Giáo trình

Lập Trình C Căn Bản

CHƯƠNG I: CÁC KHÁI NIỆM CƠ BẢN VỀ C

Đây là chương tiền đề đóng vai trò rất quan trọng khi bắt đầu học về lập trình C. Học viên cần nắm vững chương này để có thể học tiếp các chương sau.

I. Một số khái niệm về ngôn ngữ C:

1. Các kiểu dữ liệu cơ cở:

Kiểu (Type)	Từ khóa (Key word)	Định dạng (format)	Mô tả (description)
Số nguyên	int	%d	chỉ chứa các số nguyên (không có phần thập phân)
Số thực	double	%lf	gồm tất cả các loại số: nguyên, thập phân, …
Luận lý	Bool		Đúng (#0) hoặc Sai (0).
Kí tự	char	%c	Chỉ 1 phím trên bàn phím (Mã ASCII).

1. Các kiểu dữ liệu cơ sở

Kiểu dữ liệu có thể hiểu như là một tập hợp các giá trị và các phép toán thực hiện trên các giá trị đó. Có 4 kiểu dữ liệu cơ sở như sau:

- Kiểu số nguyên: Tập hợp các giá trị số nguyên (ví dụ như: int)
- Kiểu số thực: Tập hợp các giá trị số thực (ví dụ như: double)
- Kiểu luận lý: Tập hợp các giá trị đúng hoặc sai (ví dụ như: bool)
- Kiểu ký tự: Tập hợp các giá trị gồm 256 ký tự trong bảng mã ASCII. (ví dụ như: char)

a. Kiểu số nguyên:

- Các kiểu số nguyên có dấu

Kiểu (Type)	Kích thước (Byte)	Miền giá trị (Range)
char	1	–128 … +127
short	2	–32768 … +32767
int	4	–2147483648…+2147483.647
long	4	–2147483648…+2147483.647

- Các kiểu số nguyên không dấu

Kiểu (Type)	Kích thước (Byte)	Miền giá trị (Range)
unsigned char	1	0 … 255

unsigned short	2	0 … 65535
unsigned int	4	0 … 4294967295
unsigned long	4	0 … 4294967295

Chú ý: Tùy thuộc vào hệ điều hành sử dụng, kích thước của kiểu int và unsigned int có thể là 2 byte hoặc 4 byte, ví dụ Visual C++ 6.0 ở Window 7 thì int là 4 byte.

Hàm sizeof(kdl) trả về số byte của kdl (kiểu dữ liệu) được lưu trữ trong bộ nhớ.

Ví dụ: Chương trình sau in kích thước của các kiểu dữ liệu

```
#include "stdio.h"
void main()
{
    printf("Kích thước của các kiểu dữ liệu:\n");
    printf("\t char: %d\n", sizeof(char));
    printf("\t short: %d\n", sizeof(short));
    printf("\t int: %d\n", sizeof(int));
    printf("\t long: %d\n", sizeof(long));
    printf("\t float: %d\n", sizeof(float));
    printf("\t double: %d\n", sizeof(double));
}
```

b. Kiểu số thực

Kiểu (Type)	Kích thước (Byte)	Miền giá trị (Range)
float	4	3.4*E–38 … 3.4*E+38
double	8	1.7*E–308 … 1.7*E+308

Trong máy tính các số thực được viết theo hai dang: dạng thập phân và dạng mũ (dạng khoa học).

Ví dụ: 3.14, -245.3789, 234.0 (dạng thập phân)

123.456E-4, 0.12E3, -49.5E-2 (dạng mũ, với E là 10 mũ).

(123.456E-4 ~ $123.465*10^{-4}$, 0.12E3 ~ $0.12*10^{3}$, -49.5E-2 ~ -49.5*10-2)

Chú ý: Miền giá trị của kiểu float được hiểu theo nghĩa máy có thể lưu trữ được số có giá trị tuyệt đối từ 3.4e-38 đến 3.4e+38, Số có giá trị tuyệt đối nhỏ hơn 3.4E-38 được xem bằng 0. Miền giá trị của kiểu double được hiểu theo nghĩa tương tự.

c. Kiểu luận lý

Kiểu luận lý trong C là một kiểu nguyên đặc biệt. Một giá trị khác 0 được hiểu là đúng, một giá trị bằng 0 được hiểu là sai.

Ví dụ: 0 (false), 1 (true), 2 (true), 2.5 (true)

1 > 2 (false), 1 < 2 (true)

d. Kiểu ký tự

Có tên kiểu là char, miền giá trị là 256 ký tự trong bảng mã ASCII. Đây cũng chính là kiểu số nguyên vì kiểu này không lưu trực tiếp ký tự mà chỉ lưu mã ASCII của ký tự đó.

Ví dụ: Lưu số 65 tương đương với ký tự 'A'

Lưu số 97 tương đương với ký tự 'a'

2. Biến

Biến là một vùng trong bộ nhớ máy tính dùng để lưu trữ giá trị. Kiểu giá trị mà biến có thể lưu trữ gọi là kiểu biến. Giá trị mà biến đang lưu trữ có thể bị thay đổi nhiều lần trong suốt quá trình chương trình thi hành. Mọi biến trong chương trình cần được khai báo trước khi sử dụng.

Cú pháp:

```
<kiểu> <Tên biến>;
<kiểu> <Tên biến 1>, <Tên biến 2>;
```

Ví dụ: int i;
int j, k;
unsigned char dem;
double ketqua, delta;

3. Hằng

Hằng là đại lương mà giá trị của nó không thay đổi trong suốt quá trình thi hành chương trình. Mục đích chính của hằng số là làm cho chương trình dễ hiểu hơn và giúp hiệu chỉnh chương trình dễ dàng hơn. Cũng như biến, hằng cũng cần được khai báo trước khi sử dụng.

Cú pháp:

```
#define <Tên hằng> <Giá trị>
```

hoặc

const <kiểu> <Tên hằng> = <Giá trị>;

Ví dụ: #define MAX 100

#define PI 3.1416

const int MAX = 100;

const double PI = 3.1416;

4. Biểu thức

Biểu thức là một sự kết hợp giữa các toán tử (+, –, *, /, %….) và các toán hạng (hằng, biến, lời gọi hàm, …). Mỗi biểu thức sẽ có một giá trị và nói chung cái gì có giá trị đều được xem là biểu thức.

Ví dụ: p = (a + b + c) / 2 ; // Nửa chu vi

s = sqrt(p *(p-a) * (p-b) *(p-c)) ; //Diện tích tam giác

a. Toán tử gán

Dùng để gán giá trị cho biến.

Cú pháp:

<Biến> = <Biểu thức>;

Ví dụ: a = 12;

b = a;

c = a + b;

e = d = c; //Phép gán liên tiếp

b. Toán tử số học

- Toán tử 2 ngôi:

Kiểu (Type)	Từ khóa (Key word)	Kích thước (Size)	Phép toán (Operator)
Số nguyên	int	4 byte	+ , - , * , / (phần nguyên) , % (phần dư)
Số thực	double	8 byte	+ , - , * , /
Luận lý	Bool	1 byte	&& (and) , || (or) , ! (not)
Kí tự	char	1 byte	+ , - , * , / , % (là 1 dạng của số nguyên)

* Một số lưu ý:

- Khi tính toán giữa 2 miền giá trị giống nhau, kết quả trả về miền giá trị đó.

Ví dụ: a=7, b=2 là số nguyên. Nên kết quả a/b (7/2) = 3 ở số nguyên. → Lưu ý tránh mất dữ liệu.

- Khi tính toán giữa 2 miền giá trị khác nhau, kết quả trả về miền giá trị lớn hơn.

Ví dụ: a = 7.0 là số thực, b=2 là số nguyên. Nên kết quả a/b (7.0/2) = 3.5 ở số thực.

- Số thực: double → có đặc tả %lf

- Toán tử 1 ngôi: Chỉ có một toán hạng trong biểu thức.

Bao gồm: ++ (tăng 1 đơn vị) -- (giảm 1 đơn vị)

Chú ý: Cần phân biệt toán tử đặt trước/sau toán hạng. Khi toán tử đặt trước toán hạng thì toán hạng được tăng/giảm trước khi được dùng, còn khi toán tử đặt sau toán hạng thì toán hạng được dùng trước khi tăng/giảm.

```
Ví dụ: x = 10; y = x++;     // y = 10 và x = 11
       x = 10; y = ++x;     // x = 11 và y = 11
```

- Toán tử 2 ngôi: Có hai toán hạng trong biểu thức.

  Bao gồm: + − * / % (chia lấy phần dư) += -= *= /= %=

```
Ví dụ 1: int a = 9, b = 2,  x, y;
    double c = 9.0, z;
    x = a/b;     // x = 4 (chia nguyên)
    y = a%b;     // y = 1
    z = c/b;     // z = 4.5 (chia thực)
Ví dụ 2:  n = 10;
    n += 10;     // n = n + 10 , n nhận 20
    n -= 5;      // n = n – 5, n nhận 15
    n *= 2;      // n = n * 2, n nhận 30
    n /= 4;      //n = n / 4, n nhận 7
    n %= 2;      //n = n % 2, n nhận 1
```

c. Toán tử trên bít

Các toán tử trên bít cho phép tác động lên các bit của toán hạng (nguyên). Bao gồm:

& (and) | (or) ^ (xor) ~ (not hay lấy số bù 1)

>> (shift right), << (shift left)

&= |= ^= ~= >>= <<=

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

```
Ví dụ 1: int a = 5;          // 0000 0000 0000 0101
         int b = 6;          // 0000 0000 0000 0110
         int z1, z2, z3, z4, z5, z6;
         z1 = a & b;         // 0000 0000 0000 0100
         z2 = a | b;         // 0000 0000 0000 0111
```

```
z3 = a ^ b;      // 0000 0000 0000 0011
z4 = ~a;         // 1111 1111 1111 1010
z5 = a >> 2;     // 0000 0000 0000 0001
z6 = a << 2;     // 0000 0000 0001 0100
```

Ví dụ 2: Để xem xét bít thứ n của biến nguyên a bằng 0 hay khác 0 có thể dùng đoạn chương trình sau:

```
b = a >> n;
if(0x01 & b)
  printf("Khac khong\n");
else
  printf("Bang khong\n");
```

Ví dụ 3: Để trích byte thấp và byte cao của biến nguyên a có thể dùng đoạn chương trình sau:

```
b_thap = a && 0xff;
b_cao = a >> 8;
printf("Byte thap = %x\n", b_thap);
printf("Byte cao = %x\n", b_cao);
```

Ví dụ 4: Để xóa biến nguyên a có thể dùng lệnh

```
a = a^a; //hoặc a ^= a
```

d. Toán tử quan hệ

Toán tử quan hệ cho ra giá trị đúng hoặc sai. Bao gồm:

== (bằng) > < >= < <= != (khác)

Ví dụ:

```
b1 = (1 == 2); //b1 nhận giá trị false
b2 = (1 != 2); //b2 nhận giá trị true
b3 = (1 > 2); //b3 nhận giá trị false
b4 = (1 >= 2); //b4 nhận giá trị false
b5 = (1 < 2); //b5 nhận giá trị true
b6 = (1 <= 2); //b6 nhận giá trị true
```

e. Toán tử luận lý

Toán tử quan hệ cho ra giá trị đúng hoặc sai. Bao gồm:

&& (and) || (or) ! (not)

a	b	a && b	a || b	!a
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0

1	1	1	1	0

Ví dụ: b1 = (1 > 2) && (3 > 4); //b1 nhận giá trị false

b2 = (1 < 2) || (3 > 4); //b1 nhận giá trị true

b3 = !(1 > 2); //b1 nhận giá trị true

f. Toán tử biểu thức điều kiện

Đây là toán tử 3 ngôi (gồm có 3 toán hạng), ký hiệu ?:

Cú pháp:

<biểu thức 1> ? <biểu thức 2> : <biểu thức 3>

biểu thức này cho kết quả là <biểu thức 2> nếu <biểu thức 1> đúng, ngược lại cho kết quả là <biểu thức 3>

Ví dụ: int a = 10, b = 20;

max = (a > b) ? a : b; //20

min = (a < b) ? a : b; // 10

g. Toán tử chuyển kiểu

Toán tử chuyển kiểu dùng để chuyển một kiểu dữ liệu bất kỳ sang một kiểu dữ liệu mong muốn.

Cú pháp:

<Kiểu> Biến

<Kiểu> là kiểu dữ liệu mong muốn.

Ví dụ: int a = 9;

int b = 2;

int c = a / b; //4

double d = (double)a / b; //4.5

h. Toán tử phẩy

Các biểu thức con đặt cách nhau bằng dấu phẩy và lần lượt được tính từ trái sang phải. Biểu thức mới nhận được là giá trị của biểu thức bên phải cuối cùng.

Ví dụ: x = (a++, b = b + 2); ⇔ a++; b = b + 2; x = b;

i. Độ ưu tiên của các toán tử

Các toán tử có độ ưu tiên khác nhau, điều này có nghĩa là trong cùng một biểu thức một số toán tử này được thực hiện trước một số các toán tử khác. Thứ tự ưu tiên của các toán tử được trình bày trong bảng sau:

Toán tử	Độ ưu tiên

() [] -> .	→
! ~ & * - ++ -- (type) & sizeof	←
* / %	→
+ -	→
<< >>	→
< <= > >=	→
== !=	→
&	→
|	→
^	→
&&	→
||	→
?:	←
= += -= *= /= %= &= … <<= >>= &= ^ = |=	←
,	→

Chú ý:

- Các toán tử trên một dòng có cùng thứ tự ưu tiên, các toán tử ở dòng trên có độ ưu tiên cao hơn các toán tử ở dòng dưới.
- Đối với các toán tử có cùng độ ưu tiên thì trình tự tính toán có thể từ trái qua phải (→) hoặc ngược lại (←).
- Để viết biểu thức một cách chính xác nên sử dụng các dấu ngoặc tròn.

Ví dụ: *--a = *(--a) (phải qua trái)

a > b ? a : x ? y : z = a > b ? a : (x ? y : z) (phải qua trái)

8/4*6 = (8/4)*6 (trái qua phải)

5. Các hàm thư viện C chuẩn

Thư viện C chuẩn tập hợp các hằng, các hàm đã được định nghĩa sẵn trong những tập tin tiêu đề *.h (header file). Để sử dụng được các hằng, các hàm trong thư viện này, chúng ta phải đặt các chỉ thị #**include** ở đầu tập tin chương trình.

a. Hàm toán học

Các hàm toán học được định nghĩa sẵn trong tập tin tiêu đề **math.h**, do đó để sử dụng được các hàm này cần thêm vào đầu chương trình chỉ thị **#include <math.h>**.

Tên hàm	Trả về
ceil(x)	Phần nguyên của x cộng 1
floor(x)	Phần nguyên của x
abs(x)	Trị tuyệt đối của x
fabs(x)	Trị tuyệt đối của x
sqrt(x)	Căn bậc hai của x
pow(x, y)	x lũy thừa y
exp(x)	exponential của x
log(x)	Logarithm tự nhiên của x

Ví dụ: double x = 3.7;

int y = ceil(x); //y = 4

int z = floor(x); //z = 3

int a = -10; int b = abs(a); //b = 10;

double u = -20.5; double v = fabs(u); //v = 20.5

double t = pow(2.0, 10.0); // t = 2^{10} = 1024

double p = exp(log(100.0)/3) ; //y = $e^{(1/3)\log(100)} = 100^{1/3}$

double q = exp(100.0*log(3.0)) ; //q = $e^{100\log(3)} = 3^{100}$

b. Hàm xuất dữ liệu

Các hàm nhập/xuất được định nghĩa sẵn trong tập tin tiêu đề **stdio.h**. Để sử dụng được các hàm này cần thêm vào đầu chương trình chỉ thị **#include <stdio.h>**

Cú pháp:

printf(<chuỗi định dạng>[, <đs1>, <đs2>, ...]);

- <chuỗi định dạng> là cách trình bày thông tin xuất và được đặt trong cặp nháy kép " ", gồm 3 loại:
  - + Văn bản thường (literal text): Được xuất y hệt như lúc gõ trong chuỗi định dạng.
  - + Ký tự điều khiển (escape sequence): Gồm dấu \ và một trong các ký tự như trong bảng sau:

Ký tự điều khiển	Ý nghĩa

\a	Tiếng chuông
\n	Xuống dòng
\t	Dấu Tab
\\	In dấu \
\"	In dấu "

+ Đặc tả (conversion specifier): Gồm dấu % và một ký tự, nó xác định kiểu của biến/giá trị muốn xuất.

Đặc tả	**Ý nghĩa**	
%c	Ký tự	char
%d, %ld	Số nguyên có dấu	char, int, short, long
%f, %lf	Số thực	float, double
%s	Chuỗi ký tự	char [], char*
%u	Số nguyên không dấu	unsigned int/short/long

– Các đối số <đs1>, <đs2>, . . . chính là các biến/giá trị muốn xuất, được liệt kê theo thứ tự cách nhau dấu phẩy.

```
Ví dụ: int a = 10, b = 20;
        printf("%d", a);            //10
        printf("%d", b);            //20
        printf("%d %d", a, b);      //10 20
        double x = 15.06;
        printf("%lf", x);           //15.060000
        printf("%0.2lf", x);        //15.06
        printf("%lf", 1.0/3);       //0.333333
        printf("%0.1lf", 1.0/3);    //0.3
```

c. Hàm nhập dữ liệu

Cú pháp:

scanf(<chuỗi định dạng>[, <đs1>, <đs1>, ...]);

– <chuỗi định dạng> giống định dạng xuất nhưng chỉ có các đặc tả.

– Các đối số <đs1>, <đs2>, . . . là tên các biến sẽ chứa giá trị nhập và được đặt trước bởi dấu & (toán tử địa chỉ)

```
Ví dụ: int a, b;
  scanf("%d", &a);  // Nhập giá trị cho biến a
  scanf("%d", &b);
  scanf("%d%d", &a, &b); // Nhập giá trị cho hai biến a và b
```

```
double c, d;
scanf("%lf", &c);  // Nhập giá trị cho biến c
scanf("%lf", &d);
scanf("%lf%lf", &c, &d); // Nhập giá trị cho hai biến c và d
char e, f;
scanf("%c", &e);  // Nhập giá trị cho biến e
scanf("%c", &lf);
scanf("%c%c", &e, &f); // Nhập giá trị cho hai biến e và f
```

CHƯƠNG 3:

CÁC CÂU LỆNH RẼ NHÁNH

Các câu lệnh rẽ nhánh được sử dụng trong trường hợp việc tính toán trong chương trình có phụ thuộc vào giá trị của một biểu thức. Khi giá trị của biểu thức là đúng thì thực hiện một số lệnh nào đó và nếu giá trị của biểu thức là sai thì lại thực hiện một số lệnh khác. Ngôn ngữ lập trình C có cung cấp 2 câu lệnh rẽ nhánh: **if** và **switch**.

1. Câu lệnh if

a. Câu lệnh if (thiếu)

Cú pháp:

```
if (<BT>)
    <Lệnh >;
```

<Lệnh> : Lệnh đơn hoặc lệnh phức.

Hoạt động:

Đầu tiên máy sẽ tính giá trị của biểu thức <BT>. Nếu giá trị của <BT> là đúng thì máy sẽ thực hiện <Lệnh>, ngược lại nếu giá trị của <BT> là sai thì máy bỏ qua <Lệnh>

b. Câu lệnh if (đủ)

Cú pháp:

```
if (<BT>)
    <Lệnh 1>;
else
    <Lệnh 2>;
```

<Lệnh 1>, <Lệnh 2> : Lệnh đơn hoặc lệnh phức.

Hoạt động:

Đầu tiên máy sẽ tính giá trị của biểu thức <BT>. Nếu giá trị của <BT> là đúng thì máy sẽ thực hiện <Lệnh 1>, ngược lại nếu giá trị của <BT> là sai thì máy sẽ thực hiện <Lệnh 2>.

c. Một số lưu ý

- Câu lệnh if và câu lệnh if…else là một câu lệnh đơn.

- Câu lệnh if...else có thể lồng vào nhau và else sẽ tương ứng với if gần nó nhất.
- Nên dùng else để loại trừ trường hợp.

Ví dụ 1: Nhập vào hai số nguyên và in ra số lớn nhất

Cách 1: Sử dụng câu lệnh if thiếu

```
void main()
{
    int a, b, max;
    printf("Nhap hai so nguyen:");
    scanf("%d%d", &a, &b);
    max = a;
    if(b > max)
    max = b;
    printf("max(%d, %d) = %d\n", a, b, max);
}
```

Cách 2: Sử dụng câu lệnh if đủ

```
void main()
{
    int a, b, max;
    printf("Nhap hai so nguyen:");
    scanf("%d%d", &a, &b);
    if(a >b)
        max = a;
    else
        max = b;
    printf("max(%d, %d) = %d\n", a, b, max);
}
```

Ví dụ 2: Nhập vào ba số nguyên và in ra số lớn nhất

Cách 1: Sử dụng câu lệnh if thiếu

```
void main()
{
    int a, b, c, max;
    printf("Nhap ba so nguyen:");
    scanf("%d%d%d", &a, &b, &c);
    max = a;
    if(b > max)
        max = b;
    if(c > max)
        max = c;
    printf("max(%d, %d, %d) = %d\n", a, b, c, max);
}
```

Cách 2: Sử dụng câu lệnh if thiếu và if đủ

```
void main()
{
    int a, b, c, max;
    printf("Nhap ba so nguyen:");
    scanf("%d%d%d", &a, &b, &c);
    // Tìm max(a, b)
    if(a >b)
        max = a;
    else
        max = b;
    // Tìm max(max, c)
    if(c > max)
```

```
        max = c;
    printf("max(%d, %d, %d) = %d\n", a, b, c, max);
}
```

Ví dụ 3**:** Nhập vào điểm của một học sinh (giả sử điểm là một giá trị nguyên). Tùy theo điểm học sinh hãy phân loại học sinh theo tiêu chuẩn như sau:

0 ≤ điểm ≤ 4: Yếu

5 ≤ điểm ≤ 6: Trung bình

7 ≤ điểm ≤ 8: Khá

9 ≤ điểm ≤ 10: Giỏi

Ở đây ta sử dụng các câu lệnh if...else lồng nhau

```
void main()
{
    int diem;
    printf("Nhap diem:");
    scanf("%d", &diem);
    if(diem >= 0 && diem <= 4)
        printf("Yeu\n");
    else if(diem >= 5 && diem <= 6)
        printf("Trung binh\n");
    else if(diem >= 7 && diem <= 8)
        printf("Kha\n");
    else if(diem >= 9 && diem <= 10)
        printf("Gioi\n");
    else
        printf("Diem nhap không hop le\n");
}
```

2. Câu lệnh switch

a. Câu lệnh switch (thiếu)

Cú pháp:

```
switch (<BT Nguyên>)
{
    case <Hằng 1>: <Lệnh 1>; break;
    case < Hằng 2>: <Lệnh 2>; break;
    …
    case < Hằng n> : <Lệnh n>; break;
}
```

Hoạt động:

Trước tiên máy sẽ tính giá trị của <BT Nguyên>. Nếu giá trị này bằng <Hằng i> thì máy sẽ nhảy tới câu lệnh có nhãn case <Hằng i> (i = 1, 2, . . ., n) tương ứng. Nếu giá trị <BT Nguyên> khác tất cả các <Hằng i> thì máy sẽ nhảy ra khỏi câu lệnh switch mà không làm gì. Máy cũng ra khỏi câu lệnh switch nếu gặp câu lệnh break.

b. Câu lệnh switch (đủ)

Cú pháp:

```
switch (<BT Nguyên>)
{
    case < Hằng 1>: <Lệnh 1>; break;
    case < Hằng 2> :<Lệnh 2>; break;
    …
    case < Hằng n> :<Lệnh n>; break;
    default: <Lệnh n + 1>;
}
```

Hoạt động:

Tương tự như câu lệnh switch (thiếu) chỉ khác một điều là nếu giá trị của <BT Nguyên> khác tất cả các <Hằng i> thì máy sẽ nhảy tới câu lệnh có nhãn default.

c. Một số lưu ý:

- Câu lệnh switch là một câu lệnh đơn và có thể lồng nhau.
- Các <Hằng i> trong mỗi trường hợp phải khác nhau.
- switch sẽ nhảy đến case tương ứng và thực hiện cho đến khi nào gặp câu lệnh break hoặc cuối switch sẽ kết thúc.
- Tận dụng tính chất khi bỏ break

Ví dụ 1: Nhập vào một số nguyên dương n (1 ≤ n ≤ 7). Tùy theo giá trị n, hãy in ra các từ Sunday, Monday, . . ., Saturday tương ứng.

Cách 1: Sử dụng câu lệnh switch thiếu

```
void main()
{
    int n;
    printf("Nhap mot so nguyen:");
    scanf("%d", &n);
    if(n >= 1 && n <= 7) {
        switch(n) {
            case 1: printf("Sunday"); break;
            case 2: printf("Monday"); break;
            case 3: printf("Tuesday"); break;
            case 4: printf("Wednesday"); break;
            case 5: printf("Thursday"); break;
            case 6: printf("Friday"); break;
            case 7: printf("Saturday"); break;
        }
    }
    else
        printf("Du lieu nhap khong hop le\n");
}
```

Cách 2: Sử dụng câu lệnh switch đủ

```
void main()
{
    int n;
    printf("Nhap mot so nguyen:");      scanf("%d", &n);
    switch(n)
    {
        case 1: printf("Sunday"); break;
        case 2: printf("Monday"); break;
        case 3: printf("Tuesday"); break;
        case 4: printf("Wednesday"); break;
        case 5: printf("Thursday"); break;
        case 6: printf("Friday"); break;
        case 7: printf("Saturday"); break;
        default: printf("Du lieu nhap khong hop le\n");
    }
}
```

Ví dụ 2: làm lại ví dụ phân loại học sinh ở trên bằng câu lệnh switch.

```
void main()
{
    int diem;
    printf("Nhap diem:");
    scanf("%d", &diem);
    switch(diem) {
        case 0:
        case 1:
        case 2:
```

```
        case 3:
        case 4: printf("Yeu"); break;
        case 5:
        case 6: printf("Trung bình"); break;
        case 7:
        case 8: printf("Kha"); break;
        case 9:
        case 10: printf("Gioi"); break;
        default: printf("Diem nhap khong hop le \n");
    }

}
```

CHƯƠNG 4:

CÁC CÂU LỆNH LẶP

Trong trường hợp một số đoạn chương trình gần như hoàn toàn giống nhau được lặp đi lặp lại nhiều lần thì người lập trình có thể sử dụng các câu lệnh lặp trong ngôn ngữ lập trình. Ngôn ngữ lập trình C có cung cấp 3 câu lệnh lặp: **for**, **while** và **do** … **while**.

1. Câu lệnh for

Cú pháp:

```
for (<BT1>; <BT2>; <BT3>)
        <Lệnh>;
```

<Lệnh>: có thể là lệnh đơn hoặc là lệnh phức.

Hoạt động:

Bước 1: Tính giá trị của biểu thức <BT1>.

Bước 2: Tính giá trị của biểu thức <BT2>.

Bước 3: Nếu giá trị của <BT2> là đúng thì máy sẽ thực hiện <Lệnh> và nhảy đến bước 4, ngược lại nếu giá trị của <BT2> là sai thì máy sẽ ra khỏi câu lệnh for.

Bước 4: Tính giá trị biểu thức <BT3> và quay về bước 2.

Một số Lưu ý:

- Câu lệnh for là một câu lệnh đơn và có thể lồng nhau.
- Trong câu lệnh for, có thể sẽ không có phần <BT1>, <BT2> và <BT3>. Khi không có phần <BT2> thì nó được xem là đúng.
- <BT1>, <BT2>, <BT3> có thể bao gồm nhiều biểu thức con phân cách nhau bởi dấu phẩy. Khi <BT2> gồm nhiều biểu thức thì tính đúng sai của nó được xem là tính đúng sai của biểu thức cuối cùng.

Ví dụ: Nhập vào một số nguyên dương n, hãy tính và in ra tổng

s = 1 + 2 + … + n

```
void main()
{
```

```
    int n, i, s ;
    printf("Nhap mot so nguyen:");
    scanf("%d", &n);
    if(n <= 0)
        printf("Du lieu nhap khong hop le\n");
    else
    {
        s = 0;
        for(i = 1; i <= n; i++)
            s += i;
        printf("Tong s = %d\n", s);
    }
}
```

Chú ý: Đoạn mã trong phần else ở trên cũng có thể viết cách khác như sau:

```
for(s = 0, i = 1; i <= n; i++)
        s += i;
```

2. Câu lệnh while

Cú pháp:

```
while (<BT>)
        <Lệnh>;
```

<Lệnh>: có thể là lệnh đơn hoặc là lệnh phức.

Hoạt động:

Bước 1: Tính giá trị của biểu thức <BT>.

Bước 2: Nếu giá trị của <BT> là đúng thì máy sẽ thực hiện <Lệnh> và quay lại bước 1, ngược lại nếu giá trị của <BT> là sai thì máy sẽ ra khỏi câu lệnh while.

Một số Lưu ý:

- Câu lệnh while là một câu lệnh đơn và có thể lồng nhau.
- <Lệnh> trong câu lệnh câu lệnh while có thể không thực hiện lần nào nếu như biểu thức <BT> ngay từ lần đầu đã là sai.

Ví dụ: Làm lại ví dụ trên bằng câu lệnh while

```
void main()
{
    int n, i, s;
    printf("Nhap mot so nguyen:");
    scanf("%d", &n);
    if(n <= 0)
        printf("Du lieu nhap khong hop le\n");
    else {
        s = 0;
        i = 1;
        while(i <= n) {
            s += i;
            i++;
        }
        printf("Tong s = %d\n", s);
    }
}
```

3. Câu lệnh do ... while

Cú pháp:

```
do
    <Lệnh>;
while (<BT>);
```

<Lệnh>: có thể là lệnh đơn hoặc là lệnh phức.

Hoạt động:

Bước 1: Thực hiện <Lệnh> .

Bước 2: Tính giá trị của biểu thức <BT>. Nếu giá trị của <BT> là đúng thì máy sẽ quay lại bước 1, ngược lại nếu giá trị của <BT> là sai thì máy sẽ ra khỏi câu lệnh do while.

Một số Lưu ý:

- Câu lệnh do… while là một câu lệnh đơn và có thể lồng nhau.
- <Lệnh> trong câu lệnh do… while sẽ được thực hiện ít nhất 1 lần do biểu thức <BT> được kiểm tra ở cuối.

Ví dụ: Làm lại ví dụ trên nhưng cho phép người sử dụng chạy chương trình nhiều lần, mỗi lần kết thúc tính toán thì nhắc người sử dụng "Tiếp tục hay ngừng chương trình ?".

```
void main()
{
    char traloi;
    int n, i, s
    do {
        printf("Nhap mot so nguyen:");
        scanf("%d", &n);
        if(n <= 0)
            printf("Du lieu nhap khong hop le\n");
        else {
            s = 0;
            i =1;
            while(i <= n)  {
                s += i;
                i++;
            }
            printf("Tong s = %d\n", s);
        }
```

```
        printf("Tiep tuc hay ngung chuong trinh ? (Y/N)");
        fflush(stdin);
        scanf("%c", &traloi);
    } while(traloi == 'Y' || traloi == 'y');
}
```

4. Câu lệnh break

Cú pháp:

```
break;
```

Hoạt động:

Khi gặp câu lệnh này trong các câu lệnh lặp máy sẽ ra khỏi các câu lệnh lặp. Khi có nhiều câu lệnh lặp lồng nhau thì câu lệnh break sẽ nhảy ra khỏi câu lệnh lặp sâu nhất chứa nó.

Ví dụ: Làm lại ví dụ trên bằng câu lệnh while và break

```
void main()
{
    int n, i, s;
    printf("Nhap mot so nguyen:");
    scanf("%d", &n);
    if(n <= 0)
        printf("Du lieu nhap khong hop le\n");
    else {
        s = 0;
        i = 1;
        while(1) {   //Biểu thức trong ngoặc luôn luôn đúng
            s += i;
            i++;
            if(i > n) break;
```

```
        }
        printf("Tong s = %d\n", s);
    }}
```

Chú ý: Câu lệnh while(1) tương đương câu lệnh for(; ;)

5. Câu lệnh continue

Cú pháp:

```
continue;
```

Hoạt động:

Khi gặp câu lệnh này trong các câu lệnh lặp máy sẽ bỏ qua phần còn lại trong vòng lặp và tiếp tục thực hiện vòng lặp tiếp theo. Đối với câu lệnh **for** máy sẽ tính lại giá trị của biểu thức <BT3> và quay lại bước 1, đối với câu lệnh **while**, **do** … **while** máy sẽ tính lại giá trị của biểu thức <BT> và quay lại bước 1.

Ví dụ:

```
void main()
{
    int n;
    for(; ;) {
        printf("Nhap mot so nguyen:");
        scanf("%d", &n);
        if(n % 2 == 0) continue;
        else if(n % 3 == 0) break;
        printf("%d khong chia het cho 2 va 3\n", n);
    }
    printf("Da tim dược mot so chia het cho 3, do la %d\n", n);
}
```

CHƯƠNG 5: HÀM

1. Khái niệm

Hàm là một đoạn chương trình có tên, có thể có dữ liệu đầu vào/đầu ra nhằm thực hiện trọn vẹn một công việc nhất định. Chức năng của hàm là nhằm chia cắt các công việc lớn thành các công việc nhỏ hơn.

2. Khai báo hàm

Bên cạnh các hàm thư viện chuẩn C, chúng ta cũng có thể khai báo các hàm của riêng mình. Những hàm này thường được gọi là **hàm do người dùng định nghĩa**.

Cú pháp:

```
<Kiểu hàm> <Tên hàm> (<Danh sách đối số>)
{
    <Khai báo thêm các biến>
    <Các câu lệnh>
    return <biểu thức>
}
```

Trong đó:

- <Kiểu hàm> có thể là một kiểu dữ liệu nào đó (char, int, float, double, . . .) hoặc là kiểu void.
- <Tên hàm> bắt buộc phải có đối với mọi hàm.
- <Danh sách đối số> có thể có hoặc không tùy thuộc ta định dùng hàm đó làm gì.
- Phần bao trong cặp dấu ngoặc {} được gọi là thân hàm, dấu {} là bắt buộc đối với mọi hàm.
- Khi cần thêm một số biến thì phải <Khai báo thêm các biến>. Các biến này gọi là **biến cục bộ** vì chỉ riêng hàm này sử dụng mà thôi.
- <Các câu lệnh> là phần thực hiện nhiệm vụ của hàm.
- Câu lệnh return <biểu thức> có thể có hoặc không, khi kiểu hàm không phải là void thì nó bắt buộc phải có. Câu lệnh này có nhiệm vụ trả về một giá trị cho nơi gọi hàm.

3. Lời gọi hàm

Hàm được sử dụng thông qua lời gọi tới nó. Số và kiểu tham số thực trong lời gọi hàm phải tương ứng với số và kiểu đối số trong khai báo hàm.

Cú pháp:

<Tên hàm> (<Danh sách tham số thực>)

4. Nguyên tắc hoạt động của hàm

Khi gặp một lời gọi hàm ở một chỗ nào đó trong chương trình, thì máy sẽ tạm rời chỗ đó và chuyển đến chỗ có khai báo hàm tương ứng. Quá trình đó sẽ diễn ra theo trình tự 4 bước như sau:

- Bước 1: Cấp phát bộ nhớ cho các đối số và các biến cục bộ.
- Bước 2: Truyền giá trị của các tham số thực cho các đối số tương ứng.
- Bước 3: Thực hiện các câu lệnh trong thân hàm.
- Bước 4: Khi gặp câu lệnh return hoặc dấu } cuối cùng của thân hàm thì máy sẽ giải phóng các đối số, các biến cục bộ và thoát khỏi hàm.

Ví dụ: Hàm sau đây trả về giá trị lớn nhất của hai số thực

```
double Max(double x, double y)
{
	double ret; //Khai báo biến cục bộ ret
	if(x > y)
		ret = x;
	else
		ret = y;
	return ret;
}
void main()
{
    double a = 3.6, b = 7.2;
    double kq;
    kq = Max(a, b);  //Lời gọi hàm Max() với hai tham số thực a và b
    printf("max(%0.2lf, %0.2lf) = %0.2lf\n", a, b, kq);
}
```

5. Biến cục bộ và biến toàn cục

- Biến cục bộ là biến được khai báo bên trong một hàm. Biến này chỉ có thể được truy cập bên trong hàm mà nó khai báo và chỉ tồn tại khi hàm được gọi tới, nó sẽ biến mất khi hàm được gọi thực hiện xong.
- Biến toàn cục là biến được khai báo bên ngoài các hàm. Biến này có thể được truy cập ở mọi nơi trong chương trình và tồn tại trong suốt thời gian chương trình thực hiện.

Ví dụ:

```
double a = 3.6, b = 7.2; //Khai báo hai biến toàn cục a và b
//Hàm trả về giá trị lớn nhất của hai số thực (hàm không có đối số)
double Max()
{
    double ret; //Khai báo biến cục bộ ret
    if(a > b)
        ret = a;
    else
        ret = b;
    return ret;
}
void main()
{
    double kq;
    kq = Max();  //Lời gọi hàm Max() không có tham số thực
    printf("So lon nhat của %0.2lf va %0.2lf la %0.2lf\n", a, b, kq);
}
```

Chú ý: Cả hàm Max() và hàm main() đều có thể truy cập được hai biến toàn cục a và b.

6. Hàm kiểu void

- Khi một hàm không trả về một giá trị nào, hàm đó được gọi là hàm kiểu void.

Ví dụ:

```
void Xuat(int x, int y)
```

```
{
    printf("%d, %d\n", x, y);
}
void main()
{
    int i;
    for(i = 1; i <= 10; i++)
            Xuat (i, 2*i);     //Lời gọi hàm với hai tham số thực i và 2*i
}
```

7. Con trỏ

Kiến thức về con trỏ sẽ được sử dụng trong mục truyền tham số cho hàm theo địa chỉ.

a. Khái niệm

Con trỏ là một kiểu dữ liệu đặc biệt, được sử dụng để chứa địa chỉ của các biến (trỏ tới các biến) trong bộ nhớ. Ứng với mỗi kiểu dữ liệu sẽ có con trỏ tương ứng.

b. Khai báo

- Con trỏ có kiểu: Dùng để chứa địa chỉ của biến có kiểu tương ứng với kiểu con trỏ.
  Cú pháp:

  <Kiểu> *<Tên biến>;

- Con trỏ không kiểu: Dùng để chứa địa chỉ của biến có kiểu bất kỳ.
  Cú pháp:

  void *<Tên biến>;

```
Ví dụ:  int *p1; //Con trỏ kiểu int dùng để trỏ tới biến kiểu int
double *p2; //Con trỏ kiểu double dùng để trỏ tới biến kiểu double
char *p3; //Con trỏ kiểu char dùng để trỏ tới biến kiểu char
void *p4; //Con trỏ kiểu void dùng để trỏ tới biến kiểu bất kỳ
```

c. Toán tử nội dung (*) và toán tử địa chỉ (&)

Vì con trỏ chứa địa chỉ của biến nên có thể truy cập đến biến gián tiếp thông qua con trỏ. Để truy cập nội dung của biến mà con trỏ đang trỏ tới ta sử dụng toán tử *, để gán địa chỉ của một biến cho con trỏ ta sử dụng toán tử &.

Ví dụ:

```
void main()
{
    int *p1, *p2;
    int a = 10;
    p1 = &a; //p1 trỏ tới a
    printf("Giá trị biến a: %d\n", *p1);
    (*p1)++;
    printf("Giá trị biến a: %d\n", a);
    p2 = p1; //p1 và p2 cùng trỏ tới a
    printf("Giá trị biến a: %d\n", *p2);
}
```

d. Hằng NULL

Hằng **NULL** là một giá trị hằng đặc biệt để gán cho bất kỳ con trỏ nào. Nó được dùng để bảo rằng con trỏ không trỏ vào đâu cả.

Ví dụ:

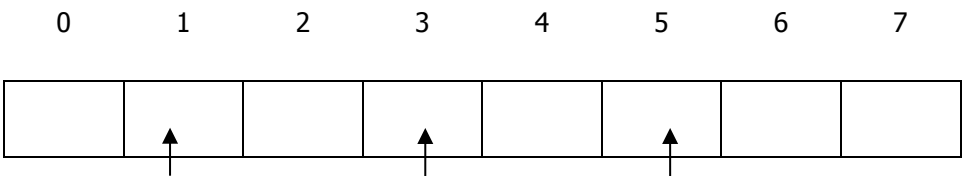
```
int *p;
p = NULL; //p Không trỏ vào đâu cả
```

e. Các phép toán trên con trỏ

Giả sử p là con trỏ đang trỏ tới một ô nhớ nào đó và k là số nguyên dương.

- **Phép cộng con trỏ với số nguyên**: Phép toán p + k sẽ tạo ra một con trỏ tới ô nhớ sau ô nhớ trước đó k ô nhớ.
- **Phép trừ con trỏ với số nguyên**: Phép toán p - k sẽ tạo ra một con trỏ tới ô nhớ trước ô nhớ trước đó k ô nhớ.

  (Kích thước của mỗi ô nhớ bằng kích thước của biến mà p đang trỏ tới).



p - 2 p p + 2

8. Các cách truyền tham số cho hàm

Mọi dữ liệu đầu vào/đầu ra của hàm đều có thể được truyền thông qua các tham số của hàm

a. Truyền tham số theo giá trị

Trong cách truyền này, mọi thay đổi về mặt giá trị của các đối số trong khai báo hàm đều không làm thay đổi giá trị của các tham số thực tương ứng trong lời gọi hàm, vì thực chất lúc này hàm chỉ thao tác trên các bản sao của các tham số thực chứ không làm việc trực tiếp trên các tham số thực.

Ví dụ:

```
//Hàm hoán vị hai số thực
void HoanVi(double a, double b)
{
    double tam;          //khai báo biến cục bộ tam
    tam  = a;
    a = b;
    b = tam;
    printf("%0.2f, %0.2f\n", a, b); //a và b có thay đổi
}
void main()
{
    double x = 3.5, y = 7.6;
    HoanVi(x, y); //Lời gọi hàm HoanVi với hai tham số thực x và y
    printf("%0.2lf, %0.2lf\n", x, y); //x và y không thay đổi
}
```

b. Truyền tham số theo địa chỉ

Trong cách truyền này, mọi thay đổi về mặt giá trị của các đối số trong khai báo hàm đều làm thay đổi giá trị của các tham số thực tương ứng trong lời gọi hàm, vì thực chất lúc này hàm làm

việc trực tiếp trên vùng nhớ của tham số thực. Khi sử dụng cách truyền này thì các đối số tương ứng trong khai báo hàm phải là các biến con trỏ.

Ví dụ:

```
//Hàm hoán vị hai số thực (có hai đối số là con trỏ)
void HoanVi(double *a, double *b)
{
    double tam;
    tam = *a;
    *a = *b;
    *b = tam;
}
void main()
{
    double x = 3.5, y = 7.6;
    HoanVi(&x, &y); //Gọi hàm HoanVi với hai tham số thực &x và &y
    printf("%0.2lf, %0.2lf\n", x, y); //x và y có thay đổi
}
```

c. Một số lưu ý

- Các đối số trong khai báo hàm có thể chia làm hai loại: loại thứ nhất tương ứng với các **dữ liệu đầu vào** của hàm gọi là các **đối vào**, loại thứ hai tương ứng với các **dữ liệu đầu ra** của hàm gọi là các **đối ra**. Các đối ra phải là con trỏ.
- Nên dùng hàm có giá trị trả về đối với các hàm cần đưa ra một giá trị duy nhất hoặc đối với các hàm mà lời gọi tới nó cần xuất hiện trong một biểu thức.
- Nên dùng hàm kiểu void đối với các hàm có tính chất thực hiện một hành động nào đó hoặc có tính chất tính toán nhưng đưa ra cùng lúc nhiều giá trị.

9. Nguyên mẫu hàm

Về nguyên tắc khi gọi một hàm thì hàm đó phải được khai báo trước, nếu không chương trình sẽ báo lỗi. Tuy nhiên cũng có thể gọi một hàm chưa được khai báo trước bằng cách khai báo trước nguyên mẫu hàm. Nguyên mẫu hàm thực chất là dòng đầu của hàm có thêm dấu chấm phẩy.

Ví dụ:

```
void HoanVi(double *a, double *b); //Khai báo nguyên mẫu hàm

void main()

{

    double x = 3.5, y = 7.6;

    HoanVi(&x, &y); /*Gọi HoanVi() nhưng hàm này được khai báo sau*/     printf("%0.2f, %0.2f\n", x, y);

}

void HoanVi(double *a, double *b)

{

    double tam;

    tam = *a;

    *a = *b;

    *b = tam;

}
```

CHƯƠNG 6:

MẢNG MỘT CHIỀU

1. Khái niệm mảng

Mảng là một tập hợp nhiều phần tử (biến) có cùng một kiểu và chung một tên được sắp xếp liên tiếp nhau trong bộ nhớ. Mỗi phần tử của mảng có một đại lượng xác định vị trí tương đối của phần tử đó so với các phần tử khác trong mảng, gọi là *chỉ số*.

2. Khai báo mảng một chiều

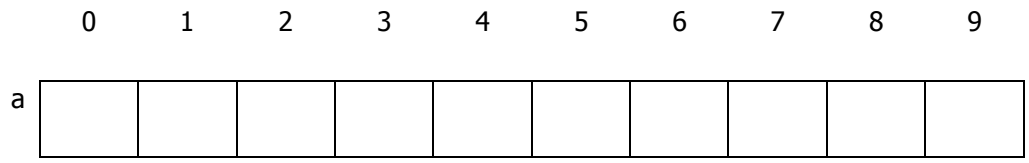
Cú pháp:

```
<Kiểu> <Tên mảng>[<Kích thước>];
```

Mỗi phần tử của mảng được truy nhập trực tiếp thông qua tên mảng cùng với chỉ số đặt giữa hai ngoặc vuông [ ]. Chỉ số là một số nguyên đươc đánh số từ 0 đến <Kích thước> - 1.

Ví dụ: int a[10]; //khai báo một mảng gồm 10 phần tử kiểu int

Các phần tử của mảng a được phân bố trong bộ nhớ như sau:



3. Khởi tạo mảng một chiều

Để khởi tạo mảng ta liệt kê danh sách các giá trị của chúng trong cặp dấu ngoặc "{" và "}", kích thước mảng không được nhỏ hơn số giá trị có trong danh sách khởi tạo.

Ví dụ **:** int a[10] = {20, -5, 8, 40, 10};

Các phần tử của mảng a được phân bố trong bộ nhớ như sau:

	0	1	2	3	4	5	6	7	8	9
a	20	-5	8	40	10					

Khi khởi tạo mảng cũng có thể không cần chỉ ra kích thước của nó, lúc đó máy sẽ dành cho mảng một vùng nhớ đủ để thu nhận danh sách giá trị khởi tạo.

Ví dụ**:** int b[] = {100, -25, 18, 10, 18};

Các phần tử của mảng b được phân bố trong bộ nhớ như sau:

	0	1	2	3	4
b	100	-25	18	10	18

Trong một số trường hợp việc sử dụng mảng khởi tạo như bảng tra sẽ giải quyết vấn đề rất nhanh gọn mà không cần phải sử dụng câu lệnh *if*() hoặc câu lệnh *switch*() phải xét rất nhiều trường hợp.

Ví dụ 1**:** Nhập vào một số nguyên dương n (1 ≤ n ≤ 7). Tùy theo giá trị n, hãy in ra các từ Sunday, Monday, . . ., Saturday tương ứng.

```
void main()
{
    char *a[7]= {"Sunday", "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday" };
    int n;
    <Nhập n>
    if(n >= 1 && n <= 7)
        printf("%s\n", a[n – 1]);
    else
        printf("Gia tri nhap khong hop le\n");
}
```

Ví dụ 2: Nhập tháng và năm. Hãy tính và in ra số ngày trong tháng.

```
int Nhuan(int y)
{
    if(y % 400 == 0 || (y % 4 == 0 && y % 100 != 0))
        return 1 ;
    return 0 ;
}
void main()
{
    int m, y;
    <Nhap m, y>
    int a[12] = {31, Nhuan(y) ? 29 : 28, 31, 30, 31, 30, 31, 31, 30,          31, 30, 31 };
    if(m > 0 && m < 13 && y > 0)
        printf("So ngay trong thang %d  nam %d la %d\n",m,y,a[m– 1]);
    else
        printf("Du lieu nhap khong hop le\n");
}
```

Ví dụ 3: Nhập vào năm Dương Lịch. Hãy tính và xuất ra năm Âm lịch tương ứng.

```
void main()
{
    char *can[10] = {"Canh", "Tan", "Nham", "Quy", "Giap", "At", "Binh",                    "Dinh",
"Mau", "Ky" };
    char *chi[12] = {"Than", "Dau", "Tuat", "Hoi", "Ti", "Suu", "Dan",                      "Meo",
"Thin", "Ti", Ngo", Mui" };
    int namdl;
    <Nhập namdl>
    if(namdl > 0)
```

```
        printf("Nam am lich: %s %s\n",can[namdl%10],chi[namdl% 12]);
    else
        printf("Gia tri nhap khong hop le\n");
}
```

4. Nhập xuất mảng một chiều

Ví dụ:

```
#define SIZE 20
void main()
{
    int a[SIZE];      //Khai báo mảng gồm SIZE phần tử kiểu int
    int n;            //Lưu số phần tử mảng
    int i;
    do {
        printf("Nhap so phan tu:");
        scanf("%d", &n);
    } while(n < 1 || n > SIZE);
    //Nhập dữ liệu cho mảng từ bàn phím
    for(i = 0; i < n; i++) {
        printf("pt thu %d:", i);
        scanf("%d", &a[i]);
    }
    //Xuất dữ liệu của mảng ra màn hình
    for(i = 0; i < n; i++)
        printf("%d\t", a[i]);
    printf("\n");
}
```

5. Con trỏ và mảng một chiều

Giả sử có khai báo :

int a[SIZE] ;

int *p ;

- Theo quan điểm của C tên mảng là một hằng địa chỉ và nó chính là địa chỉ phần tử đầu tiên của mảng. Như vậy:
  a chính là &a[0]

  a + 1 chính là &a[1]

  ...

  a + i chính là &a[i]

  ==> *(a + i) chính là a[i]

- Nếu có câu lệnh p = a thì:
  p trỏ tới a[0]

  p + 1 trỏ tới a[1]

  ...

  p + i trỏ tới a[i]

  ==> *(p + i) chính là a[i]

Ví dụ 1: Chương trình nhập xuất mảng một chiều sau đây sử dụng a + i thay cho &a[i] và *(a + i) thay cho a[i]

```
#define SIZE 20

void main()

{

    int a[SIZE];      //Khai báo mảng gồm SIZE phần tử kiểu int

    int n;            //Lưu số phần tử mảng

    int i;

    do {

        printf("Nhap so phan tu:");
```

```
        scanf("%d", &n);
    } while(n < 1 || n > SIZE);
    //Nhập dữ liệu cho mảng từ bàn phím
    for(i = 0; i < n; i++) {
        printf("pt thu %d:", i);
        scanf("%d", a + i);
    }
    //Xuất dữ liệu của mảng ra màn hình
    for(i = 0; i < n; i++)
        printf("%d\t", *(a + i));
    printf("\n");
}
```

Ví dụ 2: Chương trình nhập xuất mảng một chiều sau đây dùng thêm con trỏ p với câu lệnh p = a

```
#define SIZE 20
void main()
{
    int a[SIZE];      //Khai báo mảng gồm SIZE phần tử kiểu int
    int n;            // Lưu số phần tử mảng
    int i, *p;
    p = a;
    do {
        printf("Nhap so phan tu:");
        scanf("%d", &n);
    } while(n < 1 || n > SIZE);
    //Nhập dữ liệu cho mảng từ bàn phím
    for(i = 0; i < n; i++) {
```

```
        printf("pt thu %d:", i);
        scanf("%d", &p[i]);          //hoặc scanf("%d", p + i)
    }
    //Xuất dữ liệu của mảng ra màn hình
    for(i = 0; i < n; i++)
        printf("%d\t", p[i]);        //printf("%d\t", *(p + i))
    printf("\n");
}
```

6. Dùng mảng một chiều làm tham số truyền cho hàm

Như mục trên đã nói tên mảng là một hằng địa chỉ và nó chính là địa chỉ phần tử đầu tiên của mảng, do đó khi dùng mảng làm tham số truyền cho hàm thì thực chất là địa chỉ phần tử đầu tiên của mảng được truyền cho hàm và như vậy đối số tương ứng trong khai báo hàm phải viết dưới dạng con trỏ.

Ví dụ:

```
#define SIZE 20
// khai báo các nguyên mẫu hàm
void Nhap(int *a, int *n);
void Xuat(int *a, int n);
// Định nghĩa các hàm
void Nhap(int *a, int *n)
{
    do {
        printf("Nhap so phan tu:");
        scanf("%d", &(*n));
    } while(*n < 1 || *n > SIZE);
    for(int i = 0; i < *n; i++)  {
        printf("pt thu %d:", i);
        scanf("%d", &a[i]);
```

```
    }
}
void Xuat(int *a, int n)
{
    for(int i = 0; i < n; i++)
        printf("%d\t", a[i]);
    printf("\n");
}
void main()
{
    int a[SIZE];
    int n;
    Nhap(a, &n);
    Xuat(a, n);
}
```

Chú ý: Đối số tương ứng với tham số truyền là mảng cũng có thể khai báo như một mảng hình thức như sau:

```
void Nhap(int a[], int *n); /*Không cần chỉ ra kích thước trong ngoặc vuông*/
void Xuat(int a[], int n);
```

7. Các bài toán cơ bản trên mảng một chiều

Xét một mảng nguyên a gồm n phần tử kiểu int

Ví dụ 1: Tính tổng các phần tử trong mảng

```
int Tông(int a[], int n)
{
    int ret = 0;
    int i;
```

```
    for(i = 0; i < n; i++)
        ret += a[i];
    return ret;
}
```

Ví dụ 2: Tính tổng các phần tử là số chính phương trong mảng

```
int TôngCP(int a[], int n)
{
    int ret = 0;
    int i;
    for(i = 0; i < n; i++)
        if(ChinhPhuong(a[i]))    ret += a[i];
    return ret;
}
```

Ví dụ 3: Đếm các phần tử là số nguyên tố trong mảng

```
int DemNT(int a[], int n)
{
    int ret = 0;
    int i;
    for(i = 0; i < n; i++)
        if(NguyenTo(a[i]))   ret++;
    return ret;
}
```

Ví dụ 4: Tìm phần tử lớn nhất trong mảng

```
int TimMax(int a[], int n)
{
    int ret = a[0];
```

```
    int i;
    for(i = 1; i < n; i++)
        if(a[i] > ret)          ret = a[i];
    return ret;
}
```

Ví dụ 5: Tìm vị trí phần tử lớn nhất trong mảng

```
int TimVTMax(int a[], int n)
{
    int ret = 0;
    int i;
    for(i = 1; i < n; i++)
        if(a[i] > a[ret])       ret = i;
    return ret;
}
```

Ví dụ 6: Tìm vị trí phần tử dương nhỏ nhất trong mảng

```
int TimVTDuongMin(int a[], int n)
{
    int i, j, ret;
    //Tim vị trí phần tử dương đầu tiên
    for(i = 0; i < n; i++)
        if(a[i] > 0) break;
    if(i >= n)          //Không có phần tử dương
        ret = -1;
    else {              //Có phần tử dương
        //Tìm vị trí phần tử dương min
```

```
        ret = i;
        for(j = i + 1; j < n; j++)
            if(a[j] > 0 && a[j] < a[ret])
                ret = j;
    }
    return ret;
}
```

Ví dụ 7: Kiểm tra mảng có thứ tự tăng không. Hàm sau trả về 1 nếu mảng có thứ tự tăng, ngược lại hàm trả về 0.

```
int Tăng(int a[], int n)
{
    int ret = 1;
    int i;
    for(i = 0; i < n - 1; i++)
        if(a[i] > a[i+1]) {
            ret = 0;
            break;
        }
    return ret;
}
```

Ví dụ 8: Tìm các cặp phần tử có quan hệ sao cho phần tử này gấp 3 lần phần tử kia trong mảng.

```
void TimCapQuanHe(int a[], int n)
{
    int qh = 0;
    int i, j;
    for(i = 0; i < n - 1; i++)
```

```
        for(j = i + 1; j < n; j++)
            if(a[i] == 3 * a[j] || a[j] == 3 * a[i]) {
                printf("(%d, %d)\n", a[i], a[j]);
                qh = 1;
            }
    if(qh == 0)
        printf("Khong tim thay cap phan tu nao co quan he\n");
}
```

Ví dụ 9: Thêm phần tử x tại vị trí vt (0 <= vt <= n; vt = 0: thêm phần tử vào đầu mảng, vt = n: thêm phần tử vào cuối mảng), khi đó các phần tử từ vị trí n-1 xuống vt bị dời lên một vị trí để x chèn vào vị trí vt.

```
void Them(int a[], int *n, int vt, int x)
{
    int i;
    if(vt  < 0 || vt > *n)
        printf("Vi tri %d khong hợp le\n", vt);
    else {
        if(*n == SIZE)
            printf("Mang da day\n")
        else {
            // Dời các phần tử từ vị trí n-1 tới vt xuống một vị trí
            for(i = *n – 1; i >= vt; i--)
                a[i + 1] = a[i];
            //Thêm x vào vị trí vt
            a[vt] = x;
            //Tăng số phần tử lên 1
            (*n)++;
```

```
        }
    }
}
```

Ví dụ 10: Xóa phần tử tại ví trí vt (0 <= vt <= n - 1; vt = 0: xóa phần tử đầu mảng, vt = n - 1: xóa phần tử cuối mảng), khi đó các phần tử từ vị trí vt + 1 đến n -1 bị dời xuống một vị trí.

```
void xoa(int a[], int * n, int vt)
{
    int i;
    if(vt  < 0 || vt >= *n)
        printf("Vi tri %d khong hợp le\n", vt);
    else {
        if(*n == 0)
            printf("Mang rong\n")
        else {
            // Dời các phần tử từ vị trí vt + 1 đến n - 1 lên một vị trí
            for(i = vt; i <=* n - 1; i++)
                a[i] = a[i + 1];
            //Giảm số phần tử xuống 1
            (*n)--;
        }
    }
}
```

Ví dụ 11: Thay thế phần tử tại vi trí vt bằng phần tử khác (0 <= vt <= n -1; vt = 0: thay thế phần tử đầu mảng, vt = n-1: thay thế phần tử cuối mảng).

```
void Thay(int a[], int n, int vt, int x)
{
    if(vt  < 0 || vt >= n)
```

```
        printf("Vi tri %d khong hợp le\n", vt);
    else {
        if(n == 0)
            printf("Mang rong\n")
        else
            a[vt] = x;
    }
}
```

Ví dụ 12: Sắp xếp mảng theo thứ tự tăng dần

Có rất nhiều phương pháp sắp xếp, nói chung các phương pháp này thường được thực hiện bằng cách so sánh và đổi chỗ. Ở đây chúng ta chỉ trình bày một phương pháp sắp xếp đơn giản như sau:

Bắt đầu từ phần tử đầu tiên so sánh với các phần tử còn lại nếu thỏa điều kiện so sánh thì đổi chỗ hai phần tử đó với nhau, tiếp tục đến phần tử kế tiếp so sánh với các phần tử còn lại, . . . cho đến khi gặp phần tử cuối cùng. Việc sắp xếp hoàn tất.

Sau đây là hình ảnh minh họa các bước sắp xếp tăng một mảng gồm 5 phần tử

Danh sách ban đầu:	25	55	45	40	10
Bước 0:	10	55	45	40	25
Bước 1:	10	25	55	45	40
Bước 2:	10	25	40	55	45
Bước 3:	10	25	40	45	55

Chúng ta có nhận xét rằng với một mảng có n phần tử thì sẽ có n – 1 bước so sánh và đổi chỗ.

```
void SapXepTang(int a[], int n)
{
    int i, j;
    for(i = 0; i < n - 1; i++)
        for(j = i + 1; j <n; j++) {
            if(a[i] > a[j]) { //so sánh a[i] và a[j]
```

```
                //hoán vị a[i] và a[j]
                int tam = a[i];
                a[i] = a[j];
                a[j] = tam;
            }
        }
}
```

Ví dụ 13: Sắp xếp một bộ phận của mảng (dãy con) gồm a[s] a[s+1], . . . , a[e] với 0 ≤ s ≤ e ≤ n - 1 tăng dần mà không làm ảnh hưởng đến các phần tử còn lại của mảng

```
void SapXepDayConTang(int a[], int n, int s, int e)
{
    int i, j;
    for(i = s; i < e; i++)
        for(j = i + 1; j <= e; j++) {
            if(a[i] > a[j]) {  //so sánh a[i] và a[j]
                //hoán vị a[i] và a[j]
                int tam = a[i];
                a[i] = a[j];
                a[j] = tam;
            }
        }
}
```

Ví dụ 14: Tìm một phần tử có giá trị là x trong mảng (x được gọi là khóa tìm kiếm).

- **Tìm kiếm tuyến tính**

Phương pháp này sẽ tiến hành so sánh x với các phần tử thứ 0, thứ 1, . . . của mảng cho đến khi gặp được phần tử có khóa cần tìm hoặc đã tìm hết mảng mà không thấy x.

```
int TKTuyenTinh(int a[], int n, int x)
```

```
{
    int i;
    int ret = -1; //Không tìm thấy
    for(i = 0; i < n; i++)
        if(a[i] == x) {
            ret = i; // Tìm thấy tại ví trí i
            break;
        }
    return ret;
}
```

- **Tìm kiếm nhị phân**

Phương pháp này chỉ áp dụng cho **mảng đã có thứ tự tăng**. Ý tưởng của phương pháp này là tại mỗi bước ta tiến hành so sánh x với phần tử nằm ở vị trí giữa của dãy tìm kiếm hiện hành, dựa vào kết quả so sánh này để quyết định giới hạn dãy tìm kiếm kế tiếp là nửa trên hay nửa dưới của dãy tìm kiếm hiện hành.

```
int TKNhiPhan(int a[], int n, int x)
{
    int dau = 0, cuoi = n-1, giua;
    int ret = -1; //Không tìm thấy
    while(dau <= cuoi) {
        giua = (dau+cuoi)/2;
        if(a[giua] == x)  {
            ret = giua;       // Tìm thấy tại vị trí giữa
            break;
        }
        if(x < a[giua])       cuoi = giua – 1;
        else          dau = giua + 1;
```

```
    }

    return ret;

}
```

CHƯƠNG 7:

MẢNG HAI CHIỀU

1. Khai báo mảng hai chiều

Mảng hai chiều, còn gọi là ma trận, là sự mở rộng trực tiếp của mảng một chiều.

Cú pháp:

<Kiểu> <tên mảng>[<Kich thước dòng>][<Kích thước cột>];

Mỗi phần tử của màng được truy nhập trực tiếp thông qua tên mảng cùng với hai chỉ số đặt trong hai dấu ngoặc vuông [ ]. Chỉ số thứ nhất là chỉ số dòng (bắt đầu từ 0 đến <Kích thước dòng> – 1) và chỉ số thứ hai là chỉ số cột (bắt đầu từ 0 đến <Kích thước cột> – 1).

Ví dụ 1: int a[2][3]

Khai báo mảng hai chiều gồm 6 phần tử kiểu int

	0	1	2
0	a[0][0]	a[0][1]	a[0][2]
1	a[1][0]	a[1][1]	a[1][2]

Ví dụ 2: char b[3][4]

Khai báo mảng hai chiều gồm 12 phần tử kiểu char

	0	1	2	3
0	b[0][0]	b[0][1]	b[0][2]	b[0][3]
1	b[1][0]	b[1][1]	b[0][2]	b[0][3]
2	b[2][0]	b[2][1]	b[2][2]	b[2][3]

2. Khởi tạo mảng hai chiều

Khởi tạo mảng hai chiều tương tự như khởi tạo mảng một chiều bằng cách liệt kê các giá trị trong một danh sách.

Ví dụ 1: int a[2][3] = { {11, 12, 13}, {21, 22, 23} };

Khi đó a là mảng hai chiều gồm 6 phần tử kiểu int và có nội dung như sau:

	0	1	2
0	11	12	13
1	21	22	23

Ví dụ 2: Khởi tạo không cần chỉ ra kích thước dòng

- double b[][4] = { {6, 7. 8, 8, 9}, {3, 12.6, 4, 14}, {6.5, 20,7, 8} };
- Khi đó b là mảng hai chiều gồm 12 phần tử kiểu double và có nội dung như sau:

	0	1	2	3
0	6	8.7	8	9
1	3	12.6	4	14
2	6.5	20	7	8

Ví dụ 3: Sử dụng mảng hai chiều khởi tạo để vẽ chữ H hoa bằng các dấu "*"

```
#define SIZE1 5

#define SIZE2 5

void main()

{

    int a[SIZE1][ SIZE2] = {

        {1, 0, 0, 0, 1},

        {1, 0, 0, 0, 1},

        {1, 1, 1, 1, 1},

        {1, 0, 0, 0, 1},

        {1, 0, 0, 0, 1},

    };

    for(i = 0; i < SIZE1; i++) {

        for(j = 0; j < SIZE2; j++)

            if(a[i][j])

                printf("*")

            else printf(" ");            //In khoảng trắng

        printf("\n");
```

```
    }
}
```

3. Nhập xuất mảng hai chiều

Ví dụ:

```
#define SIZE1 4
#define SIZE2 6
void main()
{
    double a[SIZE1][SIZE2]; /* Khai báo mảng hai chiều gồm SIZE1*SIZE2 phần tử kiểu thực*/
    int sd, sc;         //Số dòng và số cột
    int i, j;
    double tam;
    do {
        printf("Nhap so dong va so cot:");
        scanf("%d%d", &sd, &sc);
    } while(sd < 1 || sd > SIZE1 || sc < 1 || sc > SIZE2);
    // Nhập dữ liệu mảng từ bàn phím
    for(i = 0; i < sd; i++)
        for(j = 0; j < sc; j++)  {
            printf("pt thu [%d][%d]:", i, j);
            scanf("%lf", &tam);
            a[i][j] = tam;
        }
    //Xuất dữ liệu mảng ra màn hình
    for(i = 0; i < sd; i++)  {
        for(j = 0; j < sc; j++)
```

```
            printf("%0.2f\t", a[i][j]);

        printf("\n");

    }

}
```

Lưu ý: Trong C, đối với các phần tử mảng hai chiều không nguyên ta không thể sử dụng toán tử lấy địa chỉ. Do đó trong ví dụ trên vì mảng hai chiều là thực nên ta không thể dùng hàm scanf() để nhập trực tiếp vào các phần tử mảng.

4. Con trỏ và mảng hai chiều

Giả sử xét khai báo sau:

int a[2][3] ; //a là địa chỉ kiểu int[3]

int (*p)[3] ; //p là con trỏ kiểu int[3]

- C quan niệm mảng hai chiều là mảng (một chiều) của mảng. Như vậy có thể xem a như là mảng một chiều mà mỗi phần tử của nó là một dãy 3 số nguyên (một dòng của bảng), do đó :
  a[0] chính là &a[0][0],

  a[1] chính là &a[1][0],

  . . .

  a[i] chính là &a[i][0]

- Nếu có câu lệnh p = a thì
  p trỏ tới a[0][0],

  p + 1 trỏ tới a[1][0],

  . . .

  p + i trỏ tới a[i][0]

Ví dụ 1: Chương trình nhập xuất mảng hai chiều sau có sử dụng a[i]+j hoặc *(a + i) + j thay cho &a[i][j] và *(a[i] + j) hoặc *(*(a + i) + j) thay cho a[i][j]

```
#define SIZE1 4

#define SIZE2 6

void main()

{
```

```
    int a[SIZE1][SIZE2]; /*Khai báo mảng hai chiều gồm SIZE1*SIZE2
    phần tử kiểu int*/

    int sd, sc;         //Số dòng và số cột

    int i, j;

    do {

        printf("Nhap so dong va so cot:");

        scanf("%d%d", &sd, &sc);

    } while(sd < 1 || sd > SIZE1 || sc < 1 || sc > SIZE2);

    // Nhập dữ liệu mảng từ bàn phím

    for(i = 0; i < sd; i++)

        for(j = 0; j < sc; j++)  {

            printf("pt thu [%d][%d]:", i, j);

            scanf("%lf", *(a + i) + j); // hoặc (a[i] + j)

        }

    //Xuất dữ liệu mảng ra màn hình

    for(i = 0; i < sd; i++)  {

        for(j = 0; j < sc; j++)

            printf("%0.2f\t", *(*(a + i) + j)); // hoặc *(a[i] + j)

        printf("\n");

    }

}
```

Ví dụ 2: Chương trình nhập xuất mảng hai chiều sau có dùng thêm con trỏ p kiểu int[SIZE2] với câu lệnh p = a và sử dụng p[i]+j hoặc *(p + i) + j thay cho &a[i][j], *(p[i] + j) hoặc *(*(p + i) + j) thay cho a[i][j]

```
#define SIZE1 4

#define SIZE2 6

void main()
```

```
{
    int a[SIZE1][SIZE2]; /* Khai báo mảng hai chiều gồm SIZE1*SIZE2 phần tử kiểu int*/
    int sd, sc;          //Số dòng và số cột
    int i, j;
    int (*p)[SIZE2];  //p là con trỏ kiểu int[SIZE2]
    p = a;
    do {
        printf("Nhap so dong va so cot:");
        scanf("%d%d", &sd, &sc);
    } while(sd < 1 || sd > SIZE1 || sc < 1 || sc > SIZE2);
    // Nhập dữ liệu mảng từ bàn phím
    for(i = 0; i < sd; i++)
        for(j = 0; j < sc; j++)  {
            printf("pt thu [%d][%d]:", i, j);
            scanf("%lf", p[i] + j); //hoặc *(p + i) + j
        }
    //Xuất dữ liệu mảng ra màn hình
    for(i = 0; i < sd; i++)  {
        for(j = 0; j < sc; j++)
            printf("%0.2f\t", *(p[i] + j)); //hoặc *(*(p + i) + j)
        printf("\n");
    }
}
```

5. Dùng mảng hai chiều làm tham số truyền cho hàm

Ví dụ:

```
#define SIZE1 5

#define SIZE2 6

// khai báo các nguyên mẫu hàm

void Nhap(double (*a)[SIZE2], int *sd, int *sc);

void Xuat(double (*a)[SIZE2], int sd, int sc);

// Định nghĩa các hàm

void Nhap(double (*a)[SIZE2], int *sd, int *sc)

{

    int i, j;

    double tam;

    do {

        printf("Nhap so dong va so cot:");

        scanf("%d%d", &(*sd), &(*sc));

    } while(*sd < 1 || *sd > SIZE1 | |*sc < 1 || *sc > SIZE2);

    for(i = 0; i < *sd; i++)

        for(j = 0; j < *sc; j++)  {

            printf("pt thu [%d][%d]:", i, j);

            scanf("%lf", &tam);

            a[i][j] = tam;

        }

}

void Xuat(double (*a)[SIZE2], int sd, int sc)

{

    int i, j;
```

```
    for(i = 0; i < sd; i++) {
        for(j = 0; j < sc; j++)
            printf("%0.2f\t", a[i][j]);
        printf("\n");
    }
}
void main()
{
    double a[SIZE1][SIZE2];
    int sd, sc;
    Nhap(a, &sd, &sc);
    Xuat(a, sd, sc);
}
```

Lưu ý: Đối số tương ứng với tham số truyền là mảng hai chiều cũng có thể khai báo như một mảng hình thức như sau:

```
void Nhap(double a[][SIZE2], int *sd, int *sc);
void Xuat(double a[][SIZE2], int sd, int sc);
```

6. Các bài toán cơ bản trên mảng hai chiều

Các thao tác trên mảng hai chiều chủ yếu dựa vào việc phối hợp tinh tế giữa các thao tác trên mảng một chiều và cách thức duyệt mảng hai chiều.

- Duyệt hết các phần tử của mảng

  Duyệt theo từng dòng từ trên xuống dưới, với mỗi dòng duyệt từ trái sang phải

```
for(i = 0; i < sd; i++)
    for(j = 0; j < sc; j++) {
        //Xử lý a[i][j]
        . . .
    }
```

Duyệt theo từng cột từ trái sang phải, với mỗi cột duyệt từ trên xuống dưới

```
for(j = 0; j < sc; j++)
        for(i = 0; i < sd; i++) {
                //Xử lý a[i][j]
                . . .
        }
```

- Duyệt các phần tử nằm trên cùng một dòng hay trên cùng một cột

  Duyệt các phần tử trên dòng có chỉ số k (0 ≤ k < sd)

```
for(i = 0; i < sc; i++) {
        //Xử lý a[k][i]
        . . .
}
```

  Duyệt các phần tử trên cột có chỉ số k (0 ≤ k < sc)

```
for(i = 0; i < sd; i++) {
        //Xử lý a[i][k]
        . . .
}
```

Ví dụ 1: Tính tổng các phần tử của ma trận

```
int Tong(int a[][SIZE2], int sd, int sc)
{
    int i, j, ret = 0;
    for(i = 0; i < sd; i++)
        for(j = 0; j < sc; j++)
            ret += a[i][j];
    return ret;
}
```

Ví dụ 2: Tìm phần tử lớn nhất của ma trận

```
int TimMax(int a[][SIZE2], int sd, int sc)
{
    int i, j, ret = a[0][0];
    for(i = 0; i < sd; i++)
        for(j = 0; j < sc; j++)
            if(a[i][j] > ret) ret = a[i][j];
    return ret;
}
```

Ví dụ 3: Tính tổng các phần tử trên dòng có chỉ số k (0 ≤ k ≤ sd - 1) của ma trận.

```
int TongDong(int a[][SIZE2], int sc, int k)
{
    int i, ret = 0;
    for(i = 0; i < sc; i++)
        ret += a[k][i];
    return ret;
}
```

Ví dụ 4: Sắp xếp các phần tử trên dòng có chỉ số k (0 ≤ k ≤ sd – 1) của ma trận tăng dần từ trái qua phải.

```
void SapXepDong(int a[][SIZE2], int sc, int k)
{
    int i, j;
    for(i = 0; i < sc – 1; i++)
        for(j = i + 1; j < sc; j++)
            if(a[k][i] > a[k][j]) {
                int tam = a[k][i];
```

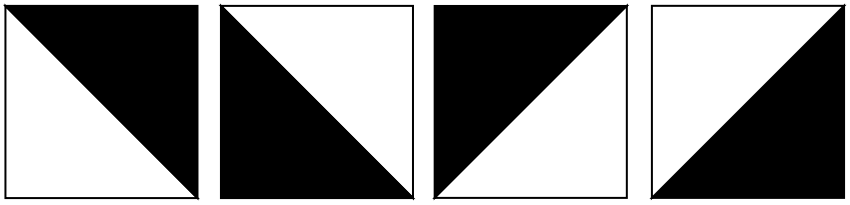
```
                a[k][i] = a[k][j];

                a[k][j] = tam;

            }

}
```

7. Mảng vuông (ma trận vuông)

a. Tính chất

Mảng vuông là mảng hai chiều có số dòng và số cột bằng nhau. Lúc đó số dòng và số cột của ma trận vuông được gọi chung là cấp ma trận, ký hiệu là n. Sau đây là một số đặc tính của ma trận vuông:

- Các phần tử nằm trên đường chéo chính là các phần tử a[i][i] ($0 \le i < n$).
- Các phần tử nằm trên đường chéo phụ là các phần tử a[i][n – 1- i] ($0 \le i < n$).
- Các phần tử nằm trong nửa mảng vuông phía trên đường chéo chính là các phần tử a[i][j] với $i \le j$.
- Các phần tử nằm trong nửa mảng vuông phía dưới đường chéo chính là các phần tử a[i][j] với $i \ge j$.
- Các phần tử nằm trong nửa mảng vuông phía trên đường chéo phụ là các phần tử a[i][j] với $i + j \le n - 1$.
- Các phần tử nằm trong nửa mảng vuông phía dưới đường chéo phụ là các phần tử a[i][j] với $i + j \ge n - 1$.



Nửa mảng vuông phía trên ĐĐC	Nửa mảng vuông phía dưới ĐĐC	Nửa mảng vuông phía trên ĐĐP	Nửa mảng vuông phía dưới ĐĐP

b. Duyệt mảng vuông

- Duyệt các phần tử nằm trên đường chéo chính.

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

        //Xử lý a[i][i]

        ...
```

```
}
```

- Duyệt các phần tử nằm trên đường chéo phụ.

```
for(i = 0; i < n; i++) {
        //Xử lý a[i][n – 1 – i]
        ...
}
```

- Duyệt các phần tử nằm trong nửa mảng vuông.
  + Duyệt các phần tử nằm trong nửa mảng vuông phía trên đường chéo chính

```
for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
                if(i <= j) {
                        <Xử lý a[i][j]>
                .}
```

  + Duyệt các phần tử nằm trong nửa mảng vuông phía dưới đường chéo chính

```
for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
                if(i >= j) {
                        <Xử lý a[i][j]>
                }
```

  + Duyệt các phần tử nằm trong nửa mảng vuông phía trên đường chéo phụ

```
for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
                if(i + j <= n – 1) {
                        <Xử lý a[i][j]>
                }
```

  + Duyệt các phần tử nằm trong nửa mảng vuông phía dưới đường chéo phụ

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

```
if(i  + j >= n – 1) {
        //Xử lý a[i][j]
}
```

Ví dụ 1: Tính tổng các phần tử là số nguyên tố trên đường chéo chính của ma trận vuông.

```
int TongNTTrenDDC(int a[][SIZE], int n)
{
    int i, ret = 0;
    for(i = 0; i < n; i++)
        if(NguyenTo(a[i][i]))
            ret += a[i][i];
    return ret;
}
```

Ví dụ 2: Tính tích các phần tử là số chính phương trên đường chéo phụ của ma trận vuông.

```
int TichCPTrenDCP(int a[][SIZE], int n)
{
    int i, ret = 1;
    int flag = 0;
    for(i = 0; i < n; i++)
        if(ChinhPhuong(a[i][n - 1 - i]))  {
            ret *= a[i][ n - 1 - i];
            flag = 1;
        }
    return (flag ? ret : -1);
}
```

Ví dụ 3: Tìm xem có phần tử x nằm trong nửa mảng vuông phía trên đường chéo chính không?

```
int TimNuaMVTrenDCC(int a[][SIZE], int n, int x)
```

```
{
    int i, j;
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
            if(i <= j && a[i][j] == x)
                return 1; //Tìm thấy
    return 0; //Không tìm thấy
}
```

Ví dụ 4: Sắp xếp các phần tử trên đường chéo chính tăng dần từ trên xuống dưới của ma trận vuông

```
void SapXepTangDDC(int a[][SIZE], int n)
{
    int i, j;
    for(i = 0; i < n - 1; i++)
        for(j = i + 1; j < n; j++)
            if(a[i][i] > a[j][j]) {
                //Hoan vi a[i][i] va a[j][j]
                int tam = a[i][i];
                a[i][i] = a[j][j];
                a[j][j] = tam;
            }
}
```

Ví dụ 5: Sắp xếp các phần tử trên đường chéo phụ tăng dần từ dưới lên trên của ma trận vuông

```
void SapXepTangDCP(int a[][SIZE], int n)
{
    int i, j;
```

```
    for(i = 0; i < n - 1; i++)
        for(j = i + 1; j < n; j++)
            if(a[i][n- 1- i] < a[j][n – 1- j]) {
                //Hoan vi a[i] [n- 1 - i] va a[j] [n – 1- j]
                int tam = a[i] [n- 1- i];
                a[i] [n- 1- i] = a[j] [n – 1- j];
                a[j] [n – 1- j] = tam;
            }
}
```

Ví dụ 6: Kiểm tra ma trận vuông có đối xứng không?

Ma trận vuông A gọi là đối xứng khi và chỉ khi A[i][j] = A[j][i] với mọi i = 0, ..., n - 1 và với mọi j = 0, ..., n – 1, chẳng hạn hai ma trận sau đây là đối xứng

A=

1	2	9
2	5	6
9	6	4

B=

4	20	-9
20	8	10
-9	10	3

```
int DoiXung(int a[][SIZE2], int n)
{
    int i, j;
    for(i = 0; i < n; i++)
        for( j = 0; j < n; j++)
            if (i >= j && a[i][j] != a[j][i])
                return 0; //Không đối xứng
    return 1; //Đối xứng
}
```

Cách khác:

```
int DoiXung(int a[][SIZE2], int n)
```

```
{
    int i, j;
    for(i = 0; i < n; i++)
        for( j = 0; j < i ; j++)
            if (a[i][j] != a[j][i])
                return 0; //Không đối xứng
    return 1; //Đối xứng
}
```

CHƯƠNG 8:

CHUỖI KÝ TỰ

1. Khái niệm

Chuỗi được xem như một mảng gồm các phần tử kiểu char. Ngoài ra ký hiệu kết thúc chuỗi được quy ước là ký tự '\0' (ký tự có mã ASCII là 0) đặt ở cuối chuỗi. Như vậy một mảng ký tự gồm n phần tử sẽ lưu được tối đa n – 1 ký tự.

2. Khai báo chuỗi

Cú pháp:

char <Tên chuỗi>[<Kích thước>];

Ví dụ: char s[10];

Khai báo một chuỗi s chứa tối đa được 9 ký tự. Khi đó trình biên dịch sẽ cấp phát một vùng nhớ 10 byte cho chuỗi s.

0	1	2	3	4	5	6	7	8	9

3. Khởi tạo chuỗi

Khi khởi tạo chuỗi có thể chỉ ra hoặc không chỉ ra kích thước chuỗi. Ngoài ra cũng có thể dùng một biến con trỏ kiểu char để khởi tạo một chuỗi.

Ví dụ: Chỉ ra kích thước khi khởi tạo chuỗi

char s[10] = "Hello";

0	1	2	3	4	5	6	7	8	9
H	e	l	l	o	\0				

Khi đó trình biên dịch sẽ cấp phát một vùng nhớ 10 byte có nội dung là chuỗi nói trên được ghép thêm một ký tự kết thúc chuỗi, 4 byte còn lại chưa sử dụng đến.

Ví dụ: Không chỉ ra kích thước khi khởi tạo chuỗi

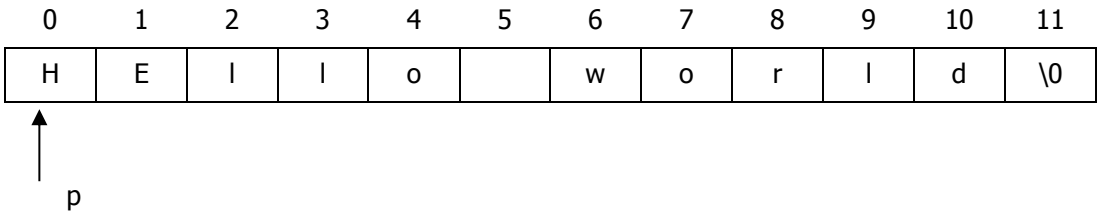
char s[] = "Hello";

0	1	2	3	4	5
H	e	l	l	o	\0

Khi đó trình biên dịch sẽ cấp phát một vùng nhớ 6 byte có nội dung là chuỗi nói trên được ghép thêm một ký tự kết thúc chuỗi.

Ví dụ: Dùng một biến con trỏ kiểu char để khởi tạo một chuỗi.

char *p = "Hello world";



Khi đó biến con trỏ p sẽ trỏ đến vùng nhớ gồm 12 byte có nội dung là chuỗi nói trên được ghép thêm một ký tự kết thúc chuỗi.

4. Hàm nhập chuỗi

a. Khái niệm về stdin

stdin là một vùng nhớ đệm dùng để lưu dữ liệu từ bàn phím. Các hàm scanf, gets và getchar đều nhận dữ liệu từ stdin. Cần phân biệt hai trường hợp:

- Nếu trên stdin có đủ dữ liệu thì các hàm trên sẽ nhận một phần dữ liệu mà chúng yêu cầu. Phần dữ liệu còn lại (chưa được nhận) vẫn ở trên stdin.
- Nếu trên stdin không đủ dữ liệu theo yêu cầu của các hàm, thì máy tạm dừng để người dùng đưa thêm dữ liệu từ bàn phím lên stdin (cho đến khi nhấn phím Enter).

b. Hàm nhập chuỗi

char *gets(char *s);

Nhập một dãy ký tự từ stdin cho đến khi gặp '\n' (ký tự '\n' bị loại khỏi stdin). Dãy ký tự được bổ sung thêm ký tự '\0' và đặt vào vùng nhớ do s trỏ đến. Hàm trả về con trỏ đến chuỗi nhận được.

Chú ý: Hàm scanf có để lại ký tự '\n' trên stdin, ký tự này sẽ làm trôi hàm gets (nếu có) sau đó. Để hàm gets không bị trôi thì phải khử ký tự '\n' trong stdin trước khi gọi hàm gets bằng cách dùng hàm fflush(stdin) để làm sạch stdin.

Ví dụ:

```
void main()
{
    int tuoi; char ten[31];
    printf("Nhap tuoi:");
```

```
    scanf("%d", &tuoi);
    printf("Nhap ten:");
    fflush(stdin);          //làm sạch stdin
    gets(ten);
}
```

5. Một số hàm thao tác chuỗi

Các hàm sau đây được định nghĩa sẵn trong tập tin tiêu đề **string.h**

- int **strlen**(char *s);
  Trả về độ dài chuỗi s.

- char ***strcat**(char *s1, char *s2);
  Ghép chuỗi s2 vào chuỗi s1 và trả về con trỏ đến chuỗi s1.

- char ***strcpy**(char *s1, char *s2);
  Chép chuỗi s2 đè lên chuỗi s1 và trả về con trỏ đến chuỗi s1.

- int **strcmp**(char *s1, char *s2);
  So sánh hai chuỗi s1 và s2. Hàm trả về giá trị âm nếu chuỗi s1 nhỏ hơn chuỗi s2, giá trị 0 nếu chuỗi s1 bằng hơn chuỗi s2 và giá tri dương nếu chuỗi s1 lớn hơn chuỗi s2.

- int **stricmp**(char *s1, char *s2);
  Hàm làm việc tương tự như strcmp nhưng không phân biệt chữ hoa với chữ thường

  Chú ý: Khi so sánh hai chuỗi, các ký tự của hai chuỗi được so sánh từng cặp một từ trái qua phải theo giá trị mã ASCII của chúng.

- char ***strchr**(char *s, char c);
  Tìm sự xuất hiện đầu tiên của ký tự c trong chuỗi s. Nếu tìm thấy hàm trả về con trỏ đến ký tự tìm được trong chuỗi s, nếu không hàm trả về giá trị NULL.

- char ***strstr**(char *s1, char *s2);
  Tìm sự xuất hiện đầu tiên của chuỗi con s2 trong chuỗi s1. Nếu tìm thấy hàm trả về con trỏ đến chuỗi con tìm được trong chuỗi s1, nếu không hàm trả về giá trị NULL.

Ví dụ 1: Nhập vào một chuỗi. Hãy viết các hàm thực hiện:

- Chuyển các chữ thường trong chuỗi thành chữ hoa.
- Chuyển các chữ hoa trong chuỗi thành chữ thường.
- Chuyển các chữ đầu một từ thành chữ hoa, các chữ còn lại trong từ thành chữ thường

```
void ChuyenHoa(char *s)

{

    int len = strlen(s);
    for(i = 0; i < len; i++)
        if(s[i] >= 'a' && s[i] <= 'z')
            s[i] -= 32;
}

void ChuyenThuong(char *s)

{

    int len = strlen(s);
    for(i = 0; i < len; i++)
        if(s[i] >= 'A' && s[i] <= 'Z')
            s[i] += 32;
}

void ChuyenDauTuHoa (char *s)

{

        int len = strlen(s);

        for(i = 0; i < len; i++) {

            if(i == 0) {            //đầu chuỗi

                if(s[i] != ' ')     //Ký tự đầu từ

                        if(s[i] >= 'a' && s[i] <= 'z')

                                s[i] -= 32; }

            else {

                if(s[i] != ' ' && s[i – 1] == ' ') {    //Ký tự đầu từ

                        if(s[i] >= 'a' && s[i] <= 'z')

                                s[i] -= 32; }

                else {              //Ký tự không phải đầu từ

                        if(s[i] >= 'A' && s[i] <= 'Z')

                                s[i] += 32;
```

```
                }

            }

        }

}

void main()

{

        char s[30];
        <Nhap s>
        ChuyenHoa(s);
        <In s>
        ChuyenThuong(s);
        <In s>
        ChuyenDauTuHoa(s);
        <In s>
}
```

Ví dụ 2: Nhập vào một chuỗi. Hãy viết các hàm thực hiện:

- Xóa các khoảng trắng bên trái chuỗi.
- Xóa các khoảng trắng bên phải chuỗi.
- Xóa các khoảng trắng dư thừa ở giữa chuỗi sao cho hai từ liên tiếp trong chuỗi chỉ cách nhau một khoảng trắng.

```
void XoaTrangTrai(char *s)

{

    while(s[0] == ' ')
    strcpy(s, s+1);
}

void XoaTrangPhai(char *s)

{

    while(s[strlen(s) – 1] == ' ')
    s[strlen(s) – 1] = 0;
}

void XoaTrangGiua(char *s)

{

    char *p;
```

```
    p = strstr(s, "  ");              //"  " là chuỗi hai khoảng trắng
    while(p != NULL) {
        strcpy(p, p + 1);
        p = strstr(s, "  ");
    }
}
void main()
{
    char s[30];
    <Nhập s>
    printf("->%s<-\n", s);
    XoaTrangTrai(s);
    printf("->%s<-\n", s);
    XoaTrangTrai(s);
    printf("->%s<-\n", s);
    XoaTrangTrai(s);
    printf("->%s<-\n", s);
}
```

CHƯƠNG 9: CẤU TRÚC

1. Khái niệm

Cấu trúc là một kiểu dữ liệu cho phép nhiều loại dữ liệu được nhóm lại với nhau, mỗi loại dữ liệu là một thành phần của cấu trúc.

2. Khai báo kiểu cấu trúc

cú pháp:

```
struct <Tên kiểu cấu trúc>{
    <Khai báo các thành phần cấu trúc>
};
```

Ví dụ 1: Khai báo cấu trúc lưu trữ thông tin của một điểm và một tam giác trong mặt phẳng

```
struct Diem {
    int x, y;
};
```

Diem là một kiểu cấu trúc gồm hai thành phần x, y kiểu int

```
struct TamGiac {
    Diem a, b, c;
};
```

TamGiac là một kiểu cấu trúc gồm 3 thành phần a, b, c kiểu Diem

Ví dụ 2: Khai báo cấu trúc lưu trữ thông tin của một sinh viên

```
struct SinhVien {
    char ma[10];
    char ten[30];
    int namsinh;
    double diem;
};
```

SinhVien là một kiểu cấu trúc gồm 4 thành phần: ma và ten kiểu chuỗi, namsinh kiểu int và diem kiểu double.

3. Truy nhập đến các thành phần cấu trúc

- Để truy nhập đến các thành phần cấu trúc từ một biến cấu trúc, ta dùng **toán tử dấu chấm**.

Ví dụ 1:

Diem d; //khai báo một biến d kiểu cấu trúc Diem

Để dùng d truy nhập đến các thành phần của Diem, ta viết:

d.x, d.y: : tọa độ điểm d

Ví dụ 2:

TamGiac tg; //khai báo một biến tg kiểu cấu trúc TamGiac

Để dùng tg truy nhập đến các thành phần của TamGiac, ta viết:

tg.a.x, tg.a.y : tọa độ điểm a

tg.b.x, tg.b.y: tọa độ điểm b

tg.c.x, tg.c.y: tọa độ điểm c

- Để truy nhập đến các thành phần cấu trúc từ một biến con trỏ cấu trúc, ta dùng **toán tử dấu mũi tên**.

Ví dụ:

Diem *p1; //khai báo một biến con trỏ p1 kiểu Diem

p1 = &d; //p1 trỏ tới d

Để dùng p1 truy nhập đến các thành phần của Diem, ta viết:

p1->x, p1->y: tọa độ điểm d

Ví dụ 2:

TamGiac *p2; //khai báo một biến con trỏ p2 kiểu TamGiac

p2 = &tg; //p2 trỏ tới tg

Để dùng p2 truy nhập đến các thành phần của TamGiac, ta viết:

p2->a.x, p2->a.y : tọa độ điểm a

p2->b.x, p2->b.y: tọa độ điểm b

p2->c.x, p2->c.y: tọa độ điểm c

4. Mảng cấu trúc

Mảng cấu trúc là một mảng mà mỗi phần tử của nó là một cấu trúc bao gồm nhiều thành phần.

Ví dụ 1:

```
#define SIZE 20

Diem darr[SIZE];
```

darr là một mảng cấu trúc gồm SIZE phần tử kiểu Diem. Khi đó:

darr[0].x, darr[0].y là tọa độ điểm thứ 0

darr[1].x, darr[1].y là tọa độ điểm thứ 1

. . .

Ví dụ 2:

```
#define SIZE 20

TamGiac tgarr[SIZE];
```

tgarr là một mảng cấu trúc gồm SIZE phần tử kiểu TamGiac. Khi đó:

tgarr[0].a.x, tgarr[0].a.y, tgarr[0].b.x, tgarr[0].b.y, tgarr[0].c.x, tgarr[0].c.y là tọa độ 3 đỉnh của tam giác thứ 0

tgarr[1].a.x, tgarr[1].a.y, tgarr[1].b.x, tgarr[1].b.y, tgarr[1].c.x, tgarr[1].c.y là tọa độ 3 đỉnh của tam giác thứ 1

...

5. Khởi tạo cấu trúc

Đế khởi tạo cho cấu trúc ta viết vào sau khai báo của chúng một danh sách giá trị cho các thành phần.

Ví dụ:

```
SinhVien sv = {"a00", "Le Minh Hung", 1980, 9.5};

SinhVien svarr[3] = {
    {"c00", "Nguyen Hoang An", 1982, 7},   //Sinh vien thứ 0
    {"c04", "Vo Thi Ly Lan", 1982, 10},    //Sinh vien thứ 1
```

```
    {"d11", "Tran Van Tung", 1981, 8.5}     //Sinh vien thứ 2
};
```

6. Nhập xuất cấu trúc

Ví dụ: Nhập xuất một sinh viên

```
//Khai báo các nguyên mẫu hàm
void Nhap1SV(SinhVien *u);
void Xuat1SV(SinhVien u);
//Định nghĩa các hàm
void Nhap1SV(SinhVien *u)
{
    double tam;
    printf("Ma sinh vien:"); fflush(stdin); gets(u->ma);
    printf("Ho ten:"); gets(u->ten);
    printf("Nam sinh:"); scanf("%d", &u->namsinh);
    printf("Diem:"); scanf("%lf", &tam); u->diem = tam;
}
void Xuat1SV(SinhVien u)
{
    printf("%s\t%s\t%d\t%0.2f\n", u.ma, u.ten, u.namsinh, u.diem);
}
//hàm chính
void main()
{
    SinhVien sv;
    Nhap1SV(&sv);
    Xuat1SV(sv);
```

```
}
```

Chú ý: Trong C đối với các thành phần không nguyên của kiểu cấu trúc ta không thể sử dụng toán tử lấy địa chỉ. Do đó trong ví dụ trên vì thành phần diem của cấu trúc SinhVien có kiểu thực nên ta không thể dùng hàm scanf() để nhập trực tiếp vào thành phần này.

Ví dụ: Nhập xuất danh sách sinh viên

```
#define SIZE 20
struct DSSV {
    int n; //Số sinh viên
    SinhVien arr[SIZE]; //Mảng các sinh viên
};
//Khai báo các nguyên mẫu hàm
void Nhap1SV(SinhVien *u);
void Xuat1SV(SinhVien u);
void NhapDSSV(DSSV *u);
void XuatDSSV(DSSV u);
//Định nghĩa các hàm
void NhapDSSV(DSSV* u)
{
    do {
        printf("Nhap số sinh vien:");
        scanf("%d", &u->n);
        if(u->n < 1 || u->n > SIZE)
        printf("So sinh vien khong hop le, xin nhap lai !!!\n");
    }while(u->n < 1 || u->n > SIZE);
    for(i = 0; i < u->n, i++) {
        printf("** Sinh vien thu %d **\n", i);
```

```
        Nhap1SV(&u->arr[i]);
    }
}
void XuatDSSV(DSSV u)
{
    for(i = 0; i < u.n, i++)
        Xuat1SV(u.arr[i]);
}
void main()
{
    DSSV dssv;
    printf("Nhap danh sach sinh vien\n");
    NhapDSSV(&dssv);
    printf("Danh sach sinh vien vua nhap\n");
    XuatDSSV(dssv);
}
```

7. Các bài toán cơ bản trên mảng cấu trúc

Các thao tác trên mảng cấu trúc được xử lý tương tự như các thao tác trên mảng một chiều.

Ví dụ 1: Tìm sinh viên có điểm cao nhất

```
int TimSVDiemMax(DSSV u)
{
    int i, ret = 0;
    for(i = 1; i < u.n; i++)
        if(u.arr[i].diem > u.arr[ret].diem)
            ret = i;
    return ret;        //Trả về vị trí sinh viên có điểm cao nhất
```

```
}
```

Ví dụ 2: Thêm một sinh viên sv vào danh sách tại vị trí vt

```
void Them(DSSV *u, int vt, SinhVien sv)
{
    int i;
    if(vt  < 0 || vt > u->n)
        printf("Vi tri %d khong hợp le\n", vt);
    else {
        if(u->n == SIZE)
            printf("Danh sach da day\n")
        else {
            for(i = u->n – 1; i >= vt; i--)
                u->arr[i + 1] = u->arr[i];
            u ->arr[vt] = sv;
            u->n++;
        }
    }
}
```

Ví dụ 3: Xóa một sinh viên tại vị trí vt khỏi danh sách

```
void xoa(DSSV *u, int vt)
{
    int i;
    if(vt  < 0 || vt >= u->n)
        printf("Vi tri %d khong hợp le\n", vt);
    else {
        if(u->n == 0)
```

```
            printf("Danh sach rong\n")
        else {
            for(i = vt; i <= u->n - 1; i++)
                u->arr[i] = u->arr[i + 1];
            u->n--;
        }
    }
}
```

Ví dụ 4: Sắp xếp danh sách sinh viên theo thứ tự tăng dần của mã sinh viên

```
void SapXep(DSSV *u)
{
    int i, j;
    for(i = 0; i < u->n - 1; i++)
        for(j = i + 1; j <u->n; j++) {
            // So sánh mã của sinh viên thứ i và sinh viên thứ j
            if(strcmp(u->arr[i].ma , u->arr[j].ma) > 0) {
                //Hoán vị u->arr[i] và u->arr[j]
                SinhVien tam = u->arr[i];
                u->arr[i] = u->arr[j];          u->arr[j] = tam;
            }
        }
}
```

CHƯƠNG 10: CON TRỎ

1. Cấp phát vùng nhớ động

a. Biến động (Dynamic variable)

- Biến động là các biến được tạo lúc chạy chương trình.
- Các biến động không có tên và việc truy nhập các biến động được thực hiện thông qua biến con trỏ chỉ đến nó.

b. Cấp phát vùng nhớ động

Cú pháp:

```
void *malloc(size_t size)
```

```
void *calloc(size_t nitems, size_t size)
```

Ý nghĩa:

Hàm malloc sẽ cấp phát một vùng nhớ động có kích thước là size bytes.

Hàm calloc sẽ cấp phát một vùng nhớ động có kích thước là nitems*size bytes.

Nếu thành công cả hai hàm này đều trả về con trỏ chỉ đến đầu vùng nhớ được cấp phát, ngược lại nếu thất bại sẽ trả về giá trị NULL.

Ví dụ:

```
void main() {
        int n, i, j, temp;
        int *p;
        printf("\n Nhập số phần tử mảng: ");
        scanf("%d", &n);
        p = (int*) malloc(n * sizeof(int));
        Nhap(p, n);      //Nhập mảng
        Xuat(p, n);      //Xuất mảng
        free(p);
}
```

Ví dụ:

```
void main() {
        int *p, sd, sc, i, j, temp;
        printf("\nsố dòng và số cột: ");
```

```
        scanf("%d%d", &sd, &sc);
        p = (int**) malloc(sd * sizeof(int *));
        for(i = 0; i < sd; i++)
                p[i] = (int *)malloc(sc * sizeof(int));
        Nhap(p, sd, sc);          //Nhập mảng
        Xuat(p, sd, sc); //Xuất mảng
        for(i = 0; i < sd; i++)
                free(p[i]);
        free(p);
}
```

c. Cấp phát lại vùng nhớ

Cú pháp:

```
void *realloc(void *p, size_t size)
```

Ý nghĩa:

Cấp phát lại 1 vùng nhớ do con trỏ p quản lý, vùng nhớ này có kích thước mới là size bytes, khi cấp phát lại thì nội dung vùng nhớ trước đó vẫn tồn tại. Nếu thành công hàm này trả về con trỏ chỉ đến đầu vùng nhớ được cấp phát, ngược lại nếu thất bại sẽ trả về giá trị NULL.

```
void main() {
        int *ptr; int i;
        ptr = (int *)calloc(5, sizeof(int *));
        if(ptr != NULL) {
                *ptr = 1; *(ptr + 1) = 2;
                ptr[2] = 4; ptr[3] = 8; ptr[4] = 16;
                ptr = (int*)realloc(ptr, 7 * sizeof(int));
                if(ptr != NULL)  {
                        ptr[5] = 32; sptr[6] = 64;
                        for(i = 0;i < 7; i++)
                                printf("ptr[%d] : %d\n", i, ptr[i]);
                        free(p);
                        printf("\n"); }}
                else  printf("Khong du bo nho - recalloc that
bai.\n"); }
```

```
else  printf("Khong du bo nho - calloc that bai\n"); }
```

d. Giải phóng vùng nhớ động

Cú pháp:

void free(void *p)

Ý nghĩa:

Giải phóng vùng nhớ được quản lý bởi con trỏ p.

BÀI TẬP

CHƯƠNG 2: CÁC KIỂU DỮ LIỆU CƠ SỞ

1. Nhập vào 2 số nguyên. Tính và in ra tổng, hiệu, tích, thương và dư của hai số nguyên này.
2. Nhập vào bán kính của hình tròn. Tính và in ra chu vi, diện tích theo công thức sau:

   CV = 2*PI*r

   DT = PI*r*r

3. Nhập vào chiều dài và chiều rộng của hình chữ nhật. Tính và in ra chu vi, diện tích theo công thức sau:

   CV = (a+b)*2

   DT = a*b

4. Nhập vào 3 cạnh của hình tam giác. Tính và in ra chu vi, diện tích theo công thức sau:

   CV = a+b+c

   DT = $(p*(p-a)*(p-b)*(p-c))^{1/2}$

   với p là nửa chu vi.

5. Nhập vào bán kính hình cầu. Tính và in ra thể tích, diện tích theo công thức sau:

   TT = $4*PI*r^3 / 3$;

   DT = $4*PI*r^2$

6. Nhập vào độ Ferenheit. Tính và in ra độ celsius theo công thức sau:

   C = 5/9 * (F – 32).

   Lưu ý: Tránh mất dữ liệu trong quá trình tính toán.

7. Nhập chiều dài con lắc đơn. Tính và in ra chu kỳ con lắc đơn theo công thức:

   $T = 2*PI*(l/g)^{1/2}$

   l: chiều dài con lắc đơn

   g: gia tốc trọng trường (9.81 m/s^2)

8. Nhập khối lượng hai vật thể và khoảng cách giữa chúng. Tính và in ra lực vạn vật hấp dẫn theo công thức:

   $F = G*(m1*m2)/r^2$

   G: hằng số hấp dẫn ($6.67*10^{-11}$ Nm^2/kg^2)

   m1, m2: Khối lượng hai vật thể.

   r: khoảng cách hai vật.

❖ Lưu ý:

```
#define G       6.67E-11 //Cách viết khoa hoc của 6.67*10^-11
```

9. Nhập vào một số nguyên n và một số thực x. Tính và in ra biểu thức $(x^2 + 1)^n$.
10. Nhập vào một số nguyên n và một số thực x. Tính và in ra biểu thức

    $A = (x^2 + x + 1)^n + (x^2 + x - 1)^n$

11. Nhập vào một số tiền nguyên dương. Đổi số tiền này ra các tờ giấy bạc 50đ, 20đ, 10đ, 5đ và 1đ. Với giả thiết ưu tiên cho tờ có mệnh giá lớn hơn, hãy in ra xem đổi được bao nhiêu tờ mỗi loại.
12. Nhập vào một số nguyên dương có đúng 3 chữ số. Tính và in ra số đảo ngược.

CHƯƠNG 3: CÁC CÂU LỆNH RẼ NHÁNH

1. Nhập vào một ký tự. Hãy in thông báo cho biết ký tự đó là ký tự số, ký tự chữ cái, ký tự phép toán hay là ký tự dạng khác với các dạng trên.
   Lưu ý: Kí tự phải để trong dấu nháy đơn. Ví dụ: 'a' là kí tự a (có mã ASCII là 97).
2. Nhập vào hai số thực a, b. Hãy tính và in ra nghiệm phương trình bậc nhất (a bất kì).

   ax + b = 0.

3. Nhập vào hai số nguyên. Hãy mô phỏng một máy tính đơn giản gồm 5 phép tính hai ngôi (+, -, *, /, %) trên hai số nguyên này.
   Lưu ý: Trước câu lệnh đọc kí tự phải thêm câu lệnh xóa vùng nhớ đệm là: fflush(stdin);
4. Nhập vào ba số thực a, b, c (a bất kì). Hãy tính và in ra nghiệm phương trình bậc hai:

   $ax^2 + bx + c = 0.$

5. Nhập vào sáu số thực a, b , c, d, m, n. Hãy tính và in ra nghiệm hệ phương trình hai ẩn số:

   $ax + by = m$

   $cx + dy = n$

   Nhắc lại:

   Nghiệm: $x = DX/D$, $y = DY/D$

   với $DX = md – nb$, $DY = an – cm$ và $D = ad – cb$.

6. Nhiệt độ F(Fahrenheit) và nhiệt độ C(Celcius) liên hệ với nhau theo công thức: $C = 5(F – 32)/9$. Viết chương trình cho phép người dùng nhập vào độ F hay độ C và đổi sang độ còn lại.
7. Nhập vào các số thực a, b, c. Hãy kiểm tra xem ba số này có lập thành 3 cạnh của một tam giác hay không? Nếu có hãy tính và in ra chu vi, diện tích, chiều dài mỗi đường cao của tam giác.
8. Nhập vào ba số nguyên d, m, y là ngày, tháng, năm. Hãy in thông báo cho biết đó là ngày thứ mấy trong tuần. Sử dụng công thức sau để biến đổi ngày, tháng, năm thành thứ trong tuần:

   $A = d + 2m + (3(m + 1)/5) + y + (y/4) – (y /100) + (y/400) + 2$

   với quy ước tháng 1, 2 của năm y được xem là tháng 13, 14 của năm y–1. Số dư trong phép chia A cho 7 cho kết quả là thứ trong tuần theo nghĩa số dư là 0: Thứ bảy, số dư là 1: Chủ nhật, số dư là 2: Thứ hai. v.v…

9. Nhập vào ngày, tháng, năm. Hãy kiểm tra tính hợp lệ của ngày tháng năm nhập vào. Nếu hợp lệ hãy cho biết tháng nhập có bao nhiêu ngày, ngày hôm sau của ngày đã nhập là ngày nào.
   (Biết rằng năm nhuận là năm (chia hết cho 400) hoặc (chia hết cho 4 và không chia hết cho 100)).

10. Nhập vào giờ vào ca, giờ ra ca. Hãy tính và in ra tiền lương ngày cho công nhân, biết rằng tiền trả cho mỗi giờ trước 12 giờ trưa là 6000đ và mỗi giờ sau 12 giờ trưa là 7500đ. Giờ vào ca sớm nhất là 6 giờ sáng và giớ ra ca trễ nhất là 18 giờ.
11. Nhập vào trọng lượng m(kg) hàng hóa bán được ($0 < m \leq 100$). Hãy tính và in ra tiền lời thu được, biết rằng

    $0 < m \leq 10$ Tiền lời là 5000đ/kg.

    $10 < m \leq 20$ Tiền lời là 5000đ/kg.

    $20 < m \leq 50$ Tiền lời là 9000đ/kg và thêm 2% tổng số tiền lời.

    $50 < m$ Tiền lời là 10000đ/kg và thêm 4% tổng số tiền lời nhưng không được quá 1000000đ.

12. Nhập vào số ngày thuê và loại phòng (một trong 3 loại A, B hoặc C). Hãy tính và in ra tiền thuê phòng với quy định như sau:

    Loại A: 250000đ/ngày

    Loại B: 200000đ/ngày

    Loại C: 150000đ/ngày

    Nếu thuê quá 12 ngày thì phần trăm được giảm trên tổng số tiền (tính theo giá quy định) là: 10% cho phòng loại A, 8% cho phòng loại B hoặc C.

13. Trong một kỳ thi tuyển, hội đồng tuyển sinh đã xem xét đề nghị một điểm chuẩn (xem như dữ kiện nhập). Mỗi thí sinh tham gia kỳ thi sẽ trúng tuyển nếu điểm tổng kết của thí sinh đó lớn hơn hoặc bằng điểm chuẩn và không có môn nào điểm 0. Điểm tổng kết của mỗi thí sinh là tổng điểm của 3 môn thi và điểm ưu tiên. Điểm ưu tiên bao gồm điểm ưu tiên theo khu vực và điểm ưu tiên theo đối tượng dự thi. Cho biết:

    Khu vực A: điểm ưu tiên là 2.

    Khu vực B: điểm ưu tiên là 1.

    Khu vực C: điểm ưu tiên là 0.5.

    Đối tượng 1: điểm ưu tiên là 2.5.

    Đối tượng 2: điểm ưu tiên là 1.5.

    Đối tượng 3: điểm ưu tiên là 1.

    Viết chương trình nhập vào điểm chuẩn, điểm của 3 môn thi của một thí sinh, khu vực và đối tượng dự thi. Cho biết thí sinh đó đậu hay rớt.

CHƯƠNG 4: CÁC CÂU LỆNH LẶP

Bài tập:

1. Nhập vào số nguyên dương n. Hãy tính và in ra các tổng sau:
   a. n! = 1*2*3*…*n.
   b. x^y = x*x*x*…*x (nhân đủ y lần).
   c. s = 1*2 + 2*3 + … + n*(n + 1)
   d. s = $1^1 + 2^2 + … + n^n$
   e. s = 1 + 1/2+ 1/3 + … +1/n
2. Nhập vào một số nguyên dương n. Hãy tính và in ra giá trị của vế trái, vế phải của mỗi công thức sau đây và cho biết kết quả tính của hai vế có bằng nhau hay không?
   a. 1.2 + 2.2 + ... + n.2 = n(n + 1)(2n + 1) / 6
   b. 1.2.3 + 2.3.4 + … + n(n + 1)(n + 2) = n(n + 1)(n + 2)(n + 3)/4
   c. 1(1!) + 2(2!) + … + n(n!) = (n + 1)! – 1
   d. 1/2! + 2/3! + … + n/(n + 1)! = 1 – (1/(n + 1)!)
3. Nhập vào x (tính theo radian). Hãy tính và in ra:
   a. e(x) = 1 + x + $x^2/2$! + ... + x^n/n ! (độ chính xác là 0.0001, tức là e(x) được tính đến số hạng sao cho trị tuyệt đối của số hạng đó nhỏ hơn 0.0001)
   b. sin(x) = x – $x^3/3!$ + $x^5/5!$ - ... + $(-1)^n(x^{2n+1}/(2n+1)!)$ (độ chính xác là 0.0001)
   c. cos(x) = 1 – $x^2/2!$ + $x^4/4!$ - ... + $(-1)^n(x^n/2n!)$ (độ chính xác là 0.0001)
4. Nhập vào số nguyên n. Hãy kiểm tra xem n có phải là số đối xứng không. Biết rằng số đối xứng là số có số nghịch đảo của nó bằng chính nó.
5. Nhập vào số nguyên n. Hãy kiểm tra xem n có phải là số hoàn chỉnh không? Biết rằng số hoàn chỉnh là số có tổng các ước số của nó bằng chính nó.
6. Nhập vào một số nguyên. Hãy kiểm tra xem số nguyên đó có phải là số nguyên tố hay không? Biết rằng một số nguyên dương (khác số 1) được gọi là số nguyên tố nếu ngoài 1 và chính nó ra thì nó không chia hết cho một số nguyên dương nào khác. Chẳng hạn các số sau là nguyên tố:

   2, 3, 5, 7, 11, 13, . . .

7. Nhập vào số nguyên dương n. Hãy tìm và in ra các số nguyên tố trong khoảng từ 1 đến n.
8. Nhập vào một số nguyên. Hãy kiểm tra xem một số nguyên đó có phải là số hoàn chỉnh hay không? Biết rằng một số nguyên dương n được gọi là số hoàn chỉnh nếu tổng các ước số của n (không kể n) bằng chính nó. Chẳng hạn các số sau là hoàn chỉnh:

   6, 28, 496, ...

9. Nhập vào số nguyên dương n. Hãy tìm và in ra các số hoàn chỉnh trong khoảng từ 1 đến n.
10. Nhập vào một số nguyên. Hãy kiểm tra xem số nguyên đó có phải là số chính phương hay không? Biết rằng một số nguyên dương n được gọi là số chính phương nếu nó bằng bình phương của một số nguyên. Chẳng hạn các số sau là chính phương:

1, 4, 9, 16, 25, ...

11. Nhập vào số nguyên dương n. Hãy tìm và in ra các số chính phương trong khoảng từ 1 đến n.
12. Nhập vào một số nguyên. Hãy kiểm tra xem số nguyên đó có phải là số đối xứng hay không? Biết rằng một số nguyên n được gọi là đối xứng nếu số đảo của n bằng chính n. Chẳng hạn các số sau là đối xứng:

    1, 44, 161, 2552, ...

13. Nhập vào số nguyên dương n. Hãy tìm và in ra các số đối xứng trong khoảng từ 1 đến n.
14. Nhập vào hai số nguyên a và b. Hãy tìm và in ra ước số chung lớn nhất của a và b.

Bài tập làm thêm:

1. Nhập vào số nguyên dương n. Hãy tính và in ra các tổng sau:
   a. s = 1 + 1/2+ 1/3 + … +1/n.
   b. s = 1/3 + 2/5 + 3/7 + … + n/(n+n+1).
2. Hãy xuất
   c. s = 1 + 1/2+ 1/3 + … +1/n.
   d. s = 1/3 + 2/5 + 3/7 + … + n/(n+n+1).
3. Nhập vào một số nguyên dương n. Hãy tính và in ra giá trị của vế trái, vế phải của mỗi công thức sau đây và cho biết kết quả tính của hai vế có bằng nhau hay không?
   a. 1.2.3 + 2.3.4 + … + n(n + 1)(n + 2) = n(n + 1)(n + 2)(n + 3)/4
   b. 1/2! + 2/3! + … + n/(n + 1)! = 1 – (1/(n + 1)!)
4. Nhập vào x (tính theo radian). Hãy tính và in ra:
   d. $\sin(x) = x - x^3/3! + x^5/5! - ... + (-1)^n(x^{2n+1}/(2n+1)!)$ (độ chính xác là 0.0001)
   e. $\cos(x) = 1 - x^2/2! + x^4/4! - ... + (-1)^n(x^n/2n!)$ (độ chính xác là 0.0001)
5. Hãy in bảng mã ASCII ra màn hình. Bảng được in ra theo từng trang màn hình sao cho người sử dụng có thể đọc được.
6. Hãy in toàn bộ bảng cửu chương từ hai đến chín ra màn hình. Bảng được in ra theo từng trang màn hình sao cho người sử dụng có thể đọc được.
7. Nhập vào một số nguyên. Hãy kiểm tra xem số nguyên đó có phải là số tiến hay không. Biết rằng số tiến là số có các chữ số không giảm từ trái qua phải.
   Ví dụ: Các số tiến: 12345, 13357,… Không phải số tiến: 12354, 41234
8. Nhập vào số nguyên dương n. Hãy tìm và in ra các số nguyên tố trong khoảng từ 1 đến n.
9. Nhập vào một số nguyên. Hãy kiểm tra xem một số nguyên đó có phải là số hoàn chỉnh hay không? Biết rằng một số nguyên dương n được gọi là số hoàn chỉnh nếu tổng các ước số của n (không kể n) bằng chính nó. Chẳng hạn các số sau là hoàn chỉnh:

    6, 28, 496, ...

10. Nhập vào số nguyên dương n. Hãy tìm và in ra các số hoàn chỉnh trong khoảng từ 1 đến n.

11. Nhập vào một số nguyên. Hãy kiểm tra xem số nguyên đó có phải là số chính phương hay không? Biết rằng một số nguyên dương n được gọi là số chính phương nếu nó bằng bình phương của một số nguyên. Chẳng hạn các số sau là chính phương:

    1, 4, 9, 16, 25, ...

12. Nhập vào số nguyên dương n. Hãy tìm và in ra các số chính phương trong khoảng từ 1 đến n.
13. Nhập vào một số nguyên. Hãy kiểm tra xem số nguyên đó có phải là số đối xứng hay không? Biết rằng một số nguyên n được gọi là đối xứng nếu số đảo của n bằng chính n. Chẳng hạn các số sau là đối xứng:

    1, 44, 161, 2552, ...

14. Nhập vào số nguyên dương n. Hãy tìm và in ra các số đối xứng trong khoảng từ 1 đến n.
15. Nhập vào hai số nguyên a và b. Hãy tìm và in ra ước số chung lớn nhất của a và b.

CHƯƠNG 5: HÀM

1. Viết hàm DienTich() sau, hàm này trả về diện tích một đường tròn với bán kính r:

   double DienTich (double r);

2. Viết hàm Min() sau, hàm này trả về giá trị nhỏ nhất của hai số nguyên x và y:

   int Min(int x, int y);

3. Viết hàm Min() sau, hàm này trả về giá trị nhỏ nhất của ba số nguyên x, y và z:

   int Min(int x, int y, int z);

4. Viết hàm LuyThua() sau, hàm này trả về giá trị x^p:

   double LuyThua (double x, int p);

5. Viết hàm ToHop() sau, hàm này trả về C(n, k) là số tổ hợp n lấy k (n > 0, 0 <= k <= n) bằng cách dựa vào công thức C(n, k) = n! / (k!(n – k)!):

   long ToHop (int n, int k);

6. Viết hàm InTGPascal() sau, hàm này xuất ra một tam giác Pascal có sd dòng. Hình sau đây là một tam giác Pascal có 7 dòng

	0	1	2	3	4	5	6
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1

   Mỗi số trong tam giác Pascal là một tổ hợp C(n, k). Nếu chúng ta tính dòng và cột bắt đầu bằng 0, thì số ở dòng n và cột k là C(n, k). Ví dụ, số ở dòng 6 và cột 4 là C(6, 4) = 15

   void InTGPascal(int sd);

7. Viết hàm Nhuan() sau, hàm này trả về giá trị 1 (đúng) nếu năm y nhuần và trả về giá trị 0 (sai) nếu năm y không nhuần:

   int Nhuan (int y);

8. Viết hàm SNTrongThang () sau, hàm này trả về số ngày trong tháng m và năm y:
   int SNTrongThang (int m, int y);

9. Viết hàm HopLe() sau, hàm này trả về giá trị 1 (đúng) nếu ngày d, tháng m, năm y hợp lệ và trả về giá trị 0 (sai) nếu d, m, y không hợp lệ:
   int HopLe(int d, int m, int y);

10. Viết hàm NguyenTo() sau, hàm này trả về giá trị 1 (đúng) nếu x là số nguyên tố và trả về giá trị 0 (sai) nếu x không là số nguyên tố:
    int NguyenTo (int x);

11. Viết hàm InNguyenTo() sau, hàm này in các số nguyên tố trong khoảng từ số đầu đến số cuối:
    void InNguyenTo(int dau, int cuoi);

12. Viết hàm HoanChinh() sau, hàm này trả về giá trị 1 (đúng) nếu x là số hoàn chỉnh và trả về giá trị 0 (sai) nếu x không là số hoàn chỉnh:
    int HoanChinh (int x);

13. Viết hàm InHoanChinh() sau, hàm này in các số hoàn chỉnh trong khoảng từ số đầu đến số cuối:
    void InHoanChinh (int dau, int cuoi);

14. Viết hàm ChinhPhuong() sau, hàm này trả về giá trị 1 (đúng) nếu x là số chính phương và trả về giá trị 0 (sai) nếu x không là số chính phương:
    int ChinhPhuong (int x);

15. Viết hàm InChinhPhuong () sau, hàm này in các số chính phương trong khoảng từ số đầu đến số cuối:
    void InChinhPhuong (int dau, int cuoi);

16. Viết hàm DoiXung() sau, hàm này trả về giá trị 1 (đúng) nếu x là số đối xứng và trả về giá trị 0 (sai) nếu x không là số đối xứng:
    int DoiXung (int x);

17. Viết hàm InDoiXung () sau, hàm này in các số đối xứng trong khoảng từ số đầu đến số cuối
    void InDoiXung (int dau, int cuoi);

18. Viết hàm TimUSCLN() sau, hàm này trả về ước số chung lớn nhất của hai số nguyên u và v:
    int TimUSCLN (int u, int v);

19. Viết hàm RutGon() sau, hàm này rút gọn một phân số gồm ts và ms:
    void RutGon (int *ts, int *ms);

20. Viết hàm TinhDTr() sau, hàm này trả về diện tích dt và chu vi cv của một đường tròn bán kính r:
    void TinhDTr (double r, double *dt, double *cv);

21. Viết hàm TinhHCN() sau, hàm này trả về diện tích dt và chu vi cv của một hình chữ nhật với chiều dài là d và chiều rộng là r:

    void TinhHCN (double d, double r, double *dt, double *cv);

22. Viết hàm TinhTG() sau, hàm này trả về diện tích dt và chu vi cv của một tam giác với các cạnh a, b và c:

    void TinhTG (double a, double b, double c, double *dt, double *cv);

23. Viết hàm GiaiPTBậc2() sau, hàm này trả về số nghiệm sn và các nghiệm x1, x2 (nếu có) của một phương trình bậc 2 với các hệ số a, b và c:

    void GiaiPTBậc2 (double a, double b, double c, int *sn, double *x1, double *x2);

24. Viết hàm NgayKe() sau, hàm này trả về ngày kế tiếp nd, nm, ny của ngày hiện hành d, m, y:

    void NgayKe(int d, int m, int y, int *nd, int *nm, int *ny);

25. Viết hàm NgayTruoc() sau, hàm này trả về ngày kế trước pd, pm, py của ngày hiện hành d, m, y:
    void NgayTruoc(int d, int m, int y, int *pd, int *pm, int *py);

26. Viết hàm xuất ra hình vuông gồm các dấu *.
27. Viết hàm xuất ra hình vuông rỗng bên trong.
28. Cải tiến bài 26, 27 để được hình chữ nhật.
29. Viết hàm xuất ra các tam giác vuông (nửa trên, nửa dưới đường chéo chính, nửa trên, nửa dưới đường chéo phụ, tam giác cân, cân đảo đầu.
30. Xuất hình cây thông noel.

CHƯƠNG 6: MẢNG MỘT CHIỀU

Đối với các bài tập trong chương này, trong chương trình luôn phải có hai hàm nhập và xuất mảng một chiều nguyên.

1.

a. Tính Tổng các phần tử của mảng.
b. Tính Tổng các phần tử có chữ số tận cùng là 6 và chia hết cho 6.
c. Tính tổng các phần tử có chữ số đầu tiên là chữ số lẽ.
d. Tính tổng các phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.

2.

a. Đếm số phần tử có chữ số tận cùng là 6 và chia hết cho 6.
b. Đếm số phần tử có chữ số đầu tiên là chữ số lẽ.
c. Đếm số phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.
d. Đếm số phần tử phân biệt, các phần tử trùng nhau chỉ đếm một lần.

3.

a. Kiểm tra mảng có chứa số dương hay không?
b. Kiểm tra mảng có gồm toàn số dương hay không?
c. Kiểm tra mảng có thứ tự tăng hay không?
d. Kiểm tra mảng có thứ tự giảm hay không?
e. Kiểm tra mảng có được sắp thứ tự hay không?
f. Kiểm tra mảng có đan xen âm dương hay không?
g. Kiểm tra mảng có đối xứng hay không?
h. Kiểm tra mảng có tất cả các cặp phần tử đứng cạnh nhau có giá trị khác nhau hay không?
i. Kiểm tra mảng có tất cả bộ 3 phần tử đứng cạnh nhau lập thành cấp số cộng hay không?

4.

a. Sắp sếp các phần tử giảm dần.
b. Sắp sếp các phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng tăng dần, các phần tử khác giữ nguyên vị trí.
c. Trộn hai mảng tăng dần lại thành một mảng được sắp thứ tự tăng dần.

5.

a. Tìm vị trí của phần tử lớn nhất/nhỏ nhất.
b. Tìm vị trí của phần tử dương nhỏ nhất/âm lớn nhất.

6.

a. Liệt kê các phần tử có chữ số tận cùng là 6 và chia hết cho 6.
b. Liệt kê các phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.

c. Liệt kê tần suất xuất hiện của các phần tử. Chẳng hạn với mảng 12 34 12 34 43 12 5, thì tần suất xuất hiện các phần tử được cho trong bảng sau:

Phần tử	Tần suất
12	3
34	2
43	1
5	1

7.

a. Thêm một phần tử có giá trị x vào mảng tại vị trí k
b. Thêm một giá trị x vào trong mảng tăng mà vẫn giữ nguyên tính tăng của mảng.

8.

a. Xóa phần tử có chỉ số k.
b. Xóa tất cả phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng trong mảng.

9.

a. Tạo mảng b từ mảng a sao cho mảng b chỉ chứa các phần tử có giá trị lẻ.
b. Tạo mảng b từ mảng a sao cho mảng b chỉ chứa các phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/ số đối xứng.

CHƯƠNG 7: MẢNG HAI CHIỀU

Đối với các bài tập trong chương này, trong chương trình luôn phải có hai hàm nhập và xuất mảng hai chiều nguyên.

1.
   a. Tính tổng/tích các phần tử trên dòng/cột thứ k.
   b. Tính trung bình cộng các phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.
2.
   a. Đếm số phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.
   b. Đếm số phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng trên dòng/cột thứ k.
3.
   a. Kiểm tra ma trận có phần tử dương hay không.
   b. Kiểm tra ma trận có gồm toàn phần tử dương hay không.
   c. Kiểm tra dòng/cột thứ k của ma trận có tăng dần/giảm dần hay không.
4.
   a. Liệt kê các dòng/cột có chứa phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.
   b. Liệt kê các dòng/cột có chứa phần tử gồm toàn là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.
5.
   a. Tìm vị trí phần tử lớn nhất/nhỏ nhất.
   b. Tìm vị trí phần tử lớn nhất/nhỏ nhất trên dòng/cột thứ k.
   c. Tìm vị trí phần tử dương nhỏ nhất/âm lớn nhất trên dòng/cột thứ k.
6.
   a. Sắp xếp các phần tử trên dòng thứ k tăng dần từ trái sang phải.
   b. Sắp xếp các phần tử trên cột thứ k tăng dần từ trên xuống dưới.
   c. Sắp xếp các cột của ma trận sao cho tổng giá trị của mỗi cột tăng dần từ trái sang phải.

Ma trận nhập vào

2	4	6	0
9	1	2	6
Σ 11	**5**	**8**	**6**

Ma trận sau khi sxếp

4	0	6	2
1	6	2	9
5	**6**	**8**	**11**

   d. Sắp xếp ma trận tăng dần trên mỗi dòng từ trái sang phải và từ trên xuống dưới.

Ma trận nhập vào _____ Ma trận sau khi sxếp _____

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

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

7. Mảng vuông

a. Tính tổng/tích các phần tử là là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng nằm trên đường chéo chính/đường chéo phụ.
b. Đếm các phần tử là là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng nằm trên đường chéo chính/đường chéo phụ.
c. Tính tổng/tích các phần tử là là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng nằm trong nửa mảng vuông phía trên đường chéo chính/nửa mảng vuông phía dưới đường chéo chính/nửa mảng vuông phía trên đường chéo phụ/ nửa mảng vuông phía dưới đường chéo phụ.
d. Tìm phần tử lớn nhất/nhỏ nhất trên đường chéo chính/đường chéo phụ.
e. Tìm phần tử âm lớn nhất/dương nhỏ nhất trên đường chéo chính/đường chéo phụ.
f. Sắp xếp các phần tử trên đường chéo chính/đường chéo phụ theo thứ tự tăng dần/giảm dần từ trên hướng xuống/từ dưới hướng lên.

CHƯƠNG 8: CHUỖI

1. Viết các hàm thực hiện:
   - Chuyển đổi một chuỗi sang dạng thường
   - Chuyển đổi một chuỗi sang dạng hoa.
   - Chuyển đổi một chuỗi sang dạng Title case (ký tự đầu của mỗi từ là chữ hoa, các ký tự còn lại là chữ thường).
2. Viết hàm kiểm tra một chuỗi có đối xứng hay không?
   Ví dụ: Các chuỗi level, radar, dad, … là các chuỗi đối xứng
3. Viết hàm tìm chuỗi đảo ngược của một chuỗi.
   Ví dụ: Chuỗi "Lap trinh C" có chuỗi đảo ngược là "C hnirt paL"
4. Viết hàm đếm số từ trong một chuỗi.
   Ví dụ: Chuỗi " Lap trinh C " có 3 từ
5. Viết hàm in mỗi từ của chuỗi trên một dòng.
   Ví dụ: Chuỗi " Lập trình C" sẽ được in ra:

   Lập

   trinh

   C
6. Viết hàm xóa các khoảng trắng dư thừa ở bên trái, bên phải và giữa chuỗi.
7. Viết hàm thống kê số lần xuất hiện của mỗi ký tự từ A đến Z trong một chuỗi (không phân biệt chữ hoa hay chữ thường).
   Ví dụ: Với chuỗi "Hello World" ta sẽ có:

   D: 1 lần

   E: 1 lần

   H: 1 lần

   L: 3 lần

   ...

CHƯƠNG 9: CẤU TRÚC

CẤU TRÚC

1. Khai báo kiểu dữ liệu biểu diễn thông tin của một phân số. Hãy viết hàm thực hiện các yêu cầu sau:
   a. Nhập phân số.
   b. Xuất phân số.
   c. Rút gọn phân số.
   d. Tính tổng hai phân số.
   e. Tính hiệu hai phân số.
   f. Tính tích hai phân số.
   g. Tính Thương hai phân số.
   h. So sánh hai phân số (hàm trả về một trong 3 giá trị: -1, 0, 1).
   i. Kiểm tra phân số âm.
   j. Kiểm tra phân số dương.
2. Khai báo kiểu dữ liệu biểu diễn thông tin của một số phức. Hãy viết hàm thực hiện các yêu cầu sau:
   a. Nhập số phức.
   b. Xuất số phức (theo định dạng Re + Im*i).
   c. Tính modul số phức.
   d. So sánh hai số phức (hàm trả về một trong 3 giá trị: 0, -1, 1).
   e. Tính tổng hai số phức.
   f. Tính hiệu hai số phức.
   g. Tính tích hai số phức.
   h. Tính thương hai số phức.
3. Khai báo kiểu dữ liệu biểu diễn thông tin điểm trong mặt phẳng. Hãy viết hàm thực hiện các yêu cầu sau:
   a. Nhập điểm
   b. Xuất điểm theo định dạng (x, y)
   c. Tính khoảng cách giữa hai điểm.
   d. Tính khoảng cách giữa hai điểm theo phương Ox
   e. Tính khoảng cách giữa hai điểm theo phương Oy
   f. Tìm điểm đối xứng qua gốc tọa độ.
   g. Tìm điểm đối xứng qua trục hoành.
   h. Tìm điểm đối xứng qua trục tung.
   i. Tìm điểm đối xứng qua đường phân giác thứ nhất (y=x).
   j. Tìm điểm đối xứng qua đường phân giác thứ hai (y=-x).
   k. Kiểm tra điểm có thuộc phần tư thứ I không?
   l. Kiểm tra điểm có thuộc phần tư thứ II không?
   m. Kiểm tra điểm có thuộc phần tư thứ III không?
   n. Kiểm tra điểm có thuộc phần tư thứ IV không?

4. Khai báo kiểu dữ liệu biểu diễn thông tin đường tròn. Hãy viết hàm thực hiện các yêu cầu sau:
   a. Nhập đường tròn.
   b. Xuất đường tròn theo định dạng ((x, y), r).
   c. Tính chu vi đường tròn.
   d. Tính diện tích đường tròn.
   e. Xét vị trí tương đối giữa hai đường tròn (trùng nhau, ở trong nhau, tiếp xúc, cắt nhau, nằm ngoài nhau).
   f. Kiểm tra một điểm có nằm trên đường tròn không?

   Nhắc lại:

   Nếu kc(o1, o2) = 0 và o1 = o2 thì hai đường tròn trùng nhau.

   Nếu abs(r1 – r2) > kc(o1, o2) thì hai đường tròn ở trong nhau.

   Nếu r1 + r2 = kc(o1, o2) hoặc abs(r1 – r2) = kc(o1, o2) thì hai đường tròn tiếp xúc nhau.

   Nếu r1 + r2 > kc(o1, o2) và abs(r1 – r2) < kc(o1, o2) thì hai đường tròn cắt nhau.

   Nếu abs(r1 – r2) < kc(o1, o2) thì hai đường tròn ở ngoài nhau

5. Khai báo kiểu dữ liệu biểu diễn thông tin một tam giác. Hãy viết hàm thực hiện các yêu cầu sau:
   a. Nhập tam giác.
   b. Xuất tam giác theo định dạng ((x1, y1); (x2, y2); (x3, y3)).
   c. Kiểm tra 3 đỉnh có lập thành 3 đỉnh của một tam giác không?
   d. Tính chu vi tam giác.
   e. Tính diện tích tam giác.
   f. Tính tọa độ trọng tâm tam giác.
   g. Phân loại tam giác.
   h. Tìm một đỉnh có hoành độ lớn nhất.
   i. Tìm một đỉnh có tung độ nhỏ nhất.
   j. Tính tổng khoảng cách từ điểm p(x, y) tới 3 đỉnh tam giác.
6. Khai báo kiểu dữ liệu biểu diễn thông tin ngày. Hãy viết hàm thực hiện các yêu cầu sau:
   a. Nhập ngày.
   b. Xuất ngày theo định dạng (dd/mm/yy).
   c. Kiểm tra ngày hợp lệ.
   d. Tìm ngày kế tiếp.
   e. Tìm ngày trước đó.
7. Khai báo kiểu dữ liệu biểu diễn thông tin giờ. Hãy viết hàm thực hiện các yêu cầu sau:
   a. Nhập giờ.
   b. Xuất giờ theo định dạng (hh:mm:ss).
   c. Kiểm tra giờ hợp lệ.
   d. Tính tổng hai giờ.

8. Khai báo kiểu dữ liệu để biểu diễn thông tin của một chuyến bay. Biết rằng một chuyến bay gồm các thành phần như sau:
   Mã chuyến bay: chuỗi tối đa 5 ký tự.

   Ngày bay: kiểu ngày.

   Giờ bay: Kiểu giờ.

   Nơi đi: chuỗi tối đa 20 ký tự.

   Nơi đến: chuỗi tối đa 20 ký tự.

   Sau đó viết hàm nhập và hàm xuất cho kiểu dữ liệu này.

9. Khai báo kiểu dữ liệu để biểu diễn thông tin của một học sinh. Biết rằng một học sinh gồm các thành phần như sau:
   Mã sinh viên: chuỗi tối đa 10 ký tự.

   Tên sinh viên: chuỗi tối đa 30 ký tự.

   Ngày sinh: Kiểu dữ liệu ngày.

   Điểm: kiểu thực

   Sau đó viết hàm nhập và hàm xuất cho kiểu dữ liệu này.

10. Khai báo kiểu dữ liệu để biểu diễn thông tin của một phiếu hàng. Biết rằng một phiếu hàng gồm các thành phần như sau:
    Người mua: chuỗi tối đa 30 ký tự.

    Mã hàng: chuỗi tối đa 10 ký tự.

    Đơn giá: Kiểu số thực.

    Số lượng: kiểu nguyên.

    Sau đó viết hàm nhập và hàm xuất cho kiểu dữ liệu này.

MẢNG CẤU TRÚC

11. Mảng điểm
    a. Viết hàm nhập mảng.
    b. Viết hàm xuất mảng.
    c. Đếm số lượng điểm có hoành độ dương.
    d. Tìm một điểm có tung độ lớn nhất trong mảng.
    e. Tìm một điểm trong mảng gần gốc tọa độ nhất.
    f. Tìm hai điểm gần nhau nhất trong mảng.

g. Tìm hai điểm xa nhau nhất trong mảng.

12. Mảng phân số
    a. Viết hàm nhập mảng.
    b. Viết hàm xuất mảng.
    c. Chuyển tất cả các phân số về dạng tối giản.
    d. Tìm phân số dương nhỏ nhất/phân số âm lớn nhất.
    e. Sắp xếp các phân số theo thứ tự mẫu số giảm dần.
13. Mảng các chuyến bay
    a. Nhập mảng.
    b. Xuất mảng.
    c. Sắp xếp các chuyến bay tăng dần theo ngày bay.
14. Mảng các học sinh
    a. Nhập mảng.
    b. Xuất mảng.
    c. Tìm và in các học sinh có điểm cao nhất
    d. Sắp xếp các học sinh tăng dần theo mã.
15. Mảng các phiếu hàng
    a. Nhập mảng.
    b. Xuất mảng.
    c. Tính tổng số tiền bán được theo từng mã hàng.

HƯỚNG DẪN GIẢI MỘT SỐ BÀI

CHƯƠNG 2: CÁC KIỂU DỮ LIỆU CƠ SỞ

1. Nhập vào 2 số nguyên. Tính và in ra tổng, hiệu, tích, thương và dư của hai số nguyên này.
2. Nhập vào bán kính của hình tròn. Tính và in ra chu vi, diện tích theo công thức sau:
   CV = 2*PI*r

   DT = PI*r*r
3. Nhập vào chiều dài và chiều rộng của hình chữ nhật. Tính và in ra chu vi, diện tích theo công thức sau:
   CV = (a+b)*2

   DT = a*b
4. Nhập vào 3 cạnh của hình tam giác. Tính và in ra chu vi, diện tích theo công thức sau:
   CV = a+b+c

   DT = $(p*(p-a)*(p-b)*(p-c))^{1/2}$

   với p là nửa chu vi.
5. Nhập vào bán kính hình cầu. Tính và in ra thể tích, diện tích theo công thức sau:
   TT = $4*PI*r^3 / 3$;

   DT = $4*PI*r^2$
6. Nhập vào độ Ferenheit. Tính và in ra độ celsius theo công thức sau:
   C = 5/9 * (F – 32).

   Lưu ý: Tránh mất dữ liệu trong quá trình tính toán.
7. Nhập chiều dài con lắc đơn. Tính và in ra chu kỳ con lắc đơn theo công thức:
   $T = 2*PI*(l/g)^{1/2}$

   l: chiều dài con lắc đơn

   g: gia tốc trọng trường (9.81 m/s^2)
8. Nhập khối lượng hai vật thể và khoảng cách giữa chúng. Tính và in ra lực vạn vật hấp dẫn theo công thức:

$F = G*(m1*m2)/r^2$

G: hằng số hấp dẫn ($6.67*10^{-11}$ Nm^2/kg^2)

m1, m2: Khối lượng hai vật thể.

r: khoảng cách hai vật.

a. Lưu ý:

#define G 6.67E-11 //Cách viết khoa hoc của $6.67*10^{-11}$

9. Nhập vào một số nguyên n và một số thực x. Tính và in ra biểu thức $(x^2 + 1)^n$.
10. Nhập vào một số nguyên n và một số thực x. Tính và in ra biểu thức
    $A = (x^2 + x + 1)^n + (x^2 + x - 1)^n$
11. Nhập vào một số tiền nguyên dương. Đổi số tiền này ra các tờ giấy bạc 50đ, 20đ, 10đ, 5đ và 1đ. Với giả thiết ưu tiên cho tờ có mệnh giá lớn hơn, hãy in ra xem đổi được bao nhiêu tờ mỗi loại.
12. Nhập vào một số nguyên dương có đúng 3 chữ số. Tính và in ra số đảo ngược.

CHƯƠNG 3: CÁC CÂU LỆNH RẼ NHÁNH

1. Nhập vào một ký tự. Hãy in thông báo cho biết ký tự đó là ký tự số, ký tự chữ cái, ký tự phép toán hay là ký tự dạng khác với các dạng trên.
   Lưu ý: Kí tự phải để trong dấu nháy đơn. Ví dụ: 'a' là kí tự a (có mã ASCII là 97).
2. Nhập vào hai số thực a, b. Hãy tính và in ra nghiệm phương trình bậc nhất (a bất kì).

   ax + b = 0.

3. Nhập vào hai số nguyên. Hãy mô phỏng một máy tính đơn giản gồm 5 phép tính hai ngôi (+, -, *, /, %) trên hai số nguyên này.
   Lưu ý: Trước câu lệnh đọc kí tự phải thêm câu lệnh xóa vùng nhớ đệm là: fflush(stdin);
4. Nhập vào ba số thực a, b, c (a bất kì). Hãy tính và in ra nghiệm phương trình bậc hai:

   $ax^2 + bx + c = 0.$

5. Nhập vào sáu số thực a, b , c, d, m, n. Hãy tính và in ra nghiệm hệ phương trình hai ẩn số:

   $ax + by = m$

   $cx + dy = n$

   Nhắc lại:

   Nghiệm: $x = DX/D$, $y = DY/D$

   với $DX = md - nb$, $DY = an - cm$ và $D = ad - cb$.

6. Nhiệt độ F(Fahrenheit) và nhiệt độ C(Celcius) liên hệ với nhau theo công thức: $C = 5(F - 32)/9$. Viết chương trình cho phép người dùng nhập vào độ F hay độ C và đổi sang độ còn lại.

   a. Hướng dẫn:

```
double f, c;
int chon;          //Chức năng chọn
printf("BANG CHON\n");
printf("1: Nhap F, tinh C\n");
printf("2: Nhap C, tinh F\n");
printf("Moi chon:"); scanf("%d", &chon);
switch(chon) {
case 1:
        <Nhap f>;
        <Tinh c>;
        <In c>;
        break;
case 2:
        <Nhap c>;
```

```
            <Tinh f>;
            <In f>;
            break;
    default: <Ban chon khong hop le>
    }
```

7. Nhập vào các số thực a, b, c. Hãy kiểm tra xem ba số này có lập thành 3 cạnh của một tam giác hay không? Nếu có hãy tính và in ra chu vi, diện tích, chiều dài mỗi đường cao của tam giác.
8. Nhập vào ba số nguyên d, m, y là ngày, tháng, năm. Hãy in thông báo cho biết đó là ngày thứ mấy trong tuần. Sử dụng công thức sau để biến đổi ngày, tháng, năm thành thứ trong tuần:

   $A = d + 2m + (3(m + 1)/5) + y + (y/4) – (y /100) + (y/400) + 2$

   với quy ước tháng 1, 2 của năm y được xem là tháng 13, 14 của năm y–1. Số dư trong phép chia A cho 7 cho kết quả là thứ trong tuần theo nghĩa số dư là 0: Thứ bảy, số dư là 1: Chủ nhật, số dư là 2: Thứ hai. v.v…

9. Nhập vào ngày, tháng, năm. Hãy kiểm tra tính hợp lệ của ngày tháng năm nhập vào. Nếu hợp lệ hãy cho biết tháng nhập có bao nhiêu ngày, ngày hôm sau của ngày đã nhập là ngày nào.
   (Biết rằng năm nhuận là năm (chia hết cho 400) hoặc (chia hết cho 4 và không chia hết cho 100)).

10. Nhập vào giờ vào ca, giờ ra ca. Hãy tính và in ra tiền lương ngày cho công nhân, biết rằng tiền trả cho mỗi giờ trước 12 giờ trưa là 6000đ và mỗi giờ sau 12 giờ trưa là 7500đ. Giờ vào ca sớm nhất là 6 giờ sáng và giớ ra ca trễ nhất là 18 giờ.
11. Nhập vào trọng lượng m(kg) hàng hóa bán được ($0 < m \leq 100$). Hãy tính và in ra tiền lời thu được, biết rằng

    $0 < m \leq 10$ Tiền lời là 5000đ/kg.

    $10 < m \leq 20$ Tiền lời là 5000đ/kg.

    $20 < m \leq 50$ Tiền lời là 9000đ/kg và thêm 2% tổng số tiền lời.

    $50 < m$ Tiền lời là 10000đ/kg và thêm 4% tổng số tiền lời nhưng không được quá 1000000đ.

12. Nhập vào số ngày thuê và loại phòng (một trong 3 loại A, B hoặc C). Hãy tính và in ra tiền thuê phòng với quy định như sau:

    Loại A: 250000đ/ngày

    Loại B: 200000đ/ngày

    Loại C: 150000đ/ngày

    Nếu thuê quá 12 ngày thì phần trăm được giảm trên tổng số tiền (tính theo giá quy định) là: 10% cho phòng loại A, 8% cho phòng loại B hoặc C.

b. Hướng dẫn:

```
#define DON_GIA_A     250000.0
#define DON_GIA_B     200000.0
#define DON_GIA_C     150000.0
#define GIAM_A        0.1
#define GIAM_B_C      0.08
#define GIOI_HAN      12
int snt;          //Số ngày thuê
char lp;          //Loại phòng
double tien;      //Tiền thuê phòng
<Nhap snt>
fflush(stdin); //Xóa dữ liệu trong vùng đệm
<Nhap lp>
if(snt > 0) {
        switch(lp) {
        case 'a': case 'A':
                tien =  DON_GIA_A * snt;
                if(snt > GIOI_HAN)
                        tien *= (1 - GIAM_A);
                <In tien>
                break;
        case 'b': case 'B': ...
        case 'c': case 'C': ...
        default: <Loai phong khong hop le>
        }
}
else <So ngay thue khong hop le>
```

13. Trong một kỳ thi tuyển, hội đồng tuyển sinh đã xem xét đề nghị một điểm chuẩn (xem như dữ kiện nhập). Mỗi thí sinh tham gia kỳ thi sẽ trúng tuyển nếu điểm tổng kết của thí sinh đó lớn hơn hoặc bằng điểm chuẩn và không có môn nào điểm 0. Điểm tổng kết của mỗi thí sinh là tổng điểm của 3 môn thi và điểm ưu tiên. Điểm ưu tiên bao gồm điểm ưu tiên theo khu vực và điểm ưu tiên theo đối tượng dự thi. Cho biết:

    Khu vực A: điểm ưu tiên là 2.

    Khu vực B: điểm ưu tiên là 1.

    Khu vực C: điểm ưu tiên là 0.5.

    Đối tượng 1: điểm ưu tiên là 2.5.

    Đối tượng 2: điểm ưu tiên là 1.5.

    Đối tượng 3: điểm ưu tiên là 1.

Viết chương trình nhập vào điểm chuẩn, điểm của 3 môn thi của một thí sinh, khu vực và đối tượng dự thi. Cho biết thí sinh đó đậu hay rớt.

CHƯƠNG 4: CÁC CÂU LỆNH LẶP

16. Nhập vào số nguyên dương n. Hãy tính và in ra các tổng sau:
    e. s = 1*2 + 2*3 + … + n*(n + 1)
    f. s = $1^1 + 2^2 + … + n^n$
    g. s = 1 + 1/2+ 1/3 + … +1/n

c. Hướng dẫn câu 1 b:

```
s = 0;
for(i = 1; i <= n; i++)
        s += pow(i, i);
```

17. Nhập vào một số nguyên dương n. Hãy tính và in ra giá trị của vế trái, vế phải của mỗi công thức sau đây và cho biết kết quả tính của hai vế có bằng nhau hay không?
    e. 1.2 + 2.2 + ... + n.2 = n(n + 1)(2n + 1) / 6
    f. 1.2.3 + 2.3.4 + … + n(n + 1)(n + 2) = n(n + 1)(n + 2)(n + 3)/4
    g. 1(1!) + 2(2!) + … + n(n!) = (n + 1)! – 1
    h. 1/2! + 2/3! + … + n/(n + 1)! = 1 – (1/(n + 1)!)

d. Hướng dẫn câu 2a

```
//Tính vế trái

vt = 0;

for(i = 1; i <= n; i++)

        vt += i * 2;

//Tính vế phải

vp = n * (n + 1) * (2*n + 1) / 6;

<In vt, vp>;

if(vt == vp)

        <Hai ve bang nhau>

else <Hai ve khong bang nhau>
```

e. Hướng dẫn câu 2d

```
#define E       0.0001 //Độ chính xác
//Tính vế trái
vt = 0;
gt = 1;
for(i = 1; i <= n; i++) {
        gt *= (i + 1);
```

```
        vt += (double)i/gt;
    }
    //Tính vế phải
    vp = 1 – (1.0/gt);
    //So sánh vế trái và vế phải
    if(fabs(vt – vp) < E) /*Không nên so sánh theo cách vt == vp vì vt và
    vp mang giá trị thực*/
        <Hai ve bang nhau>
    else <Hai ve khong bang nhau>
```

18. Nhập vào x (tính theo radian). Hãy tính và in ra:
    f. $e(x) = 1 + x + x^2/2 ! + ... + x^n/n !$ (độ chính xác là 0.0001, tức là e(x) được tính đến số hạng sao cho trị tuyệt đối của số hạng đó nhỏ hơn 0.0001)
    g. $\sin(x) = x – x^3/3! + x^5/5! - ... + (-1)^n(x^{2n+1}/(2n+1)!)$ (độ chính xác là 0.0001)
    h. $\cos(x) = 1 – x^2/2! + x^4/4! - ... + (-1)^n(x^n/2n!)$ (độ chính xác là 0.0001)

f. Hướng dẫn câu 3b

```
#define E       0.0001 //độ chính xác
<Nhập x>
k = 1; y = x; sin = x;
while(fabs(y) >= E) {
        k += 2;
        y = -y * x * x / ((k – 1) * k);
        sin += y;
 }
<In sin>
```

19. Hãy in bảng mã ASCII ra màn hình. Bảng được in ra theo từng trang màn hình sao cho người sử dụng có thể đọc được.

g. Hướng dẫn:

```
#define NUM_LINES  40
for(i = 0; i < 256; i++) {
      printf("%c: %d\n", i, i);
      if((i + 1) % NUM_LINES == 0) {
          printf("** Nhan phim bat ky de tiep tuc **\n";
          getch(); //Dừng màn hình
      }
}
```

20. Hãy in toàn bộ bảng cửu chương từ hai đến chín ra màn hình. Bảng được in ra theo từng trang màn hình sao cho người sử dụng có thể đọc được.

h. Hướng dẫn:

```
for(i = 2; i < 10; i++) {
    printf("** BANG CUU CHUONG %d**\n", i);
```

```
        for(j = 1; j < 10; j++)
            printf("%d * %d = %d\n", i, j, i * j);
    }
```

21. Nhập vào một số nguyên. Hãy kiểm tra xem số nguyên đó có phải là số nguyên tố hay không? Biết rằng một số nguyên dương (khác số 1) được gọi là số nguyên tố nếu ngoài 1 và chính nó ra thì nó không chia hết cho một số nguyên dương nào khác. Chẳng hạn các số sau là nguyên tố:

    2, 3, 5, 7, 11, 13, . . .

    i. Hướng dẫn:

```
int n;
int nt;   //0: Không là nguyên tố, 1: Là nguyên tố (cờ hiệu)
<Nhap n>
if(n < 2)
        nt = 0; //Không là số nguyên tố
else {
        nt = 1; //Giả sử là số nguyên tố
        for(i = 2; i < n; i++)
                if(n % i == 0) {
                        nt = 0; //Không là số nguyên tố
                        break;
                }
}
//In kết quả
if(nt) <Day la so nguyen to>
else <Day khong la so nguyen to>
```

22. Nhập vào số nguyên dương n. Hãy tìm và in ra các số nguyên tố trong khoảng từ 1 đến n.

    j. Hướng dẫn:

```
for(i = 1; i <= n; i++) {
    //Kiểm tra i có là nguyên tố
    if(i < 2)
        nt = 0;   //Không là số nguyên tố
    else {
        nt = 1; //Giả sử là số nguyên tố
        for(j = 2; j < i; j++)
                if(i % j == 0) {
                        nt = 0; //Không là số nguyên tố
                        break;
                }
    }
    if(nt)  printf("%d\t", i);
}
```

23. Nhập vào một số nguyên. Hãy kiểm tra xem một số nguyên đó có phải là số hoàn chỉnh hay không? Biết rằng một số nguyên dương n được gọi là số hoàn chỉnh nếu tổng các ước số của n (không kể n) bằng chính nó. Chẳng hạn các số sau là hoàn chỉnh:

    6, 28, 496, ...

    k. Hướng dẫn:

```
int n;
int hc; //0: Không là số hoàn chỉnh, 1: Là số hoàn chỉnh (cờ hiệu)
<Nhap n>
if(n < 1)
    hc = 0; //Không là số hoàn chỉnh
else {
        //Tính tổng các ước số của n
        s = 0;
        for(i = 1; i < n; i++)
                if(n % i == 0)   s += i;
        //Kiểm tra
        hc = (s == n ? 1 : 0);
}
//In kết quả
if(hc)
        <Day la so hoan chinh>
else <day khong la so hoan chinh>
```

24. Nhập vào số nguyên dương n. Hãy tìm và in ra các số hoàn chỉnh trong khoảng từ 1 đến n.
25. Nhập vào một số nguyên. Hãy kiểm tra xem số nguyên đó có phải là số chính phương hay không? Biết rằng một số nguyên dương n được gọi là số chính phương nếu nó bằng bình phương của một số nguyên. Chẳng hạn các số sau là chính phương:

    1, 4, 9, 16, 25, ...

    l. Hướng dẫn:

```
int n;
int cp;  //0: Không là số cphương, 1: Là số cphương (cờ hiệu)
<Nhap n>
if(n < 1)
        cp = 0; // Không là số chính phương
else {
    cp = 0; //Giả sử không là số chính phương
    for(i = 1; i <= sqrt(n); i++)
                if(n == i * i) {
                        cp = 1; //Là số chính phương
                        break;
```

```
                }
    }
    //In kết quả
    if(cp) <Day la so chinh phuong >
    else <Day khong la so chinh phuong >
```

26. Nhập vào số nguyên dương n. Hãy tìm và in ra các số chính phương trong khoảng từ 1 đến n.
27. Nhập vào một số nguyên. Hãy kiểm tra xem số nguyên đó có phải là số đối xứng hay không? Biết rằng một số nguyên n được gọi là đối xứng nếu số đảo của n bằng chính n. Chẳng hạn các số sau là đối xứng:

    1, 44, 161, 2552, ...

    m. Hướng dẫn:

```
int n;
int m;   //Lưu lại giá trị của n
int dao; //Số đảo
int dx;  //0: Không là số đối xứng, 1: Là số đối xứng (cờ hiệu)
int d;
<nhap n>

m = n;
//Tìm số đảo của m
dao = 0;
while(m > 0) {
    d = m % 10;  // Lấy chữ số phải nhất của m
    dao = 10*dao + d; // Thêm chữ số phải nhất vào dao
    m /= 10; // Xóa chữ số phải nhất của m
}
dx = (dao == n ? 1: 0);
//Kiểm tra đối xứng
if(dx == 1)
    <Day la so doi xung>
else <Day la so khong doi xung>
```

28. Nhập vào số nguyên dương n. Hãy tìm và in ra các số đối xứng trong khoảng từ 1 đến n.
29. Nhập vào hai số nguyên a và b. Hãy tìm và in ra ước số chung lớn nhất của a và b. Áp dụng giải thuật **Euclide** sau (chỉ áp dụng cho hai số nguyên dương):

    (1) Lấy số lớn trừ số nhỏ

    (2) Đặt số lớn bằng hiệu số của phép trừ ở bước (1)

    (3) Lặp lại bước (1) cho đến khi hai số bằng nhau

    Giá trị bằng nhau đạt được chính là ước số chung lớn nhất của hai số đã cho.

n. Hướng dẫn:

```
int a, b;
int uscln;          //Ước số chung lớn nhất
<Nhap a, b>
if(a < 0)           a = -a;
if(b < 0)           b= - b;
if(a == 0 || b == 0)
        uscln = a + b;
else {
        while(a != b) {
                if(a > b) a = a – b;
                else b = b – a;
        }
        uscln = a; //hoăc b
}
<In uscln>
```

CHƯƠNG 5: HÀM

24. Viết hàm DienTich() sau, hàm này trả về diện tích một đường tròn với bán kính r:

```
double DienTich (double r);
```

o. Hướng dẫn:

```
double DienTich (double r) {
        return   r * r * PI;
}
```

25. Viết hàm Min() sau, hàm này trả về giá trị nhỏ nhất của hai số nguyên x và y:

```
int Min(int x, int y);
```

26. Viết hàm Min() sau, hàm này trả về giá trị nhỏ nhất của ba số nguyên x, y và z:

```
int Min(int x, int y, int z);
```

27. Viết hàm LuyThua() sau, hàm này trả về giá trị x^p:

```
double LuyThua (double x, int p);
```

p. Hướng dẫn:

```
double LuyThua(double x, int p) {
        int i ;
        double ret = 1 ;
        for(i = 1 ; i <= p ; i++)
                ret *= x ;
        return ret ;
}
```

28. Viết hàm ToHop() sau, hàm này trả về C(n, k) là số tổ hợp n lấy k (n > 0, 0 <= k <= n) bằng cách dựa vào công thức C(n, k) = n! / (k!(n – k)!):

```
long ToHop (int n, int k);
```

q. Hướng dẫn:

```
long GiaiThua(int u) {
        long ret = 1;
        int i;
        for(i = 1; i <= u; i++)
                ret *= i;
        return ret;
}
long ToHop(int n, int k) {
        return GiaiThua(n)/(GiaiThua(k)*GiaiThua(n-k));
}
```

29. Viết hàm InTGPascal() sau, hàm này xuất ra một tam giác Pascal có sd dòng. Hình sau đây là một tam giác Pascal có 7 dòng

	0	1	2	3	4	5	6
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1

Mỗi số trong tam giác Pascal là một tổ hợp C(n, k). Nếu chúng ta tính dòng và cột bắt đầu bằng 0, thì số ở dòng n và cột k là C(n, k). Ví dụ, số ở dòng 6 và cột 4 là C(6, 4) = 15

```
void InTGPascal(int sd);
```

r. Hướng dẫn:

```
void InTGPascal(int sd) {
    int i, j;
    for(i = 0; i < sd; i++) {
        for(j = 0; j <= i; j++)
            printf("%d\t", ToHop(i, j));
        printf("\n");
    }
}
```

30. Viết hàm Nhuan() sau, hàm này trả về giá trị 1 (đúng) nếu năm y nhuần và trả về giá trị 0 (sai) nếu năm y không nhuần:

```
int Nhuan (int y);
```

s. Hướng dẫn:

```
int Nhuan(int y) {
        if(y % 400 == 0 || (y % 4 == 0 && y % 100 != 0))
                return 1 ;
        else return 0 ;
}
```

31. Viết hàm SNTrongThang () sau, hàm này trả về số ngày trong tháng m và năm y:

int SNTrongThang (int m, int y);

32. Viết hàm HopLe() sau, hàm này trả về giá trị 1 (đúng) nếu ngày d, tháng m, năm y hợp lệ và trả về giá trị 0 (sai) nếu d, m, y không hợp lệ:

int HopLe(int d, int m, int y);

t. Hướng dẫn :

```
int HopLe(int d, int m, int y) {
        if(d > 0 && d <= SNTrongThang(m, y) &&
                m > 0 && m < 13 && y > 0)
                return 1;
        else return 0;
}
```

33. Viết hàm NguyenTo() sau, hàm này trả về giá trị 1 (đúng) nếu x là số nguyên tố và trả về giá trị 0 (sai) nếu x không là số nguyên tố:

int NguyenTo (int x);

u. Hướng dẫn:

```
int NguyenTo(int x) {
  int nt; //biến cờ = 1: x là nguyên tố, 0: x không là nguyên tố
  int i;
  if(x < 2)
        nt = 0; //Không là nguyên tố
  else {
        nt = 1; //Là nguyên tố
        for(i = 2; i < x; i++)
                if(x % i == 0) {
                        nt = 0; //Không là nguyên tố
                        break;
                }
  }
  return nt;
}
```

34. Viết hàm InNguyenTo() sau, hàm này in các số nguyên tố trong khoảng từ số đầu đến số cuối:

void InNguyenTo(int dau, int cuoi);

v. Hướng dẫn:

```
void InNguyenTo(int dau, int cuoi) {
        int i;
        for(i = dau; i <= cuoi; i++)
                if(NguyenTo(i))  printf("%d\t", i);
}
```

35. Viết hàm HoanChinh() sau, hàm này trả về giá trị 1 (đúng) nếu x là số hoàn chỉnh và trả về giá trị 0 (sai) nếu x không là số hoàn chỉnh:

```
int HoanChinh (int x);
```

w. Hướng dẫn:

```
int Hoanchinh(int x) {
    int hc; //Biến cờ
    int s;
    if(x <= 0)
        hc = 0;
    else {
        s = 0;
        for(i = 1; i < x; i++)
            if(x % i == 0) s += i;
        hc = (s == x ? 1: 0);
    }
    return hc;
}
```

36. Viết hàm InHoanChinh() sau, hàm này in các số hoàn chỉnh trong khoảng từ số đầu đến số cuối:

```
void InHoanChinh (int dau, int cuoi);
```

37. Viết hàm ChinhPhuong() sau, hàm này trả về giá trị 1 (đúng) nếu x là số chính phương và trả về giá trị 0 (sai) nếu x không là số chính phương:

```
int ChinhPhuong (int x);
```

38. Viết hàm InChinhPhuong () sau, hàm này in các số chính phương trong khoảng từ số đầu đến số cuối:

```
void InChinhPhuong (int dau, int cuoi);
```

39. Viết hàm DoiXung() sau, hàm này trả về giá trị 1 (đúng) nếu x là số đối xứng và trả về giá trị 0 (sai) nếu x không là số đối xứng:

```
int DoiXung (int x);
```

x. Hướng dẫn:

```
int DoiXung(int x) {
    int dx;       //Biến Cờ
    int dao;      //Lưu số đảo
    int y;
    y = x;
    //Tìm số đảo của y
    dao = 0;
    while(y > 0) {
```

```
        d = y % 10;          // Lấy chữ số phải nhất của y
        dao = 10*dao + d;    // Thêm chữ số phải nhất vào dao
        y /= 10;             // Xóa chữ số phải nhất của y
    }
    dx = (dao == x ? 1: 0);
    return dx ;
}
```

40. Viết hàm InDoiXung () sau, hàm này in các số đối xứng trong khoảng từ số đầu đến số cuối

    void InDoiXung (int dau, int cuoi);

41. Viết hàm TimUSCLN() sau, hàm này trả về ước số chung lớn nhất của hai số nguyên u và v:

    int TimUSCLN (int u, int v);

42. Viết hàm RutGon() sau, hàm này rút gọn một phân số gồm ts và ms:

    void RutGon (int *ts, int *ms);

y. Hướng dẫn:

```
void RutGon (int *ts, int *ms) {
    int uscln = TimUSCLN(*ts, *ms);
    *ts /= uscln;
    *ms /= uscln;
}
```

43. Viết hàm TinhDTr() sau, hàm này trả về diện tích dt và chu vi cv của một đường tròn bán kính r:

    void TinhDTr (double r, double *dt, double *cv);

z. Hướng dẫn:

```
void TinhDTr (double r, double *dt, double *cv) {
    *dt = r * r * PI;
    *cv = 2 * r * PI;
}
```

44. Viết hàm TinhHCN() sau, hàm này trả về diện tích dt và chu vi cv của một hình chữ nhật với chiều dài là d và chiều rộng là r:

    void TinhHCN (double d, double r, double *dt, double *cv);

45. Viết hàm TinhTG() sau, hàm này trả về diện tích dt và chu vi cv của một tam giác với các cạnh a, b và c:

    void TinhTG (double a, double b, double c, double *dt, double *cv);

46. Viết hàm GiaiPTBậc2() sau, hàm này trả về số nghiệm sn và các nghiệm x1, x2 (nếu có) của một phương trình bậc 2 với các hệ số a, b và c:

    void GiaiPTBậc2 (double a, double b, double c, int *sn, double *x1, double *x2);

aa. Hướng dẫn:

```
void GiaiPTBậc2 (double a, double b, double c, int *sn, double *x1, double *x2)  {
    double delta = b*b – 4*a*c ;
    if(delta < 0)
        *sn = 0
    else if(delta == 0) {
        *sn = 1;
        *x1 = *x2 = -b / (2*a);
    }
    else {
        *sn = 2;
        *x1 = (-b + sqrt(delta)) / (2*a);
        *x2 = (-b - sqrt(delta)) / (2*a);
    }
}
```

31. Viết hàm NgayKe() sau, hàm này trả về ngày kế tiếp nd, nm, ny của ngày hiện hành d, m, y:

    void NgayKe(int d, int m, int y, int *nd, int *nm, int *ny);

Hướng dẫn:

```
void NgayKe(int d, int m, int y, int *nd, int *nm, int *ny) {
    *nd = d + 1, *nm = m; *ny = y;
    if(*nd > SNTrongThang(*nm, *ny)) {
        *nd =1;
        (*nm)++ ;
        if(*nm > 12) {
                *nm = 1 ;
                (*ny)++ ;
        }
    }
}
```

32. Viết hàm NgayTruoc() sau, hàm này trả về ngày kế trước pd, pm, py của ngày hiện hành d, m, y:

    void NgayTruoc(int d, int m, int y, int *pd, int *pm, int *py);

Hướng dẫn:

```
void NgayTruoc(int d, int m, int y, int *pd, int *pm, int *py) {
    *pd = d - 1, *pm = m; *py = y;
    if(*pd < 1) {
                (*pm)-- ;
                if(*pm < 1) {
                        *pm = 12 ;
                        (*py)-- ;
```

```
                }
            *pd = SNTrongThang(*pm, *py);
        }
    }
```

CHƯƠNG 6: MẢNG MỘT CHIỀU

Đối với các bài tập trong chương này, trong chương trình luôn phải có hai hàm nhập và xuất mảng một chiều nguyên.

1.

e. Tính Tổng các phần tử.
f. Tính Tổng các phần tử có chữ số tận cùng là 6 và chia hết cho 6.
g. Tính tổng các phần tử có chữ số đầu tiên là chữ số lẽ.
h. Tính tổng các phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.

Hướng dẫn 1c

```
int ChuSoDauTienle(int x) {
        while (x >= 10)
                x /= 10 ;
        return (x % 2 != 0 ? 1 : 0) ;
}
```

2.

e. Đếm số phần tử có chữ số tận cùng là 6 và chia hết cho 6.
f. Đếm số phần tử có chữ số đầu tiên là chữ số lẽ.
g. Đếm số phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.
h. Đếm số phần tử phân biệt, các phần tử trùng nhau chỉ đếm một lần.

Hướng dẫn 2d

```
int DemPhanBiet(int a[], int n) {
        int ret = 0;
        for(i = 0; i < n; i++) {
                if(TimKiem(a, n, i+1, a[i]) == -1) /*Không tìm thấy a[i] trong mảng a gồm n phần tử từ vị trí i+1 trở đi*/
                        ret++;
        }
        return ret;
}
```

3.

j. Kiểm tra mảng có chứa số dương hay không?
k. Kiểm tra mảng có gồm toàn số dương hay không?
l. Kiểm tra mảng có thứ tự tăng hay không?
m. Kiểm tra mảng có thứ tự giảm hay không?

n. Kiểm tra mảng có được sắp thứ tự hay không?

o. Kiểm tra mảng có đan xen âm dương hay không?

p. Kiểm tra mảng có đối xứng hay không?

q. Kiểm tra mảng có tất cả các cặp phần tử đứng cạnh nhau có giá trị khác nhau hay không?

r. Kiểm tra mảng có tất cả bộ 3 phần tử đứng cạnh nhau lập thành cấp số cộng hay không?

Hướng dẫn 3g

```
int DoiXung(int a[], int n)  {
        int ret = 1, i;
        for(i = 0; i < n/2; i++)
                if(a[i] != a[n – 1 - i]) {
                        ret = 0; break;
                }
        return ret;
}
```

4.

d. Sắp sếp các phần tử giảm dần.

e. Sắp sếp các phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng tăng dần, các phần tử khác giữ nguyên vị trí.

f. Trộn hai mảng tăng dần lại thành một mảng được sắp thứ tự tăng dần.

Hướng dẫn 4b

```
void SapXepCPTang(int a[], int n) {
    for(int i = 0; i < n-1; i++)
      for(int j = i+1; j < n; j++)
        if(ChinhPhuong(a[i]) && ChinhPhuong (a[j]) && a[i] > a[j]) {
            int tam = a[i]; a[i] = a[j]; a[j] = tam;
        }
}
```

5.

c. Tìm vị trí của phần tử lớn nhất/nhỏ nhất.

d. Tìm vị trí của phần tử dương nhỏ nhất/âm lớn nhất.

6.

d. Liệt kê các phần tử có chữ số tận cùng là 6 và chia hết cho 6.

e. Liệt kê các phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.

f. Liệt kê tần suất xuất hiện của các phần tử. Chẳng hạn với mảng 12 34 12 34 43 12 5, thì tần suất xuất hiện các phần tử được cho trong bảng sau:

Phần tử	Tần suất
12	3
34	2
43	1
5	1

7.

c. Thêm một phần tử có giá trị x vào mảng tại vị trí k

d. Thêm một giá trị x vào trong mảng tăng mà vẫn giữ nguyên tính tăng của mảng.

8.

c. Xóa phần tử có chỉ số k.

d. Xóa tất cả phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng trong mảng.

9.

c. Tạo mảng b từ mảng a sao cho mảng b chỉ chứa các phần tử có giá trị lẻ.

d. Tạo mảng b từ mảng a sao cho mảng b chỉ chứa các phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/ số đối xứng.

Hướng dẫn 9b

```
void TaoMangb(int a[], int n, int b[], int *m) {
    *m = 0;
    for(int i = 0; i < n; i++)
        if(NguyenTo(a[i])) {
                b[*m] = a[i];
                (*m)++;
        }
}
```

CHƯƠNG 7: MẢNG HAI CHIỀU

Đối với các bài tập trong chương này, trong chương trình luôn phải có hai hàm nhập và xuất mảng hai chiều nguyên.

7.

c. Tính tổng/tích các phần tử trên dòng/cột thứ k.

d. Tính trung bình cộng các phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.

Hướng dẫn 1b

```
double TBCong(int a[][SIZE2], int sd, int sc)
{
    int dem = 0, s = 0;
    double ret;
    for(int i = 0; i < sd; i++)
        for(int j = 0; j < sc; j++)
                if(chinhPhuong(a[i][j]) {
                        s += a[i][j];
                        dem++;
                }
    if(dem > 0)
        ret = (double)s / dem;
    else ret = -1; //Ma trận không chứa số chính phương
    return ret;
}
```

8.

c. Đếm số phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.

d. Đếm số phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng trên dòng/cột thứ k.

Hướng dẫn 2b

```
int DemCPTrenCot(int a[][SIZE2], int sd, int k)
{
    int ret = 0;
    for(int i = 0; i < sd; i++)
        if(chinhPhuong(a[i][k]))
                ret++;
    return ret;
}
```

9.

d. Kiểm tra ma trận có phần tử dương hay không.

e. Kiểm tra ma trận có gồm toàn phần tử dương hay không.

f. Kiểm tra dòng/cột thứ k của ma trận có tăng dần/giảm dần hay không.

Hướng dẫn 3c

```
int CotTang(int a[][SIZE2], int sd, int k)
{
    int ret = 1;
    for(int i = 0; i < sd - 1; i++)
        if(a[i][k] > a[i+1][k])  {
                ret = 0; break;
    }
    return ret;
}
```

10.

c. Liệt kê các dòng/cột có chứa phần tử là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.

d. Liệt kê các dòng/cột có chứa phần tử gồm toàn là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng.

Hướng dẫn 4a

```
void LietKeDongCoCP(int a[][SIZE2], int sd, int sc)
{
    int cp; //Cờ
    int i, j;
    for(i = 0; i < sd; i++) {
        //Kiểm tra dòng thứ i có số chính phương không?
        cp = 0;
        for(j = 0; j < sc; j++)
                if(ChinhPhuong(a[i][j])) {
                        cp = 1; break
                }
        //In dòng có số chính phương
        if(cp) {
                for(j = 0; j < sc; j++)
                        printf("%d\t", a[i][j]);
                printf("\n");
        }
    }
}
```

11.

d. Tìm vị trí phần tử lớn nhất/nhỏ nhất.

e. Tìm vị trí phần tử lớn nhất/nhỏ nhất trên dòng/cột thứ k.

f. Tìm vị trí phần tử dương nhỏ nhất/âm lớn nhất trên dòng/cột thứ k.

Hướng dẫn 5b

```
int TimVTMaxTrenCot(int a[][SIZE2], int sd, int k)
{
    int ret = 0;
    for(int i = 1; i < sd; i++)
        if(a[i][k] > a[ret][k])
                ret = i;
    return ret; //Vị trí phần tử lớn nhất trên cột thứ k
}
```

12.

e. Sắp xếp các phần tử trên dòng thứ k tăng dần từ trái sang phải.

f. Sắp xếp các phần tử trên cột thứ k tăng dần từ trên xuống dưới.

g. Sắp xếp các cột của ma trận sao cho tổng giá trị của mỗi cột tăng dần từ trái sang phải.

	Ma trận nhập vào				Ma trận sau khi sxếp			
	2	4	6	0	4	0	6	2
	9	1	2	6	1	6	2	9
Σ	**11**	**5**	**8**	**6**	**5**	**6**	**8**	**11**

Hướng dẫn 6c

```
void HoanViCot(int a[][SIZE2], int sd, int i, int j)
{
    for(int k = 0; k < sd; k++) {
        int tam = a[k][i];
        a[k][i] = a[k][j];
        a[k][j] = tam;
    }
}
void SXTongCotTang(int a[][SIZE2], int sd, int sc)
{
    for(int i = 0; i < sc-1; i++)
      for(int j = i+1; j < sc; j++) {
          int s1 = TongCot(a, sd, i); //Tổng các phần tử trên cột i
          int s2 = TongCot(a, sd, j); //Tổng các phần tử trên cột j
```

```
            if(s1 > s2)
              HoanViCot(a, sd, i, j); //Hoán vị cột i và cột j
        }
    }
```

h. Sắp xếp ma trận tăng dần trên mỗi dòng từ trái sang phải và từ trên xuống dưới.

Ma trận nhập vào

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

Ma trận sau khi sxếp

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

Hướng dẫn 6d:

```
void SapXepCau6d(int a[][SIZE2], int sd, int sc)
{
        int b[SIZE1*SIZE2];
        int m = 0;
        // Xây dựng mảng một chiều b từ mảng hai chiều a cho trước
        for(i = 0; i < sd; i++)
                for(j = 0; j < sc; j++) {
                        b[m] = a[i][j]; m++;
                }
        // Sắp xếp mảng một chiều b tăng dần
        SapXepTang(b, m);
        // Đặt các phần tử trong b ngược trở về a
        m = 0;
        for(i = 0; i < sd; i++)
                for(j = 0; j < sc; j++) {
                        a[i][j] = b[m]; m++;
                }
}
```

7. Mảng vuông

g. Tính tổng/tích các phần tử là là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng nằm trên đường chéo chính/đường chéo phụ.

h. Đếm các phần tử là là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng nằm trên đường chéo chính/đường chéo phụ.

i. Tính tổng/tích các phần tử là là số chính phương/số hoàn chỉnh/số nguyên tố/số đối xứng nằm trong nửa mảng vuông phía trên đường chéo chính/nửa mảng vuông phía dưới đường chéo chính/nửa mảng vuông phía trên đường chéo phụ/ nửa mảng vuông phía dưới đường chéo phụ.
j. Tìm phần tử lớn nhất/nhỏ nhất trên đường chéo chính/đường chéo phụ.
k. Tìm phần tử âm lớn nhất/dương nhỏ nhất trên đường chéo chính/đường chéo phụ.
l. Sắp xếp các phần tử trên đường chéo chính/đường chéo phụ theo thứ tự tăng dần/giảm dần từ trên hướng xuống/từ dưới hướng lên.

CHƯƠNG 8: CHUỖI

8. Viết các hàm thực hiện:
   - Chuyển đổi một chuỗi sang dạng thường
   - Chuyển đổi một chuỗi sang dạng hoa.
   - Chuyển đổi một chuỗi sang dạng Title case (ký tự đầu của mỗi từ là chữ hoa, các ký tự còn lại là chữ thường).
9. Viết hàm kiểm tra một chuỗi có đối xứng hay không?
   Ví dụ: Các chuỗi level, radar, dad, … là các chuỗi đối xứng

Hướng dẫn bài 2

```
int DoiXung(char *s)
{
        int len = strlen(s);
        int ret = 1;
        for(int i = 0; i <= len/2; i++)
                if(s[i] != s[len – 1 – i) {
                        ret = 0; break’
                }
        return ret;
}
```

10. Viết hàm tìm chuỗi đảo ngược của một chuỗi.
    Ví dụ: Chuỗi “Lap trinh C” có chuỗi đảo ngược là “C hnirt paL”

Hướng dẫn bài 3

```
void DaoNguoc(char *s1, char *s2)
{
        int len = strlen(s1);
        int j = 0;
        for(int i = len - 1; i >= 0; i--)  {
                s2[j] = s1[i];
                j++;
        }
        s2[j] = 0; //Thêm ký tự ‘\0’ và cuối chuỗi s2
}
```

11. Viết hàm đếm số từ trong một chuỗi.

Ví dụ: Chuỗi " Lap trinh C " có 3 từ

Hướng dẫn bài 4

```
int DemTu(char *s)
{
	int len = strlen(s);
	int ret = 0;
	int i = 0;
	while(i < len) {
		//Đi qua các khoảng trắng
		while(s[i] == ' ' && i < len)
			i++;
		if(i < len) {
			//Đi qua các ký tự khác trắng
			while(s[i] != ' ' && i < len)
				i++;
			ret++;
		}
	}
	return ret;
}
```

12. Viết hàm in mỗi từ của chuỗi trên một dòng.

Ví dụ: Chuỗi " Lập trình C" sẽ được in ra:

Lập

trinh

C

Hướng dẫn bài 5

```
void InTu(char *s)
{
	int i = 0, j;
	char tu[20];
	int len = strlen(s);
	while(i < len) {
		//Đi qua các khoảng trắng
		while(s[i] == ' ' && i < len)
			i++;
```

```
                if(i < len) {
                        //Đi qua các ký tự khác trắng
                        j = 0;
                        while(s[i] != ' ' && i < len) {
                                tu[j] = s[i];
                                j++; i++;
                        }
                        tu[j] = 0;
                        printf("%s\n" tu);
                }
        }
}
```

13. Viết hàm xóa các khoảng trắng dư thừa ở bên trái, bên phải và giữa chuỗi.
14. Viết hàm thống kê số lần xuất hiện của mỗi ký tự từ A đến Z trong một chuỗi (không phân biệt chữ hoa hay chữ thường).

    Ví dụ: Với chuỗi "Hello World" ta sẽ có:

    D: 1 lần

    E: 1 lần

    H: 1 lần

    L: 3 lần

    ...

Hướng dẫn bài 7

```
#include <ctype.h>
#define NUM_ALPHABET        26
void ThongKe(char *s)
{
    int dem[NUM_ALPHABET]; //Mảng đếm số lần xuất hiện ký tự
    //Khởi tạo mảng đếm
    for(i = 0; i < NUM_ALPHABET; i++)
        dem[i] = 0;
    //Đếm số lần xuất hiện ký tự
    int len = strlen(s);
    for(i = 0; i < len; i++)
        if(toupper(s[i]) >= 'A' && toupper(s[i]) <= 'Z')
            dem[toupper(s[i]) – 'A']++;
    //In bảng thống kê
    for(i = 0; i < NUM_ALPHABET; i++)
        printf("ký tự %c: %d lan\n", i + 'A', dem[i]);
```

```
}
```

CHƯƠNG 9: CẤU TRÚC

CẤU TRÚC

16. Khai báo kiểu dữ liệu biểu diễn thông tin của một phân số. Hãy viết hàm thực hiện các yêu cầu sau:
    k. Nhập phân số.
    l. Xuất phân số.
    m. Rút gọn phân số.
    n. Tính tổng hai phân số.
    o. Tính hiệu hai phân số.
    p. Tính tích hai phân số.
    q. Tính Thương hai phân số.
    r. So sánh hai phân số (hàm trả về một trong 3 giá trị: -1, 0, 1).
    s. Kiểm tra phân số âm.
    t. Kiểm tra phân số dương.

Hướng dẫn 1c

```
void RutGon (PS *u) {
        int uscln = TimUSCLN(u ->ts, u->ms);
        u->ts /= uscln;
        u->ms /= uscln;
}
```

Hướng dẫn 1d

```
PS Tong(PS u, PS v)
{
        PS ret;
        ret.ts = u.ts * v.ms + u.ms * v.ts;
        ret.ms = u.ms * v.ms;
        RutGon(&ret);
        return ret;
}
```

Hướng dẫn 1h

```
int SoSanh(PS u, PS v) {
        int ret;
        if(u.ts * v.ms > u.ms * v.ts)
                ret = 1;
        else if(u.ts * v.ms == u.ms * v.ts)
                ret = 0;
        else ret = -1;
        return ret;
}
```

17. Khai báo kiểu dữ liệu biểu diễn thông tin của một số phức. Hãy viết hàm thực hiện các yêu cầu sau:
    i. Nhập số phức.
    j. Xuất số phức (theo định dạng Re + Im*i).
    k. Tính modul số phức.
    l. So sánh hai số phức (hàm trả về một trong 3 giá trị: 0, -1, 1).
    m. Tính tổng hai số phức.
    n. Tính hiệu hai số phức.
    o. Tính tích hai số phức.
    p. Tính thương hai số phức.
18. Khai báo kiểu dữ liệu biểu diễn thông tin điểm trong mặt phẳng. Hãy viết hàm thực hiện các yêu cầu sau:
    o. Nhập điểm
    p. Xuất điểm theo định dạng (x, y)
    q. Tính khoảng cách giữa hai điểm.
    r. Tính khoảng cách giữa hai điểm theo phương Ox
    s. Tính khoảng cách giữa hai điểm theo phương Oy
    t. Tìm điểm đối xứng qua gốc tọa độ.
    u. Tìm điểm đối xứng qua trục hoành.
    v. Tìm điểm đối xứng qua trục tung.
    w. Tìm điểm đối xứng qua đường phân giác thứ nhất (y=x).

x. Tìm điểm đối xứng qua đường phân giác thứ hai (y=-x).

y. Kiểm tra điểm có thuộc phần tư thứ I không?

z. Kiểm tra điểm có thuộc phần tư thứ II không?

aa. Kiểm tra điểm có thuộc phần tư thứ III không?

bb. Kiểm tra điểm có thuộc phần tư thứ IV không?

19. Khai báo kiểu dữ liệu biểu diễn thông tin đường tròn. Hãy viết hàm thực hiện các yêu cầu sau:

g. Nhập đường tròn.

h. Xuất đường tròn theo định dạng ((x, y), r).

i. Tính chu vi đường tròn.

j. Tính diện tích đường tròn.

k. Xét vị trí tương đối giữa hai đường tròn (trùng nhau, ở trong nhau, tiếp xúc, cắt nhau, nằm ngoài nhau).

l. Kiểm tra một điểm có nằm trên đường tròn không?

Nhắc lại:

Nếu kc(o1, o2) = 0 và o1 = o2 thì hai đường tròn trùng nhau.

Nếu abs(r1 – r2) > kc(o1, o2) thì hai đường tròn ở trong nhau.

Nếu r1 + r2 = kc(o1, o2) hoặc abs(r1 – r2) = kc(o1, o2) thì hai đường tròn tiếp xúc nhau.

Nếu r1 + r2 > kc(o1, o2) và abs(r1 – r2) < kc(o1, o2) thì hai đường tròn cắt nhau.

Nếu abs(r1 – r2) < kc(o1, o2) thì hai đường tròn ở ngoài nhau

20. Khai báo kiểu dữ liệu biểu diễn thông tin một tam giác. Hãy viết hàm thực hiện các yêu cầu sau:

k. Nhập tam giác.

l. Xuất tam giác theo định dạng ((x1, y1); (x2, y2); (x3, y3)).

m. Kiểm tra 3 đỉnh có lập thành 3 đỉnh của một tam giác không?

n. Tính chu vi tam giác.

o. Tính diện tích tam giác.

p. Tính tọa độ trọng tâm tam giác.

q. Phân loại tam giác.

r. Tìm một đỉnh có hoành độ lớn nhất.

s. Tìm một đỉnh có tung độ nhỏ nhất.

t. Tính tổng khoảng cách từ điểm p(x, y) tới 3 đỉnh tam giác.

21. Khai báo kiểu dữ liệu biểu diễn thông tin ngày. Hãy viết hàm thực hiện các yêu cầu sau:

f. Nhập ngày.

g. Xuất ngày theo định dạng (dd/mm/yy).

h. Kiểm tra ngày hợp lệ.

i. Tìm ngày kế tiếp.

j. Tìm ngày trước đó.

22. Khai báo kiểu dữ liệu biểu diễn thông tin giờ. Hãy viết hàm thực hiện các yêu cầu sau:

e. Nhập giờ.
f. Xuất giờ theo định dạng (hh:mm:ss).
g. Kiểm tra giờ hợp lệ.
h. Tính tổng hai giờ.

23. Khai báo kiểu dữ liệu để biểu diễn thông tin của một chuyến bay. Biết rằng một chuyến bay gồm các thành phần như sau:
Mã chuyến bay: chuỗi tối đa 5 ký tự.

Ngày bay: kiểu ngày.

Giờ bay: Kiểu giờ.

Nơi đi: chuỗi tối đa 20 ký tự.

Nơi đến: chuỗi tối đa 20 ký tự.

Sau đó viết hàm nhập và hàm xuất cho kiểu dữ liệu này.

24. Khai báo kiểu dữ liệu để biểu diễn thông tin của một học sinh. Biết rằng một học sinh gồm các thành phần như sau:
Mã sinh viên: chuỗi tối đa 10 ký tự.

Tên sinh viên: chuỗi tối đa 30 ký tự.

Ngày sinh: Kiểu dữ liệu ngày.

Điểm: kiểu thực

Sau đó viết hàm nhập và hàm xuất cho kiểu dữ liệu này.

25. Khai báo kiểu dữ liệu để biểu diễn thông tin của một phiếu hàng. Biết rằng một phiếu hàng gồm các thành phần như sau:
Người mua: chuỗi tối đa 30 ký tự.

Mã hàng: chuỗi tối đa 10 ký tự.

Đơn giá: Kiểu số thực.

Số lượng: kiểu nguyên.

Sau đó viết hàm nhập và hàm xuất cho kiểu dữ liệu này.

MẢNG CẤU TRÚC

26. Mảng điểm
h. Viết hàm nhập mảng.
i. Viết hàm xuất mảng.

j. Đếm số lượng điểm có hoành độ dương.

k. Tìm một điểm có tung độ lớn nhất trong mảng.

l. Tìm một điểm trong mảng gần gốc tọa độ nhất.

m. Tìm hai điểm gần nhau nhất trong mảng.

n. Tìm hai điểm xa nhau nhất trong mảng.

❖ **Hướng dẫn câu 11e**

```
int TimDiemGanONhat(DSD u)
{
  Diem o = {0, 0}; //Gốc tọa độ
  int ret = 0;
  double kc1 = KhoangCach(u.arr[ret], o);
  for(int i = 1; i < u.n; i++) {
        double kc2 = KhoangCach(u.arr[i], o);
        if(kc2 < kc1) {
              ret = i;
              kc1 = kc2;
        } }
  return ret;     //Trả về vi trị điểm gần gốc o nhất
}
```

❖ **Hướng dẫn câu 11f**

```
//Hàm trả về vị trí hai điểm gần nhau nhất
void Tim2DiemGanNhauNhat(DSD u, int *vt1, int *vt2)
{
    *vt1 = 0; *vt2 = 1;
    double kc1 = KhoangCach(u.arr[*vt1], u.arr[*vt2]);
    double kc2;
    for(int i = 0; i < u.n - 1; i++)
        for(int j = i + 1; j < u.n; j++) {
            kc2 = KhoangCach(u.arr[i], u.arr[j]);
            if(kc2 < kc1) {
                *vt1 = i;
                *vt2 = j;
                kc1 = kc2;
        }
    }
}
```

27. Mảng phân số

f. Viết hàm nhập mảng.

g. Viết hàm xuất mảng.

h. Chuyển tất cả các phân số về dạng tối giản.

i. Tìm phân số dương nhỏ nhất/phân số âm lớn nhất.

j. Sắp xếp các phân số theo thứ tự mẫu số giảm dần.

Hướng dẫn 1c

```
void ChuyenDangToiGian(DSPS *u) {
        int i;
        for(i = 0; i < u->n; i++)
                RutGon(&u->arr[i]);
}
```

Hướng dẫn 1d

```
int TimPSDuongMin(DSPS u)

{

    int i, j, ret;

    //Tim vị trí phân số dương đầu tiên

    for(i = 0; i < u->n; i++)

        if(u.arr[i].ts * u.arr[i].ms > 0)        break;

    if(i >= u->n)    //Không có phân số dương

        ret = -1;

    else {            //Có phân số dương

        //Tìm vị trí phân số dương min

        ret = i;

        for(j = i + 1; j < u->n; j++)

            if(u.arr[j].ts * u.arr[j].ms > 0 && SoSanh(u.arr[j], u.arr[ret]
                                    == -1))

                ret = j;

    }

    return ret;

}
```

28. Mảng các chuyến bay
    d. Nhập mảng.
    e. Xuất mảng.
    f. Sắp xếp các chuyến bay tăng dần theo ngày bay.
29. Mảng các học sinh
    e. Nhập mảng.
    f. Xuất mảng.
    g. Tìm và in các học sinh có điểm cao nhất
    h. Sắp xếp các học sinh tăng dần theo mã.
30. Mảng các phiếu hàng
    d. Nhập mảng.
    e. Xuất mảng.
    f. Tính tổng số tiền bán được theo từng mã hàng.