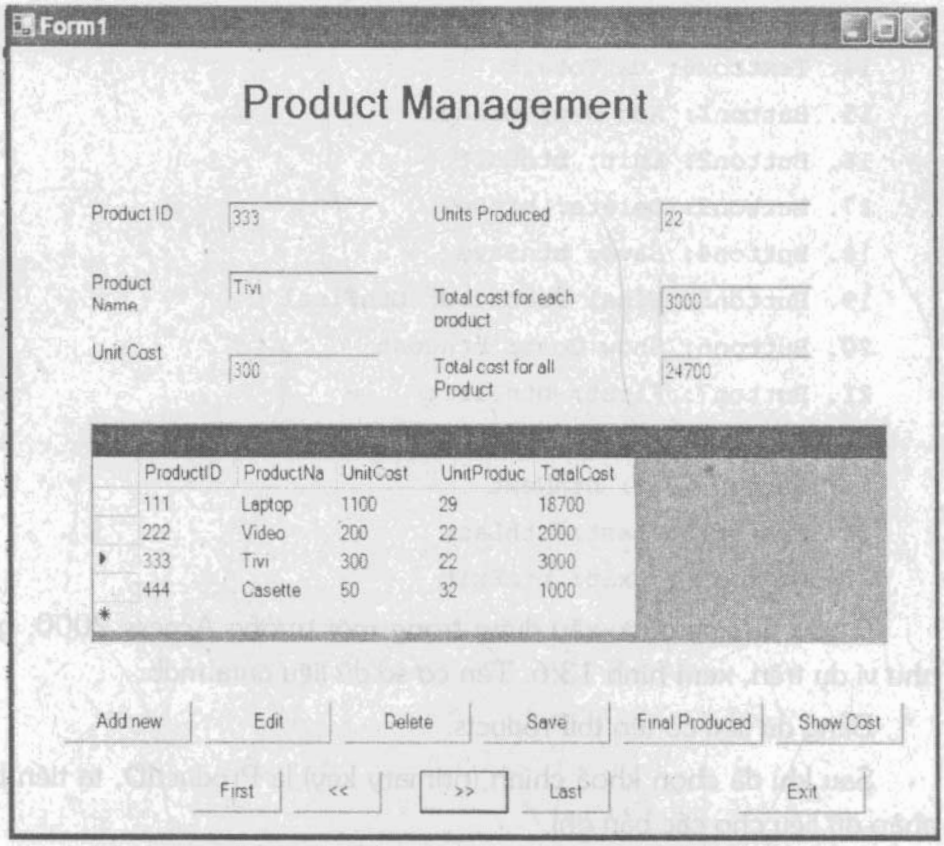


Hình 13.4

Bằng cách chọn lần lượt các mục trong Menu, ta có kết quả tương ứng. Khi đóng chương trình C#, nếu mở CSDL trong môi trường Access, ta có thông tin dữ liệu hoàn toàn nhất quán.

13.3. XÂY DỰNG CHƯƠNG TRÌNH TRONG WINDOWS APPLICATION

Ví dụ 13.2. Yêu cầu đặt ra là hãy xây dựng một giao diện Windows như hình 13.5, để quản trị CSDL là một bảng như đã liệt kê trong Datagrid.



Hình 13.5

Các thuộc tính của các điều khiển:

1. Label1: Product Management
2. Label2: Product ID
3. Label3: Product Name
4. Label4: Unit Cost
5. Label5: Units Produced
6. Label6: Total cost for each product
7. Label7: Total cost for all products
8. Datagrid1: dg
9. Textbox1: txtPId
10. Textbox2: txtName
11. Textbox3: txtUcost
12. Textbox4: txtUpro

```
13. Textbox5: txtTotalA
14. Textbox6: txtTotalE
15. Button1: Add New; btnAdd
16. Button2: Edit; btnEdit
17. Button3: Delete; btnDel
18. Button4: Save; btnSave
19. Button5: Final Produced; btnFinal
20. Button6: Show Cost; btnCost
21. Button7: First; btnFirst
22. Button8: <<; btnPrev
23. Button9: >>; btnNext
24. Button10: Last; btnLast
25. Button11: Exit; btnExit
```

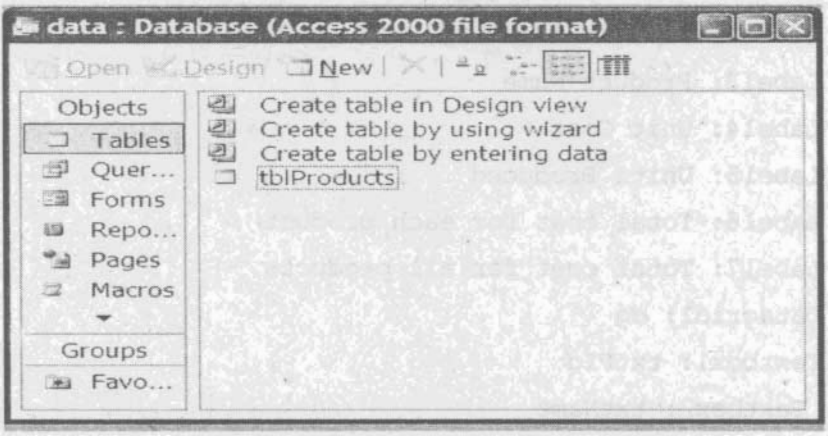
Cơ sở dữ liệu được xây dựng trong môi trường Access 2000, giống như ví dụ trên, xem hình 13.6. Tên cơ sở dữ liệu data.mdb.

Bảng dữ liệu có tên tblProducts.

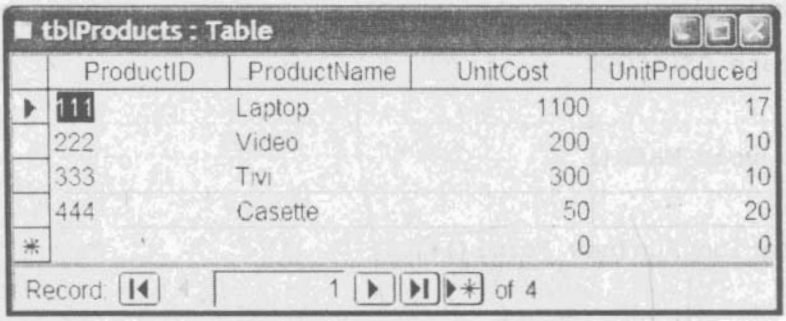
Sau khi đã chọn khoá chính (primary key) là ProductID, ta tiến hành nhập dữ liệu cho các bản ghi.

Để viết mã lệnh cho chương trình, ta cần phải khai báo tệp tiêu đề cho việc kết nối dữ liệu.

```
using System.Data.OleDb;
```



Hình 13.6



Hình 13.7

Sau đây là toàn bộ chương trình để quản trị cơ sở dữ liệu.

Thao tác các điều khiển trên giao diện.

1. Nhập dữ liệu: Nhấn nút Add New trên hình 13.5, các Textbox được xoá sạch. Sau đó ta nhập dữ liệu lần lượt vào các Textbox. Cuối cùng ta nhấn nút Save.

2. Sửa đổi dữ liệu: Chọn một bản ghi trong Datagrid, nhấn nút Edit. Rồi tiến hành sửa đổi. Chú ý trường ProductID không được sửa. Cuối cùng nhấn nút Save.

3. Muốn xoá một bản ghi trong CSDL, ta chọn một bản ghi trong Datagrid và nhấn nút Delete. Sau đó nhấn nút Save.

4. Khi ta đã nhập được 4 trường ProductID, ProductName, UnitCost, UnitProduced, ta nhấn nút Show Cost và nhấn nút Final Produced ta sẽ nhìn thấy kết quả tính toán trong hai Textbox còn lại.

5. Trở về bản ghi đầu tiên, nhấn nút First; Về bản ghi cuối cùng nhấn nút Last; Về bản ghi trước, nhấn nút << và bản ghi sau nhấn nút >> .

6. Thoát khỏi chương trình, nhấn nút Exit.

Toàn bộ mã chương trình liệt kê sau đây.

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
```

```
using System.Data.OleDb;
namespace CSDLCODE
{
static void Main()
{
Application.Run(new Form1());
}
OleDbConnection conn;
OleDbDataAdapter da;
DataSet ds; //Bien dataSet11
//----------------------------------------
private void Form1_Load(object sender, System.EventArgs e)
{
conn = new OleDbConnection(@"Provider =
Microsoft.Jet.Oledb.4.0; Data Source=..\..\data.mdb");
da = new OleDbDataAdapter("SELECT * FROM tblProducts", conn);
OleDbCommandBuilder cb = new OleDbCommandBuilder(da);
da.InsertCommand = cb.GetInsertCommand();
da.DeleteCommand = cb.GetDeleteCommand();
da.UpdateCommand = cb.GetUpdateCommand();
ds = new DataSet();
da.Fill(ds, "tblProducts");
dg.DataSource = ds;
dg.DataMember = "tblProducts";
txtPId.DataBindings.Add("Text", ds, "tblProducts.ProductId");
txtPName.DataBindings.Add("Text", ds, "tblProducts.ProductName");
txtUcost.DataBindings.Add("Text", ds, "tblProducts.UnitCost");
txtUpro.DataBindings.Add("Text", ds, "tblProducts.UnitProduced");
}
//------------------------------------------------
private void btnFirst_Click(object sender, System.EventArgs e)
{
this.BindingContext[ ds, "tblProducts"].Position = 0;
```

```
    }
    //-----------------------------------------------------
    private void btnPrev_Click(object sender, System.EventArgs e)
    {
    this.BindingContext[ ds, "tblProducts"] .Position--;
    }
    //-----------------------------------------------------
    private void btnNext_Click(object sender, System.EventArgs e)
    {
    this.BindingContext[ ds, "tblProducts"] .Position++;
    }
    //-----------------------------------------------------
    private void btnLast_Click(object sender, System.EventArgs e)
    {
    BindingContext[ ds, "tblProducts"] .Position =
this.BindingContext[ ds, "tblProducts"] .Count - 1;
    }
    //-----------------------------------------------------
    private void btnAdd_Click(object sender, System.EventArgs e)
    {
    this.BindingContext[ ds, "tblProducts"] .AddNew();
    }
    //-----------------------------------------------------
    private void btnDel_Click(object sender, System.EventArgs e)
    {
    BindingContext[ ds, "tblProducts"] .RemoveAt(this.
BindingContext[ ds, "tblProducts"] .Position);
    }
    //-----------------------------------------------------
    private void btnSave_Click(object sender, System.EventArgs e)
    {
    BindingContext[ ds, "tblProducts"] .EndCurrentEdit();
    da.Update(ds, "tblProducts");
```

```
    }
    //-------------------------------------------------
    private void btnExit_Click(object sender, System.EventArgs e)
    {
    this.Close();
    }
    //-------------------------------------------------------
    private void btnFinal_Click(object sender,
System.EventArgs e)
    {
    DataTable tbl = ds.Tables["tblProducts"];
    for (int i = 0; i < tbl.Rows.Count; i++)
    {
    int Upro;
    int.Parse(tbl.Rows[ i]["UnitProduced"].ToString());
    tbl.Rows[ i]["UnitProduced"] = Upro + 12;
    }
    }
    //-------------------------------------------------------
    private void btnShowCost_Click(object sender,
System.EventArgs e)
    {
    DataTable tbl = ds.Tables["tblProducts"];
    tbl.Columns.Add("TotalCost");
    double TotalA = 0;
    for (int i = 0; i < tbl.Rows.Count; i++)
    {
    double UCost = double.Parse(tbl.Rows[ i]["UnitCost"].ToString());
    int Upro = int.Parse(tbl.Rows[ i]["UnitProduced"].ToString());
    double TotalE = UCost * Upro;
    tbl.Rows[ i]["TotalCost"] = TotalE;
    TotalA+ = TotalE;
    }
```

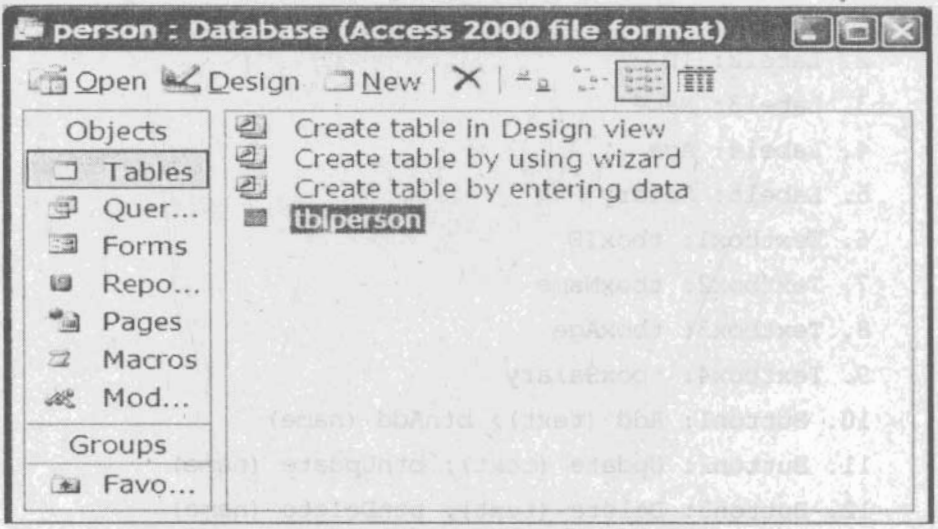
```
txtTotalA.Text = TotalA.ToString();
txtTotalE.DataBindings.Add("Text",ds, "tblProducts.TotalCost");
}
}
}
```

13.4. CHƯƠNG TRÌNH QUẢN TRỊ CSDL SỬ DỤNG WIZARD

Ở mục trên ta đã thực hiện chương trình quản trị cơ sở dữ liệu trong môi trường Access 2000 bằng cách viết mã chương trình. Ở đây ta sẽ xét bằng một ví dụ về quản trị CSDL bằng cách kéo thả và kết nối CSDL bằng Wizard.

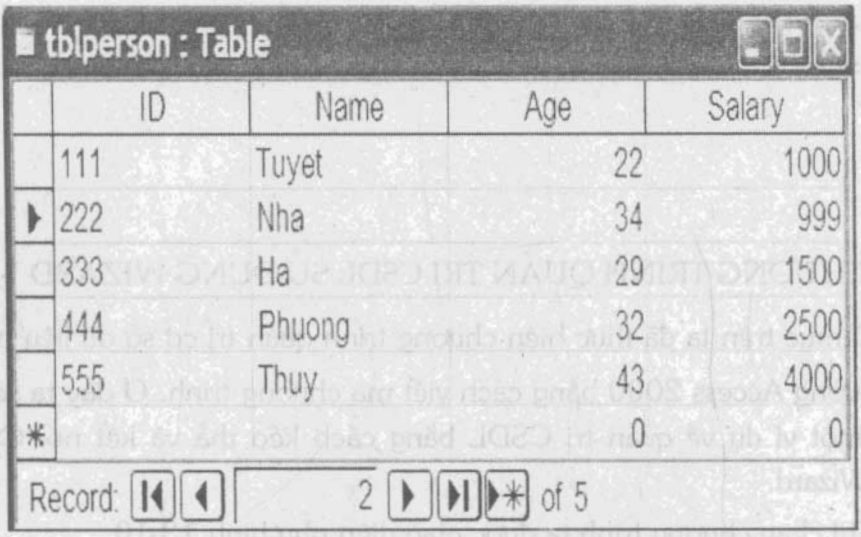
Khi chạy chương trình ta được giao diện như hình 13.10.

CSDL được lập trong Access 2000 với tệp dữ liệu person.mdb được đặt trong thư mục QLPerson của C# trong ổ đĩa C.



Hình 13.8

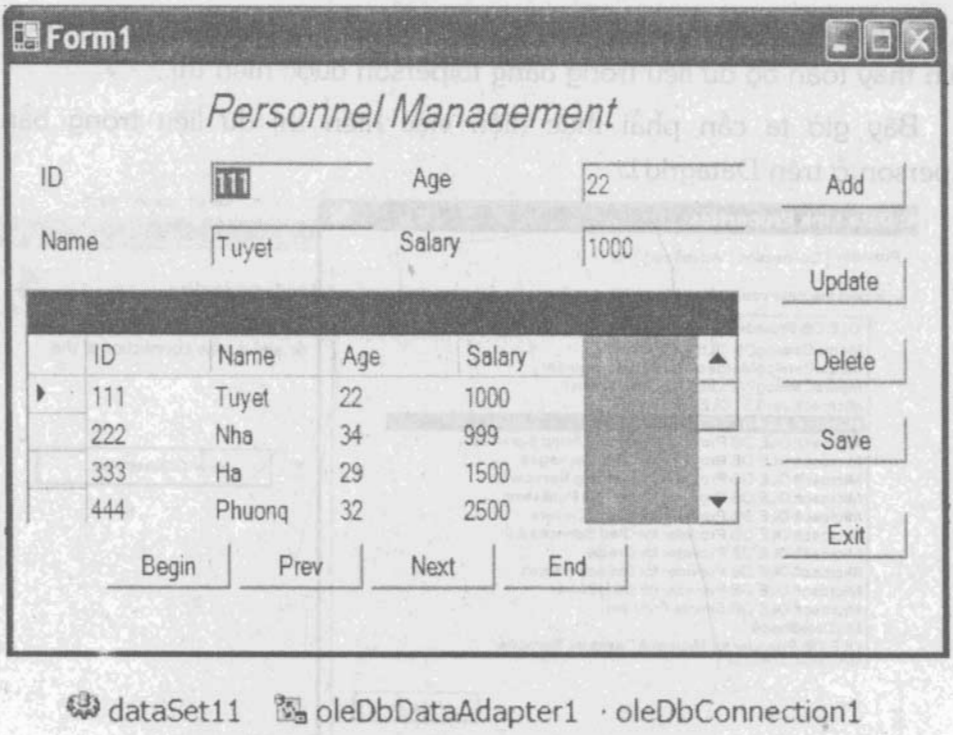
Tên bảng dữ liệu tblperson trong hình 13.8 và có nội dung như trên hình 13.9.



Hình 13.9

Giao diện hình 13.10 với các điều khiển có thuộc tính như sau.

```
1. Label1: text: Personnel Management
2. Label2: ID
3. Label3: Name
4. Label4: Age
5. Label5: Salary
6. Textbox1: tboxID
7. Textbox2: tboxName
8. Textbox3: tboxAge
9. Textbox4: tboxSalary
10. Button1: Add (text); btnAdd (name)
11. Button2: Update (text); btnUpdate (name)
12. Button3: Delete (text); btnDelete (name)
13. Button4: Save (text); btnSave (name)
14. Button5: Begin (text); btnBeg (name)
15. Button6: Prev (text); btnPrev (name)
16. Button7: Next (text); btnNext (name)
17. Button8: End (text); btnEnd (name)
18. Button9: Exit (text); btnExit (name)
19. Datagrid1 (name)
```



Hình 13.10

Kết nối CSDL:

Kéo điều khiển oleDbDataAdapter1 vào giao diện như hình 13.11.

Chọn Tab Provider → Microsoft Jet 4.0 OLE DB Provider → Next → OK.

Ta có tiếp cửa sổ như hình 13.12. Tại nút ... , ta chọn thư mục C:\QLPerson0\person.mdb và nhấn nút Test Connection và có kết quả Succeded. Nhấn tiếp nút OK ta nhận được oleDbConnection1 ở dưới hình 13.10.

- Để tạo Dataset1, ấn chuột phải vào oleDbDataAdapter1 dưới hình 13.10, ta có cửa sổ mới như hình 13.13. Ở đây ta chọn:

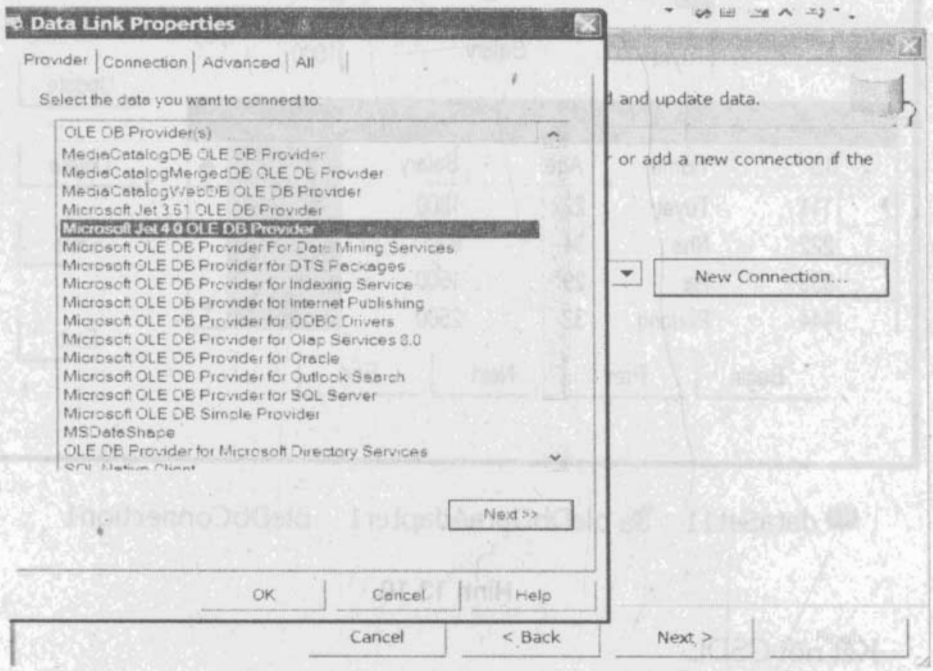
Generate Dataset, ta nhận được cửa sổ mới hình 13.14.

Sau đó nhấn nút OK, phía dưới hình 13.10 ta thấy thêm Dataset11.

- Để điền dữ liệu vào dataGrid, từ hình 13.3, ta chọn Preview Data. Và ta có cửa sổ mới hình 13.15.

Nhấn nút FillDataset ở ngay trong hình 13.15, tại cửa sổ Result, ta nhìn thấy toàn bộ dữ liệu trong bảng tblperson được hiển thị.

Bây giờ ta cần phải thực hiện việc hiển thị dữ liệu trong bảng tblperson ở trên Datagrid1.

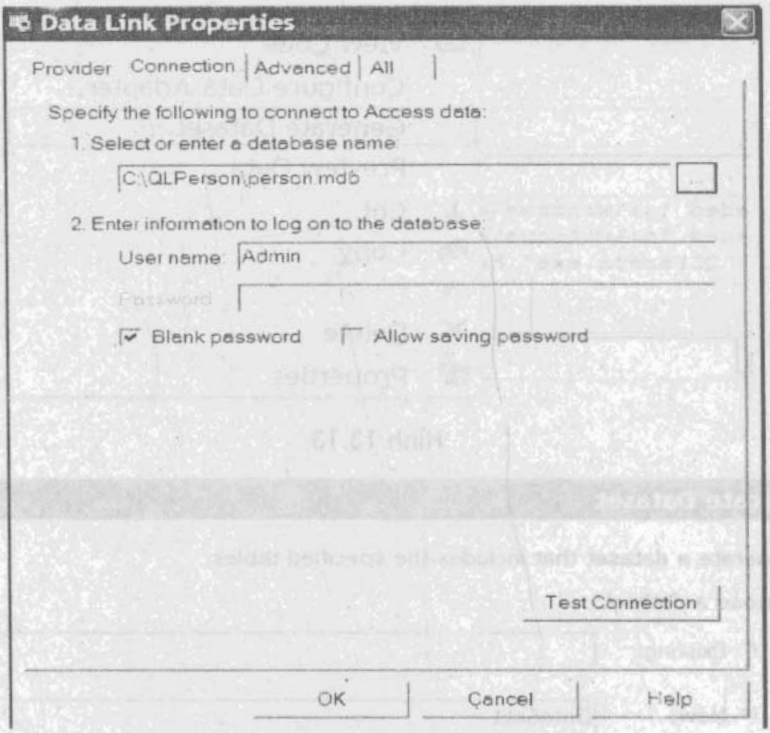


Hình 13.11

Bấm chọn Datagrid1 trên hình 13.10. Chọn Property của điều khiển này (nhấn chuột phải). Trong cửa sổ hình 13.16, tại hàng Datasource, chọn Dataset1, tại hàng Datamember, chọn tên bảng tblperson.

- Liên kết giữa các Textbox với các cột trong DataGrid. Nhấn chuột trái (single click) vào TextboxID. Sau đó chọn DataBinding trong Property của Textbox này. Mở hàng Text và nhấn chuột vào tên tBoxID. Cũng làm động tác tương tự với các Textbox còn lại: tboxName, tboxAge, tboxSalary.

Save toàn bộ chương trình và biên dịch.

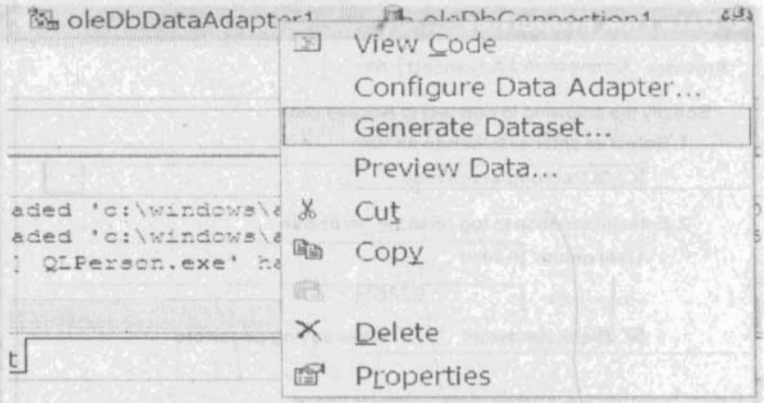


Hình 13.12

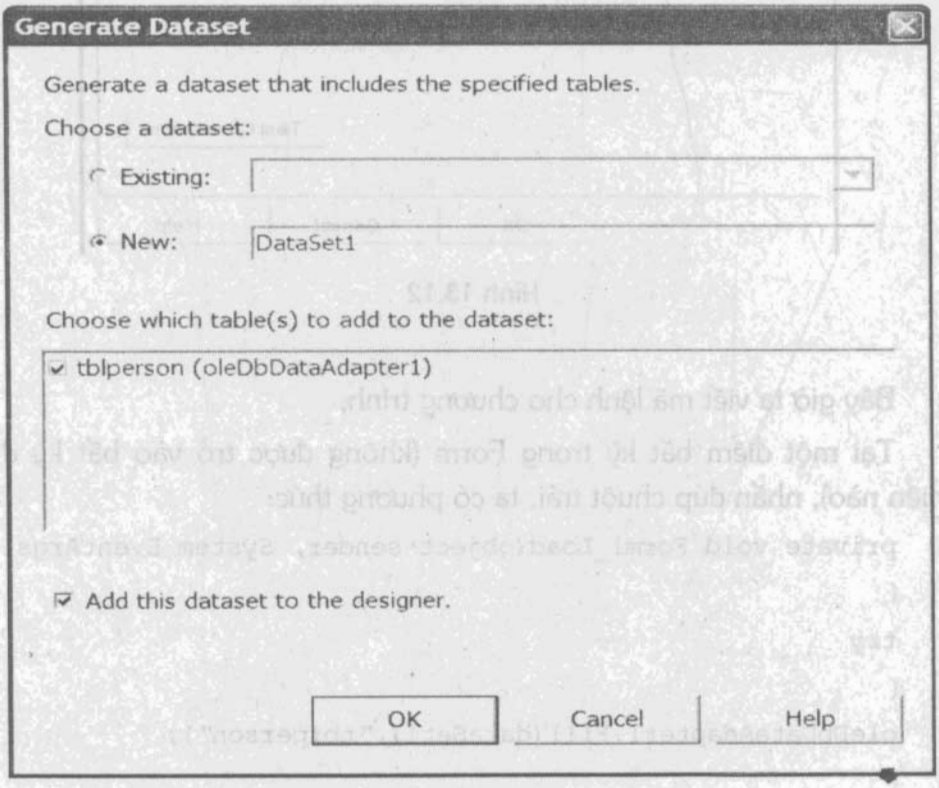
Bây giờ ta viết mã lệnh cho chương trình.

Tại một điểm bất kỳ trong Form (không được trỏ vào bất kỳ điều khiển nào), nhấn đúp chuột trái, ta có phương thức:

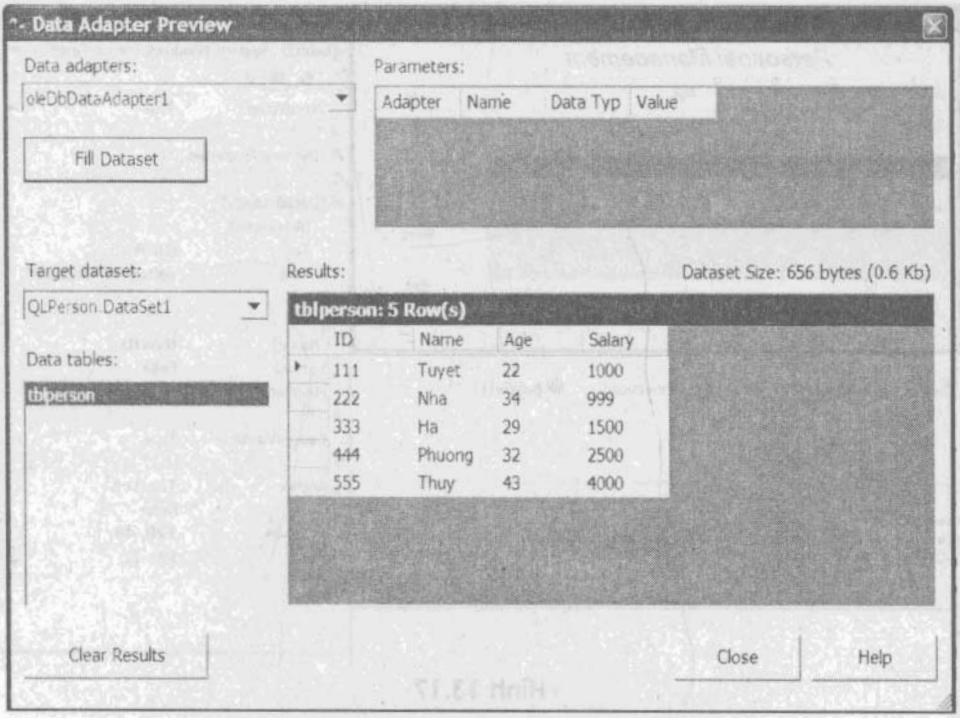
```
private void Form1_Load(object sender, System.EventArgs e)
{
try
{
oleDbDataAdapter1.Fill(dataSet11,"tblperson");
}
catch (Exception ex)
{
MessageBox.Show(ex.Message);
}
}
```



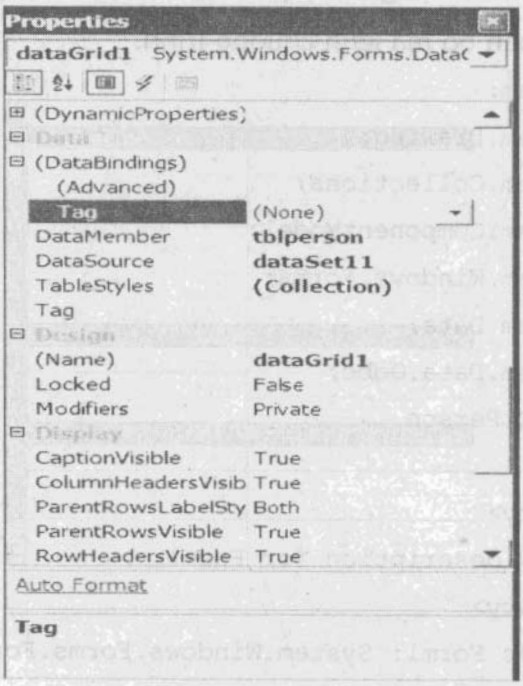
Hình 13.13



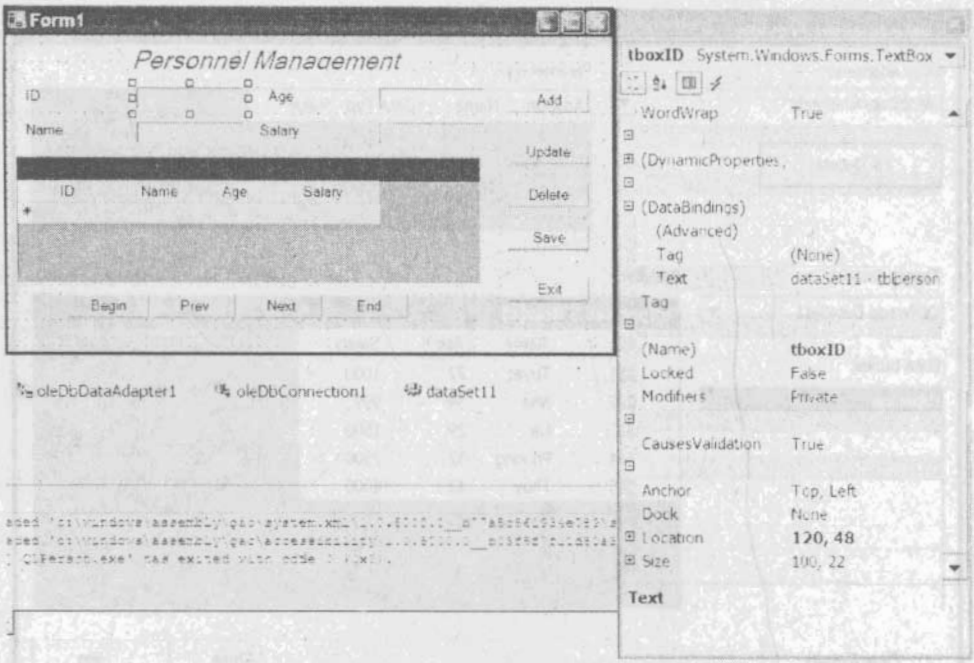
Hình 13.14



Hình 13.15



Hình 13.16



Hình 13.17

Ví dụ 13.3. Viết chương trình quản trị CSDL sử dụng Wizard.

Sau đây là toàn bộ mã lệnh chương trình.

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Data.Odbc;
namespace QLPerson
{
/// <summary>
/// Summary description for Form1
/// </summary>
public class Form1: System.Windows.Forms.Form
{ [STAThread]
```

```
    static void Main()
    {
    Application.Run(new Form1());
    }
    //-----------------------------------------------------
    private void Form1_Load(object sender, System.EventArgs e)
    {
    this.oleDbDataAdapter1.Fill(dataSet11,"tblperson");
    }
    //-----------------------------------------------------
    private void btnClose_Click(object sender, System.EventArgs e)
    {
    this.Close();
    }
    //-----------------------------------------------------
    private void btnAdd_Click(object sender, System.EventArgs e)
    {
    this.BindingContext[ dataSet11,"tblperson"].AddNew();
    }
    //-----------------------------------------------------
    private void btnExit_Click(object sender,System.EventArgs e)
    {
    this.BindingContext[ dataSet11,"tblperson"].
EndCurrentEdit();
    this.oleDbDataAdapter1.Update(dataSet11, "tblperson");
    }
    //-----------------------------------------------------
    private void button3_Click(object sender,System.EventArgs e)
    {
    this.BindingContext[ dataSet11,"tblperson"].
    RemoveAt(this.BindingContext[ dataSet11,
"tblperson"].Position);
    }
```

```
    //-----------------------------------------------------
    private void btnUpdate_Click(object sender,
System.EventArgs e)
    {
    string id = tboxID.Text;
    int n_row = BindingContext[ dataSet11,"tblperson"] .Count;
    int i = 0;
    for(i = 0; i < n_row; i++)
    {
    if(id == dataGrid1[ i,0] .ToString())
    {
    this.BindingContext[ dataSet11,tblperson"] .Position = i;
    dataGrid1.Select(i);
    break;
    }
    }
    if(i == n_row)
    {
    MessageBox.Show("Not found the record !");
    }
    }
    //-----------------------------------------------------
    private void btnBeg_Click(object sender, System.EventArgs e)
    {
    this.BindingContext[ dataSet11,"tblperson"] .Position = 0;
    }
    //-----------------------------------------------------
    private void btnEnd_Click(object sender, System.EventArgs e)
    {
    this.BindingContext[ dataSet11,"tblperson"] .Position=
BindingContext[ dataSet11,"tblperson"] .Count - 1;
    }
    //-----------------------------------------------------
```

```
private void btnPrev_Click(object sender, System.EventArgs e)
{
this.BindingContext[ dataSet11,"tblperson"] .Position--;
}
//-----------------------------------------------------
private void btnNext_Click(object sender, System.EventArgs e)
{
this.BindingContext[ dataSet11, "tblperson"] .Position++;
}
}
}
```

Tài Liệu Tham Khảo

1. Charles Petzold. *Programming Microsoft Windows with C#*. Microsoft Press, 2002.

2. Morgan Skinner, Karli Watson, Bill Evjen, Allen Jones. *Professional C#*. Third Edition. Wiley Publishing, Inc, 2004.

3. Radley L. Jones. *Teach Yourself The C# Language in 21 days*. Sams Publishing, 2004.

4. Jeff Ferguson, Brian Patterson, Jason Beres, Pierre Boutquin, Meeta Gupta. *C# Bible*. Wiley Publishing, Inc, 2002.

5. Kevin Hoffman , Lonny Kruger. *Microsoft Visual C# .NET 2003 Unleashed*. Sams Publishing, 2004.

6. Rebecca M. Riordan. *Microsoft ADO .NET Step by Step*. Microsoft Press, 2002.

7. Kevin Hoffman. *Microsoft Visual C Sharp 2005 Unleashed*. Microsoft Press, 2005.

8. Chrischian Nagelet Al. *Professional C Sharp 2005*. Microsoft Press, 2005.

9. Jesse Liberty. *Visual C# 2005: A Developer's Notebook*. Microsoft Press, 2005.

10. Phạm Công Ngô. *Tự học ngôn ngữ lập trình Visual C++ 6.0 từ cơ bản đến nâng cao*. Nhà xuất bản Thống kê, Hà Nội, 2002.

Mục lục

Chịu trách nhiệm xuất bản:
Chủ tịch HĐQT kiêm Tổng Giám đốc NGÔ TRẦN ÁI
Phó Tổng Giám đốc kiêm Tổng biên tập NGUYỄN QUÝ THAO

Tổ chức bản thảo và chịu trách nhiệm nội dung:
Chủ tịch HĐQT kiêm Giám đốc CTCP Sách ĐH–DN
TRẦN NHẬT TÂN

Biên tập và sửa bản in:
PHẠM THỊ PHƯỢNG
Trình bày bìa:
LƯU CHÍ ĐỒNG
Chế bản:
PHẠM ĐÌNH PHONG

Lập trình C# từ cơ bản đến nâng cao

Mã số: 7B656M7 - DAI

In 1000 bản (QĐ 64), khổ 16 x 24 cm, tại Xí nghiệp in Hà Tây.

Số ĐKKH xuất bản: 17 - 2007/ CXB/67 – 2217/ GD.

In xong và nộp lưu chiểu tháng 9 năm 2007.

CÔNG TY CỔ PHẦN SÁCH ĐẠI HỌC - DẠY NGHỀ
HEVOBCO
25 HÀN THUYÊN – HÀ NỘI
Website : www.hevobco.com.vn

VƯƠNG MIỆN KIM CƯƠNG
CHẤT LƯỢNG QUỐC TẾ

TÌM ĐỌC SÁCH THAM KHẢO KỸ THUẬT
CỦA NHÀ XUẤT BẢN GIÁO DỤC

1.	Giáo trình Vật lý đại cương tập một	Lương Duyên Bình
2.	Giáo trình Vật lý đại cương tập hai	Lương Duyên Bình
3.	Bài tập Vật lý đại cương tập một	Lương Duyên Bình
4.	Bài tập Vật lý đại cương tập hai	Lương Duyên Bình
5.	Giáo trình Kỹ thuật xung số	Đặng Văn Chuyết (CB)
6.	Giáo trình Kỹ thuật lập trình C	Nguyễn Linh Giang (CB)
7.	Giáo trình Kỹ thuật mạch điện tử	Đặng Văn Chuyết (CB)
8.	Giáo trình Linh kiện điện tử	Nguyễn Viết Nguyên (CB)
9.	Giáo trình Lý thuyết điều khiển tự động	Phan Xuân Minh (CB)
10.	Giáo trình Xử lý số tín hiệu	Nguyễn Quốc Trung (CB)
11.	Giáo trình Vi xử lý và cấu trúc máy tính	Ngô Diên Tập (CB)
12.	Giáo trình Khí cụ điện	Phạm Văn Chới
13.	Lập trình C# từ cơ bản đến nâng cao	Phạm Công Ngô
14.	Cơ sở tự động điều khiển quá trình	Nguyễn Văn Hoà

Bạn đọc có thể mua tại các Công ti Sách - Thiết bị trường học ở các địa phương hoặc các Cửa hàng của Nhà xuất bản Giáo dục :

Tại Hà Nội : 25 Hàn Thuyên ; 187B Giảng Võ ; 232 Tây Sơn ; 23 Tràng Tiền ;

Tại Đà Nẵng : Số 15 Nguyễn Chí Thanh ; Số 62 Nguyễn Chí Thanh ;

Tại Thành phố Hồ Chí Minh : 104 Mai Thị Lựu, Quận 1 ; Cửa hàng 451B - 453, Hai Bà Trưng, Quận 3 ; 240 Trần Bình Trọng – Quận 5.

Tại Thành phố Cần Thơ : Số 5/5, đường 30/4 ;

Website : www.nxbgd.com.vn

8 934980 763148

Giá: **26.000 đ**