

# MỤC LỤC

Chương 1: LÀM QUEN VỚI NGÔN NGỮ LẬP TRÌNH C/C++.....	4
1.1. GIỚI THIỆU.....	4
1.2. LÀM QUEN MÔI TRƯỜNG LẬP TRÌNH TURBO C++ 3.0.....	4
1.2.1. Khởi động và thoát khỏi Turbo C (TC).....	4
1.2.2. Các bước cơ bản khi lập một chương trình C/C++ .....	5
1.2.3. Cấu trúc chung của một chương trình C/C++ .....	5
1.2.4. Một số phím nóng thông dụng .....	6
1.2.5. Các thao tác cơ bản khi soạn thảo chương trình.....	6
1.3. CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ C .....	7
1.3.1. Từ khóa.....	7
1.3.2. Lời chú thích.....	7
1.3.3. Các kiểu dữ liệu cơ bản .....	7
1.3.4. Biến.....	10
1.3.5. Hằng.....	11
1.3.6. Biểu thức và các phép toán.....	13
1.3.7. Hàm .....	19
BÀI TẬP.....	20
Chương 2: CÁC LỆNH XUẤT NHẬP DỮ LIỆU VÀ CÁC CẤU TRÚC ĐIỀU KHIỂN.....	21
2.1. CÁC LỆNH XUẤT NHẬP DỮ LIỆU .....	21
2.1.1. Xuất dữ liệu với hàm printf() .....	21
2.1.2. Nhập dữ liệu với hàm scanf() .....	22
2.1.3. Nhập chuỗi với hàm gets() .....	23
2.1.4. Nhập ký tự với hàm getchar() và getch() .....	24
2.2. CÁC CẤU TRÚC ĐIỀU KHIỂN.....	25
2.2.1. Cấu trúc if ... else.....	25
2.2.2. Cấu trúc switch .....	27
2.2.3. Các cấu trúc lặp .....	28
2.2.4. Câu lệnh <i>break</i> , <i>continue</i> và <i>goto</i> .....	31
BÀI TẬP.....	35
Chương 3: HÀM.....	39
3.1. KHÁI NIỆM VỀ HÀM .....	39
3.2. XÂY DỰNG HÀM .....	39
3.3. ĐỆ QUY .....	42
3.3.1. Khái niệm đệ qui .....	42
3.3.2. Phương pháp thiết kế giải thuật đệ qui .....	42
BÀI TẬP.....	43
Chương 4: MẢNG VÀ CON TRỎ.....	45
4.1. KHAI BÁO MẢNG .....	45
4.2. XUẤT NHẬP TRÊN DỮ LIỆU KIỂU MẢNG.....	45
4.3. MẢNG VÀ CON TRỎ.....	45
BÀI TẬP.....	47
CHƯƠNG 5: XÂU KÝ TỰ.....	51
5.1. KHÁI NIỆM.....	51
5.2. KHAI BÁO VÀ KHỞI GÁN CHUỖI .....	51

5.3. PHÉP GÁN CHUỖI.....	51
5.4. CHUỖI VÀ KÝ TỰ .....	52
5.5. NHẬP VÀ XUẤT CHUỖI.....	53
5.5.1 Nhập chuỗi.....	53
5.5.2 Xuất chuỗi .....	53
5.6. SO SÁNH CHUỖI .....	53
5.7. MỘT SỐ HÀM VỀ CHUỖI.....	54
5.7.1. Các hàm trong thư viện <string.h>.....	54
5.7.2. Các hàm trong thư viện <stdlib.h>.....	55
5.8. MỘT SỐ VÍ DỤ VỀ XỬ LÝ CHUỖI.....	55
BÀI TẬP.....	57
CHƯƠNG 6: KIỂU CẤU TRÚC VÀ HỢP NHẤT .....	60
6.1. KIỂU CẤU TRÚC .....	60
6.1.1. Định nghĩa cấu trúc .....	60
6.1.2. Định nghĩa cấu trúc với typedef.....	60
6.1.3. Khai báo biến cấu trúc.....	61
6.1.4. Khởi động các biến cấu trúc .....	62
6.1.5. Truy cập vào các thành phần của cấu trúc.....	62
6.2. DANH SÁCH LIÊN KẾT .....	63
6.2.1. Khai báo danh sách liên kết.....	64
6.2.2. Các thao tác thường gặp trên danh sách liên kết .....	65
6.3. KIỂU UNION.....	68
BÀI TẬP.....	69
CHƯƠNG 7: KIỂU TẬP TIN.....	72
7.1. KHAI BÁO.....	72
7.2. MỞ FILE .....	72
7.3. ĐÓNG FILE .....	73
7.4. ĐỌC VÀ GHI DỮ LIỆU.....	73
7.4.1 Đọc/ghi từng ký tự.....	73
7.4.2. Đọc ghi từng dòng .....	74
7.4.3 Đọc ghi từng block .....	75
7.5. XUẤT NHẬP CÓ ĐỊNH DẠNG.....	76
7.6. TRUY XUẤT NGẪU NHIÊN .....	76
7.7. MỘT SỐ HÀM QUẢN LÝ FILE .....	78
BÁI TẬP.....	78
CHƯƠNG 8: ĐỒ HỌA .....	84
8.1. MÀN HÌNH TRONG CHẾ ĐỘ ĐỒ HỌA.....	84
8.2. KHỞI TẠO VÀ THOÁT KHỎI CHẾ ĐỘ ĐỒ HỌA .....	84
8.2.1. Khởi tạo chế độ đồ họa.....	84
8.2.2. Thoát khỏi chế độ đồ họa .....	84
8.3. TỌA ĐỘ VÀ CON TRỎ TRÊN MÀN HÌNH ĐỒ HỌA .....	85
8.3.1. Lấy kích thước màn hình.....	85
8.3.2. Di chuyển con trỏ .....	85
8.3.3. Vẽ điểm .....	85
8.4. ĐẶT MÀU TRÊN MÀN HÌNH ĐỒ HỌA .....	85
8.4.1. Đặt màu cho đối tượng cần vẽ.....	85

8.4.2. Đặt màu nền.....	86
8.5. CỬA SỔ TRONG CHẾ ĐỘ ĐỒ HỌA .....	86
8.5.1. Đặt cửa sổ trên màn hình.....	86
8.6. VIẾT CHỮ TRONG ĐỒ HỌA .....	86
8.6.1. Thiết lập font chữ .....	86
8.6.2. Thiết lập phân bố chữ.....	86
8.6.3. Viết một xâu ký tự lên màn hình .....	87
8.7. VẼ CÁC HÌNH CƠ BẢN.....	87
8.7.1. Chọn kiểu đường .....	87
8.7.2. Vẽ đoạn thẳng.....	87
8.7.3. Vẽ hình chữ nhật.....	91
8.7.4. Vẽ cung tròn .....	92
8.7.5. Vẽ đường tròn - Ellipse .....	92
8.7.6. Định MODE vẽ cho đoạn thẳng.....	92
8.8. TÔ MÀU CÁC HÌNH .....	93
8.8.1. Chọn kiểu tô .....	93
8.8.2. Vẽ hình chữ nhật có tô màu ở bên trong .....	93
8.8.3. Vẽ hình hộp chữ nhật .....	97
8.8.4. Vẽ và tô màu Ellipse.....	98
8.8.5. Vẽ hình quạt tròn .....	98
8.8.6. Vẽ hình quạt Ellipse .....	98
8.8.7. Làm loang màu một vùng kín.....	98
8.8.8. Vẽ đa giác .....	98
8.9. CÁC KỸ THUẬT TẠO HÌNH CHUYỂN ĐỘNG .....	100
8.9.1. Kỹ thuật lật trang màn hình.....	100
8.9.2. Lưu và di chuyển một vùng màn hình.....	102
BÀI TẬP.....	105

## Chương 1: LÀM QUEN VỚI NGÔN NGỮ LẬP TRÌNH C/C++

### 1.1. GIỚI THIỆU

Ngôn ngữ lập trình C do Dennis Ritchie tạo lập vào năm 1972 tại Bell Telephone Laboratories. Mục đích ban đầu của ngôn ngữ này là để thiết kế hệ điều hành UNIX.

C là một ngôn ngữ lập trình mạnh và mềm dẻo nên đã được nhiều người sử dụng. Tuy nhiên, các tổ chức khác nhau đã sử dụng các version khác nhau của C và đã nên nhiều sự khác biệt trong các version khác nhau của C gây khó khăn cho nhiều người lập trình. Để khắc phục vấn đề này, năm 1983, *Viện Tiêu chuẩn Quốc gia Hoa Kỳ* (ANSI) đã thành lập một ủy ban để đưa ra một định nghĩa chuẩn cho ngôn ngữ C, gọi là *ANSI Standard C*.

Ngôn ngữ này có tên là C vì trước nó đã có một ngôn ngữ được gọi là B. Ngôn ngữ B do Ken Thompson phát triển cũng tại Bell Labs. Có thể, B là để chỉ tên của phòng thí nghiệm, còn C chắc là do đứng sau B.

C là ngôn ngữ lập trình được đánh giá cao với các đặc điểm sau:

- ◇ *C là một ngôn ngữ mạnh và mềm dẻo.* Hạn chế duy nhất của C chính là sự hạn chế trong tư duy trừu tượng của chính người lập trình mà thôi. C được sử dụng cho nhiều mục đích khác nhau, như thiết kế các hệ điều hành, các bộ soạn thảo văn bản, đồ họa, trang tính, và thậm chí làm các chương trình dịch cho các ngôn ngữ khác.
- ◇ *C là một ngôn ngữ bình dân* được nhiều người lập trình chuyên nghiệp ưa dùng. Bằng chứng là đã có rất nhiều chương trình dịch khác nhau và nhiều tiện ích kèm theo.
- ◇ *C là một ngôn ngữ chuyển đổi được* tức là một chương trình C được viết cho một hệ máy vi tính (ví dụ IBM PC) có thể được dịch và chạy trên một hệ thống khác (ví dụ DEC VAX) mà chỉ cần thay đổi chút ít hoặc không cần thay đổi gì cả.
- ◇ *C là một ngôn ngữ ngắn gọn*, nó chỉ bao gồm một số các từ được gọi là từ khóa (keyword) làm cơ sở để tạo ra các câu lệnh của ngôn ngữ.

Với các đặc điểm trên, C là sự lựa chọn tuyệt vời đối với một ngôn ngữ lập trình. Nhưng có lẽ người ta còn nghe nói về C++ và một kỹ thuật lập trình mới được gọi là *lập trình hướng đối tượng*. Cũng không có gì đáng hoang mang cho lắm vì thực ra C++ chỉ là một ngôn ngữ siêu C với tất cả những gì mà C có cộng thêm với kỹ thuật lập trình hướng đối tượng. Nếu ai đó bắt đầu ngay bằng việc học C++, thì tất cả những gì đã được dạy khi học C vẫn còn được áp dụng cho C++. Trong khi học C, không những chỉ học một ngôn ngữ lập trình phổ thông và mạnh nhất hôm nay mà còn tự chuẩn bị cho mình một kỹ thuật lập trình hướng đối tượng cho ngày mai.

### 1.2. LÀM QUEN MÔI TRƯỜNG LẬP TRÌNH TURBO C++ 3.0

#### 1.2.1. Khởi động và thoát khỏi Turbo C (TC)

### 1.2.2. Các bước cơ bản khi lập một chương trình C/C++

**Bước 1:** Soạn thảo chương trình.

**Bước 2:** Dịch chương trình (nhấn phím **F9**), nếu có lỗi thì phải sửa lỗi.

**Bước 3:** Chạy chương trình (nhấn phím **Ctrl-F9**).

### 1.2.3. Cấu trúc chung của một chương trình C/C++

```
#include <thư viện.h> //khai báo các thư viện
[Khai báo các hằng biến, biến, kiểu, hàm...]
void main() //hàm chính
{
    ...
    <các câu lệnh>;
    ...
}
```

Ví dụ 1: Chương trình C đơn giản nhất

**Cách 1:**

```
#include <stdio.h> //khai báo thư viện chuẩn
void main()
{
    printf("Hello");
}
```

**Cách 2:**

```
#include <iostream.h> //khai báo thư viện xuất nhập C++
void main()
{
    cout<<"Hello";
}
```

Ví dụ 2: Chương trình tính diện tích hình tròn.

```
#include<conio.h>
#include<iostream.h>
#define PI 3.1416
void main()
{
```

```
float r,s;
cout<<"Nhap ban kinh R=\n";
cin >> r;
s=r*r*PI;
cout<<"Dien tich hinh tron="<<s;
getch();
}
```

#### 1.2.4. Một số phím nóng thông dụng

- **F2:** Lưu chương trình đang soạn thảo vào đĩa.
- **F3:** Mở file mới hoặc file đã tồn tại trên đĩa để soạn thảo.
- **Alt-F3:** Đóng file đang soạn thảo.
- **Alt-F5:** Xem kết quả chạy chương trình.
- **F8:** Chạy từng câu lệnh một trong chương trình.
- **Alt-X:** Thoát khỏi TC.
- **Alt-<Số thứ tự của file đang mở>:** Dịch chuyển qua lại giữa các file đang mở.
- **F10:** Vào hệ thống Menu của TC.

#### 1.2.5. Các thao tác cơ bản khi soạn thảo chương trình

##### 1.2.5.1. Các phím thông dụng

- **Insert:** Chuyển qua lại giữa chế độ đè và chế độ chèn.
- **Home:** Đưa con trỏ về đầu dòng.
- **End:** Đưa con trỏ về cuối dòng.
- **Page Up:** Đưa con trỏ lên một trang màn hình.
- **Page Down:** Đưa con trỏ xuống một trang màn hình.
- **Del:** Xóa ký tự ngay tại vị trí con trỏ.
- **Back Space (←):** Xóa ký tự bên trái con trỏ.
- **Ctrl-PgUp:** Đưa con trỏ về đầu văn bản.
- **Ctrl-PgDn:** Đưa con trỏ về cuối văn bản.
- **Ctrl-Y:** Xóa dòng tại vị trí con trỏ.

##### 1.2.5.2. Các thao tác trên khối văn bản

- Chọn khối văn bản: **Shift + <Các phím ←↑→↓>**
- **Ctrl-KY:** Xóa khối văn bản đang chọn

- **Ctrl-Insert**: Đưa khối văn bản đang chọn vào Clipboard
- **Shift-Insert**: Dán khối văn từ Clipboard xuống vị trí con trỏ.

### 1.3. CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ C

#### 1.3.1. Từ khóa

Từ khóa là những từ có một ý nghĩa hoàn toàn xác định, các từ khóa của C phải được viết chữ thường. Sau đây là các từ khóa của Turbo C:

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

#### 1.3.2. Lời chú thích

Các chú thích trong C được đặt trong cặp dấu */\* .... \*/*. Dòng chú thích có thể trên nhiều dòng.

*Ví dụ: /\* Bắt đầu lập \*/*

Trong C++ có thể dùng hai dấu *//* để tạo chú thích trên từng dòng.

*Ví dụ: // Bắt đầu lập*

Khi gặp các chú thích, C không dịch chúng sang ngôn ngữ máy.

#### 1.3.3. Các kiểu dữ liệu cơ bản

##### 1.3.3.1 Kiểu char

Kiểu **char** có kích thước 1 byte (8 bit) và biểu diễn được 1 ký tự trong bảng mã ASCII.

Có hai kiểu **char** là **signed char** (**char** có dấu) và **unsigned char** (**char** không dấu). Kiểu **signed char** biểu diễn một số nguyên từ -128 đến 127 và **unsigned char** biểu diễn số nguyên có giá trị từ 0 đến 255.

### 1.3.3.2. Kiểu số nguyên

Tên kiểu	Kích cỡ (byte)	Phạm vi biểu diễn
int	2	-32,768 .. 32,767
short int	2	-32,768 .. 32,767
long (int)	4	-2,147,483,468 .. -2,147,483,467
unsigned int	2	0 .. 65,535
unsigned long (int)	4	0 .. 4,294,967,295

#### Chú ý:

- Đối với hầu hết các ngôn ngữ lập trình, luôn luôn phân biệt giữa dữ liệu kiểu số và kiểu ký tự. Tuy nhiên, cho dù ký tự hay số đều được lưu trong bộ nhớ máy tính là **số** bởi vì mỗi một ký tự đều có **một mã duy nhất**.
- Trong C, việc phân biệt ký tự và số chỉ là một cách tương đối, thực ra khi C xem xét ký tự là xem xét đến mã ASCII của nó. **char** là một kiểu số nguyên một byte, nó được dùng để lưu trữ vừa ký tự, vừa số nguyên.

Ví dụ: Sau khi ta khai báo:

```
char c;
```

ta có thể thực hiện các phép gán sau:

```
c = 'A';
```

```
c = 65;
```

- Trong các chương trình, để đọc một ký tự vào từ thiết bị thì trong chuỗi định dạng của hàm *scanf()* ta dùng **%c**, cũng tương tự vậy đối với hàm *printf()*, nếu dùng **%d** trong hàm *printf()* thì sẽ in ra mã ASCII của ký tự đó.

#### Ví dụ:

1. 

```
#include <stdio.h>
#include <conio.h>
void main()
{
    char c;
    printf("\nNhập vào ký tự: "); scanf("%c", &c);
    printf("\nMã ASCII của %c là %d", c, c);
    getch();
}
```
2. 

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



```
void main()
{
    char c1, c2;
    printf("Nhập hai ký tự c1 và c2:\n ");
    printf("\nNhập ký tự thứ nhất: "); scanf("%c",&c1);
    printf("\nNhập ký tự thứ hai: "); scanf("%c",&c2);
    printf("Giá trị số của c1 là %d\n",c1);
    printf("Giá trị số của c2 là %d\n",c2);
    getch();
}
```

### 1.3.3.3. Kiểu số thực

Tên kiểu	Kích cỡ (byte)	Phạm vi biểu diễn
float	4	3.4E-38 .. 3.4E+38
double	8	1.7E-308 .. 1.7E+308
long double	10	3.4E-4932 .. 3.4E+4932

Kiểu **float** có độ chính xác là **6 chữ số sau dấu chấm thập phân**.

Kiểu **double** có độ chính xác là **15 chữ số sau dấu chấm thập phân**.

### 1.3.3.4. Kiểu liệt kê

Cú pháp: **enum** {<các\_giá\_trị\_liệt\_kê>} <biến>;

Ví dụ: enum {red, blue, green, yellow } color;

enum {bright, medium, dark } intensity;

Các thành phần của kiểu **enum** thường được đánh số từ 0, kiểu **enum** có kích thước là 1 byte như kiểu nguyên 1 byte..

Tuy nhiên cũng có thể xác định các giá trị mặc định bằng cách đưa các giá trị trực tiếp vào khi khai báo. Nếu ta đưa vào một giá trị thì các giá trị tiếp theo sau sẽ được tuần tự đánh tăng dần.

Ví dụ:

enum {APPLES, ORANGE = 10, LEMONS, GRAPES = -5, MELONS};

enum {APPLES=0, ORANGE=10, LEMONS=11, GRAPES=-5, MELONS=-4};

### 1.3.3.5. Kiểu void

Kiểu dữ liệu **void** không có trong tiêu chuẩn K&R, nhưng trong những năm gần đây nó đã được chấp nhận trong ngôn ngữ C.

Kiểu **void** có hai tính chất quan trọng:

- Thứ nhất, nó chỉ ra rằng một hàm không trả về giá trị.

Ví dụ : có thể định nghĩa một hàm như sau:

```
void func( int a, int b )
{
    ....
}
```

Khi sử dụng hàm, chỉ cần gọi như sau:

```
func( x, y );
```

- Tính chất thứ hai của **void** là để khai báo cho một con trỏ không kiểu.

### 1.3.3.6. Định nghĩa kiểu dữ liệu mới

Cú pháp: **typedef** <Tên\_kiểu> <Tên\_kiểu\_định\_nghĩa>;

Ví dụ: **typedef long int SONGUYEN;**

Với định nghĩa này thì các khai báo sau là giống nhau:

```
long int j;
```

```
SONGUYEN j;
```

### 1.3.4. Biến

Biến là một đại lượng có thể biểu diễn nhiều giá trị khác nhau trong một chương trình. Chúng được lưu trữ trong bộ nhớ ở một địa chỉ nào đó.

#### 1.3.4.1. Tên biến

Mỗi biến khi sử dụng trong chương trình đều **phải được đặt tên** theo quy định như sau:

- Tên biến là một dãy ký tự bao gồm: chữ cái, chữ số và dấu gạch ngang dưới ( \_ )
- Ký tự đầu tiên phải là chữ cái hoặc ký tự \_

**Chú ý:**

- Khi đặt tên biến không được đặt trùng tên với các từ khóa của C.
- Tên biến trong C có sự phân biệt giữa chữ hoa và chữ thường.

Ví dụ: **aBc**, **abc** hay **ABC** là khác nhau.

- Độ dài mặc định của tên biến là 32.
- Các tên hằng, tên mảng, tên hàm, tên kiểu, tên con trỏ,... được đặt theo quy định của tên biến

✎ Để thuận tiện trong việc lập trình, các tên biến chúng ta nên đặt in thường (lower\_case), các tên hằng nên đặt in hoa (UPPER\_CASE), các tên kiểu, tên hàm nên đặt theo dạng in hoa ký tự đầu (Title\_Case).

### 1.3.4.2. Khai báo biến

Mọi biến đều phải được khai báo trước khi sử dụng. Việc khai báo biến thực hiện theo cú pháp sau đây:

*<Tên\_kiểu\_dữ\_liệu>    <Danh\_sách\_biến>;*

*Ví dụ:*

`int j;    /* khai báo biến j kiểu nguyên */`

`float x, y;    /* khai báo hai biến x,y kiểu thực*/`

Biến có thể được khai báo ở mọi nơi trong chương trình, phạm vi ảnh hưởng của biến tùy thuộc vào vị trí của nó trong chương trình.

**Chú ý:**

- Trong khi khai báo các biến, chúng ta có thể gán giá trị cho các biến. Cú pháp để khởi gán một hay nhiều biến là:

*<Tên\_kiểu> <Tên\_biến> = <Giá\_trị\_khởi\_động>;*

*Ví dụ:*

`char c = 65;    /* khởi gán c là 65 */`

`int a = 6, b = 7;    /* khởi gán cho a và b */`

`float x = 6.5, y;    /* khởi gán cho x */`

- Lấy địa chỉ biến: Mỗi biến khi được khai báo sẽ được cấp phát một vùng nhớ, có kích thước là kích thước của kiểu, gồm một số byte liên tiếp. Để lấy địa chỉ của biến (tức là địa chỉ vùng nhớ chứa giá trị của biến) ta dùng phép toán **&** , cú pháp như sau :

***&**<tên\_biến>*

### 1.3.5. Hằng

Hằng là một đại lượng mà giá trị của nó không thay đổi trong khi thực hiện chương trình.

#### 1.3.5.1. Khai báo hằng

*Cú pháp khai báo:*

**const**    *<kiểu> <tên\_hằng> = <giá\_trị>;*

hoặc    **#define**   *<Tên\_hằng> <giá\_trị>*

Câu lệnh **#define** là câu lệnh tiền xử lý nên chỉ có thể đặt ở ngoài các hàm ở đầu chương trình hoặc bắt đầu của một khối.

Ví dụ: `#define PI 3.1416`

`const float R = 10.5;`

#### 1.3.5.2. Hằng nguyên

Nếu viết bình thường thì ngầm hiểu là số nguyên có giá trị trong khoảng từ -32768 đến 32767.

Ví dụ: 12, -25 là các hằng kiểu **int**.

123456L hay 123456l là các hằng kiểu **long int**.

- *Hằng hệ thập phân*: không bắt đầu bằng con số 0.
- *Hằng hệ bát phân*: bắt đầu bằng số 0 với dạng như sau: **0n<sub>1</sub>n<sub>2</sub>n<sub>3</sub>...**, trong đó  $n_i$  là số nguyên trong khoảng từ 0 đến 7.

Ví dụ:    `const int i = 010;       /* i = 8 */`

- *Hằng hệ thập lục*: hằng bắt đầu bằng 0x (0X), được viết theo dạng sau:

**0xn<sub>1</sub>n<sub>2</sub>n<sub>3</sub>...**    hay    **0Xn<sub>1</sub>n<sub>2</sub>n<sub>3</sub>...**

Trong đó  $n_i$  là một chữ số từ 0 đến 9 hoặc chữ cái từ a(A) đến f(F).

Ví dụ:    `const int i = 0x10;   /* i = 16 */`  
              `const int j = 0xFF;   /* j = 255 */`

### 1.3.5.3. Hằng thập phân

Các hằng dấu phẩy động được viết theo hai cách:

- *Dạng thập phân*: Số bao gồm phần nguyên, dấu chấm thập phân và phần thập phân.

Ví dụ:    23.45    -12.34

- *Dạng khoa học*: Số gồm hai phần:

- Phần định trị: là số nguyên hoặc thực dạng thập phân
- Phần bậc: là một số nguyên.
- Hai phần này cách nhau bởi ký tự *e* hoặc *E*.

Ví dụ:    12e+3       (biểu diễn số 12000)  
              12e-2       (biểu diễn số 0.12)

**Chú ý:** Các trường hợp sau đây khai báo hằng với các kiểu số thực khác nhau:

3.5            /\* Hằng double \*/  
 3.5f          /\* Hằng float \*/  
 3.5e3L       /\* Hằng long double \*/

### 1.3.5.4. Hằng ký tự

Hằng ký tự là một ký tự được viết trong hai dấu nháy đơn ‘...’. *Ví dụ:* ‘a’

**Chú ý:**

- Hằng ký tự cũng được xem là những trị nguyên vì mỗi ký tự tương ứng với một giá trị nguyên trong bảng mã ASCII. Vậy nó có thể tham gia vào các phép toán như mọi số nguyên khác.

Ví dụ: 'a' - 'A' cho kết quả là 32.

- Hằng ký tự còn có cách viết khác như sau: '\x<sub>1</sub>x<sub>2</sub>x<sub>3</sub>'. Trong đó x<sub>1</sub>x<sub>2</sub>x<sub>3</sub> là số hệ bát phân (cơ số 8) mà giá trị của nó bằng mã ASCII của ký tự.

Ví dụ: hằng ký tự 'a' có mã ASCII là 97, đổi sang bát phân là 0141, vậy hằng ký tự 'a' có thể viết cách khác là '\141'.

- **Hằng đa ký tự:** mã điều khiển

\a	alert	Chuông
\b	backspace	Đưa con trỏ lùi 1 ký tự
\f	form feed	Đưa con trỏ đến trang logic kế tiếp
\n	newline	Dòng mới (xuống dòng)
\r	carriage return	Đưa con trỏ về đầu dòng
\t	tab	Đặt dấu tab

#### 1.3.5.5. Hằng chuỗi ký tự

Hằng chuỗi ký tự (chuỗi) là một dãy ký tự bất kỳ đặt trong cặp dấu nháy kép "...".

*Ví dụ:* "Truong Dai hoc Khoa hoc" hay "" (chuỗi rỗng)

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

**Chú ý:** Cần phân biệt ký tự 'a' và chuỗi "a".

### 1.3.6. Biểu thức và các phép toán

#### 1.3.6.1. Biểu thức

Biểu thức là một dãy các toán hạng được kết nối với nhau bằng các toán tử, có thể sử dụng dấu ngoặc đơn ( ), và cho kết quả là một giá trị gọi là giá trị của biểu thức.

- Toán hạng có thể gồm các biến, các hằng, các hàm (lời gọi hàm).
- Toán tử có thể là các phép toán số học, logic, quan hệ, ...

*Ví dụ:* Các dãy biểu diễn dưới đây là các biểu thức

5 + j

5\*j + 6

5 >= (a^2) \* (b >> 2)

f()/(4\*x)

#### 1.3.6.2. Các phép toán

##### 1.3.6.2.1. Phép toán số học

- **Các phép toán hai ngôi**

Phép toán	Ý nghĩa	Dạng thức
+	Cộng	$a + b$
-	Trừ	$a - b$
*	Nhân	$a * b$
/	Chia	$a / b$
%	Lấy phần dư	$a \% b$

**Chú ý:** `int / int`  $\rightarrow$  `int` ( $2/3 \rightarrow 0$ ; nhưng (float)  $2/3 \rightarrow 0.6666667$  )

- **Phép toán một ngôi:** Phép toán - (âm) đứng trước một toán hạng, chỉ rõ là trả về giá trị trái dấu với toán hạng.

Ví dụ: -10

#### 1.3.6.2.2. Phép Toán Quan hệ

Phép toán	Ý nghĩa	Dạng thức
>	Lớn hơn	$a > b$
>=	Lớn hơn hay bằng	$a >= b$
<	Bé hơn	$a < b$
<=	Bé hơn hay bằng	$a <= b$
==	Bằng nhau	$a == b$
!=	Khác nhau	$a != b$

**Chú ý:**

- Kết quả của phép toán quan hệ luôn luôn là một số nguyên: **1** nếu đúng và **0** nếu sai.
- Các phép toán >, >=, <, <= là cùng thứ tự ưu tiên, hai phép toán ==, != cùng thứ tự ưu tiên nhưng thấp hơn thứ tự ưu tiên của bốn phép toán đầu.

#### 1.3.6.2.3. Phép toán logic

Phép toán	Ý nghĩa	Dạng thức
!	Phủ định	!a
&&	Phép giao	a && b
	Phép hoặc	a    b

Kết quả của phép toán logic trả về một trong hai giá trị: **1** nếu đúng và **0** nếu sai. Ý nghĩa của các phép toán được cho bởi các bảng sau:

A	b	!a	a && b	a    b
1	1	0	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	0

#### 1.3.6.2.4. Các phép toán xử lý bit (BitWise)

Phép toán	Ý nghĩa	Dạng thức
~	Lấy phần bù theo bit	~a
&	Phép AND theo bit	a & b
	Phép OR theo bit	a   b
^	Phép XOR theo bit	a ^ b
<<	Dịch trái	a << b
>>	Dịch phải	a >> b

**Chú ý:** Các phép toán xử lý bit chỉ thực hiện trên các toán hạng có kiểu dữ liệu là số nguyên như: **char**, **int**, **long** (kể cả **signed** hoặc **unsigned**).

Bảng sau đây cho kết quả của các phép toán:

a	b	~a	a & b	a   b	a ^ b
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

#### 1.3.6.2.5. Phép gán

**Ký hiệu:** =

C có hai kiểu gán giá trị như sau:

- Phép gán đơn giản có dạng  

$$\langle \text{biến} \rangle = \langle \text{biểu thức} \rangle$$

- Phép gán phức tạp có dạng

$\langle \text{biến} \rangle \text{pt} = \langle \text{biểu thức} \rangle$

trong đó  $pt$  có thể là một trong các phép toán sau: +, -, \*, /, %, &, |, ^, >>, <<

Ví dụ 1:  $a = b;$

$c = d = e = 1;$

$x = (y = 2) * (z = 3);$

Ví dụ 2:  $a += b;$  tương đương  $a = a + b;$

$a -= b;$  tương đương  $a = a - b;$

$a *= b;$  tương đương  $a = a * b;$

$a /= b;$  tương đương  $a = a / b;$

$a \% = b;$  tương đương  $a = a \% b;$

**Chú ý:** Trong phép gán đơn giản thì giá trị của toán hạng bên phải được gán nguyên vẹn cho biến bên trái nếu chúng cùng kiểu. Nếu chúng khác kiểu thì C sẽ thực hiện việc chuyển kiểu tự động.

#### 1.3.6.2.6. Phép toán tăng giảm

C đưa ra hai phép toán một ngôi dùng để tăng giảm giá trị của các biến (nguyên, thực và con trỏ) gọi là **phép toán tăng ++** và **phép toán giảm --**.

Phép toán tăng ++ sẽ cộng thêm 1.

Phép toán giảm -- sẽ trừ đi 1.

Các phép toán ++ hay -- có thể đặt trước hay sau biến và sự khác nhau là ở chỗ:

- Nếu đặt trước ( $++n$  hay  $--n$ ) có nghĩa là tăng/giảm giá trị  $n$  rồi mới sử dụng
- Nếu đặt sau ( $n++$  hay  $n--$ ) có nghĩa là sử dụng giá trị  $n$  trước, sau đó mới tăng/giảm giá trị của  $n$ .

Ví dụ 1: Nếu  $n = 5$  thì câu lệnh

$x = ++n;$  sẽ cho ra giá trị  $x = 6$

còn  $x = n++;$  sẽ cho ra giá trị  $x = 5$

Sau khi thực hiện hai câu lệnh này thì  $n$  đều có giá trị là 6.

Ví dụ 2: Câu lệnh  $j = ++n$  tương đương  $\begin{cases} n = n + 1 \\ j = n \end{cases}$

Câu lệnh  $j = n++$  tương đương  $\begin{cases} j = n \\ n = n + 1 \end{cases}$

Ví dụ 3: Xem xét kết quả của chương trình sau:

```
void main()
{
```



```
int x=2;
printf("%d %d\n",x++,sqr(x)); // 2 4
printf("%d %d\n",++x,sqr(x)); // 4 9
}
```

### 1.3.6.2.7. Phép toán điều kiện 3 ngôi ( ? : )

**Cú pháp:** *biểu\_thức\_logic ? biểu\_thức\_1 : biểu\_thức\_2*

**Diễn giải:** Nếu *biểu\_thức\_logic* đúng (khác 0) thì kết quả của phép toán là giá trị của *biểu\_thức\_1*, ngược lại kết quả toán tử là giá trị của *biểu\_thức\_2*.

Kiểu của phép toán điều kiện là kiểu lớn nhất trong các kiểu của *biểu\_thức\_1* và *biểu\_thức\_2*.

Toán tử này thực chất là cách viết tắt của cấu trúc **if ... else** (sẽ học sau).

Như vậy, câu lệnh:

***z = ( biểu\_thức\_logic ? biểu\_thức\_1 : biểu\_thức\_2 );***

tương đương với:

***if ( biểu\_thức\_logic ) z = biểu\_thức\_1;***  
***else z = biểu\_thức\_2;***

*Ví dụ:* Câu lệnh ***z = ((x < y) ? x : y);*** sẽ gán giá trị nhỏ nhất của *x* và *y* cho biến *z*.

### 1.3.6.2.8. Chuyển đổi kiểu

Trong một biểu thức, các toán hạng có thể có các kiểu dữ liệu khác nhau, để việc tính toán được chính xác, đôi lúc cần phải chuyển đổi kiểu dữ liệu cho phù hợp. Có hai cách chuyển đổi kiểu dữ liệu:

1. *Chuyển kiểu tự động:* Thực hiện theo quy tắc sau:

- ☐ Khi hai toán hạng trong một phép toán có kiểu khác nhau thì toán hạng có kiểu bé hơn sẽ tự động chuyển thành kiểu lớn hơn của toán hạng kia trước khi thực hiện phép toán.
- ☐ Đối với lệnh gán thì sự chuyển kiểu căn cứ vào kiểu dữ liệu của biến ở vế trái.

*Ví dụ 1:* ***1.2\*(10/3)*** cho kết quả là ***3.6***.

*Ví dụ 2:* Giả sử *n* là biến nguyên thì sau lệnh gán ***n = 2.7*** sẽ cho kết quả là ***n = 2***.

2. *Chuyển kiểu bắt buộc:* Thực hiện theo cú pháp sau

**( Kiểu ) Biểu\_thức**

Kiểu của giá trị biểu thức sẽ được chuyển thành *Kiểu*.

*Ví dụ 1:* ***(int)(1.4)\*10*** cho kết quả là ***10***

*Ví dụ 2:* ***int a=10, b=3;***

***float x;***

***x = (float) a / (float)b;***

Ví dụ 3: Sau đây là một chương trình minh họa

```
#include <stdio.h>

void main()
{
    float a = 66.7;
    printf( "%d %c\n", (int)a, (char)a ); // 66 B
}
```

**Chú ý:**

- Đối với các hàm toán học của thư viện chuẩn thì **giá trị của các đối và giá trị của hàm** đều có kiểu **double**, vì vậy để tính toán được chính xác ta cần phải chuyển đổi kiểu cho đối số.


Ví dụ: Để tính căn bậc hai của một biến nguyên  $n$ , ta nên viết :

```
sqrt((double)n)
```

- Phép chuyển đổi kiểu có cùng độ ưu tiên như các toán tử một ngôi.

#### 1.3.6.2.9. Độ ưu tiên của các phép toán

Bảng sau đây liệt kê độ ưu tiên của tất cả các phép toán trong cùng nhóm và các nhóm khác nhau:

Các Phép toán	Ưu tiên nhóm	Độ ưu tiên
() [] ->	Trái sang Phải	<b>CAO NHẤT</b> 
đổi_kiểu & * - + ~ ++ -- !	Phải sang Trái	
*/ %	Trái sang Phải	
+ -	Trái sang Phải	
<< >>	Trái sang Phải	
< <= > >=	Trái sang Phải	
== !=	Trái sang Phải	
&	Trái sang Phải	
^	Trái sang Phải	
	Trái sang Phải	
&&	Trái sang Phải	
	Trái sang Phải	

?:	Phải sang Trái	THẤP NHẤT
= += -= *= /= %= >>= <<= &= ^=	Phải sang Trái	
,	Trái sang Phải	

### 1.3.7. Hàm

Hàm là một khái niệm quan trọng bậc nhất ẩn dưới các ngôn ngữ bậc cao. Trong C không có khuynh hướng phân chia rành mạch thành thủ tục hoặc hàm như các ngôn ngữ bậc cao khác, tất cả mọi thủ tục và hàm đều được gọi một tên duy nhất là **hàm**.

Các chương trình được phát triển từ việc cài đặt các hàm. Các hàm ở mức thấp biểu diễn các phép toán đơn giản nhất, các hàm ở mức cao được tạo ra từ những hàm mức thấp.

*Ví dụ:* Sau đây là một hàm (mức thấp) đơn giản để tính bình phương của một số, nó biểu diễn một phép toán không được xây dựng trong C.

```
int sqr( int so )
{
    int k_qua;
    k_qua = so * so;
    return k_qua;
}
```

Một hàm giống như một cỗ máy được chuyên môn hoá, nó nhận dữ liệu vào, xử lý chúng trong những bộ phận đã được xác định, và trả về kết quả. Ví dụ như hàm *sqr()* nhận một số nguyên như dữ liệu vào và trả về bình phương của số đó như là kết quả (dữ liệu ra).

#### Hàm *main()*

Một chương trình C muốn thực hiện được phải có một hàm đặc biệt có tên là ***main()***. Để gọi thực hiện hàm *sqr()*, có thể viết:

```
main()
{
    extern int sqr();
    int kqua;
    kqua = sqr( 5 );
}
```

Tổ chức của hàm *main()* cũng giống như bất kỳ một hàm nào trong C, tuy nhiên với *main()* thường chúng ta không xác định kiểu của hàm và không khai báo các đối số. Thực ra *main()* không trả về giá trị và bản thân nó có hai đối số đặc biệt, gọi là đối số dòng lệnh.

## BÀI TẬP

**Bài tập 1.1:** Viết chương trình nhập vào độ dài hai cạnh của tam giác và góc giữa hai cạnh đó, sau đó tính và in ra màn hình diện tích của tam giác.

**Ý tưởng:**

Công thức tính diện tích tam giác:  $S = \frac{1}{2} a.b.\sin(\theta)$  với a,b là độ dài 2 cạnh và  $\theta$  là góc kẹp giữa 2 cạnh a và b.

**Bài tập 1.2:** Viết chương trình tính  $\sqrt[n]{x}$ ,  $x > 0$ .

**Ý tưởng:**

Ta có:  $\sqrt[n]{x} = x^{\frac{1}{n}} = e^{\frac{1}{n} \ln x}$

**Bài tập 1.3:** Viết chương trình nhập vào 2 số a, b. Sau đó hoán đổi giá trị của 2 số đó:

a/ Cho phép dùng biến trung gian.

b/ Không được phép dùng biến trung gian.

**Bài tập 1.4:** Viết chương trình nhập vào các số nguyên: a, b, x, y, ... sau đó in ra màn hình kết quả của các biểu thức sau:

$$\begin{array}{llll} \text{a/ } \frac{x+y}{2+\frac{x}{y}} & \text{b/ } \frac{(a+4)(b-2c+3)}{\frac{r}{2h}-9(a-1)} & \text{c/ } x^y, x>0 & \text{d/ } e^{\sqrt{|a+\sin^2(x)-x|}} \end{array}$$

**Bài tập 1.5:** Viết chương trình tính diện tích tam giác theo công thức sau:

$$S = \sqrt{p(p-a)(p-b)(p-c)} \text{ với } p = \frac{1}{2}(a+b+c)$$

**Bài tập 1.6:** Viết chương trình tính khoảng cách từ một điểm  $I(x_i, y_i)$  đến đường thẳng có phương trình  $D: Ax + By + C = 0$ .

**Gợi ý:**

$$\text{Công thức tính khoảng cách: } h = \frac{A.x_i + B.y_i + C}{\sqrt{A^2 + B^2}}$$

**Bài tập 1.7:** Viết chương trình tách một số n thành 2 số a, b sao cho tích  $P=a*b^2$  đạt cực đại với n được nhập vào từ bàn phím.

**Gợi ý:**

Gọi x là số thứ hai thì số thứ nhất là:  $(n-x)$ . Theo đề ta có:  $P(x) = x^2.(n-x)$ .

Hàm P đạt cực đại khi  $P'(x) = -3x^2 + 2nx = 0 \Rightarrow x = 2n/3$ .

## Chương 2: CÁC LỆNH XUẤT NHẬP DỮ LIỆU VÀ CÁC CẤU TRÚC ĐIỀU KHIỂN

### 2.1. CÁC LỆNH XUẤT NHẬP DỮ LIỆU

#### 2.1.1. Xuất dữ liệu với hàm printf()

Cú pháp của hàm printf() như sau:

```
int printf( const char *format [,arg,...] );
```

Chuỗi format có dạng: %[flags][width][.prec][l,L]<type>

- flags: nếu không có: dữ liệu được in ra canh phải
  - : dữ liệu được in ra canh trái
  - + : dữ liệu được in ra có dấu phụ.

blank : dữ liệu được in ra có dấu âm, nếu là dương thì dấu + được thay bằng khoảng trắng

# : đổi dạng biểu diễn (chuyển đúng kiểu với mã đổi kiểu ở đối số sau).

- width: chỉ độ rộng tối thiểu để in dữ liệu. Nếu số chữ số của dữ liệu bé hơn width thì các khoảng thừa được lấp đầy bởi các khoảng trắng. Nếu width là 0n: tương tự như trên nhưng thay vì khoảng trắng, bây giờ là số 0.
- prec: dạng số thực. Số con số có ý nghĩa sau dấu chấm thập phân.
- [l,L]: hai đối số để đổi dữ liệu thành long, L dùng cho kiểu double.
- type: mã định dạng, được liệt kê bởi bảng sau:

Kiểu	Đối số	Dạng xuất
d	số nguyên	int (decimal)
u	“	unsigned int (decimal)
o	“	int (octal)
x	“	hexadecimal (a, b, c, d, e, f)
X	“	Hexadecimal (A, B, C, D, E, F)
f	số thực	float, double (dấu . tĩnh)
e	“	float, double (biểu diễn khoa học) (e)
E	“	float, double (biểu diễn khoa học) (E)
		biểu diễn khoa học hoặc dấu . thập phân tùy

g	“	theo dạng nào ngắn nhất (e)
G	“	biểu diễn khoa học hoặc dấu . thập phân tùy theo dạng nào ngắn nhất (E)
c	ký tự	char
s	chuỗi ký tự	chuỗi ký tự
p	con trỏ	địa chỉ

Ví dụ 1: Với các lệnh:

```
printf("Nhap mon\n");
printf("TURBO C\n");
```

sẽ in ra màn hình:

```
Nhap mon
TURBO C
```

Ví dụ 2: Với khai báo:

```
int a=5, b=10;
```

thì lệnh:

```
printf("Tong cua %d va %d la %d",a,b,a+b);
```

sẽ in ra màn hình

```
Tong cua 5 va 10 la 15
```

### 2.1.2. Nhập dữ liệu với hàm scanf()

Cú pháp của hàm scanf() như sau:

```
int scanf( const char *format [,adds,...] );
```

Chuỗi format có dạng: %[width][l,L]<type>.

adds có dạng như sau &<biến>. Các đối số này là địa chỉ các biến tương ứng với chuỗi định dạng.

Toán tử địa chỉ: & để lấy địa chỉ của một biến. Giả sử x là biến thì &x địa chỉ của biến x.

Ví dụ: Nhập dữ liệu cho hai biến x và y:

```
int x; float y;
scanf("%d%f",&x,&y);
```

**Chú ý:**

- Khi dùng hàm scanf(), điều cần chú ý là việc làm sạch bộ đệm. Để thực hiện việc này, có thể dùng hàm fflush(stdin).

- Không nên sử dụng hàm scanf() để nhập chuỗi ký tự. Sau đây là một ví dụ minh họa cho các vấn đề trên.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    char st [50];
    gets(st);      /* nhập vào dòng TURBO C */
    puts(st);
    scanf("%s",st); /* nhập vào dòng TURBO C */
    puts(st);
    scanf("%s",st); /* không dùng lại để nhập */
    puts(st);
}
```

TURBO C?

TURBO C

TURBO C?

TURBO

C

### 2.1.3. Nhập chuỗi với hàm gets()

Cú pháp của hàm gets() như sau:

**char \*gets( char \*s );**

Trong đó đối số s là con trỏ (kiểu char) trỏ đến vùng nhớ chứa chuỗi ký tự nhận được. Hàm gets() nhận một chuỗi ký tự từ thiết bị vào chuẩn là bàn phím cho đến khi gặp “\n” (nhấn ENTER). Ký tự “\n” không được đặt vào chuỗi, thay vào đó là ký tự kết thúc “\0” và đặt vào vùng nhớ do s trỏ đến.

Hàm trả về địa chỉ chuỗi nhận được.

Ví dụ:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    char ht [30];
    printf("\nHọ và tên: ");
```

```

    gets(ht);
    printf("\nHọ và tên là %s.", ht);
}

```

#### 2.1.4. Nhập ký tự với hàm getchar() và getch()

**int getchar( void );**

Hàm getchar() nhận một ký tự từ bàn phím và trả về màn hình ký tự nhận được.

Ví dụ:

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int ch;
    printf("\nNhập một ký tự: ");
    getchar(ch);
    printf("\nKý tự đó là %c.", ch);
}

```

**Chú ý:** Để tránh làm việc sai sót với gets() và getchar() chúng ta nên làm sạch bộ đệm bàn phím bằng cách dùng hàm *fflush(stdin)* như sau:

Ví dụ:

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int ht[30];
    int tuoi;
    printf("\nNhập tuổi: ");
    scanf("%d",&tuoi);
    printf("\nNhập họ tên: ");
    fflush(stdin);
    gets(ht);
}

```

**Hàm getch()**



Hàm getch() hoạt động hoàn toàn giống như getchar() chỉ khác ở hai điểm: thứ nhất ký tự sẽ được nhận ngay không chờ nhấn ENTER. Thứ hai, ký tự nhập vào sẽ không được in ra màn hình.

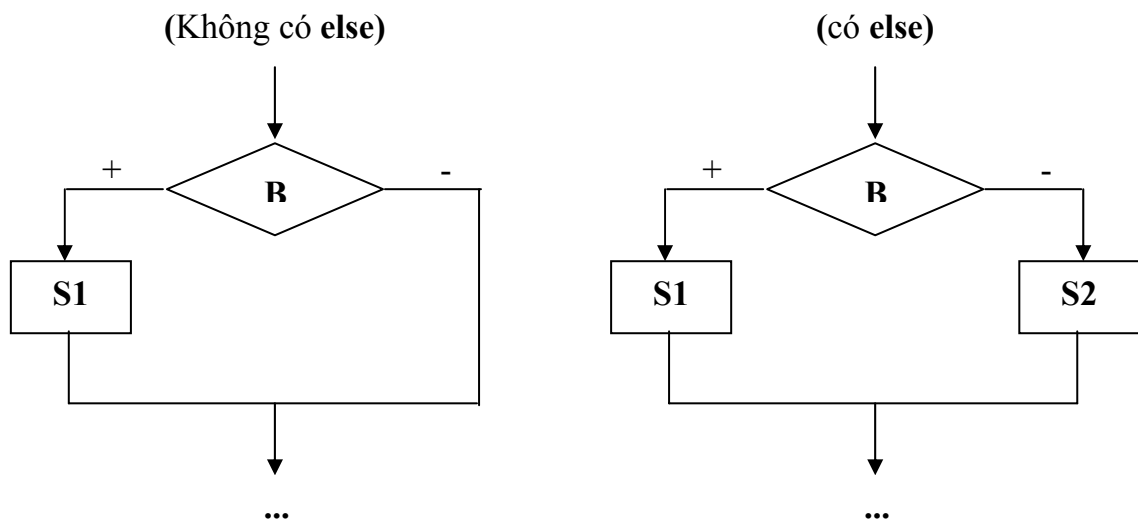
## 2.2. CÁC CẤU TRÚC ĐIỀU KHIỂN

Các cấu trúc điều khiển của C thường làm việc với các biểu thức điều kiện (biểu thức logic), các biểu thức điều kiện trong C có hai giá trị là 0 và 1 (đúng hơn là khác 0) tương ứng với hai trị TRUE và FALSE.

### 2.2.1. Cấu trúc if ... else

```
if (<biểu_thức_logic B>)<Lệnh S1>;
[ else <Lệnh S2>;]
```

Sơ đồ thực hiện:



Các câu lệnh sau if hoặc else có thể là đơn hoặc phức hợp. Nếu các câu lệnh if lồng nhau trong phạm vi một khối, thì else sẽ đi với if gần nhất.

Ví dụ 1: Chương trình nhập vào một ký tự và xem xét đó có phải là một ký tự alphabet không.

```
#include <stdio.h>
#include <ctype.h> /* thư viện chứa hàm isalpha() */
void main()
{
    char ch;
    printf( "Nhập một ký tự: " );
    scanf( "%c", &ch);
    if (isalpha( ch )) printf( "%c", ch );
    else printf( "%c không phải là chữ cái\n",ch);
}
```

}

Ví dụ 2: Giải phương trình bậc hai  $ax^2 + bx + c = 0$  ( $a \neq 0$ ).

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{
    float a,b,c,delta,x1,x2;
    printf( "\nNhập các hệ số a,b,c: " );
    printf( "\nNhập a= " ); scanf( "%f",&a);
    printf( "\nNhập b= " ); scanf( "%f",&b);
    printf( "\nNhập c= " ); scanf( "%f",&c);
    delta = b*b + 4*a*c;
    if (delta < 0)
        printf( "\nPhương trình vô nghiệm" );
    else if ( delta == 0 )
        printf( "\nPhương trình có nghiệm kép %.2f", -b/(2*a)
);
    else
    {
        x1 = (-b - sqrt(delta))/(2*a);
        x2 = (-b + sqrt(delta))/(2*a);
        printf("\nP.trình có 2 nghiệm x1=%f và x2= %f",x1,x2);
    }
    getch();
}
```

### 2.2.2. Cấu trúc switch

```
switch ( <biểu_thức> )
{
    case giá_trị 1 : [ Lệnh 1 ]
    case giá_trị 2 : [ Lệnh 2 ]
        .
        .
    case giá_trị n : [ Lệnh n ]
    [default : [ Lệnh n+1]]
}
```

<Biểu\_thức> phải có kết quả là một giá trị nguyên và *giá\_trị 1*, ..., *giá\_trị n* phải khác nhau và thuộc kiểu nguyên.

Nếu <biểu\_thức> có giá trị bằng với một *giá\_trị k* nào đó thì C sẽ chuyển điều khiển đến **case** *giá\_trị k* và thực hiện từ *Lệnh k* cho đến khi gặp lệnh **break** mới thoát khỏi **switch**, ngược lại nếu giá trị của <biểu\_thức> không có trong các giá trị được liệt kê sau **case** thì C sẽ thực hiện câu lệnh sau **default** (nếu có).

Ví dụ:

```
#include <stdio.h>

void main()
{
    int x;
    printf("Nhap so nguyen x= "); scanf("%d",&x);
    switch (x)
    {
        case 0: printf ("Zero\n");
        case 1:
        case 2:
        case 3: printf ("0, 1, 2 hoac 3\n");
                break;
        default: printf("Cac so khac\n");
    }
}
```

### 2.2.3. Các cấu trúc lặp

#### 2.2.3.1. Cấu trúc for

**For** (*biểu\_thức\_1*; *biểu\_thức\_2*; *biểu\_thức\_3*) <*câu\_lệnh*>;

Cấu trúc **for** làm việc như sau:

1. Đầu tiên, *biểu\_thức\_1* được thực hiện. Đây thường là phép gán để khởi động một hay nhiều biến.
2. Tiếp theo, *biểu\_thức\_2* được thực hiện. Đây là phần điều kiện thực hiện của cấu trúc **for**.
3. Nếu *biểu\_thức\_2* sai (bằng 0), chương trình sẽ thoát khỏi cấu trúc **for**. Nếu *biểu\_thức\_2* là đúng (khác 0) thì <*câu\_lệnh*> sẽ được thực hiện.
4. Sau khi thực hiện <*câu\_lệnh*>, thì *biểu\_thức\_3* mới được tính đến, và cấu trúc sẽ lặp lại việc kiểm tra lại *biểu\_thức\_2*.

**Chú ý:**

- *biểu\_thức\_1* chỉ được thực hiện một lần là khi bắt đầu cấu trúc, ngược lại *biểu\_thức\_2* và *biểu\_thức\_3* sẽ được thực hiện vào mỗi lần lặp.
- *biểu\_thức\_1, 2, 3* có thể vắng mặt, tuy nhiên dấu chấm phẩy bắt buộc phải đặt vào đúng vị trí của biểu thức đó.
- *biểu\_thức\_1, 2, 3* có thể là các biểu thức phẩy.

Ví dụ 1: Tính tổng  $n$  số nguyên liên tiếp:  $S = 1 + 2 + \dots + n$ .

```
#include <stdio.h>

void main()
{
    int n, i;
    long s;
    printf( "Nhập n: " ); scanf( "%d", &n);
    s = 0;
    for (i=0; i<=n; i++) s += i;
    printf( "Tổng là: %d\n", s );
}
```

Ví dụ 2:

```
#include <stdio.h>
#include <conio.h>
int a[]={4,5,3,2,6,9,1}; /*mang mot chieu*/
void main()
```

```
{
    int i,j,tmp;
    for (i=0,j=6;i<j;++i,--j)
    {
        tmp=a[i];
        a[i]=a[j];
        a[j]=tmp;
    }
    printf("\nDay so da dao nguoc");
    for (i=0;i<7;++i)
        printf("%d ",a[i]);
    getch();
}
```

### 2.2.3.2. Cấu trúc while

```
while (<biểu_thức_logic>)
{
    < câu_lệnh>;
}
```

Trong khi <biểu\_thức\_logic> vẫn còn đúng thì <câu\_lệnh> sẽ được thực hiện.

#### Ví dụ 1:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i=10;
    while (i>0)
    {
        printf("\ni= %d ",i); /*9 8 7 6 5 4 3 2 1*/
        i--;
    }
    getch();
}
```

#### Ví dụ 2:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int ch, ky_tu_trang=0;
    printf( "Nhap mot cau:\n" );
    ch = getchar();
    while (ch != '\n')
    {
        if (ch == ' ') ky_tu_trang++;
        ch = getchar();
    }
    printf( "So ky tu trang: %d\n",ky_tu_trang);
    getch();
}
```

### 2.2.3.3. Cấu trúc do ... while

<pre>Do {     &lt; câu_lệnh&gt;; } <b>while</b> (&lt;biểu_thức_logic&gt;;</pre>
---

Cấu trúc **do ... while** sẽ thực hiện <câu\_lệnh> cho đến khi nào <biểu\_thức\_logic> sai thì kết thúc.

Ví dụ 1: Viết lại chương trình ở ví dụ 2 của phần 2.3.2 dùng cấu trúc **do ... while**

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int ch, ky_tu_trang =0;
    clrscr();
    printf( "Nhap mot cau:\n" );
    do
    {
```

```

    ch = getchar();
    if (ch == ' ') ky_tu_trang ++;
}
while (ch != '\n');
printf( "So ky tu trang: %d\n", ky_tu_trang);
getch();
}

```

#### Ví dụ 2:

```

# include <stdio.h>
# include <conio.h>
void main()
{
    int s=0,i=1,n;
    clrscr();
    printf("\nTong binh phuong may so nguyen:");
    scanf("%d",&n);
    do
    {
        s+=(i*i);
        ++i;
    }
    while (i<=n);
    printf("\nTong binh phuong la : %d ",s);
    getch();
}

```

#### 2.2.4. Câu lệnh *break* , *continue* và *goto*

☒ **Câu lệnh *break*** : thoát ra khỏi các cấu trúc lặp hoặc cấu trúc **switch**.

Ví dụ: Tính tổng các giá trị được nhập vào từ bàn phím.

```

#include <stdio.h>
#include <conio.h>
void main()
{
    int total = 0, j;

```

```
while (1)
{
    printf("\nNhap so bat ky hoac so 0 de dung: ");
    scanf("%d", &j);
    if (j == 0) break;
    total += j;
    printf("Tong: %d\n", total);
}
printf("Tong cuoi: %d\n", total);
getch();
}
```

☒ **Câu lệnh *continue*** : Câu lệnh **continue** chỉ được sử dụng trong các cấu trúc lặp. Nó sẽ chuyển điều khiển chương trình trở lại đầu vòng lặp đang chứa nó và bỏ qua các câu lệnh sau nó.

Ví dụ 1: Tính tổng các giá trị (không âm) được nhập vào từ bàn phím.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int total = 0, m = 0, j;
    while (1)
    {
        printf("\nNhap so bat ky hoac so 0 de dung: ");
        scanf("%d", &j);
        if (j == 0) break;
        if (j < 0)
        {
            ++m;
            continue;
        }
        total += j;
        printf("Tong: %d\n", total);
    }
    printf("Tong cuoi: %d\n", total);
}
```



```
printf("voi %d so am khong tinh\n", m);
getch();
}
```

**Ví dụ 2:** Chương trình cắt các khoảng trắng cuối chuỗi.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    int len;
    char line[100];
    printf("Nhap 1 dong: "); gets(line);
    len = strlen(line);
    while (len)
    {
        len--;
        if ((line[len]==' ') || (line[len]=='\t'))
            continue;
        line[len+1] = '\0'; //chèn dấu kết thúc chuỗi
        break;
    }
    puts(line);
    getch();
}
```

☒ **Câu lệnh goto** : Lệnh goto dùng để chuyển quyền điều khiển tới một câu lệnh nào đó được chỉ định bởi nhãn. Cú pháp như sau:

**goto nhãn;**

Trong đó **nhãn** là một tên hợp lệ và tên này phải đặt trước lệnh mà ta muốn nhảy đến, cú pháp như sau :

**nhãn : lệnh;**

**Chú ý:**

- Nếu lệnh **goto** và **nhãn** nằm trong cùng một hàm thì lệnh **goto** chỉ cho phép nhảy từ vị trí này sang vị trí khác trong thân hàm đó, *không được nhảy từ hàm này sang hàm khác*.

- Không cho phép dùng lệnh **goto** để nhảy từ ngoài vào trong một khối lệnh nhưng cho phép nhảy từ trong khối lệnh ra ngoài.

Ví dụ: Tìm tất cả số nguyên tố trong khoảng hai số nguyên dương.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i,j,a,b;
    clrscr();
    printf("\nTim cac so nguyen to trong [a,b]:");
    printf("\nNhap a="); scanf("%d",&a);
    printf("\nNhap b="); scanf("%d",&b);
    for (i=a;i<=b;i++)
    {
        for (j=2;j<=i/2;j++)
        {
            if (i%j==0) goto tt;
            printf("%d\t",i);
        }
        tt: ;
    }
    getch();
}
```

## BÀI TẬP

Bài tập 2.1: Viết chương trình nhập vào một số nguyên và kiểm tra xem số vừa nhập là số chẵn hay số lẻ.

Bài tập 2.2: Viết chương trình giải phương trình bậc nhất  $ax+b=0$ .

Bài tập 2.3: Viết chương trình nhập vào tuổi của một người và cho biết người đó là thiếu niên, thanh niên, trung niên hay lão niên. Biết rằng: nếu tuổi nhỏ hơn 18 là thiếu niên, từ 18 đến 39 là thanh niên, từ 40 đến 60 là trung niên và lớn hơn 60 là lão niên.

Bài tập 2.4: Viết chương trình tính tổng  $S = 1+2+...+N$  theo 3 cách:

**Cách 1:** Dùng cấu trúc *for*.

**Cách 2:** Dùng cấu trúc *do...while*.

**Cách 3:** Dùng cấu trúc *while*.

Bài tập 2.5: Viết chương trình nhập vào N số nguyên từ bàn phím. Hãy tính và in ra màn hình tổng của các số vừa được nhập vào.

**Ý tưởng:**

Dùng phương pháp cộng dồn. Cho vòng lặp **for** chạy từ 1 tới N, ứng với lần lặp thứ *i*, ta nhập vào số nguyên X và đồng thời cộng dồn X vào biến S.

Bài tập 2.6: Viết chương trình nhập vào các số nguyên cho đến khi nào gặp số 0 thì kết thúc. Hãy đếm xem có bao nhiêu số chẵn vừa được nhập vào.

**Ý tưởng:**

Bài toán này không biết chính xác số lần lặp nên ta không thể dùng vòng lặp FOR. Vì phải nhập vào số nguyên N trước, sau đó mới kiểm tra xem  $N=0$ ? Do đó ta nên dùng vòng lặp **do...while**.

Bài tập 2.7: Viết chương trình tính số Pi với độ chính xác Epsilon, biết:

$$Pi/4 = 1-1/3+1/5-1/7+...$$

**Ý tưởng:**

Ta thấy rằng, mẫu số là các số lẻ có qui luật:  $2*i+1$  với  $i=1,...,n$ . Do đó ta dùng *i* làm biến chạy.

Vì tính số Pi với độ chính xác **Epsilon** nên không biết trước được cụ thể số lần lặp, do đó ta phải dùng vòng lặp **while** hoặc **do...while**. Có nghĩa là phải lặp cho tới khi  $t=4/(2*i+1) \leq \text{Epsilon}$  thì dừng.

Bài tập 2.8: Viết chương trình nhập vào số nguyên N. In ra màn hình tất cả các ước số của N.

**Ý tưởng:**

Cho biến *i* chạy từ 1 tới N. Nếu  $N \text{ MOD } i=0$  thì viết *i* ra màn hình.

Bài tập 2.9: Viết chương trình tìm USCLN và BSCNN của 2 số a, b được nhập vào từ bàn phím.

**Ý tưởng:**

- Tìm USCLN: Lấy số lớn trừ số nhỏ cho đến khi  $a=b$  thì dừng. Lúc đó:  $USCLN=a$ .
- $BSCNN(a,b) = a*b \text{ DIV } USCLN(a,b)$ .

Bài tập 2.10: Viết chương trình tìm các số có 3 chữ số  $\overline{abc}$  sao cho:  $\overline{abc} = a^3 + b^3 + c^3$ .

**Ý tưởng:**

Dùng phương pháp vét cạn. Ta biết rằng:  $a$  có thể có giá trị từ  $1 \rightarrow 9$  (vì  $a$  là số hàng trăm),  $b, c$  có thể có giá trị từ  $0 \rightarrow 9$ . Ta sẽ dùng 3 vòng lặp **for** lồng nhau để duyệt qua tất cả các trường hợp của  $a, b, c$ .

Ứng với mỗi bộ  $abc$ , ta sẽ kiểm tra: Nếu  $100.a + 10.b + c = a^3 + b^3 + c^3$  thì in ra bộ  $abc$  đó.

Bài tập 2.11: Viết chương trình nhập vào số tự nhiên  $N$  rồi thông báo lên màn hình số đó có phải là số nguyên tố hay không.

**Ý tưởng:**

$N$  là số nguyên tố nếu  $N$  không có ước số nào từ  $2 \rightarrow N \text{ div } 2$ . Từ định nghĩa này ta đưa ra giải thuật:

- Đếm số ước số của  $N$  từ  $2 \rightarrow N \text{ div } 2$  lưu vào biến  $d$ .
- Nếu  $d=0$  thì  $N$  là số nguyên tố.

Bài tập 2.12: Viết chương trình nhập vào từ bàn phím: giờ, phút, giây. Cộng thêm một số giây cũng được nhập từ bàn phím. Hãy in ra kết quả sau khi cộng xong.

**Gợi ý:**

- Gọi số giây được cộng thêm là:  $ss$ . Gán  $giây:=giây+ss$ .
- Nếu  $giây \geq 60$  thì:  $phút:=phút + giây \text{ DIV } 60$  và  $giây:=giây \text{ MOD } 60$ .
- Nếu  $phút \geq 60$  thì:  $giờ:=giờ + phút \text{ DIV } 60$  và  $phút:=phút \text{ MOD } 60$ .

Bài tập 2.13: Viết chương trình tìm Max, Min của 4 số:  $a, b, c, d$ .

Bài tập 2.14: Viết chương trình nhập vào số tự nhiên  $N$  rồi thông báo lên màn hình số đó có bao nhiêu chữ số và tổng của các chữ số của  $N$ .

Bài tập 2.15: Viết chương trình nhập vào ngày, tháng, năm. Máy sẽ hiện lên ngày, tháng, năm hôm sau.

**Gợi ý:**

Biện luận theo tháng. Gom tháng thành 3 nhóm: tháng có 31 ngày (1,3,5,7,8,10,12), tháng có 30 ngày (4,6,9,11) và tháng 2 (có 28 hoặc 29 ngày tùy theo năm nhuận).

Dùng lệnh lựa chọn:

**switch** (tháng)

{

case 1,3,5,7,8,10,12: .....

```

    case 4,6,9,11: .....
    case 2: .....
}

```

**Bài tập 2.16:** Viết chương trình in ra màn hình các giá trị của bảng mã ASCII từ 0→255.

**Bài tập 2.17:** Viết chương trình in ra màn hình các số nguyên từ 1 đến 100 sao cho cứ 10 số thì xuống dòng.

**Gợi ý:**

Cho biến *i* chạy từ 1 → 100. In ra màn hình *i* và kiểm tra: nếu *i* MOD 10=0 thì “\n”.

**Bài tập 2.18:** Viết chương trình in ra màn hình bảng cửu chương.

**Gợi ý:**

Dùng 2 vòng lặp **for** lồng nhau: *i* là số bảng cửu chương (2...9), *j* là số thứ tự trong từng bảng cửu chương (1...10).

```
For ( :=2;i<= 9;i++)
```

```
    For (j=1; j<= 10;j++) cout<<i<<"x"<<j<<"="<<i*j<<"\n";
```

**Bài tập 2.19:** Viết chương trình tính các tổng sau:

$S_0 = n! = 1*2*...*n$       {*n* giai thừa}

$S_1 = 1 + 1/2 + ... + 1/n$

$S_2 = 1 + 1/2! + ... + 1/n!$

$S_3 = 1 + x + x^2/2! + x^3/3! + ... + x^n/n!$

$S_4 = 1 - x + x^2/2! - x^3/3! + ... + (-1)^n x^n/n!$

$S_5 = 1 + \sin(x) + \sin^2(x) + ... + \sin^n(x).$

**Bài tập 2.20:** Viết chương trình để tìm lời giải cho bài toán sau:

Trong giỏ vừa thỏ vừa gà,

Một trăm cái cẳng bốn ba cái đầu.

Hỏi có mấy gà mấy thỏ?

**Bài tập 2.21:** Viết chương trình để tìm lời giải cho bài toán sau:

Trăm trâu trăm bó cỏ

Bó lại cho tròn

Trâu đứng ăn năm

Trâu nằm ăn ba

Năm trâu nghé ăn một.

Hỏi có bao nhiêu trâu đứng, trâu nằm, trâu nghé?

**Bài tập 2.22:** Viết chương trình nhập vào các số nguyên từ bàn phím cho đến khi nào gặp số nguyên tố thì kết thúc nhập. Tính tổng các số chẵn và trung bình cộng các số lẻ.

**Bài tập 2.23:** Viết chương trình in ra màn hình tất cả các số nguyên tố từ 2 đến N. Với N được nhập từ bàn phím.

**Bài tập 2.24:** Viết chương trình phân tích một số ra thừa số nguyên tố. Ví dụ: N=100 sẽ in ra màn hình:

```
100 | 2
    | 2
    | 5
    | 5
    |
```

**Bài tập 2.25:** Số hoàn thiện là số tự nhiên có tổng các ước của nó (không kể chính nó) bằng chính nó. Viết chương trình kiểm tra xem một số được nhập vào từ bàn phím có phải là số hoàn thiện hay không? Ví dụ: 6, 28 là các số hoàn thiện.

**Gợi ý:**

- Tính tổng các ước số của N: từ 1  $\rightarrow$  N div 2 lưu vào biến S.
- Nếu S=N thì N là số hoàn thiện.

**Bài tập 2.26:** Viết chương trình in ra các số nguyên từ 1 đến  $N^2$  theo hình xoắn ốc với N được nhập vào từ bàn phím. Ví dụ, với N=5 ta có:

```

1   2   3   4   5
16  17  18  19  6
15  24  25  20  7
14  23  22  21  8
13  12  11  10  9
```

## Chương 3: HÀM

### 3.1. KHÁI NIỆM VỀ HÀM

Hàm là một bộ phận quan trọng trong quá trình phát triển chương trình. Hàm là một công cụ để thực hiện việc modul hoá chương trình. Trong ngôn ngữ C, một hàm có các đặc trưng sau:

- Có thể nằm ngay trong module văn bản chính, hoặc được đưa vào từ các thư viện được khai báo bằng các câu lệnh `#include` hoặc được biên dịch riêng rẽ.
- Hàm được gọi từ chương trình chính, từ một hàm khác, hoặc từ chính nó.
- Hàm có thể nhận hay không nhận các đối số và có thể có giá trị trả về hoặc không có.

Trong C, không cho phép các hàm định nghĩa lồng nhau, nhưng một hàm có thể gọi nhiều hàm khác thực hiện.

### 3.2. XÂY DỰNG HÀM

Một hàm được xây dựng theo cú pháp sau:

```
<Kiểu dữ liệu>  tên_hàm ( [<Danh sách đối số>] )  
{  
    <Thân hàm>;  
}
```

Giải thích:

- <Kiểu dữ liệu>: thuộc kiểu vô hướng hoặc các kiểu được định nghĩa.
- <Danh sách đối số>: cần khai báo kiểu dữ liệu cụ thể cho từng đối số. Khi các đối số có cùng kiểu thì cũng không được viết gộp lại.

*Ví dụ:*

```
void func(int a,b)    // không hợp lệ  
void func(int a, int b) // hợp lệ
```

- <Thân hàm>: bao gồm các khai báo biến cục bộ và các câu lệnh. Nếu hàm có trả về giá trị thì trong thân hàm phải sử dụng câu lệnh **return** <biểu thức>;

Ví dụ 1: Viết hàm tính  $x^2$ .

```
#include <stdio.h>  
#include <conio.h>  
int sqr( int x )
```

```
{
    return x*x;
}

void main()
{
    int t = 10;
    printf("%d binh phuong = %d\n", t, sqr(t));
    getch();
}
```

**Ví dụ 2:** Chương trình sau **không thể** hoán vị hai số  $x, y$  mặc dù cú pháp là đúng :

```
#include <stdio.h>
#include <conio.h>
void swap(int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}

void main()
{
    int x, y;
    x = 10; y = 30;
    printf("Trước khi gọi hàm x=%d, y=%d", x, y);
    swap(x, y);
    printf("Sau khi thay đổi x=%d, y=%d", x, y);
    getch();
}
```

Trước khi gọi hàm swap:  $x=10, y=30$

Sau khi gọi hàm swap:  $x=10, y=30$

Hàm swap ở trên không thực hiện được việc hoán vị hai số  $x, y$ .

✗ Truyền theo địa chỉ (theo con trỏ): Tận dụng một đặc tính của các đối số trong các hàm C khi chúng là các con trỏ (con trỏ là một biến chứa địa chỉ của một biến khác),



nó không thay đổi giá trị các con trỏ mà chỉ thay đổi các giá trị mà chúng trỏ đến nếu ta có các thao tác nhằm thay đổi các giá trị đó.

Với cách viết hàm **swap()** như ví dụ 2 thì ta không thể hoán đổi được giá trị của 2 đối tượng truyền vào, do đó phải sửa lại hàm swap() như sau:

```
#include <stdio.h>
#include <conio.h>
void swap( int *a, int *b )
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
void main()
{
    int x = 10, y = 30;
    printf("Trước khi đổi x=%d, y=%d", x, y);
    swap(&x, &y);
    printf("Sau khi đổi x=%d, y=%d", x, y);
    getch();
}
```

Trước khi đổi    x=10, y=30

Sau khi đổi        x=30, y=10

☛\***Chú ý:** Với C++, có thêm một cách viết nữa, đó là khai báo theo địa chỉ.

```
void swap( int &a, int &b )
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

Khi đó, trong lời gọi hàm chỉ gọi **swap( x, y );** là được.

### 3.3. ĐỆ QUY

#### 3.3.1. Khái niệm đệ quy

Một hàm có thể gọi một hàm khác vào làm việc. Nếu như hàm đó *gọi lại chính nó* thì gọi là sự đệ quy.

#### 3.3.2. Phương pháp thiết kế giải thuật đệ quy

- Tham số hóa bài toán
- Tìm trường hợp suy biến.
- Phân tích các trường hợp chung (đưa về các bài toán cùng loại nhưng nhỏ hơn).

Ví dụ 1: Viết hàm đệ quy để tính  $n! = 1.2...n$ .

- Tham số hóa:  $n! = \text{Factorial}(n)$ ;
- $\text{Factorial}(0) = 1$  (trường hợp suy biến)
- $\text{Factorial}(n) = n * \text{Factorial}(n-1)$  (trường hợp chung)

```
int Factorial(int n)
{
    if (n == 0) return 1;
    else return n * Factorial(n-1);
}
```

Ví dụ 2: Viết hàm in ra biểu diễn nhị phân của một số nguyên.

- Tham số hóa:  $\text{bin}(n)$ ;
- $n=0$  kết thúc (trường hợp suy biến)
- $n>0$ : lưu lại **`printf("%d", n%2)`** và thực hiện tiếp **`bin(n/2)`** (trường hợp chung)

```
void bin(int n)
{
    if (n > 0)
    {
        bin(n/2);
        printf("%d", n%2);
    }
}
```

Ví dụ 3: Viết hàm để tìm điểm đối xứng của điểm (x,y) qua gốc tọa độ.

**Ý tưởng:** Vì bài toán này trả về tọa độ điểm đối xứng (xx,yy) gồm 2 giá trị nên ta dùng hàm kiểu void và có thể viết như sau:

```
void doixung(int x, int y, int &xx, int &yy)
{
    xx = -x;
    yy = -y;
}
```

}

## BÀI TẬP

Bài tập 3.1: Viết hàm tìm Max của 2 số thực x,y.

Bài tập 3.2: Viết hàm LOWCASE(c:char):char; để đổi chữ cái hoa c thành chữ thường.

Bài tập 3.3: Viết hàm NGTO(int n) để kiểm tra n có phải là số nguyên tố hay không ?

Bài tập 3.4: Viết hàm XMU(x:Real;n:Byte):Real; để tính giá trị  $x^n$ .

Bài tập 3.5: Viết hàm KHUNG(int x1,int y1,int x2,int y2) để vẽ một khung hình chữ nhật có đỉnh trên bên trái là (x1,y1) và đỉnh dưới bên phải là (x2,y2).

**Ý tưởng:**

Dùng các ký tự mở rộng trong bảng mã ASCII: | (#179), —(#196), ⌈(#218), ⌋(#192), ⌌(#191), ⌑(#217).

Bài tập 3.6: Viết hàm PHANTICH(int n); để phân tích số nguyên n ra thừa số nguyên tố.

Bài tập 3.7: Viết 2 hàm tìm Max , Min của 3 số thực.

Bài tập 3.8: Viết hàm PERFECT(int n) để kiểm tra số nguyên n có phải là số hoàn thiện hay không?

Bài tập 3.9: Viết hàm FILL(int x1,int y1,int x2,int y2, char ch) để tô một vùng màn hình hình chữ nhật có đỉnh trên bên trái là (x1,y1) và đỉnh dưới bên phải là (x2,y2) bằng các ký tự ch.

Bài tập 3.10: Viết các hàm để tìm ước chung lớn nhất và bội chung nhỏ nhất của 2 số nguyên a,b.

Bài tập 3.11: Viết hàm để tối giản phân số a/b , với a, b là 2 số nguyên.

Bài tập 3.12: Viết các hàm đệ quy để tính:

$$S_1 = 1+2+3+\dots+n ;$$

$$S_2 = 1+1/2 + \dots + 1/n ;$$

$$S_3 = 1-1/2 + \dots + (-1)^{n+1} 1/n$$

$$S_4 = 1 + \sin(x) + \sin^2(x) + \dots + \sin^n(x)$$

Bài tập 3.13: Viết hàm đệ quy để tính  $C_n^k$  biết :

$$C_n^n = 1, C_n^0 = 1, C_n^k = C_{n-1}^{k-1} + C_{n-1}^k.$$

Bài tập 3.14: Cho m , n nguyên dương . Lập hàm đệ quy tính:

$$A(m,n) = \begin{cases} n+1 & , m=0 \\ A(m-1,1) & , n=0 \\ A(m-1, A(m,n-1)) & , m>0 \wedge n>0 \end{cases}$$

Bài tập 3.15: Lập hàm đệ quy để tính dãy Fibonacci:

$$F(n) = \begin{cases} 1 & , \quad n = 1 \vee n = 2 \\ F(n-1) + F(n-2) & , \quad n > 2 \end{cases}$$

Bài tập 3.16: Viết hàm đệ qui tìm USCLN của 2 số.

Bài tập 3.17: Viết hàm để in ra màn hình số đảo ngược của một số nguyên cho trước theo 2 cách: đệ qui và không đệ qui.

## Chương 4: MẢNG VÀ CON TRỎ

### 4.1. KHAI BÁO MẢNG

**<Kiểu> <Tên\_mảng> [<Số\_phần\_tử>] [= {  
<Các\_giá\_trị\_khởi\_tạo>}];**

Ví dụ:

```
float a[3];
char b[100];
int x[4]={2,5,-8,6}
char c[] = {'a', 'b', 'c', 'd'};
```

Chú ý

- Chỉ số của mảng được đánh từ 0, tức là nếu ta khai báo một mảng 3 phần tử thì ta sẽ có các phần tử  $a[0]$ ,  $a[1]$  và  $a[2]$ .
- Các phần tử của mảng có địa chỉ liên tiếp nhau trong bộ nhớ. Tên mảng biểu thị địa chỉ phần tử đầu của mảng. Như vậy ta có  $a = \&a[0]$ .

### 4.2. XUẤT NHẬP TRÊN DỮ LIỆU KIỂU MẢNG

- Để truy cập đến phần tử thứ  $k$  trong mảng một chiều  $A$ , ta sử dụng cú pháp:  $A[k]$ .
- Để truy cập đến phần tử  $(i,j)$  trong mảng hai chiều  $M$ , ta sử dụng cú pháp:  $M[i,j]$ .
- Có thể sử dụng các thủ tục `printf()` / `scanf()` đối với các phần tử của biến kiểu mảng.

### 4.3. MẢNG VÀ CON TRỎ

Ví dụ: Ta có các khai báo sau:

```
int a[10];
int *p;
```

Với các khai báo trên, có thể gán:

**$p = a;$  //cho con trỏ  $p$  trỏ đến đầu mảng  $a \Leftrightarrow p = \&a[0]$ .**

và các cách viết sau là tương đương:

```
*p      ⇔ a[0]
*(p + i) ⇔ a[i]
p + i    ⇔ &a[i]
p = p + 4 ⇔ *pa = a[4]
a[i]     ⇔ *(a + i)
```

Các cách viết trên là cách khai thác mảng thông qua con trỏ. Mặt khác, ta có thể khai thác con trỏ thông qua mảng:

`p[i]` thay cho `*(p + i)`

Tức là lúc này *p* đóng vai trò như một mảng, nhưng vẫn có thể thay đổi giá trị trỏ:

`p[1] == a[1]`

**Chú ý :**

- *Khi khai báo mảng*: Thì địa chỉ của mảng được cấp rõ ràng và đủ kích thước, địa chỉ này không thay đổi. Tên mảng được xem như là một con trỏ hằng trỏ đến vị trí đầu mảng
- *Khi khai báo con trỏ*: Thì vị trí được cấp là chỗ của con trỏ ở, còn vị trí con trỏ trỏ đến là tùy ý, và nó chỉ trao đổi địa chỉ, giá trị khi nó đã có địa chỉ.

Ví dụ:

```
int a[10];
```

`a[i] = 0;` hoặc `*(a + i) = 0;` là hoàn toàn hợp lệ

Còn

```
int *p;
```

`p[i] = 0;` hoặc `*(p + i) = 0;` là vô nghĩa khi *p* chưa được gán địa chỉ của một đối tượng cụ thể nào đó.

Mặt khác, **&a** là không hợp lệ, nhưng **&p** là hợp lệ.

## BÀI TẬP

**Bài tập 4.1:** Viết chương trình tìm giá trị lớn nhất của một mảng chứa các số nguyên gồm N phần tử.

**Ý tưởng:**

- Cho số lớn nhất là số đầu tiên:  $\text{Max} = a[1]$ .
- Duyệt qua các phần tử  $a[i]$ , với  $i$  chạy từ 2 tới N:

Nếu  $a[i] > \text{Max}$  thì thay  $\text{Max} = a[i]$ ;

**Bài tập 4.2:** Viết chương trình tính tổng bình phương của các số âm trong một mảng gồm N phần tử.

**Ý tưởng:**

Duyệt qua tất cả các phần tử  $A[i]$  trong mảng: Nếu  $A[i] < 0$  thì cộng dồn  $(A[i])^2$  vào biến S.

**Bài tập 4.3:** Viết chương trình nhập vào một mảng gồm N số nguyên. Sắp xếp lại mảng theo thứ tự tăng dần và in kết quả ra màn hình.

**Ý tưởng:**

Cho biến  $i$  chạy từ 1 đến  $N-1$ , đồng thời cho biến  $j$  chạy từ  $i+1$  đến  $N$ : Nếu  $A[i] > A[j]$  thì đổi chỗ  $A[i]$ ,  $A[j]$ .

**Bài tập 4.4:** Viết chương trình nhập vào một mảng A gồm N số nguyên và nhập thêm vào một số nguyên X. Hãy kiểm tra xem phần tử X có trong mảng A hay không?

**Ý tưởng:**

Dùng thuật toán tìm kiếm tuần tự. So sánh  $x$  với từng phần tử của mảng A. Thuật toán dừng lại khi  $x = A[i]$  hoặc  $i > N$ .

Nếu  $x = A[i]$  thì vị trí cần tìm là  $i$ , ngược lại thì kết quả tìm là 0 (không tìm thấy).

**Bài tập 4.5:** Giả sử mảng A đã được sắp xếp theo thứ tự tăng dần. Viết hàm để kiểm tra xem phần tử X có trong mảng A hay không?

**Ý tưởng:**

So sánh  $x$  với phần tử ở giữa mảng  $A[\text{giua}]$ . Nếu  $x = A[\text{giua}]$  thì dừng (vị trí cần tìm là chỉ số của phần tử giữa của mảng). Ngược lại, nếu  $x > A[\text{giua}]$  thì tìm ở đoạn sau của mảng  $[\text{giua}+1, \text{cuoi}]$ , ngược lại thì tìm ở đoạn đầu của mảng  $[\text{dau}, \text{giua}-1]$ .

Sau đây là hàm cài đặt cho thuật toán này:

**Bài tập 4.6:** Viết chương trình tìm ma trận chuyển vị của ma trận A.

**Ý tưởng:**

Dùng mảng 2 chiều để lưu trữ ma trận. Gọi B là ma trận chuyển vị của ma trận A, ta có:  $B_{ij} = A_{ji}$ .

**Bài tập 4.7:** Cho một mảng 2 chiều A cấp  $m \times n$  gồm các số nguyên và một số nguyên x. Viết chương trình thực hiện các công việc sau:

- a/ Đếm số lần xuất hiện của x trong A và vị trí của chúng.
- b/ Tính tổng các phần tử lớn nhất của mỗi dòng.





**Bài tập 4.12:** Viết chương trình in ra màn hình tam giác Pascal. Ví dụ, với  $n=4$  sẽ in ra hình sau:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

**Ý tưởng:**

Tam giác Pascal được tạo ra theo qui luật sau:

- + Mỗi dòng đều bắt đầu và kết thúc bởi số 1.
- + Phần tử thứ  $j$  ở dòng  $k$  nhận được bằng cách cộng 2 phần tử thứ  $j-1$  và  $j$  ở dòng thứ  $k-1$ .

**Bài tập 4.13:** Viết chương trình nhập vào một dãy số thực và số thực  $x$ . Thông báo lên màn hình số lượng các phần tử trong dãy bằng  $x$  và vị trí của chúng.

**Bài tập 4.14:** Nhập vào một mảng các số nguyên.

- a/ Xếp lại mảng đó theo thứ tự giảm dần.
- b/ Nhập vào một số nguyên từ bàn phím. Chèn số đó vào mảng sao cho mảng vẫn có thứ tự giảm dần. (không được xếp lại mảng)

**Gợi ý:**

- Tìm vị trí cần chèn:  $i$ .
- Đẩy các phần tử từ vị trí  $i$  tới  $n$  sang phải 1 vị trí.
- Gán:  $A[i]=x$ ;

**Bài tập 4.15:** Cho 2 mảng số nguyên: Mảng  $A$  có  $m$  phần tử, mảng  $B$  có  $n$  phần tử.

- a/ Sắp xếp lại các mảng đó theo thứ tự giảm dần.
- b/ Trộn 2 mảng đó lại thành mảng  $C$  sao cho mảng  $C$  vẫn có thứ tự giảm dần (Không được xếp lại mảng  $C$ ).

**Gợi ý:**

- Dùng 2 chỉ số  $i, j$  để duyệt qua các phần tử của 2 mảng  $A, B$  và  $k$  là chỉ số cho mảng  $C$ .
- Trong khi ( $i \leq m$ ) và ( $j \leq n$ ) thì:
  - {Tức là khi đồng thời cả 2 dãy  $A, B$  đều chưa duyệt hết}
  - + Nếu  $A[i] > B[j]$  thì:  $C[k] := A[i]$ ;  $i := i + 1$ ;
  - + Ngược lại:  $C[k] := B[j]$ ;  $j := j + 1$ ;
- Nếu dãy nào hết trước thì đem phần còn lại của dãy kia bổ sung vào cuối dãy  $C$ .

**Bài tập 4.16:** Viết chương trình tính tổng và tích 2 ma trận vuông  $A, B$  cấp  $n$ .

**Gợi ý:**

Công thức tính tổng 2 ma trận:  $C_{ij} = A_{ij} + B_{ij}$

Công thức tính tích 2 ma trận:  $C_{ij} = \sum_{k=1}^n A_{ik} * B_{kj}$

**Bài tập 4.17:** Viết chương trình nhập vào 2 dãy số nguyên  $(a)_n$  và  $(b)_m$ ,  $m \leq n$ . Kiểm tra xem dãy  $\{b\}$  có phải là dãy con của dãy  $\{a\}$  không?

**Bài tập 4.18:** Viết chương trình nhập vào một dãy số nguyên  $a_1, a_2, \dots, a_n$ . Tìm trong dãy  $\{a\}$  một dãy con tăng dần dài nhất (có số phần tử lớn nhất) và in ra màn hình dãy con đó.

**Bài tập 4.19:** Cho mảng 2 chiều A cấp  $m \times n$ . Viết chương trình sắp xếp lại mảng A theo yêu cầu sau:

- a/ Các phần tử trên mỗi dòng được sắp xếp theo thứ tự giảm dần.
- b/ Các dòng được sắp xếp lại theo thứ tự tăng dần của tổng các phần tử trên mỗi dòng.

**Bài tập 4.20:** Viết chương trình để kiểm tra một dãy các số nguyên được nhập vào từ bàn phím đã được sắp theo thứ tự tăng dần hay chưa theo 2 cách: Đệ quy và không đệ quy.

**Gợi ý:**

- Nếu dãy có 1 phần tử thì dãy tăng dần.
- Ngược lại:
  - + Nếu  $A[n-1] > A[n]$  thì dãy không tăng dần.
  - + Ngược lại: Gọi đệ quy với dãy có  $n-1$  phần tử (bỏ bớt đi phần tử cuối cùng).

**Bài tập 4.21:** Viết chương trình nhập vào 2 mảng số nguyên A, B đại diện cho 2 tập hợp (không thể có 2 phần tử trùng nhau trong một tập hợp). Trong quá trình nhập, phải kiểm tra: nếu phần tử vừa nhập vào đã có trong mảng thì không bổ sung vào mảng.

- a/ In ra màn hình hợp của 2 tập hợp A, B.
- b/ In ra màn hình hiệu của 2 tập hợp A, B.

**Gợi ý:**

- a/ - In ra màn hình tất cả các phần tử của tập hợp A.
- Duyệt qua tất cả các phần tử  $b_i \in B$ . Nếu  $b_i \notin A$  thì in  $b_i$  ra màn hình.
- b/ Duyệt qua tất cả các phần tử  $a_i \in A$ . Nếu  $a_i \notin B$  thì in  $a_i$  ra màn hình.

**Bài tập 4.22:** Viết chương trình tính tổng của 2 đa thức  $h(x) = f(x) + g(x)$ . Trong đó, mỗi đa thức có dạng:  $a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ .

**Gợi ý:**

Dùng các mảng A, B, C để lưu trữ các hệ số  $a_i$  của các đa thức  $f(x)$ ,  $g(x)$  và  $h(x)$ .

**Bài tập 4.23:** Viết chương trình tính định thức của ma trận vuông cấp n.

**Gợi ý:**

Dùng cách tính định thức theo phương pháp GAUSE.

## CHƯƠNG 5: XÂU KÝ TỰ

### 5.1. KHÁI NIỆM

- Xâu ký tự là mảng 1 chiều của các ký tự, kết thúc bằng ký tự **NULL** có mã là `'\0'`.
- Hằng chuỗi là dãy ký tự được đặt trong cặp dấu nháy kép `"..."`. Một hằng chuỗi là một mảng ký tự, trình biên dịch sẽ tự động thêm vào cuối một ký tự **NULL** để làm dấu kết thúc chuỗi.

Ví dụ: chuỗi `"abcd"` được xem như mảng một chiều kích thước 5 bytes, byte thứ năm chứa ký tự `'\0'` như sau :

a	b	c	d	\0
---	---	---	---	----

### 5.2. KHAI BÁO VÀ KHỞI GÁN CHUỖI

Một chuỗi được khai báo giống như một mảng kiểu `char`.

Ví dụ: `char s1[255];`

Cũng có thể khởi động một mảng kiểu `char` bằng một hằng chuỗi.

Ví dụ: `char s2[] = "some text";` (1)

Và cũng có thể khởi động một con trỏ `char` bằng một hằng chuỗi.

Ví dụ: `char *s3 = "more text";` (2)

Chú ý hai trường hợp sau:

```
char s[3] = "four"; /* Không hợp lệ */  
char s[4] = "four"; /*Không có ký tự NULL*/
```

Hai cách khai báo chuỗi (1) và (2) thì độ dài của chuỗi không biết được, C sẽ lấy độ dài của hằng chuỗi ở vế phải để định độ dài cho hai biến `s2`, `s3` đó.

### 5.3. PHÉP GÁN CHUỖI

Vì chuỗi được khai báo giống như mảng nên việc gán giá trị cho nó không giống như các biến thông thường mà phải thông qua hàm có tên là **`strcpy()`**.

Cú pháp:

```
char * strcpy(char *s1, const char *s2);
```

Khi hàm kết thúc, chuỗi `s1` sẽ nhận giá trị là giá trị của chuỗi `s2`.

Ở đây phải chú ý một điều hết sức quan trọng là ta phải quan tâm đến độ dài của s2 và s1, nếu độ dài của s2 lớn hơn độ dài của s1 thì đôi lúc sẽ dẫn đến chương trình sai, tốt hơn hết là độ dài của s2 không vượt quá độ dài của s1.

Nhưng việc gán giá trị cho một con trỏ char thì có thể thực hiện được một cách bình thường. Sau đây là một ví dụ:

```
void main()
{
    char a[10];
    char b[10]="Hello"; //Khởi gán chuỗi
    char *ptr1 = "10 spaces";
    char *ptr2;

    a="not OK";//Không hợp lệ, phải viết strcpy(a,"not OK");
    a[5] = 'A';      /* hợp lệ */
    ptr1 [5] = 'B'; /* hợp lệ */
    ptr1 = "OK";     /* hợp lệ */
    ptr1[5] = 'C';   /*không lỗi nhưng sẽ cho kết quả sai*/
    *ptr2 = "not OK"; /*không tương thích, sẽ có cảnh báo*/
    ptr2 = "OK"; // hợp lệ, ptr2 trỏ đến đầu chuỗi "OK"
}
```

## 5.4. CHUỖI VÀ KÝ TỰ

Một điều quan trọng là phải nhận ra sự khác biệt giữa hằng chuỗi và các hằng ký tự. Trong hai khai báo sau đây, 1 byte được cấp phát cho ch nhưng lại cấp phát 2 byte cho chuỗi "a" (một byte dành cho ký tự null – '\0'), và thành một phần bộ nhớ cấp phát cho con trỏ ps.

```
char ch = 'a'; /* 1 byte cho 'a' */
char *ps = "a";
```

Với một con trỏ **char \*p** thì:

```
*p = 'a'; //hợp lệ
*p = "a"; //không hợp lệ
p = "a"; //hợp lệ
p = 'a'; //không hợp lệ vì p là con trỏ, không phải ký tự
```

Ta cũng có thể viết:

```
char *p = "string";
```

nhưng lại không thể viết:

```
*p = "string";
```

## 5.5. NHẬP VÀ XUẤT CHUỖI

### 5.5.1 Nhập chuỗi

Như đã có chú ý trong trường hợp nhập chuỗi với hàm `scanf()` ở Chương 1, ở đây ta nhắc lại, để nhập chuỗi với hàm `scanf()` thì chuỗi nhập vào phải không có ký tự trắng (space). Ở đây ta làm quen với một hàm dùng để nhập chuỗi mà bản thân C đã cung cấp sẵn đó là hàm `gets()`, cú pháp như sau:

```
char * gets ( char *s );
```

Hàm `gets()` không giữ lại ký tự trong bộ đệm bàn phím như `scanf()`.

### 5.5.2 Xuất chuỗi

Với việc xuất chuỗi ra màn hình, ta có thể sử dụng hàm `printf()` với mã định dạng `%s`. Ở đây ta có một hàm chỉ có một nhiệm vụ là xuất một chuỗi lên màn hình, hàm `puts()`, có cú pháp như sau:

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

Cả hai hàm này được khai báo nguyên mẫu trong `<string.h>`.

Ví dụ: Chương trình minh họa cho việc nhập xuất chuỗi.

```
#include <stdio.h>
#include <string.h>
main()
{
    char s[50];
    printf("Nhap mot chuoi: "); gets(s);
    printf("Chuoi vua nhap la: "); puts(s);
    getch();
}
```

## 5.6. SO SÁNH CHUỖI

Trong C không thể so sánh trực tiếp hai chuỗi ký tự. Khi so sánh hai chuỗi ta không thể so sánh trực tiếp dựa trên tên biến, vì lúc đó nó chỉ so sánh giá trị của hai con trỏ hằng là tên hai biến chuỗi. Vì vậy C cung cấp các hàm so sánh chuỗi như sau đây :

```
• int strcmp ( const char *s1, const char *s2 );
```

Hàm trả về các giá trị như sau:

- > 0: nếu s1 lớn hơn s2.
- = 0: nếu s1 bằng s2.
- < 0: nếu s1 bé hơn s2.

• **int strncmp(const char \*s1, const char \*s2, int n);**

Tương tự trên nhưng chỉ so sánh n ký tự đầu.

• **int stricmp ( const char \*s1, const char \*s2 );**

Tương tự strcmp() nhưng khi so sánh thì không phân biệt chữ hoa thường.

• **int strincmp(const char \*s1, const char \*s2, int n);**

Tương tự strncmp() nhưng khi so sánh thì không phân biệt chữ hoa thường.

## 5.7. MỘT SỐ HÀM VỀ CHUỖI

### 5.7.1. Các hàm trong thư viện <string.h>

#### • Hàm strlen()

**size\_t strlen ( const char \* s );**

Trả về độ dài của chuỗi s (không kể ký tự null)

#### • Hàm strcpy()

**char \* strcpy ( char \* s1, const char \* s2 );**

Copy nội dung của chuỗi s2 vào s1. Hàm trả về con trỏ trỏ đến s1.

#### • Hàm strcat()

**char \* strcat ( char \* s1, const char \* s2 );**

Bổ sung nội dung chuỗi s2 vào cuối chuỗi s1. Độ dài của chuỗi kết quả bằng strlen(s1) + strlen(s2). Hàm trả về con trỏ trỏ đến chuỗi kết quả.

#### • Hàm strchr()

**char \* strchr ( char \* s, int c );**

Hàm quét toàn bộ chuỗi s từ trái sang phải để tìm ký tự c xuất hiện lần đầu trong chuỗi. Hàm trả về con trỏ trỏ đến vị trí xuất hiện đầu tiên của c trong s. Nếu không tìm thấy, hàm trả về con trỏ null.

#### • Hàm strstr()

**char \* strstr ( char \* s1, char \* s2 );**

Hàm quét toàn bộ chuỗi s1 để tìm vị trí xuất hiện đầu tiên của s2. Hàm trả về con trỏ trỏ đến vị trí xuất hiện đầu tiên của s2 trong s1. Nếu không tìm thấy, hàm trả về con trỏ null.

- **Hàm `strlwr()`**

**`char *strlwr(char *st)`**

Hàm đổi xâu ký tự sang chữ thường.

- **Hàm `strupr()`**

**`char *strupr(char *st)`**

Hàm đổi xâu ký tự sang chữ in hoa.

### 5.7.2. Các hàm trong thư viện `<stdlib.h>`

- **Hàm `itoa()`, `ltoa()`, `ultoa()`**

**`char *itoa(int value, char *st, int radix)`**

**`char *ltoa(long value, char *st, int radix)`**

**`char *ultoa(unsigned long value, char *st, int radix)`**

Chuyển số nguyên (int, long, unsigned long) **value** thành xâu ký tự **st** trong hệ cơ số **radix**.

- **Hàm `atoi()`, `atof()`, `atol()`**

**`int atoi(char *st)`**

**`double atof(char *st)`**

**`long atol(char *st)`**

Chuyển xâu ký tự thành số (int, float, long).

## 5.8. MỘT SỐ VÍ DỤ VỀ XỬ LÝ CHUỖI

Ví dụ 1: Viết lại hàm `strlen()`

```
int strlen(char *st)
{
    int i = 0;
    while (st[i]!='\0') i++;
    return i;
}
```

Ví dụ 2: Viết lại hàm `strcpy()`

```
char *strcpy(char *s1, char *s2)
{
    int i=0;
    while (s2[i]!='\0')
    {
```

```
        s1[i]=s2[i];
        i++;
    }
    s1[i]='\0';
    return s1;
}
```

**Ví dụ 3:** Viết lại hàm strcat()

```
char *strcat(char *s1, char *s2)
{
    int i=strlen(s1), j=0;
    while(s2[j]!='\0')
    {
        s1[i]=s2[j];
        i++; j++;
    }
    s1[i]='\0';
    return s1;
}
```

**Ví dụ 4:** Viết lại hàm strchr()

```
char *strchr(char *s, char c)
{
    while((s!=NULL) && (*s!=c)) s++;
    return s;
}
```



## BÀI TẬP

Bài tập 5.1: Viết một hàm **int POS(char \*st1, char \*st2)** để trả về vị trí xuất hiện của chuỗi st2 trong chuỗi st1, nếu st2 không có trong st1 thì hàm trả về giá trị -1.

Bài tập 5.2: Viết hàm **char \*Copy(char \*st, int pos, int n)** để trích một chuỗi con có n ký tự của chuỗi st bắt đầu tại vị trí pos.

Bài tập 5.3: Viết hàm **char \*Insert(char \*s, char \*st, int pos)** để chèn chuỗi s vào chuỗi st tại vị trí pos.

Bài tập 5.4: Viết hàm **char \*Delete(char \*st, int pos, int n)** để xóa n ký tự trong chuỗi st bắt đầu từ vị trí pos.

Bài tập 5.5: Viết hàm **char \*Left(char \*st, int n)** để lấy ra từ bên trái của chuỗi st n ký tự.

Tương tự viết hàm **char \*Right(char \*st, int n)** cũng để lấy ra từ bên phải của chuỗi st n ký tự.

Bài tập 5.6: Viết hàm **char \*str(int n)** để đổi số nguyên n thành chuỗi.

Bài tập 5.7: Viết hàm **int value(char \*s, int k)** để đổi chuỗi số nguyên s thành một số nguyên.

Nếu việc chuyển đổi thành công thì k nhận giá trị 0, ngược lại k nhận giá trị -1.

Bài tập 5.8: Viết chương trình nhập vào một chuỗi ký tự từ bàn phím. Đổi chuỗi ký tự đó sang chữ in hoa rồi in kết quả ra màn hình.

Ví dụ :Chuỗi abcdAbcD sẽ cho ra chuỗi ABCDABCD.

Bài tập 5.9: Viết hàm **char \*lower(char \*st)** để đổi chuỗi ký tự st sang chữ thường.

Ví dụ :Chuỗi abCdAbcD sẽ cho ra chuỗi abcdabcd.

Bài tập 5.10: Viết chương trình đếm số ký tự chữ số trong một chuỗi ký tự được nhập vào từ bàn phím.

Bài tập 5.11: Viết chương trình nhập một chuỗi từ bàn phím. In ra chuỗi đó sau khi xóa hết các ký tự trắng thừa trong chuỗi. (Ký tự trắng thừa là các ký tự trắng đầu chuỗi, cuối chuỗi và nếu ở giữa chuỗi có 2 ký tự trắng liên tiếp nhau thì có 1 ký tự trắng thừa).

Bài tập 5.12: Viết chương trình liệt kê các từ của một chuỗi ký tự được nhập vào từ bàn phím, mỗi từ phải được viết trên một dòng.

Bài tập 5.13: Viết chương trình nhập vào một chuỗi ký tự từ bàn phím. Tìm chuỗi đảo ngược của chuỗi đó rồi in kết quả ra màn hình theo 2 cách: Đệ quy và không đệ quy.

**Ý tưởng:**

- Nếu chuỗi St có 1 ký tự thì chuỗi đảo = St.
- Ngược lại: Chuỗi đảo = Ký tự cuối + Đệ quy(Phần còn lại của chuỗi St).

Bài tập 5.14: Viết chương trình nhập vào một chuỗi ký tự từ bàn phím. Thông báo lên màn hình các chữ cái có trong chuỗi và số lượng của chúng (Không phân biệt chữ hoa hay chữ thường).

**Ý tưởng:**

- Dùng một mảng dem với chỉ số là các chữ cái để lưu trữ số lượng của các chữ cái trong chuỗi.
- Duyệt qua tất cả các ký tự của chuỗi St: Nếu ký tự đó là chữ cái thì tăng ô biến mảng dem[St[i]] lên 1 đơn vị.

Bài tập 5.15: Viết chương trình xóa các ký tự chữ số trong một chuỗi ký tự được nhập vào từ bàn phím.

Bài tập 5.16: Viết chương trình để mã hóa và giải mã một chuỗi ký tự bằng cách đảo ngược các bit của từng ký tự trong chuỗi.

Bài tập 5.17: Viết chương trình thực hiện phép cộng 2 số tự nhiên lớn (không quá 255 chữ số).

Bài tập 5.18: Viết chương trình nhập vào một chuỗi ký tự từ bàn phím. Tìm và in ra màn hình một từ có độ dài lớn nhất trong chuỗi.

**Gợi ý:**

Tách từng từ để so sánh (xem bài tập 5.12).

Bài tập 5.19: Viết chương trình nhập một chuỗi ký tự St từ bàn phím và một ký tự ch. In ra màn hình chuỗi St sau khi xóa hết các ký tự ch trong chuỗi đó.

Bài tập 5.20: Viết chương trình nhập một chuỗi vào từ bàn phím và thông báo lên màn hình chuỗi đó có phải đối xứng không theo 2 cách: Đệ quy và không đệ quy. (Ví dụ: abba, abcba là các chuỗi đối xứng).

**Gợi ý:**

- Nếu  $\text{strlen}(st) \leq 1$  thì st là chuỗi đối xứng
- Ngược lại:
  - + Nếu  $st[0] \neq st[\text{strlen}(st)-1]$  thì st không đối xứng
  - + Ngược lại: Gọi đệ quy với chuỗi st sau khi bỏ đi ký tự đầu và ký tự cuối.

Bài tập 5.21: Viết chương trình đảo ngược thứ tự các từ trong một chuỗi được nhập vào từ bàn phím.

Ví dụ: Chuỗi *Nguyen Van An* sẽ thành *An Van Nguyen*.

**Gợi ý:**

Tách từng từ nối vào đầu chuỗi mới (xem bài tập 5.12).

Bài tập 5.22: Viết chương trình nhập vào 2 chuỗi ký tự s1 và s2. Kiểm tra xem chuỗi s2 xuất hiện bao nhiêu lần trong chuỗi s1. (Lưu ý:  $\text{strlen}(s2) \leq \text{strlen}(s1)$ ).

**Gợi ý:**

Dùng hàm POS để kiểm tra và thủ tục DELETE để xóa bớt sau mỗi lần kiểm tra.

Bài tập 5.23: Viết chương trình nhập vào một dòng văn bản, hiệu chỉnh văn bản theo những yêu cầu sau đây và in văn bản sau khi hiệu chỉnh ra màn hình:

- a. Xóa tất cả các ký tự trắng thừa.
- b. Trước các dấu câu không có các ký tự trắng, sau các dấu câu có một ký tự trắng.
- c. Đầu câu in hoa.

Bài tập 5.24: Viết chương trình thực hiện phép nhân 2 số nguyên lớn.

**Gợi ý:**

- Viết hàm để nhân một số lớn với số có 1 chữ số.
- Áp dụng hàm tính tổng 2 số lớn (xem bài tập 5.17).

Bài tập 5.25: Viết chương trình để nén và giải nén một chuỗi ký tự .

Ví dụ: Chuỗi 'AAAABBBBCDDDDDDDEEF' sau khi nén sẽ trở thành '4A3BC7D2EF'.

Bài tập 5.26: Viết chương trình nhập vào họ tên đầy đủ của các học viên một lớp học (không quá 50 người). Hãy sắp xếp lại họ tên của các học viên đó theo thứ tự Alphabet (Nếu tên trùng nhau thì xếp thứ tự theo họ lót, nếu họ lót cũng trùng nhau thì xếp thứ tự theo họ). In ra màn hình danh sách của lớp học sau khi đã sắp xếp theo thứ tự Alphabet.

**Gợi ý:**

- Dùng mảng chuỗi ký tự để lưu trữ họ tên học viên.
- Đảo ngược các từ của họ tên trước khi sắp xếp.

## CHƯƠNG 6: KIỂU CẤU TRÚC VÀ HỢP NHẤT

### 6.1. KIỂU CẤU TRÚC

Cấu trúc (structure) là một tập hợp các biến, mảng hoặc con trỏ có liên quan với nhau. Nói cách khác, cấu trúc giống như mảng (giống về cả cách lưu trữ trong bộ nhớ), nhưng mỗi phần tử của nó có thể nhận các kiểu dữ liệu khác nhau.

#### 6.1.1. Định nghĩa cấu trúc

```
struct [<Tên_cấu_trúc>]
{
    <type1> <field_1>;
    <type2> <field_2>;
    . . .
};
```

Trường hợp định nghĩa không có <Tên\_cấu\_trúc> thì cấu trúc gọi là cấu trúc ẩn danh.

#### 6.1.2. Định nghĩa cấu trúc với typedef

Nếu một cấu trúc đã được định nghĩa với <Tên\_cấu\_trúc>, ta có thể định nghĩa:

```
typedef struct <Tên_cấu_trúc> <Tên_kiểu>;
```

Nếu một cấu trúc chưa định nghĩa, ta cũng có thể dùng typedef như sau:

```
typedef struct [<Tên_cấu_trúc>]
{
    <kiểu1> <field_1>;
    <kiểu2> <field_2>;
    . . .
} <Tên_kiểu>;
```

Ví dụ:

```
struct NHANVIEN
{
    int maso;
    char hoten[40];
    float lcb;
    char dv[20]
    float pc;
};
```

hoặc

```
struct nhanvien
```

```

{
    int maso;
    char hoten[40];
    float lcb;
    char dv[20]
    float pc;
};
typedef struct nhanvien NHANVIEN;

```

hoặc

```

typedef struct
{
    int maso;
    char hoten[40];
    float lcb;
    char dv[20]
    float pc;
}NHANVIEN;

```

### 6.1.3. Khai báo biến cấu trúc

Với nhiều cách định nghĩa cấu trúc thì cũng có nhiều cách khai báo biến cấu trúc:

**Khai báo kết hợp: Là khai báo ngay trong khi định nghĩa cấu trúc.**

```

struct [<Tên_cấu_trúc>]
{
    <kiểu_1> <field_1>;
    <kiểu_2> <field_2>;
    . . .
} <danh_sách_các_biến>;

```

Ví dụ:

```

struct nhanvien
{
    int maso;
    char hoten[40];
    float lcb;
    char dv[20]
    float pc;
}nv, *pnv, nva[10];

```

**Khai báo riêng lẻ: Dùng nhãn cấu trúc hoặc thông qua tên cấu trúc được định nghĩa bằng typedef.**

- Dùng tên cấu trúc:

```

<Tên_cấu_trúc> <danh_sách_các_biến>;

```

Ví dụ:

```

nhanvien nv, *pnv, nva[10];

```

- Dùng tên đã được định nghĩa bằng typedef:

```
<Tên_kiểu> <danh_sách_các_biến>;
```

Ví dụ:

```
NHANVIEN nv, *pnv, nva[10];
```

#### 6.1.4. Khởi động các biến cấu trúc

Ta có thể khởi động một cấu trúc theo phương cách như là khởi động mảng: Theo sau tên biến cấu trúc là dấu bằng (=), sau đó là danh sách các giá trị khởi động được đặt trong các dấu móc {}. Các giá trị khởi động có cùng kiểu với các trường tương ứng trong cấu trúc.

Ví dụ 1:

```
NHANVIEN nv = {245, "Hoàng Văn Kháng",  
               120000.0, "Khoa CNTT", 20000.0};
```

Ví dụ 2: Khởi động hai biến cấu trúc và in dữ liệu ra màn hình

```
#include <stdio.h>  
  
struct person  
{  
    int agent;  
    char name[30];  
};  
  
typedef struct person PS;  
  
PS ps1 = {102, "Trần Văn An"};  
PS ps2 = {103, "Lê Vũ Cầu"};  
  
void main()  
{  
    printf("\nDanh sách nhân viên:\n");  
    printf("Tên: %s, Số thứ tự: %03d", ps1.name, ps1.agent);  
    printf("Tên: %s, Số thứ tự: %03d", ps2.name, ps2.agent);  
}
```

#### 6.1.5. Truy cập vào các thành phần của cấu trúc

Có hai cách để tham chiếu đến các thành phần của cấu trúc tương ứng với hai trường hợp sau:

- ☐ **Nếu nó là biến cấu trúc:** dùng toán tử dấu chấm (.) để tham chiếu đến các trường (thành phần) của cấu trúc:

```
<Tên_Cấu_Trúc>.<Tên_Trường>
```

- **Nếu nó là biến con trỏ trỏ đến cấu trúc:** dùng toán tử mũi tên (->) để tham chiếu đến các trường:

`<Tên_Con_Trỏ_Cấu_Trúc> -> <Tên_Trường>`

Ví dụ: Ta có các khai báo sau:

```
struct Date
{
    int day;
    int month;
    int year;
}date;

typedef struct Date DATE;
DATE *ps;
```

Các cách tham chiếu sau là hợp lệ:

```
date.day = 31;
date.month = 5;
date.year = 1997
```

hoặc:

```
ps->day = 31;
ps->month = 5;
ps->year = 1997;
```

các phép toán con trỏ trên tương đương với

```
(*ps).day = 31;
(*ps).month = 5;
(*ps).year = 1997;
```

**Gán hai biến cấu trúc**

```
struct Date d = {31,5,1997};
struct Date today;
today = d;
```

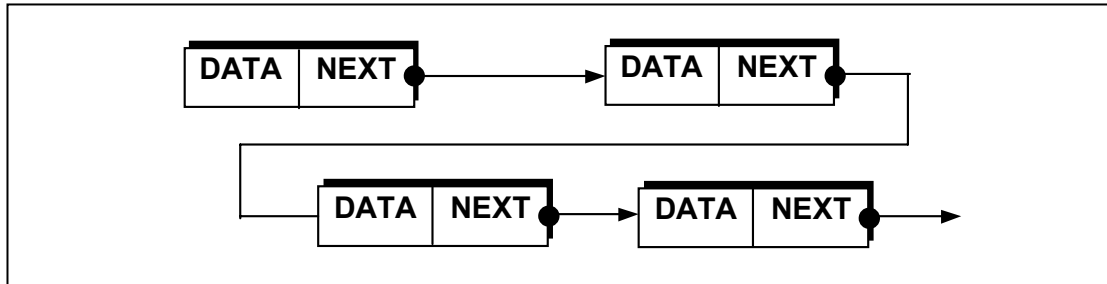
## 6.2. DANH SÁCH LIÊN KẾT

Trong các chương trước, khi sử dụng mảng để lưu trữ dữ liệu sẽ trở nên khá tốn kém về không gian nhớ vì chúng phải được cấp phát đủ bộ nhớ để hoạt động. Bộ nhớ này được đăng ký và sẽ không dành cho những tác vụ khác ngay cả khi ta chỉ dùng một số nhỏ các phần tử mảng.

Phương hướng giải quyết cho vấn đề này là cho phép cấp phát bộ nhớ mới mỗi khi cần thiết. C cho phép ta thực hiện điều này thông qua cách cấp phát vùng nhớ động bằng các hàm ***malloc()*** và ***calloc()*** cũng như giải phóng vùng nhớ động bằng hàm ***free()***. Các vùng nhớ được cấp phát sẽ không đảm bảo rằng chúng được đặt liên tiếp nhau trong bộ nhớ. Do đó điều cần thiết là kỹ thuật để nối kết tất cả các vùng nhớ đó lại với nhau.

Giải pháp thông dụng nhất để thực hiện điều này là sử dụng một kiến trúc gọi là **danh sách liên kết (linked list)**. Một danh sách liên kết là một dãy các vùng nhớ được liên kết với nhau giống như một đoàn tàu. Cách liên kết đơn giản nhất là trong mỗi vùng nhớ có một trường liên kết (**con trỏ**) trỏ đến địa chỉ của vùng nhớ tiếp theo trong danh sách.

Một cách hình thức, một danh sách liên kết có dạng như sau:



Trong các ứng dụng của danh sách liên kết, cần phải thực hiện được các thao tác cơ bản sau:

- ☐ Tạo một phần tử của danh sách
- ☐ Bổ sung các phần tử vào danh sách
- ☐ Xóa một phần tử khỏi danh sách
- ☐ Tìm kiếm một phần tử trong danh sách
- ☐ ...

### 6.2.1. Khai báo danh sách liên kết

Để định nghĩa một danh sách liên kết trước hết ta định nghĩa kiểu của mỗi **nút** trong danh sách.

```

struct <NUT>
{
    <Kiểu_dữ_liệu> <Data>; // chứa dữ liệu
    NUT *next;             // con trỏ trỏ đến nút tiếp theo
};
typedef NUT* <Trỏ_Nút>;
  
```

Sau đó ta có thể khai báo một danh sách liên kết như sau:

```
<Trỏ_Nút> First;
```

**First** là địa chỉ của nút đầu tiên trong danh sách, dựa vào trường **next** của nút này để biết được địa chỉ của các nút tiếp theo trong danh sách. Danh sách dạng này được gọi là **danh sách liên kết (móc nối) đơn**.

Ví dụ: Tạo một danh sách liên kết chứa các số nguyên:

```

struct NUT
{
    int x;                // chứa số nguyên
  
```



```

    NUT *next;
};

typedef NUT* TroDS;
TroDS First; //Danh sách được trỏ bởi con trỏ đầu First

```

## 6.2.2. Các thao tác thường gặp trên danh sách liên kết

### 6.2.2.1. Khởi tạo danh sách

```
First=NULL;
```

### 6.2.2.2. Bổ sung một nút vào đầu danh sách

//1. Tạo ra nút mới

```

<Trỏ_Nút> p;
p=(Trỏ_Nút)malloc(sizeof(Trỏ_Nút));
p->Data = <Giá trị>;

```

//2. Bổ sung vào đầu danh sách

```

p->next=First;
First=p;

```

### 6.2.2.3. Duyệt danh sách

Duyệt qua và xử lý từng nút trong danh sách.

```

void DuyệtDS (Trỏ_Nút First)
{
    Trỏ_Nút p;
    p=First;
    while (p!=NULL)
    {
        <Xử lý p->x>;
        p=p->next;
    }
}

```

### 6.2.2.4. Bổ sung một nút vào sau nút được trỏ bởi p

Hàm sau thực hiện việc bổ sung một nút mới có nội dung x vào sau nút được trỏ bởi p.

```

void Bosung(int x, Trỏ_Nút p, Trỏ_Nút &First)
{
    //Tạo nút mới q
    Trỏ_Nút q;
    q=(Trỏ_Nút)malloc(sizeof(Trỏ_Nút));
    q->x=x;

    //Bổ sung

```

```

if (First==NULL)    //Danh sach rong
{
    q->next=NULL;
    First=q;
}
else
{
    //Bổ sung nút q vào sau nút p
    q->next=p->next;
    p->next=q;
}
}

```

#### 6.2.2.5. Xóa một nút khỏi danh sách

Thủ tục sau thực hiện việc xóa một nút trở bởi p ra khỏi danh sách.

```

void Xoa (Trở_Nút p, Trở_Nút &First)
{
    if (p==First)    //Xóa nút đầu
        First=First->next;
    else
    {
        //Tìm đến nút q: nút đứng trước nút p
        Trở_Nút q=First;
        while (q->next!=p) q=q->next;

        //liên kết nút q với nút đứng sau p
        q->next=p->next;
    }
    //Xóa nút p
    free(p);
}

```

Ví dụ: Viết chương trình thực hiện các công việc sau:

- Tạo một danh sách liên kết chứa các số nguyên.
- Nhập các số nguyên vào cho danh sách cho đến khi nào nhấn ESC thì kết thúc.
- In ra màn hình tất cả các phần tử của danh sách.
- Đếm xem danh sách có bao nhiêu phần tử.

```

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

```

```
struct DANHSACH
{
    int x;
    DANHSACH *next;
};
typedef DANHSACH* TroDS;

void TaoDS(TroDS &First)
{
    TroDS p;
    char ch;
    First=NULL;
    do
    {
        //Tao nut moi
        p=(TroDS)malloc(sizeof(TroDS));
        cout<<"\nNhap x = "; cin>>p->x;

        //Bo sung nut moi vao dau danh sach
        p->next=First;
        First=p;

        cout<<"\nNhan <ESC> de ket thuc nhap!";
        ch=getch();
    }
    while (ch!=27);
}

void XemDS(TroDS First)
{
    TroDS p;
    cout<<"\nDanh sach: \n";
    p=First;
    while(p!=NULL)
    {
        cout<<" "<<p->x;
        p=p->next;
    }
    getch();
}

int Dem(TroDS First)
{
    int d=0;
    TroDS p=First;
    while(p!=NULL)
```

```

    {
        d++;
        p=p->next;
    }
    return d;
}

void main()
{
    clrscr();
    TroDS Dau;
    TaoDS(Dau);
    XemDS(Dau);
    cout<<"\nSo phan tu cua danh sach:"<<Dem(Dau);
    getch();
}

```

### 6.3. KIỂU UNION

Hợp nhất (union) là tương tự với cấu trúc chỉ khác một điều là các thành phần của nó cùng chia sẻ nhau một vùng bộ nhớ.

Cú pháp khai báo:

```

union [<nhãn>]
{
    <Type1> <Field_1>;
    <Type2> <Field_2>;
    . . .
};

```

Các cách khai báo khác hoàn toàn giống với cấu trúc.

Ví dụ:

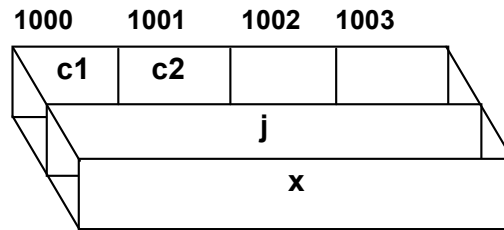
```

typedef union
{
    struct
    {
        char c1, c2;
    } s;
    long j;
    float x;
} UNI;

UNI example;

```

Lúc này việc lưu trữ bộ nhớ của *example* là:



Trình biên dịch luôn cấp phát đủ bộ nhớ để lưu giữ giá trị của thành phần lớn nhất, và tất cả các thành phần đều bắt đầu tại cùng một địa chỉ. Dữ liệu lưu giữ trong hợp nhất phụ thuộc vào thành phần đang sử dụng. Chẳng hạn, phép gán:

```
example.s.c1 = 'a';
example.s.c2 = 'b';
```

sẽ được lưu trữ như sau

1000	1001	1002	1003
'a'	'b'		

Nhưng nếu ta lại thực hiện thêm phép gán

```
example.j = 5;
```

thì nó sẽ đề lên hai ký tự này, và sử dụng hết 4 byte để lưu giá trị nguyên 5.

Với hợp nhất, có 2 ứng dụng cơ bản sau:

- ☐ Diễn dịch trên cùng một vùng bộ nhớ theo nhiều cách khác nhau
- ☐ Tạo ra các cấu trúc mềm dẻo (gọi là *các record thay đổi - variant records* trong Pascal) có thể lưu giữ các kiểu dữ liệu khác nhau.

## BÀI TẬP

**Bài tập 6.1:** Viết chương trình thực hiện phép cộng 2 số phức.

**Bài tập 6.2:** Viết chương trình nhân hai số phức c1, c2.

**Bài tập 6.3:** Viết chương trình quản lý điểm thi Tốt nghiệp của sinh viên với 2 môn thi: Cơ sở và Chuyên ngành. Nội dung công việc quản lý bao gồm:

- Nhập điểm cho từng sinh viên.
- In danh sách sinh viên ra màn hình.
- Thống kê số lượng sinh viên thi đậu.
- In ra màn hình hình danh sách những sinh viên bị thi lại.

**Bài tập 6.4:** Viết chương trình nhập vào n đỉnh của một đa giác lồi S.

a/ Tính diện tích của S biết:

$$dt(S) = \frac{1}{2} \left| \sum_{i=1}^n (x_i y_{i+1} - x_{i+1} y_i) \right|$$

trong đó:  $(x_i, y_i)$  là tọa độ đỉnh thứ i của đa giác S.

b/ Nhập vào thêm một điểm  $P(x,y)$ . Hãy kiểm tra xem  $P$  nằm trong hay ngoài đa giác  $S$ .

**Ý tưởng:**

Nối  $P$  với các đỉnh của đa giác  $S$  thì ta được  $n$  tam giác:  $S_i = PP_iP_{i+1}$ , với  $P_{n+1}=P_1$ .

Nếu  $\sum_{i=1}^n dt(S_i) = dt(S)$  thì  $P \in S$ .

**Bài tập 6.5:** Viết chương trình quản lý điểm thi của sinh viên (sử dụng danh sách liên kết) bao gồm các trường sau: Họ tên, Điểm Tin, Điểm ngoại ngữ, Điểm trung bình, Xếp loại. Thực hiện các công việc sau:

a/ Nhập vào danh sách sinh viên của một lớp (không quá 30 người), bao gồm: Họ tên, Điểm Tin, Điểm Ngoại ngữ. Tính Điểm trung bình và Xếp loại cho từng sinh viên.

b/ In ra màn hình danh sách sinh viên của lớp đó theo dạng sau:

Họ tên	Điểm Tin	Điểm Ngoại ngữ	Điểm T.Bình	Xếp loại
Trần Văn An	8	9	8.5	Giỏi
Lê Thị Béo	7	5	6.0	T.Bình
.....	.....	.....	.....	.....

c/ In ra màn hình danh sách những sinh viên phải thi lại (nợ một trong hai môn).

d/ In ra danh sách những sinh viên xếp loại Giỏi.

e/ Tìm và in ra màn hình những sinh viên có điểm trung bình cao nhất lớp.

f/ Sắp xếp lại danh sách sinh viên theo thứ tự Alphabet.

g/ Sắp xếp lại danh sách sinh viên theo thứ tự giảm dần của điểm trung bình.

h/ Viết chức năng tra cứu theo tên không đầy đủ của sinh viên. Ví dụ: Khi nhập vào tên **Phuong** thì chương trình sẽ tìm và in ra màn hình thông tin đầy đủ của những sinh viên có tên **Phuong** (chẳng hạn như: **Pham Anh Phuong, Do Ngoc Phuong, Nguyen Nam Phuong...**).

**Bài tập 6.6:** Viết chương trình quản lý sách ở thư viện gồm các trường sau: Mã số sách, Nhan đề, Tên Tác giả, Nhà Xuất bản, Năm xuất bản.

a/ Nhập vào kho sách của thư viện (gồm tất cả các trường).

b/ In ra màn hình tất cả các cuốn sách có trong thư viện.

c/ Tìm một cuốn sách có mã số được nhập vào từ bàn phím. Nếu tìm thấy thì in ra màn hình thông tin đầy đủ của cuốn sách đó, ngược lại thì thông báo không tìm thấy.

c/ Tìm và in ra màn hình tất cả các cuốn sách có cùng tác giả được nhập vào từ bàn phím.

d/ Lọc ra các cuốn sách được xuất bản trong cùng một năm nào đó.

e/ Tìm và in ra màn hình các cuốn sách mà nhan đề có chứa từ bất kỳ được nhập vào từ bàn phím.

Bài tập 6.7: Viết một hàm để xác định xem một danh sách liên kết đã cho có thứ tự tăng dần hay không theo 2 cách: Không đệ quy và đệ quy.

Bài tập 6.8: Cho 2 danh sách liên kết đơn đại diện cho 2 tập hợp được trữ bởi L1 và L2. Viết chương trình để hiển thị:

1. Phần giao của 2 danh sách trên.
2. Phần hợp của 2 danh sách trên.
3. Phần hiệu của 2 danh sách trên.

Bài tập 6.9: Cho 2 danh sách liên kết L1 và L2.

1. Sắp xếp lại 2 danh sách đó theo thứ tự tăng dần.
2. Trộn 2 danh sách đó lại thành danh sách L3 sao cho L3 vẫn có thứ tự tăng dần.

Bài tập 6.10: Dùng danh sách móc nối để biểu diễn một đa thức  $P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0$ . Trong đó, mỗi số hạng của đa thức được xác định bởi 2 thành phần: hệ số  $a_i$  và số mũ  $i$ .

Như vậy, ta có thể xây dựng danh sách liên kết để lưu trữ đa thức với các trường: HeSo(float), SoMu(int) và Next(con trỏ đến nút tiếp theo). Viết chương trình thực hiện các công việc sau:

1. Viết hàm nhập vào một đa thức.
2. Viết hàm để sắp xếp lại các số hạng của đa thức theo thứ tự số mũ giảm dần.
3. Viết hàm để cộng 2 đa thức.
4. Viết hàm để tính giá trị của đa thức theo giá trị X.

Bài tập 6.11: Cho dãy số nguyên sắp theo thứ tự tăng dần và lưu trong một danh sách liên kết đơn có địa chỉ nút đầu danh sách là First.

1. Viết hàm để xóa tất cả các nút trong danh sách có giá trị 0.
2. Viết hàm để in ra các giá trị phân biệt của danh sách (không có các phần tử trùng nhau được in ra).

Bài tập 6.12: Cho hai dãy số thực lưu trong hai danh sách liên kết đơn, có địa chỉ của các nút đầu danh sách lần lượt là First1 và First2. Giả sử trong mỗi danh sách giá trị các nút đã được sắp tăng dần. Hãy viết chương trình tạo một danh sách liên kết đơn có nút đầu trỏ bởi List chứa tất cả các phần tử của hai danh sách trên sao cho danh sách mới này cũng được sắp thứ tự tăng dần.

## CHƯƠNG 7: KIỂU TẬP TIN

### 7.1. KHAI BÁO

Để khai báo một biến file, ta dùng cú pháp sau:

**FILE \* <biến file>;**

Ví dụ:

```
#include <stdio.h>
FILE *file_var;
```

### 7.2. MỞ FILE

Trước khi đọc hay ghi dữ liệu lên một file thì cần phải mở file bằng hàm *fopen()*.

Khai báo nguyên mẫu:

**FILE \*fopen(const char \*fname, const char \*mode);**

Trong đó:

**fname:** tên file cần mở.

**mode:** các chế độ truy xuất file

- “**r**”      Mở một file văn bản đã có sẵn để đọc. Việc đọc bắt đầu tại vị trí đầu của file.
- “**w**”      Tạo mới một file văn bản để ghi. Nếu file có trên đĩa thì nó sẽ ghi đè, tức là chỉ còn file rỗng. Bộ chỉ định vị trí sẽ khởi động ở đầu file.
- “**a**”      Mở một file văn bản đã có sẵn để bổ sung. Ta chỉ có thể ghi dữ liệu chỉ ở vị trí cuối file. Ngay cả khi ta di chuyển vị trí của bộ định vị thì việc ghi cũng xuất hiện ở cuối file.
- “**r+**”      Mở một file văn bản đã có sẵn để đọc và ghi dữ liệu. Bộ định vị trí sẽ được khởi động đến đầu file.
- “**w+**”      Tạo mới một file văn bản để đọc và ghi dữ liệu. Nếu file đã có thì sẽ bị ghi đè.
- “**a+**”      Mở một file đã tồn tại hoặc tạo mới một file ở chế độ bổ sung. Ta có thể đọc dữ liệu ở bất kỳ đâu trên file nhưng chỉ có thể ghi dữ liệu ở cuối file.

❖ **Chú ý:** Đối với các file nhị phân cũng hoàn toàn tương tự, chỉ thêm vào cuối chuỗi *mode* một ký tự **b**. Ví dụ để mở một file nhị phân để đọc thì ta dùng như sau “**rb**”, còn với văn bản ta cũng có thể thêm vào ký tự **t** cuối, chẳng hạn “**rt**”.

Tóm tắt các thuộc tính chế độ của *fopen()*:



	<b>r</b>	<b>w</b>	<b>a</b>	<b>r+</b>	<b>w+</b>	<b>a+</b>
File phải tồn tại trước khi mở	✓			✓		
File cũ sẽ bị xoá hết nội dung		✓			✓	
Stream có thể đọc	✓			✓	✓	✓
Stream có thể viết		✓	✓	✓	✓	✓
Stream chỉ có thể viết ở cuối			✓			✓

Hàm *fopen()* trả về một con trỏ file nếu việc mở thành công, ngược lại sẽ trả về con trỏ NULL.

### 7.3. ĐÓNG FILE

Để đóng một file, ta dùng hàm *fclose()*:

```
int *fclose( FILE *fp );
```

Hàm trả về 0 nếu thành công, EOF (-1) nếu không thành công. *fp* là con trỏ file được trả về thông qua hàm *fopen()* trước đó. Việc đóng file sẽ giải phóng cấu trúc *FILE* mà con trỏ *fp* đang chỉ đến, vì thế hệ điều hành sẽ dùng cấu trúc này cho file khác và nó cũng xoá sạch các bộ đệm liên quan đến nó.

### 7.4. ĐỌC VÀ GHI DỮ LIỆU

#### 7.4.1 Đọc/ghi từng ký tự

**fgetc(int ch, FILE \*fp)** Hàm đọc một ký tự từ file fp.

**fputc(int ch, FILE \*fp)** Hàm ghi một ký tự ra file fp.

Ví dụ: Sử dụng *getc()* và *putc()* để sao chép file

```
#include <stdio.h>
#include <stddef.h>
#define FAIL 0
#define SUCCESS 1

int copyfile(char *infile, char *outfile)
{
    FILE *fp1, *fp2;

    if ((fp1=fopen(infile, "rb"))==NULL)
        return FAIL;
    if ((fp2=fopen(outfile, "wb"))==NULL)
    {
        fclose(fp1);
        return FAIL;
    }

    while (!feof(fp1))
```

```
fputc(fgetc(fp1), fp2);

fclose(fp1);
fclose(fp2);
return SUCCESS;
}
```

Hàm trên mở cả hai file theo chế độ nhị phân bởi vì ta không quan tâm đến cấu trúc của file. Vì vậy hàm này thực hiện được trên tất cả các loại file.

#### 7.4.2. Đọc ghi từng dòng

```
char *fgets (char *s, int n, FILE *fp);
char *fputs (char *s, FILE *fp);
```

Ở đây:

*s*: con trỏ trỏ đến phần tử đầu của chuỗi

*n*: Số nguyên đại diện cho số lớn nhất các ký tự được đọc

*fgets()* đọc các ký tự cho đến khi gặp một ký tự newline, một ký tự end-of-file, hoặc đã đến số ký tự lớn nhất chỉ ra. *fgets()* tự động thêm một ký tự null vào sau ký tự cuối cùng của mảng. Hàm *fputs()* ghi một mảng chỉ bởi đối số đầu tiên đến một stream ở đối số thứ hai.

Ví dụ: Viết lại hàm *copyfile()*

```
#include <stdio.h>
#include <stddef.h>
#define FAIL 0
#define SUCCESS 1
#define LINESIZE 100

int copyfile(char *infile, char *outfile)
{
    FILE *fp1, *fp2;
    char line[LINESIZE];

    if ((fp1=fopen(infile, "r"))==NULL)
        return FAIL;
    if ((fp2=fopen(outfile, "w"))==NULL)
    {
        fclose(fp1);
        return FAIL;
    }

    while (fgets(line, LINESIZE-1, fp1)!= NULL)
        fputs(line, fp2);
    fclose(fp1);
    fclose(fp2);
}
```

```
    return SUCCESS;
}
```

Chú ý rằng ở đây ta mở file theo dạng văn bản vì ta muốn truy xuất dữ liệu theo từng dòng một. Nếu mở file dạng nhị phân thì hàm trên có thể không làm việc đúng.

### 7.4.3 Đọc ghi từng block

Khi đọc hay ghi một block, cần phải chỉ rõ số lượng các block và kích cỡ của mỗi block. Hai hàm nhập xuất với block là *fread()* và *fwrite()*, có khai báo nguyên mẫu như sau:

```
size_t fread (void *pt, size_t size, size_t n, FILE *fp);  
size_t fwrite(void *pt, size_t size, size_t n, FILE *fp);
```

Ở đây *size\_t* là một kiểu số nguyên định nghĩa trong *<stdio.h>*.

<i>pt</i>	Con trỏ trỏ đến mảng lưu dữ liệu
<i>size</i>	Kích cỡ của mỗi phần tử trong block
<i>n</i>	Số phần tử sẽ đọc / ghi
<i>fp</i>	Con trỏ file

Hàm *fread()* trả về số phần tử đọc được.

Hàm *fwrite()* trả về số phần tử ghi được vào file.

Ví dụ: Viết lại hàm *copyfile()* dùng *fread()* và *fwrite()*.

```
#include <stdio.h>
#include <stddef.h>
#define FAIL      0
#define SUCCESS   1
#define BLOCKSIZE 512

typedef char DATA;
int copyfile(char *infile, char *outfile)
{
    FILE *fp1, *fp2;
    DATA block[BLOCKSIZE];
    int num_read;

    if ((fp1=fopen(infile, "rb"))==NULL)
        return FAIL;
    if ((fp2=fopen(outfile, "wb"))==NULL)
    {
        fclose(fp1);
        return FAIL;
    }

    while ((num_read=fread(block, sizeof(DATA),
        BLOCKSIZE, fp1))==BLOCKSIZE)
```

```
    fwrite(block, sizeof(DATA), num_read, fp2);
    fwrite(block, sizeof(DATA), num_read, fp2);

    fclose(fp1);
    fclose(fp2);
    return SUCCESS;
}
```

## 7.5. XUẤT NHẬP CÓ ĐỊNH DẠNG

Sử dụng hai hàm sau:

```
int fprintf(FILE *fp, const char *format [, arg, ...]);
int fscanf (FILE *fp, const char *format, address, ...);
```

Ví dụ: Giả sử có file văn bản chứa ma trận các số thực với cấu trúc như sau:

- Dòng đầu tiên chứa 2 số nguyên m và n tương ứng với số dòng và số cột.
- m dòng tiếp theo mỗi dòng chứa n số thực.

Hàm sau đây sẽ đọc dữ liệu từ file văn bản để lưu vào mảng 2 chiều A có kích thước  $m \times n$ .

```
void DocFile(char *FileName, int &m, int &n, float A[][])
{
    FILE *f;
    int x;
    f=fopen(FileName, "rt");
    fscanf(f, "%d", &x);    m=x;
    fscanf(f, "%d", &x);    n=x;
    for(int i=0; i<m; i++)
        for(int j=0; j<n; j++)
        {
            float xx;
            fscanf(f, "%f", &xx); A[i][j]=xx;
        }
    fclose(f);
}
```

## 7.6. TRUY XUẤT NGẪU NHIÊN

Các ví dụ đề cập ở các phần trước giới thiệu cách truy xuất file tuần tự. Trong nhiều ứng dụng ta cần truy xuất các byte đặc biệt ở giữa file, trong những trường hợp này điều thực sự hữu dụng khi dùng các hàm truy xuất ngẫu nhiên đó là *fseek()* và *ftell()*.

Hàm *fseek()* di chuyển bộ định vị trí file đến một ký tự được chỉ ra trong một stream. Khai báo nguyên mẫu *fseek()* là:

```
fseek(FILE *stream, long int distance, int pos);
```

Ba đối số là:

<i>stream</i>	Con trỏ file
<i>distance</i>	Một khoảng cách được đo tính theo ký tự (có thể dương hoặc âm)
<i>pos</i>	Vị trí khởi đầu được tính

Có ba trị số được chọn của *pos*:

<i>SEEK_SET</i>	Vị trí bắt đầu file
<i>SEEK_CUR</i>	Vị trí hiện thời của bộ định vị file
<i>SEEK_END</i>	Vị trí cuối file

Ví dụ, câu lệnh:

```
stat = fseek(fp, 10, SEEK_SET);
```

di chuyển bộ chỉ vị trí đến ký tự thứ 10 trong stream. Ký tự tiếp theo sẽ được đọc hoặc ghi. Chú ý rằng các stream, cũng giống như mảng, bắt đầu tại vị trí 0 nên ký tự 10 chính là ký tự thứ 11 trong stream.

Giá trị trả về của *fseek()* là 0 nếu yêu cầu hợp lệ; nếu yêu cầu không hợp lệ *fseek()* sẽ trả về một giá trị khác 0. Điều này xảy ra do nhiều lý do. Ví dụ nếu *fp* được mở theo chế độ chỉ đọc, sau đó nó cố gắng di chuyển bộ định vị file về sau vị trí cuối file, điều này là không hợp lệ.

```
stat = fseek(fp, 1, SEEK_END);
```

Nếu *SEEK\_END* được sử dụng trong các file chỉ đọc, thì giá trị khoảng cách phải bé hơn hay bằng 0. Cũng giống như vậy nếu dùng *SEEK\_SET* thì giá trị khoảng cách phải lớn hơn hay bằng 0.

Với các stream nhị phân, đối số khoảng cách có thể nhận giá trị nguyên âm hoặc nguyên dương nhưng đừng đẩy bộ định vị ra ngoài file. Với các stream văn bản thì đối số khoảng cách phải là 0 hoặc là giá trị trả về của *ftell()*.

Hàm *ftell()* chỉ nhận một đối số, đó là một con trỏ file. Hàm trả về vị trí hiện thời của bộ chỉ vị trí. *ftell()* được dùng thường là trả về vị trí file sau khi thực hiện một số phép toán I/O.

Chú ý rằng vị trí trả về của *ftell()* được tính từ đầu file. Với các stream nhị phân, giá trị trả về của *ftell()* đại diện cho số ký tự thật sự kể từ đầu file. Với các stream văn bản giá trị trả về của *ftell()* là các giá trị cài đặt được đã định nghĩa mà chúng chỉ có ý nghĩa khi sử dụng là giá trị khoảng cách trong một lời gọi *fseek()*.

Áp dụng xây dựng hàm *filesize()* để tính kích cỡ file:

```
long filesize(FILE *stream)
{
    long curpos, length;

    curpos = ftell(stream);
    fseek(stream, 0L, SEEK_END);
```

```
length = ftell(stream);
fseek(stream, curpos, SEEK_SET);

return length;
}
```

Ngoài hàm *fseek()* để định vị đến vị trí ký tự bất kỳ trong một stream, ta có thêm hàm *rewind()* để đưa bộ chỉ vị trí về đầu file.

```
void rewind(FILE *stream);
```

Lời gọi hàm sau

```
rewind(fp);
```

là tương đương với

```
(void)fseek( fp, 0L, SEEK_SET );
```

Nhưng ở đây có một điểm khác là: *rewind()* xoá đi mã lỗi và end-of-file cho các stream và không trả về giá trị.

## 7.7. MỘT SỐ HÀM QUẢN LÝ FILE

```
int remove( const char *file_name );
```

Xoá một file được chỉ ra bởi tên. File bị xoá phải đóng. Hàm trả về 0 nếu xoá thành công, ngược lại sẽ trả về một giá trị khác 0.

```
int rename( const char *old, const char *new );
```

Đổi tên một file được chỉ ở *old* sang tên mới *new*. Tên mới phải chưa có trên đĩa. Hàm trả về 0 nếu thành công, khác 0 nếu không thành công.

## BÀI TẬP

### Bài tập 7.1:

1. Tạo một file SINHVIEN.DAT để lưu thông tin của một lớp sinh viên. Mỗi sinh viên cần những thông tin sau: Họ tên, Ngày sinh, Quê quán, Điểm trung bình, Xếp loại (trường xếp loại do chương trình tự tính lấy dựa vào điểm trung bình như sau: nếu điểm trung bình < 5 thì xếp loại 'D', nếu 5 <= điểm trung bình < 6.5 thì xếp loại 'C', nếu 6.5 <= điểm trung bình < 8 thì xếp loại 'B', trường hợp còn lại xếp loại 'A').
2. In toàn bộ nội dung của file SINHVIEN.DAT ra màn hình nếu có, ngược lại thì thông báo "File không tồn tại".
3. In danh sách tất cả sinh viên có thông tin lưu trong file SINHVIEN.DAT xếp loại khá ('B') trở lên.
4. Thông tin về điểm của sinh viên có họ tên là Bhoten, ngày sinh là Bngay và quê quán là Bquequan bị sai lệch. Hãy sửa điểm và xếp loại của sinh viên này với dữ liệu nhập từ bàn phím.
5. Đưa nội dung của file SINHVIEN.DAT vào file văn bản SINHVIEN.TXT sao cho mỗi thông tin về sinh viên được lưu trong một dòng.

6. In toàn bộ nội dung của một file văn bản SINHVIEN.TXT ra màn hình.

**Bài tập 7.2:** Đếm số dòng, số ký tự trắng xuất hiện trong một file văn bản đã có trên đĩa, tên file được nhập từ bàn phím khi chạy chương trình.

**Bài tập 7.3:** Một ma trận  $m \times n$  được chứa trong một file văn bản có tên MT.INP gồm: dòng đầu chứa hai số  $m, n$ ;  $m$  dòng tiếp theo mỗi dòng chứa  $n$  phần tử của từng hàng của ma trận. Hãy viết chương trình đọc dữ liệu từ file MT.INP, tính tổng của từng hàng ma trận và ghi lên file văn bản có tên KQ.OUT trong đó, dòng đầu chứa số  $m$ , dòng thứ hai chứa tổng của từng hàng của ma trận ( $m, n \leq 20$ ).

MT.INP	$\Rightarrow$	KQ.OUT
5 4		5
3 8 -1 5		15 4 8 12 12
5 7 -8 0		
4 -3 1 6		
2 4 -1 7		
3 6 8 -5		

**Bài tập 7.4:** Cho 3 ma trận số nguyên  $A = (a_{ij})_{m \times n}$ ,  $B = (b_{jk})_{n \times p}$ ,  $C = (c_{kl})_{p \times q}$ , được chứa trong file MATRIX.INP gồm: dòng đầu chứa 4 số  $m, n, p, q$ .  $m+n+p$  dòng tiếp theo lần lượt chứa  $m$  hàng ma trận  $A$ ,  $n$  hàng ma trận  $B$  và  $p$  hàng ma trận  $C$ . Viết chương trình đọc dữ liệu từ file MATRIX.INP và tính ma trận tích  $D = A \times B \times C$  rồi ghi lên file văn bản có tên MATRIX.OUT trong đó: Dòng đầu chứa  $m, q$ ;  $m$  dòng tiếp theo chứa  $m$  hàng của ma trận  $D$ .

$$d_{il} = \sum_{j=1}^n \sum_{k=1}^p a_{ij} * b_{jk} * c_{kl}$$

**Bài tập 7.5:** Một ma trận  $m \times n$  số thực được chứa trong một file văn bản có tên DULIEU.INP gồm: dòng đầu chứa hai số  $m, n$ ;  $m$  dòng tiếp theo lần lượt chứa  $m$  hàng của ma trận. Hãy viết chương trình đọc dữ liệu từ file DULIEU.INP, cho biết các hàng của ma trận có tổng phần tử trên hàng đó lớn nhất. Kết quả ghi lên file văn bản có tên DULIEU.OUT, trong đó dòng đầu chứa giá trị lớn nhất của tổng các phần tử trên một hàng, dòng thứ hai chứa chỉ số các hàng đạt giá trị tổng lớn nhất đó ( $m, n \leq 20$ ).

Chẳng hạn

DULIEU.INP	$\Rightarrow$	DULIEU.OUT
6 5		34
3 6 8 12 2		2 5 6
7 5 6 10 6		
8 2 4 5 1		
3 5 6 1 3		
10 12 3 1 8		
8 8 8 9 1		

**Gợi ý:**

- Dùng mảng  $S$  để lưu tổng giá trị các phần tử trên mỗi hàng. Cụ thể,  $S[i]$  là tổng giá trị các phần tử trên hàng thứ  $i$  của ma trận đã cho.

- Tập T, GTMax lần lượt là tập chứa các chỉ số các hàng và giá trị lớn nhất của các phần tử trên mỗi hàng tại thời điểm đang xét. Xuất phát ta xem hàng thứ nhất có tổng giá trị lớn nhất. Khi xét hàng thứ i có các trường hợp sau:
  - $S[i] > GTMax$ :  $S[i]$  mới là tổng lớn nhất và lúc này chỉ có hàng i đạt được giá trị này
  - $S[i] = GTMax$ : có thêm hàng i đạt giá trị lớn nhất.
  - $S[i] < GTMax$ : không có gì thay đổi

**Bài tập 7.6:** Viết chương trình sao chép nội dung của một file cho trước vào file khác, tên của file nguồn và file đích được nhập từ bàn phím khi chạy chương trình.

**Bài tập 7.7:** Viết chương trình đổi tên một file đã có trên đĩa.

**Bài tập 7.8:** Viết chương trình xóa một file có trên đĩa.

**Bài tập 7.9:** Viết chương trình nối 2 file văn bản đã có trên đĩa thành một file thứ 3 với tên file được nhập vào từ bàn phím.

**Gợi ý:**

- Mở file 1 và file 2 để đọc dữ liệu, mở file 3 để ghi dữ liệu.
- Lần lượt đọc từng phần tử trong file 1 và 2 lưu vào file 3.
- Đóng cả ba file lại.

**Bài tập 7.10:** Viết chương trình thực hiện các công việc sau:

1. Tạo ra 2 file số nguyên và sắp xếp chúng theo thứ tự tăng dần.
2. Hãy nối 2 file đó lại với nhau thành file thứ 3 sao cho file mới vẫn có thứ tự tăng dần.

**Bài tập 7.11:** Cho đa thức  $P(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$

Trong đó n là bậc của đa thức và  $a_0, a_1, \dots, a_n$  là các hệ số của đa thức được lưu trong một file văn bản với qui ước sau:

- Dòng đầu của file văn bản chứa bậc của đa thức và giá trị của x.
- Dòng tiếp theo chứa các hệ số của đa thức.

Ví dụ:  $P(x) = 3 + 2x - 5x^2 + 4x^3$ ,  $x = 2.5$  sẽ được lưu trong file văn bản như sau:

3 2.5

3 2 -5 4

Viết chương trình đọc file văn bản trên để lấy các số liệu rồi tính giá trị của đa thức.

**Bài tập 7.12:** Viết chương trình đếm số từ có trong một file văn bản.

**Gợi ý:**

- Viết hàm COUNT để đếm số từ của 1 dòng.
- Đọc từng dòng của file văn bản, dùng hàm COUNT để cộng dồn vào biến đếm.

**Bài tập 7.13:** Tại một cửa hàng, người ta quản lý các hoạt động MUA/BÁN trong năm bằng cùng một loại hoá đơn. Mỗi hoá đơn là một bản ghi gồm các trường:

SoHoadon (số hoá đơn); Thang (tháng mua/bán); Mahang (mã hàng mua/bán);  
Loai (nhận một trong hai giá trị 'M'(mua) hoặc 'B' (bán))

Như vậy căn cứ vào trường Loai ta biết đó là hoá đơn mua hay hoá đơn bán. Viết chương trình cho phép nhập vào một dãy các hoá đơn và lưu vào file có tên



Hoadon.dat, quá trình nhập dừng khi SoHoadon = 0. Tính số dư trong tháng n (n được nhập từ bàn phím khi thực hiện chương trình). Biết rằng số dư trong một tháng được tính theo công thức:

Số dư = Tổng bán - Tổng mua,

trong đó tổng bán, tổng mua lần lượt là tổng số tiền bán, mua trong tháng đó.

**Yêu cầu:**

- Khi nhập chú ý kiểm tra để Loại chỉ nhận một trong hai giá trị 'M' hoặc 'B' và tháng chỉ nhận giá trị từ 1 đến 12.
- Không được sử dụng mảng.

**Bài tập 7.14:** Người ta quản lý các đầu sách của một thư viện bằng một bản ghi gồm có các trường: Masach, Tensach, Tentacgia, Nhaxb (nhà xuất bản), Namxb (năm xuất bản), SoLuong. Viết chương trình cho phép thực hiện các thao tác sau:

- a. Nhập vào các đầu sách có trong thư viện và lưu vào file có tên Sach.dat, quá trình nhập dừng khi mã sách đưa vào là một xâu rỗng.
- b. Duyệt và in ra tên các quyển sách được xuất bản sau năm m (m được nhập từ bàn phím khi thực hiện chương trình).
- c. Bổ sung sách vào thư viện theo yêu cầu: nếu sách đã có thì chỉ tăng số lượng sách bổ sung, ngược lại thêm một đầu sách mới vào file.

**Chú ý:**

- Không được sử dụng mảng
- Khi nhập chú ý kiểm tra để năm xuất bản  $\leq$  năm hiện tại
- Sau khi in ra danh sách các đầu sách xuất bản sau năm m, cho biết thêm danh sách đó có bao nhiêu đầu sách tất cả.

**Bài tập 7.15:** Người ta lưu thông tin các cán bộ trong cơ quan vào file có tên CANBO.DAT, mỗi cán bộ là một bản ghi gồm các trường: STT, Hoten, Ngaysinh, Diachi, HSLuong, HSPhucap, SoDT. Hãy viết chương trình thực hiện các yêu cầu sau:

- a. Nhập danh sách cán bộ và lưu vào file, quá trình nhập dừng khi họ tên nhập vào là xâu rỗng và trường STT chương trình tự gán.
- b. In ra danh sách cán bộ có hệ số lương nằm trong khoảng từ x đến y, x và y là các số thực được nhập từ bàn phím khi thực hiện chương trình.
- c. Sao chép thông tin các cán bộ có tuổi trên 50 vào một file khác.
- d. In bảng lương của tất cả cán bộ lưu trong file CANBO.DAT ra màn hình gồm các thông tin: STT, Hoten, HSLuong, Luong, trong đó Luong được tính theo công thức  $Luong = (HSLuong + HSPhucap) * 290000$ , dữ liệu in ra định dạng theo cột. Cuối bảng, in tổng lương của toàn cơ quan.
- e. Sao chép nội dung của file CANBO.DAT vào file văn bản CANBO.TXT, mỗi cán bộ tương ứng một dòng.

**Bài tập 7.16:** Viết chương trình nhập vào tên một file văn bản. Kiểm tra file này có tồn tại trên đĩa không? Nếu có, in nội dung của file từ dòng thứ m đến dòng thứ n, trong đó m và n là hai số nguyên dương bất kỳ được nhập từ bàn phím khi thực hiện chương trình.

**Bài tập 7.17:** Giả sử trong một file văn bản trên đĩa có tên là MATRIX.TXT người ta đã lưu các số liệu về một ma trận A cấp mxn và một vector X n chiều. Cách lưu trữ như sau:

- Dòng đầu tiên chứa hai số m và n
- Dòng thứ hai chứa vector X
- m dòng tiếp theo lần lượt chứa m hàng của ma trận A
- Giữa các số trong một dòng cách nhau một ký tự trắng

Viết chương trình tính giá trị vector  $Y = AX$  và đưa kết quả ra màn hình đồng thời lưu vào cuối file MATRIX.TXT (A và X được lấy từ file MATRIX.TXT)

**Yêu cầu:**

Chương trình phải thiết lập các hàm sau

- LayDulieu(A,X,m,n) thực hiện việc đọc dữ liệu từ file MATRIX.TXT và gán cho A, X, m, n
- TinhTich(A,X,m,n,Y) thực hiện việc tính vector Y
- LuuKetqua(Y,m) thực hiện việc in vector Y ra màn hình và lưu vào cuối file MATRIX.TXT
- Thành phần thứ i của vector Y được tính theo công thức

$$Y[i] = \sum_{j=1}^m A[i, j] * X[j]$$

**Bài tập 7.18:** Giả sử trong một file văn bản trên đĩa có tên là DANHBA.TXT lưu danh bạ điện thoại trong thành phố. Cách lưu như sau:

- Dòng đầu lưu hai số nguyên dương m và n, trong đó m là số máy điện thoại thuộc cơ quan nhà nước, còn n là số máy thuộc tư nhân.
- m dòng tiếp theo lưu thông tin lần lượt của m máy điện thoại thuộc cơ quan nhà nước, mỗi dòng ghi số điện thoại, một ký tự trắng và sau đó là tên cơ quan.
- n dòng tiếp theo nữa lưu thông tin lần lượt của n máy điện thoại tư nhân, mỗi dòng ghi số điện thoại, một ký tự trắng và sau đó là họ tên chủ điện thoại.

Viết chương trình đọc dữ liệu từ file DANHBA.TXT và in bảng danh bạ điện thoại ra màn hình theo thứ tự tăng dần của chủ máy điện thoại, các máy điện thoại thuộc cơ quan nhà nước in trước rồi đến các máy điện thoại tư nhân. Danh sách in ra theo 3 cột, cột 1 ghi số điện thoại, cột 2 ghi tên cơ quan hoặc tên chủ máy điện thoại, cột 3 ghi loại là TN (tư nhân) hoặc NN (nhà nước)

**Yêu cầu:**

- Khai báo kiểu bản ghi là MAYDT bao gồm 3 trường: SoDt, TenChu, Loai
- Thiết lập hàm LayDulieu(A,k) để đọc dữ liệu từ file DANHBA.TXT và lưu vào mảng A (mảng các MAYDT) với k là số phần tử của mảng.
- Thiết lập hàm SAPXEP(A,k) để sắp xếp mỗi nhóm máy điện thoại nhà nước, tư nhân theo thứ tự tăng dần của tên chủ máy điện thoại trong mảng A.
- Thiết lập hàm INKETQUA(A,k) để in ra màn hình danh bạ điện thoại từ mảng A.

**Bài tập 7.19:** Cho một file văn bản có tên là MATRIX.TXT với nội dung như sau:

- Dòng đầu tiên của file chứa hai số nguyên dương m và n lần lượt là số hàng và số cột của một ma trận cấp mxn ( $m, n \leq 50$ ).
- m dòng tiếp theo mỗi dòng chứa n số nguyên là giá trị các phần tử của mỗi hàng.

Hãy viết chương trình thực hiện các yêu cầu sau:

- Viết thủ tục LAYDULIEU để đọc dữ liệu từ file MATRIX.TXT và lưu vào mảng hai chiều A.
- Viết hàm MAXDONG(i:Byte): LongInt trả về giá trị lớn nhất của hàng i.
- Ghi các giá trị lớn nhất của mỗi hàng vào cuối file MATRIX.TXT.

**Bài tập 7.20:** Viết chương trình tạo ra hai tập tin lưu các số kiểu word mà các số trong mỗi file đã được sắp thứ tự tăng dần. Hãy tạo tập tin mới chứa tất cả các số của 2 tập tin trên sao cho thứ tự tăng dần vẫn được duy trì.

**Chú ý:** Không được dùng mảng.

**Bài tập 7.21:** Giả sử trong một file văn bản trên đĩa có tên là MT.DAT người ta đã lưu các số liệu về hai ma trận A và B cùng cấp mxn. Cách lưu trữ như sau:

- Dòng đầu tiên chứa hai số m và n  
 m dòng tiếp theo lần lượt chứa m hàng của ma trận A  
 m dòng tiếp theo nữa lần lượt chứa m hàng của ma trận B  
 Giữa các số trong một dòng cách nhau một ký tự trắng

Viết chương trình tính ma trận tổng  $C = A + B$  và ghi kết quả vào file MT.OUT với cấu trúc: dòng đầu chứa số m, m dòng tiếp theo chứa m hàng của ma trận C.

**Bài tập 7.22:** Để có thể sao chép các file có kích thước lớn lên đĩa mềm, người ta chia nhỏ file cần chép thành nhiều file có kích thước nhỏ hơn, sau đó nối các file này lại bằng lệnh copy. Hãy viết chương trình sao chép một file thành hai file có kích thước bằng nhau. Tên của tập tin nguồn và hai tập tin đích được nhập từ bàn phím khi thực hiện chương trình.

**Hướng dẫn:** Khai báo các file nguồn và đích là các file không định kiểu. Gọi Temp là một nửa kích thước của file nguồn, tính bằng byte. Thực hiện việc sao chép từ byte đầu tiên đến byte thứ Temp vào file đích thứ nhất, sau đó chép phần còn lại của file nguồn vào file đích thứ hai.

## CHƯƠNG 8: ĐỒ HỌA

### 8.1. MÀN HÌNH TRONG CHẾ ĐỘ ĐỒ HỌA

Hình ảnh trong chế độ đồ họa (Graphic) được tạo ra bằng các điểm ảnh (Pixel), số điểm ảnh của màn hình đồ họa tùy thuộc vào từng loại CARD màn hình và MODE qui định cho màn hình đó.

Việc lập trình trong chế độ đồ họa cần phải xác định được loại màn hình đang sử dụng và chương trình phải vận hành được trên nhiều loại màn hình khác nhau.

Tọa độ của một điểm ảnh trên màn hình đồ họa cũng giống như trong chế độ văn bản (TEXT) với điểm ảnh đầu tiên trên góc trái màn hình là (0,0), tọa độ đỉnh dưới phải tùy thuộc vào độ phân giải của màn hình, CARD màn hình và MODE màn hình.

Để sử dụng được chế độ đồ họa trên màn hình, ta cần phải có các File sau:

<graphics.h>	Chứa các lệnh đồ họa
*.BGI	Chứa Font màn hình
*.CHR	Chứa Font ký tự

### 8.2. KHỞI TẠO VÀ THOÁT KHỎI CHẾ ĐỘ ĐỒ HỌA

#### 8.2.1. Khởi tạo chế độ đồ họa

```
void initgraph(int *gd,int *gm,char *Path);
```

trong đó:

- gd: Chỉ CARD màn hình.

Thông thường, một chương trình phải được chạy trên nhiều loại màn hình khác nhau nên ta có thể khai báo:

```
gd = DETECT (= 0)
```

Với hằng DETECT, máy sẽ tự động tìm CARD màn hình tương ứng để chạy chương trình.

- gm: Chỉ MODE màn hình.

Trong trường hợp khai báo gd = DETECT thì không cần thiết phải khai báo gm vì máy tính sẽ tự xác định loại CARD màn hình và thiết lập chế độ MODE màn hình tương ứng với CARD màn hình đó.

- Path: Đường dẫn đến nơi chứa các file \*.BGI. Nếu Path = "" thì ta hiểu là các file \*.BGI nằm trong thư mục hiện hành.

#### 8.2.2. Thoát khỏi chế độ đồ họa

```
void closegraph();
```

Sau đây là cấu trúc chung của một chương trình đồ họa:.

```
#include <graphics.h>
```

```
void ThietLapDoHoa()
{
    int gd=0,gm;
    initGraph(&gd,&gm,"D:\\TC\\BGI");
}
void main()
{
    ThietLapDoHoa;
    . . .
    <Các lệnh vẽ>;
    ...
    Closegraph();
}
```

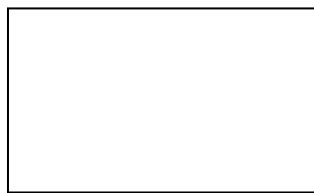
### 8.3. TỌA ĐỘ VÀ CON TRỎ TRÊN MÀN HÌNH ĐỒ HỌA

#### 8.3.1. Lấy kích thước màn hình

**int getmaxx();** và **int getmaxy();**

Cho tọa độ cột lớn nhất và dòng lớn nhất của màn hình.

(0,0)



(getmaxx,getmaxy)

#### 8.3.2. Di chuyển con trỏ

**void moveto(int x,int y);**

Di chuyển con trỏ đến tọa độ (x,y).

**void moverel(int dx,int dy);**

Di chuyển con trỏ đến tọa độ mới cách tọa độ cũ khoảng cách là dx, dy.

#### 8.3.3. Vẽ điểm

**void putpixel(int x,int y,int color);**

Vẽ một điểm tại toạ độ (x,y) với màu là color.

**unsigned getpixel(int x,int y);**

Lấy màu của một điểm tại tọa độ (x,y).

### 8.4. ĐẶT MÀU TRÊN MÀN HÌNH ĐỒ HỌA

#### 8.4.1. Đặt màu cho đối tượng cần vẽ

**void setcolor(int color);**

### 8.4.2. Đặt màu nền

```
void setbkcolor(int color);
```

## 8.5. CỬA SỔ TRONG CHẾ ĐỘ ĐỒ HỌA

### 8.5.1. Đặt cửa sổ trên màn hình

```
void setviewport(int x1,int y1,int x2,int y2;int Clip);
```

trong đó:

(x1,y1): đỉnh trên trái của cửa sổ.

(x2,y2): đỉnh dưới phải của cửa sổ.

Nếu Clip  $\neq 0$  thì những gì vượt khỏi màn hình sẽ bị cắt bỏ.

❖\* **Chú ý:** Khi tạo cửa sổ thì tọa độ trên màn hình sẽ thay đổi theo.

Tọa độ mới = Tọa độ cũ - Tọa độ đỉnh trên trái.

### 8.5.2. Xóa hình ảnh trong cửa sổ

- Xóa hình ảnh trong cửa sổ, ta dùng hàm **clearviewport()**;

- Xóa toàn bộ màn hình, ta dùng hàm **cleardevice()**;

## 8.6. VIẾT CHỮ TRONG ĐỒ HỌA

### 8.6.1. Thiết lập font chữ

```
void settextstyle(int font,int dir,int size);
```

- Các font có thể chứa các hằng sau:

DEFAULT\_FONT = 0; TRIPLEX\_FONT = 1; SMALL\_FONT = 2;

SANS\_SERIF\_FONT = 3; GOTHIC\_FONT = 4;

- Dir có các hằng sau:

HORIZ\_DIR = 0 Từ trái qua phải.

VERT\_DIR = 1 Từ dưới lên trên.

- size: độ lớn của chữ.

### 8.6.2. Thiết lập phân bố chữ

```
void settextjustify(int Hz,int Vt);
```

Chọn vị trí của chữ xung quanh tọa độ định sẵn.

- Hz là phân bố chữ theo trục ngang. Có các hằng sau:

LEFT\_TEXT = 0 Chữ viết nằm bên phải trục đứng.

CENTER\_TEXT = 1 Chữ viết nằm ở giữa trục đứng.

RIGHT\_TEXT = 2 Chữ viết nằm bên trái trục đứng.

- Vt là bố trí chữ theo hướng dọc đối với tọa độ qui định xuất chuỗi. Các hằng liên quan:

BOTTOM\_TEXT = 0    Chữ viết nằm bên trên trục ngang.  
 CENTER\_TEXT = 1    Chữ viết nằm ở giữa trục ngang.  
 TOP\_TEXT = 2        Chữ viết nằm bên dưới trục ngang.

### 8.6.3. Viết một xâu ký tự lên màn hình

- Xuất một xâu ký tự tại vị trí con trỏ:

Dùng hàm **void outtext(char \*st);**

- Xuất một xâu ký tự tại tọa độ x,y:

Dùng hàm **void outtextxy(int x,int y,char \*st);**

●\***Chú ý:** Cách xuất chuỗi của hai hàm trên được qui định trong các hàm **settextstyle** và **settextjustify**.

## 8.7. VẼ CÁC HÌNH CƠ BẢN

### 8.7.1. Chọn kiểu đường

**void setlinestyle(int Ls,int Pt,int Tk);**

**Ls:** kiểu đường vẽ, có các giá trị sau:

- 0: Đường liền nét
- 1: Nét đứt
- 2: Nét chấm gạch
- 3: Nét gạch
- 4: Đường do người thiết kế tạo ra.

**Pt:** xác định màu vẽ.

- . Nếu Ls = 0..3 thì Pt=0 (Lấy giá trị Default)
- . Nếu Ls = 4 thì Pt là số nguyên chỉ màu của kiểu đường.

**Tk:** xác định độ dày của đường.

- Tk = 1: bình thường.
- Tk = 3: đậm nét.

### 8.7.2. Vẽ đoạn thẳng

**void line(int x1,int y1,int x2,int y2);** vẽ từ điểm (x1,y1) đến điểm (x2,y2).

**void lineto(int x,int y);** vẽ từ vị trí con trỏ đến điểm (x,y).

**void linerel(int dx,int dy);** vẽ từ vị trí con trỏ đến điểm cách nó một khoảng dx,dy.

Ví dụ 1: Vẽ đồ thị hàm số:  $f(x) = ax^2 + bx + c$  trên đoạn  $[-10,10]$ .

**Ý tưởng:**

**Bước 1:** Xác định đoạn cần vẽ [Min,Max].

**Bước 2:** Đặt gốc tọa độ lên màn hình (x0,y0).

Chia tỉ lệ vẽ trên màn hình theo hệ số k.

Chọn số gia dx trên đoạn cần vẽ.

**Bước 3:** Chọn điểm xuất phát:  $x = \text{Min}$ , tính  $f(x)$ .

Đổi qua tọa độ màn hình và làm tròn:

$x1 = x0 + \text{Round}(x * k);$

$y1 = y0 - \text{Round}(f(x) * k);$

Di chuyển đến  $(x1, y1)$ :  $\text{moveto}(x1, y1);$

**Bước 4:** Tăng x lên:  $x = x + dx;$

Đổi qua tọa độ màn hình và làm tròn:

$x2 = x0 + \text{Round}(x * k);$

$y2 = y0 - \text{Round}(f(x) * k);$

Vẽ đến  $(x2, y2)$ :  $\text{lineto}(x2, y2);$

**Bước 5:** Lặp lại bước 4 cho đến khi  $x \geq \text{Max}$  thì dừng.

```
#include<conio.h>
```

```
#include<graphics.h>
```

```
float a,b,c,d,min,max;
```

```
int round(float x)
```

```
{
    if (x>0) return int (x+0.5);
    else return int (x-0.5);
}
```

```
void khoitaodohoa()
```

```
{
    int gd=0,gm=0;
    initgraph(&gd,&gm,"d:\\tc\\bgi");
}
```

```
float f(float x)
```

```
{
    return(a*x*x*x+b*x*x+c*x+d);
}
```

```
void vedothi(float min,float max)
```

```
{
    int x0,y0,x1,y1,x2,y2;
    float x,dx,k;
```



```

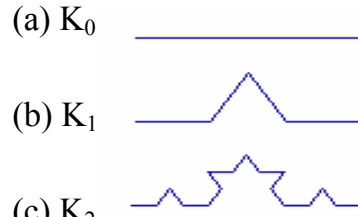
x0=getmaxx()/2;
y0=getmaxy()/2;
k=(float)getmaxx()/50;
dx=0.001;
setcolor(12);
//Vẽ trục tọa độ
line(0,y0,2*x0,y0);
line(x0,0,x0,2*y0);
//vẽ đồ thị
x=min;
setcolor(14);
x1=x0+round(x*k);
y1=y0-round(f(x)*k);
moveto(x1,y1);
while (x<max)
{
    x=x+dx;
    x2=x0+round(x*k);
    y2=y0-round(f(x)*k);
    lineto(x2,y2);
}
}

void main()
{
    khoitaodohoa();
    min=-10;max=10;
    a=1;b=-1;c=-1;d=2;
    vedothi(min,max);
    getch();
    closegraph();
}

```

**Ví dụ 2:** Viết chương trình vẽ cung Koch. Các bước phát sinh của cung Koch được thực hiện trong hình sau:

- Bắt đầu từ đường ngang  $K_0$  có độ dài bằng 1.
- Để tạo cung bậc-1(gọi là  $K_1$ ), chia đường thành ba phần và thay đoạn giữa bằng tam giác đều có cạnh dài  $1/3$ . Bây giờ, toàn bộ đường cong có độ dài  $4/3$ .
- Cung bậc-2  $K_2$  có được bằng cách dựng tiếp các tam giác đều từ 4 đoạn của  $K_1$ . Vì mỗi đoạn có độ dài tăng  $4/3$  lần nên toàn bộ cung dài ra  $4/3$  lần.



### Ý tưởng:

Từ hình (b) ta thấy rằng, đầu tiên hướng vẽ quay trái  $60^\circ$ , rồi quay phải  $120^\circ$ , cuối cùng quay trái  $60^\circ$  để trở về hướng ban đầu.

```
#include <graphics.h>
#include <conio.h>
#include <math.h>
```

```
#define PI 3.1416
```

```
void ThietLapDoHoa()
{
    int gd=0, gm;
    initgraph(&gd, &gm, "D:\\TC\\bgi");
}
```

```
int Round(float x)
{
    if (x>0) return int (x+0.5);
    else return int (x-0.5);
}
```

```
void Koch(float dir, float len, int n)
{
    if (n>0)
    {
        Koch(dir, len/3, n-1);
        dir=dir+60; //Quay phải 60 độ
        Koch(dir, len/3, n-1);
        dir=dir-120; //Quay trái 120 độ
        Koch(dir, len/3, n-1);
        dir=dir+60; //Quay phải 60 độ
        Koch(dir, len/3, n-1);
    }
    else
        linerel(Round(len*cos(dir*PI/180)), Round(len*sin(dir*PI/180)));
}
```

```
void main()
{
    float Goc=180,Length=250;
    int n=4;
    ThietLapDoHoa();
    moveto(350,200);
    Koch(Goc,Length,n);
    getch();
    closegraph();
}
```

### 8.7.3. Vẽ hình chữ nhật

**void rectangle(int x1,int y1,int x2,int y2);**

Vẽ hình chữ nhật với đỉnh trên trái là (x1,y1) và đỉnh dưới phải là (x2,y2).

Ví dụ: Vẽ các hình chữ nhật ngẫu nhiên trên màn hình.

```
#include <graphics.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>

void ThietLapDoHoa()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "d:\\tc\\bgi");
}

void Demo()
{
    int x1,y1,x2,y2;
    randomize();
    do
    {
        x1=random(getmaxx());
        y1=random(getmaxy());
        x2=random(getmaxx()-x1)+x1;
        y2=random(getmaxy()-y1)+y1;
        setcolor(random(14)+1);
        rectangle(x1,y1,x2,y2);
        delay(100);
    }
    while (kbhit()==0);
}

void main()
{

```

```
ThietLapDoHoa();
Demo();
getch();
closegraph();
}
```

#### 8.7.4. Vẽ cung tròn

**void arc(int x,int y,int g1,int g2,int R);**

Vẽ cung tròn có tâm (x,y) bán kính R, góc bắt đầu là g1 và góc kết thúc là g2.

#### 8.7.5. Vẽ đường tròn - Ellipse

Vẽ đường tròn: **void circle(int x,int y,int R);**

Vẽ ellipse: **void ellipse(x,y:integer; g1,g2,Rx,Ry:Word);**

Vẽ Ellipse có tâm (x,y) bán kính ngang Rx, bán kính dọc Ry, góc bắt đầu là g1 và góc kết thúc là g2.

#### 8.7.6. Định MODE vẽ cho đoạn thẳng

**void setwritemode(int Mode);**

Ta có thể chọn Mode bằng các hằng sau:

. COPY\_PUT = 0; đây là Mode chèn, đường mới sẽ không xóa đường cũ.

. XOR\_PUT = 1; đây là Mode xóa, đường mới sẽ xóa đường cũ.

Ví dụ: Vẽ một kim đồng hồ quay quanh tâm O(x0,y0).

```
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <dos.h>

#define PI 3.1416

void ThietLapDoHoa()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "d:\\tc\\bgi");
}

int Round(float x)
{
    if(x>0) return int(x+0.5);
    else return int(x-0.5);
}

void VeKim(int x0,int y0,float R,float Alpha)
{

```

```

    line(x0,y0,x0+Round(R*cos(PI*Alpha/180)),
          y0-Round(R*sin(PI*Alpha/180)));
}

void Demo()
{
    int x0=getmaxx()/2;
    int y0=getmaxy()/2;
    int R=100;
    float Alpha=90, Beta=6;
    setwrite mode(XOR_PUT);
    VeKim(x0,y0,R,Alpha);
    do
    {
        delay(500);
        VeKim(x0,y0,R,Alpha);
        Alpha=Alpha-Beta;
        VeKim(x0,y0,R,Alpha);
    }
    while (kbhit()==0);
}

void main()
{
    ThietLapDoHoa();
    Demo();
    getch();
    closegraph();
}

```

## 8.8. TÔ MÀU CÁC HÌNH

### 8.8.1. Chọn kiểu tô

**void setfillstyle(int Pt,int Color);**

Với:

- Pt: Mẫu tô của hình. Có các hằng từ 0 đến 12.

0: Tô bằng màu nền.

1: Tô bằng màu viền.

2: Tô bằng các dấu ---

.....

- Color: Màu tô của hình.

### 8.8.2. Vẽ hình chữ nhật có tô màu ở bên trong

**void bar(int x1,int y1,int x2,int y2);**

Vẽ hình chữ nhật có tô màu và mẫu tô được xác định bởi hàm `setfillstyle`.

Ví dụ: Viết chương trình tạo Menu cho phép chọn và thực hiện các chức năng bằng cách di chuyển mũi tên trên các hộp sáng, các thủ tục thực hiện xong quay trở lại Menu chính. Nhấn ESC để thoát khỏi chương trình.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<graphics.h>

#define TRUE 0
#define FALSE 1

const mau1=15;
const mau2=8;
const maumn=7;
const mauchu=1;
const XTop=180;
const YTop=100;
const Dy=32;
const Dx=270;

void ThietLapDoHoa()
{
    int gd=0,gm;
    initgraph(&gd,&gm,"d:\\tc\\bgi");
}

void Baitap1()
{
    cleardevice();
    outtext("\nBAI TAP TINH TONG CAC CHU SO CUA N");
    getch();
}

void Baitap2()
{
    cleardevice();
    outtext("\nBAI TAP VE MANG MOT CHIEU\n");
    getch();
}

void Baitap3()
{
    cleardevice();
    outtext("\nXin loi! Bai tap nay tui chua lam!");
    getch();
}
```

```

}

void Baitap4()
{
    cleardevice();
    outtext("\nBAI TAP TIM UOC CHUNG LON NHAT");
    getch();
}

//Vẽ nút menu
void Box(int x1,int y1,int x2,int y2,
        char MauVienTren,char MauVienDuoai,char MauNen)
{
    int i;
    setfillstyle(1,MauNen);
    bar(x1,y1,x2,y2);
    setcolor(MauVienTren);
    for(i=0;i<=1;i++)
    {
        moveto(x1-i,y2+i);
        lineto(x1-i,y1-i);
        lineto(x2+i,y1-i);
    }
    setcolor(MauVienDuoai);
    for(i=0;i<=1;i++)
    {
        moveto(x2+i,y1-i);
        lineto(x2+i,y2+i);
        lineto(x1-i,y2+i);
    }
}

//Viết xâu st ra màn hình tại dòng thứ n voi mau la color
void Write(char *st,int n,int color)
{
    setcolor(color);
    outtextxy(XTop+20,YTop+15+n*Dy,st);
}

void Ve_menu(int Xdau,int Ydau,int chon,int SoDong,
            char *DongMN[])
{
    int i;
    cleardevice();
    for(i=0;i<SoDong;i++)
    {
        if(i==chon)

```

```

        Box(Xdau,Ydau+i*Dy+6,Xdau+Dx,Ydau+i*Dy+Dy,
            mau2,mau1,maumn);
    else
        Box(Xdau,Ydau+i*Dy+6,Xdau+Dx,Ydau+i*Dy+Dy,
            mau1,mau2,maumn);
    Write(DongMN[i],i,mauchu);
}
}

void main()
{
    char ch,*st[10];
    st[0]="Bai tap tinh tong cac chu so";
    st[1]="Bai tap ve mang mot chieu";
    st[2]="Bai tap thiet ke ham De quy";
    st[3]="Bai tap tim Uoc chung lon nhat";
    st[4]="<ESC> Ket thuc chuong trinh!";
    int chon=0,luuchon,sodong=5,ok=FALSE;
    ThietLapDoHoa();
    Ve_menu(XTop,YTop,chon,sodong,st);
    do
    {
        ch=getch(); //Nhan mot phim
        switch (ch)
        {
            case 72: //phim len
                luuchon=chon;
                chon--;
                if(chon<0) chon=sodong-1;
                Box(XTop,YTop+luuchon*Dy+6,XTop+Dx,
                    YTop+luuchon*Dy+Dy,mau1,mau2,maumn);
                Write(st[luuchon],luuchon,mauchu);
                Box(XTop,YTop+chon*Dy+6,XTop+Dx,
                    YTop+chon*Dy+Dy,mau2,mau1,maumn);
                Write(st[chon],chon,mauchu);
                break;
            case 80://phim xuong
                luuchon=chon;
                chon++;
                if(chon==sodong) chon=0;
                Box(XTop,YTop+luuchon*Dy+6,XTop+Dx,
                    YTop+luuchon*Dy+Dy,mau1,mau2,maumn);
                Write(st[luuchon],luuchon,mauchu);
                Box(XTop,YTop+chon*Dy+6,XTop+Dx,
                    YTop+chon*Dy+Dy,mau2,mau1,maumn);
                Write(st[chon],chon,mauchu);
                break;
        }
    }
}

```



```

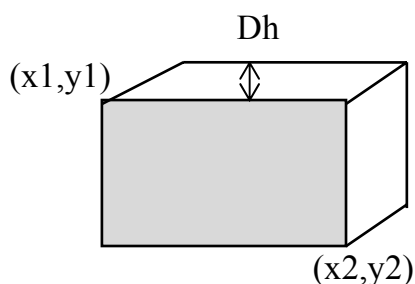
        case 13: //phim ENTER
            ok=TRUE; break;
    }
    if (ok==TRUE) //Neu phim ENTER duoc nhan
    {
        switch (chon)
        {
            case 0:
                Baitap1();
                Ve_menu(XTop,YTop,chon,sodong,st);
                break;
            case 1:
                Baitap2();
                Ve_menu(XTop,YTop,chon,sodong,st);
                break;
            case 2:
                Baitap3();
                Ve_menu(XTop,YTop,chon,sodong,st);
                break;
            case 3:
                Baitap4();
                Ve_menu(XTop,YTop,chon,sodong,st);
                break;
            case 4: exit(0);
        }
        ok=FALSE; //tra lai trang thai ENTER chua duoc nhan
    }
}
while (ch!=27); //Nhan phim ESC de thoat
closegraph();
}

```

### 8.8.3. Vẽ hình hộp chữ nhật

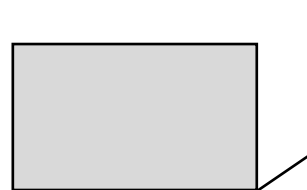
**void bar3d(int x1,int y1,int x2,int y2,int Dh,int Top) ;**

Vẽ hình hộp chữ nhật có tọa độ đỉnh trên là (x1,y1), đỉnh dưới là (x2,y2) và bề dày là Dh.



Có nắp

Top ≠ 0: Hình hộp có nắp.



Không có nắp

Top = 0: Hình hộp không có nắp.

#### 8.8.4. Vẽ và tô màu Ellipse

```
void fillellipse(int x,int y,int Rx,int Ry);
```

Vẽ hình và tô màu cho ellipse có tâm (x,y), bán kính theo 2 trục là Rx và Ry với màu và mẫu tô được xác định bởi hàm setfillstyle.

#### 8.8.5. Vẽ hình quạt tròn

```
void pieslice(int x,int y,int g1,int g2,int R);
```

Vẽ hình quạt tròn có tâm (x,y), góc đầu g1, góc cuối g2, bán kính R.

#### 8.8.6. Vẽ hình quạt Ellipse

```
void sector(int x,int y,int g1,int g2,int Rx,int Ry);
```

#### 8.8.7. Làm loang màu một vùng kín

```
void floodfill(int x,int y,int Color);
```

Trong đó:

(x,y): điểm nằm trong vùng kín.

Color: màu muốn tô.

#### 8.8.8. Vẽ đa giác

Đối với một đa giác bất kỳ có N đỉnh, ta phải khai báo N+1 đỉnh để vẽ đường gấp khúc với tọa độ điểm đầu trùng với tọa độ điểm cuối. Để vẽ đa giác, ta dùng hàm:

```
void drawpoly(int N,int P[]);
```

trong đó:

- N: số đỉnh của đa giác + 1
- P: chứa tọa độ các đỉnh, cứ hai phần tử liên tiếp trong mảng sẽ tạo thành một điểm (x,y).

Ví dụ: Viết chương trình để vẽ đa giác đều có n đỉnh.

**Ý tưởng**:

Khi vẽ một đa giác đều N đỉnh, các đỉnh này nằm trên một đường tròn tâm O bán kính R đồng thời khoảng cách giữa hai đỉnh và tâm tạo thành một góc không đổi là  $2\pi/N$ .

Giả sử đỉnh thứ nhất của đa giác nằm trên đường thẳng tạo với tâm một góc  $0^0$ , đỉnh thứ hai tạo một góc  $2\pi/N$  và đỉnh thứ i sẽ tạo một góc là  $2\pi*(i-1)/N$ .

Giả sử P0 là tọa độ tâm của đa giác, đỉnh thứ i của đa giác sẽ tạo một góc là:

$$\text{Angle} = 2\pi*(i-1)/N$$

Nhưng nếu đa giác này có đỉnh đầu tiên tạo một góc bằng A0 thì:

$$\text{Angle} = 2\pi*((i-1)/N + A0/360)$$

Và tọa độ các đỉnh này trên màn hình sẽ là:

$$P[i].x = P0.x + R \cdot \cos(\text{Angle})$$

$$P[i].y = P0.y - R \cdot \sin(\text{Angle})$$

Ta xây dựng một hàm để tự động lưu các đỉnh của đa giác đều vào mảng P. Trong đó: P0 là tọa độ tâm, A0 là góc bắt đầu, R là bán kính, N là số đỉnh của đa giác đều ( $N \geq 3$ ).

```
#include <graphics.h>
#include <conio.h>
#include <math.h>

#define Max 10
#define PI 3.1416

struct ToaDo
{
    int x,y;
};

void ThietLapDoHoa()
{
    int gd=0,gm;
    initgraph(&gd,&gm,"D:\\TC\\bgi");
}

int Round(float x)
{
    if (x>0) return int (x+0.5);
    else return int (x-0.5);
}

void TaoDinh(float R,float A0,int n,ToaDo P0,int P[])
{
    for(int i=0;i<n;i++)
    {
        float Angle=2*PI*(float(i)/n + A0/360);
        P[2*i] =P0.x + Round(R*cos(Angle));
        P[2*i+1]=P0.y - Round(R*sin(Angle));
    }
    P[2*n]=P[0];
    P[2*n+1]=P[1];
}

void main()
{
    ThietLapDoHoa();
    int n=5, P[Max];
```

```

ToaDo P0;
P0.x=getmaxx()/2;
P0.y=getmaxy()/2;
float A0=90,R=getmaxy()/4;
TaoDinh(R,A0,n,P0,P);
drawpoly(n+1,P);
getch();
closegraph();
}

```

## 8.9. CÁC KỸ THUẬT TẠO HÌNH CHUYỂN ĐỘNG

### 8.9.1. Kỹ thuật lật trang màn hình

CARD màn hình có nhiều trang, mỗi trang được đánh số 0,1,2,...

Để vẽ hình lên một **trang màn hình** (trong vùng đệm), ta dùng hàm:

```
void setactivepage(int Page);
```

Trong đó, Page là số của trang màn hình. Hàm này được đặt trước khi có lệnh vẽ ra màn hình.

Để đưa trang màn hình ra màn hình, ta dùng hàm:

```
void setvisualpage(int Page);
```

Page: trang màn hình muốn xem.

Thông thường, màn hình sẽ làm việc và hiện ra trên trang 0. Do đó, để vừa xem màn hình vừa vẽ lên trang màn hình khác, ta thường dùng hai hàm trên đi kèm với nhau.

Để thực hiện một cách tự động khi sử dụng kỹ thuật lật hình này, ta thường theo một giải thuật sau:

- Tạo 2 biến: `int page1, page2;`
- Tạo vòng lặp:
 

```

...
do
{
    setvisualpage(page1);    // Xem trang màn hình page1
    setactivepage(page2);    // Vẽ hình lên trang page2
    .....
    < Các lệnh vẽ hình lên trang page2 >;
    .....
    // Hoán vị 2 biến page1, page2
    Swap(page1, page2);
}
while <Điều kiện lặp>;

```

Ví dụ 1: Vẽ hai hình



Sau đó, viết chương trình thực hiện chuyển động của miệng cá.

```
#include <graphics.h>
#include <conio.h>
#include <dos.h>

int Xc,Yc,R;

void VeHinhCa1()
{
    setcolor(15);
    pieslice(Xc,Yc,30,330,R); //Vẽ bụng cá
    setcolor(0);
    circle(Xc + R/2,Yc - R/2,4); //Vẽ mắt cá
}

void VeHinhCa2()
{
    setcolor(15);
    pieslice(Xc,Yc,15,345,R); //Vẽ bụng cá
    setcolor(0);
    circle(Xc + R/2,Yc - R/2,4); //Vẽ mắt cá
}

void main()
{
    int gd=5,gm;
    initgraph(&gd,&gm,"D:\\TC\\BGI");
    Xc=getmaxx()/2;
    Yc=getmaxy()/2;
    R=50;
    int i=0;
    int page1=0, page2=1;
    do
    {
        setvisualpage(page1); //xem trang page1
        setactivepage(page2); //ve len trang page2
        i=1-i;
        if(i==0) VeHinhCa1();
        else VeHinhCa2();
        delay(200);
        //hoan doi 2 trang
        page1=1-page1;
    }
```

```
    page2=1-page2;
}
while (kbhit()==0);
closegraph();
}
```

**Ví dụ 2:** Viết chương trình tạo một dòng chữ chạy ngang qua màn hình.

```
#include <graphics.h>
#include <conio.h>
#include <dos.h>

void Run(char *s)
{
    int page=0;
    int x=getmaxx(), y=getmaxy()/3;
    setttextjustify(0,1);
    setwritemode(XOR_PUT);
    setactivepage(page);
    do
    {
        outtextxy(x,y,s);
        setvisualpage(page);
        page=1-page;
        setactivepage(page);
        // delay(10);
        outtextxy(x+1,y,s);
        x=x-1;
        if(x<-textwidth(s)) x=getmaxx();
    }
    while (kbhit()==0);
}

void main()
{
    int gd=5, gm;
    initgraph(&gd, &gm, "D:\\TC\\bgi");
    setcolor(14);
    setttextstyle(1,0,5);
    Run("Pham Anh Phuong");
    closegraph();
}
```

### 8.9.2. Lưu và di chuyển một vùng màn hình

Ta có thể lưu một vùng màn hình vào bộ nhớ rồi sau đó dán nó lên màn hình ở một vị trí khác.

Việc lưu một vùng màn hình vào bộ nhớ được thực hiện bằng hàm:

```
getimage(x1,y1,x2,y2:Integer; void far *Pointer);
```

trong đó Pointer là con trỏ trỏ đến vùng lưu nội dung của vùng (x1,y1,x2,y2).

Việc đăng ký một vùng nhớ động phải được khai báo dung lượng cần thiết. Dung lượng vùng nhớ được thực hiện bằng hàm:

```
unsigned imagesize(int x1,int y1,int x2,int y2);
```

Để hiện hình ảnh từ bộ nhớ ra màn hình, ta dùng hàm:

```
void putimage(int x,int y,void far *Pointer,int Mode);
```

trong đó:

(x,y): Tọa độ đỉnh trái hình chữ nhật mà ta muốn đưa ra.

Pointer: Con trỏ trỏ đến vùng lưu hình chữ nhật.

Mode: Hằng số chỉ phương thức hiện ra màn hình. Mode chứa một trong các hằng sau:

COPY\_PUT = 0: Xuất ra như đã lưu (phép MOV)

XOR\_PUT = 1: Phép XOR, xóa hình cũ nếu hai hình giao nhau.

OR\_PUT = 2: Phép OR, lấy cả hai hình nếu hai hình giao nhau.

AND\_PUT = 3: Phép AND, nếu hai hình giao nhau thì lấy phần chung.

NOT\_PUT = 4: Phép NOT, cho ra âm bản.

Về việc thực hiện được tiến hành như sau:

- Khai báo một biến con trỏ P;
- Đăng ký một vùng nhớ động do P quản lý bằng hàm  

```
farmalloc(imagesize(x1,y1,x2,y2));
```
- Lưu hình ảnh bằng hàm `getimage(x1,y1,x2,y2,P);`
- Xuất ra màn hình bằng hàm `putimage(x,y,P,Mode);`
- Xóa vùng nhớ động bằng hàm `farfree(size);`

Ví dụ: Viết chương trình vẽ chiếc đĩa bay chuyển động ngẫu nhiên trên màn hình.

```
#include <graphics.h>
#include <alloc.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
```

```
void ThietLapDoHoa()
{
    int gd=0,gm;
    initgraph(&gd,&gm,"D:\\TC\\bgi");
}
```

```
int r = 20, StartX = 100, StartY = 50;
```

```
void Move(int &x,int &y)
{
```

```

    int Step=random(2*r);
    if(Step%2==0) Step=-Step;
    x=x+Step;
    Step=random(r);
    if(Step%2==1) Step=-Step;
    y=y+Step;
}

void VeDiaBay()
{
    ellipse(StartX,StartY,0,360,r,r/3+2);
    ellipse(StartX,StartY-4,190,357,r,r/3);
    line(StartX+7,StartY-6,StartX+10,StartY-12);
    line(StartX-7,StartY-6,StartX-10,StartY-12);
    circle(StartX+10,StartY-12,2);
    circle(StartX-10,StartY-12,2);
}

void Play()
{
    unsigned int x1,y1,x2,y2,size;
    int x,y;
    void far *buff;
    VeDiaBay();
    //lấy vùng hình chữ nhật chứa hình đĩa bay
    x1=StartX - (r+1);
    y1=StartY - 14;
    x2=StartX + r + 1;
    y2=StartY + r/3 + 3;
    // Lưu và xóa ảnh
    size=imagesize(x1,y1,x2,y2);
    buff=farmalloc(size);
    getimage(x1,y1,x2,y2,buff);
    putimage(x1,y1,buff,XOR_PUT); //Xóa ảnh
    x=getmaxx()/2;
    y=getmaxy()/2;
    //Di chuyển đĩa bay
    do
    {
        putimage(x,y,buff,XOR_PUT); //Vẽ đĩa bay
        delay(200);
        putimage(x,y,buff,XOR_PUT); // Xóa đĩa bay
        Move(x,y);
    }
    while (kbhit()==0);
    farfree(buff); //Giải phóng vùng nhớ
}

```



```
void main()
{
    ThietLapDoHoa();
    Play();
    closegraph();
}
```

## BÀI TẬP

Bài tập 8.1: Viết chương trình vẽ bàn cờ quốc tế lên màn hình.

Bài tập 8.2: Viết chương trình vẽ một chiếc xe ô tô (theo hình dung của bạn) và cho nó chạy ngang qua màn hình.

**Gợi ý:**

Dùng kỹ thuật lật trang màn hình hoặc di chuyển vùng màn hình.

Bài tập 8.3: Viết chương trình vẽ lá cờ tổ quốc đang tung bay.

**Gợi ý:**

Dùng kỹ thuật lật trang màn hình.

Bài tập 8.4: Viết chương trình nhập vào n học sinh của một lớp học bao gồm 2 trường sau: Họ tên, điểm trung bình.

a/ Hãy thống kê số lượng học sinh giỏi, khá, trung bình và yếu.

b/ Vẽ biểu đồ thống kê số lượng học sinh giỏi, khá, trung bình và yếu theo 2 dạng: biểu đồ cột (column) và biểu đồ bánh tròn (Pie).

Bài tập 8.5: Viết chương trình để vẽ đồ thị của các hàm số sau:

a/  $y = ax^2 + bx + c$

b/  $y = ax^4 + bx^3 + cx^2 + dx + e$

c/  $y = \frac{ax + b}{cx + d}$

d/  $y = \frac{ax^2 + bx + c}{dx + e}$

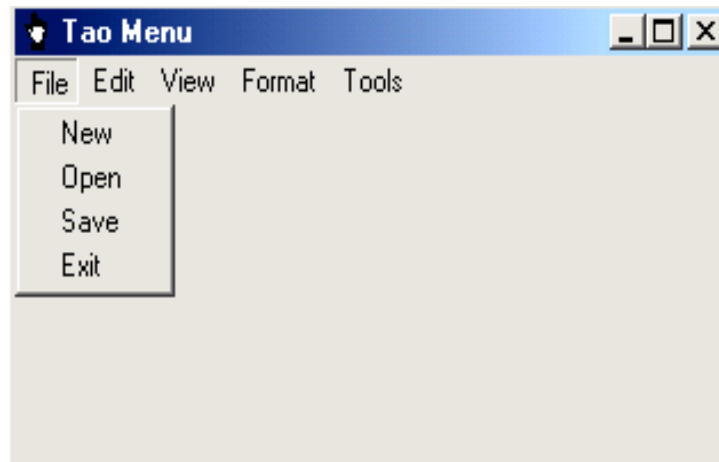
Bài tập 8.6: Viết chương trình vẽ cái đồng hồ đang hoạt động.

Bài tập 8.7: Viết chương trình mô phỏng chuyển động của trái đất xung quanh mặt trời và đồng thời chuyển động của mặt trăng xung quanh trái đất.

**Gợi ý:**

Dùng ma trận của phép quay.

Bài tập 8.8: Viết chương trình tạo Menu đồ họa giống như các Menu trong môi trường WINDOWS (xem hình).



Bài tập 8.9: Xây dựng một thư viện chứa tất cả các bài tập trong chương này.