



# LẬP TRÌNH CƠ BẢN VỚI C++

NỘI DUNG: Trang 2-127, tài liệu [1], chương 1-10 tài liệu [2].

## Phần 1. Lập trình cơ bản với C++ (Basic C++ Programming)

- 1.1. Giới thiệu về C++
- 1.2. Tập từ khóa trong C++
- 1.3. Cấu trúc chương trình trong C++
- 1.4. Cấu trúc dữ liệu cơ bản C++
- 1.5. Cấu trúc lệnh trong C++
- 1.6. Hàm và cấu trúc chương trình
- 1.7. Mảng & con trỏ
- 1.8. Cấu trúc và File
- 1.9. Lập trình hàm trên các cấu trúc dữ liệu cơ bản
- 1.10. CASE STUDY 1 : Lập trình hàm trong C++

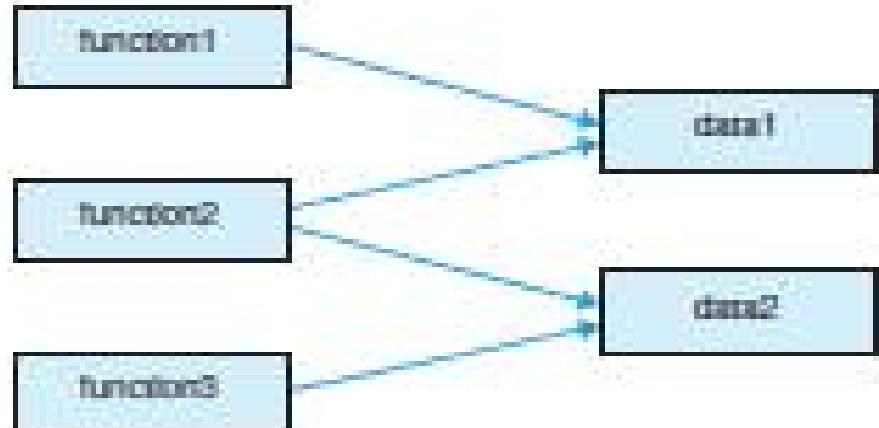
## **1.1. Giới thiệu ngôn ngữ lập trình C++**

### **Đặc điểm của ngôn ngữ C++:**

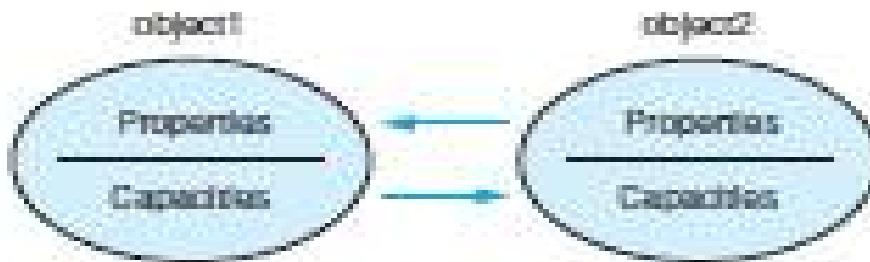
- Kế thừa các đặc tính của ngôn ngữ C.** Mọi chương trình viết bằng ngôn ngữ C đều được sử dụng trong C++: lập trình sử dụng hàm (lập trình cấu trúc), hiệu quả và gần gũi với mã máy, viết chương trình dễ dàng cho các nền tảng khác nhau.
- Hỗ trợ các nguyên lý lập trình hướng đối tượng (OOP):**
  - Trừu tượng hóa dữ liệu (data abstraction): cung cấp công cụ tạo lớp (class) để mô tả đối tượng.
  - Đóng gói dữ liệu (data encapsulation): điều khiển việc truy nhập dữ liệu đối tượng (object data).
  - Kế thừa (inheritance): kế thừa một hoặc nhiều lớp cơ sở mô tả các đối tượng khác nhau.
  - Đa hình thái (polymorphism): cài đặt cơ chế thực hiện chương trình khác nhau.

## ■ OBJECT-ORIENTED PROGRAMMING

### Traditional concept



### Object-oriented concept



## 1.2. Tập từ khóa trong C++

### Keywords in C++

asm	do	inline	short	typeid
auto	double	int	signed	typename
bool	dynamic_cast	long	sizeof	union
break	else	mutable	static	unsigned
case	enum	namespace	static_cast	using
catch	explicit	new	struct	virtual
char	extern	operator	switch	void
class	false	private	template	volatile
const	float	protected	this	wchar_t
const_cast	for	public	throw	while
continue	friend	register	true	
default	goto	reinterpret_cast	try	
delete	if	return	typedef	

### 1.3. Cấu trúc chương trình trong C++

Mỗi chương trình trong C++ được tổ chức thành 3 phần:

**Phần 1.** *Khai báo việc sử dụng thư viện (header file).* Mỗi header file chứa đựng các hàm được cung cấp bởi ngôn ngữ C++. Người dùng muốn sử dụng hàm nào thì phải được khai báo trong header file tương ứng theo cú pháp:

```
#include <Filename>
```

Dưới đây là một số header file thông dụng:

#### Header files of the C++ standard library

algorithm	ios	map	stack
bitset	iosfwd	memory	stdexcept
complex	iostream	new	streambuf
dequeue	istream	numeric	string
exception	iterator	ostream	typeinfo
fstream	limits	queue	utility
functional	list	set	valarray
iomanip	locale	sstream	vector

## Header files of the C standard library

assert.h	limits.h	stdarg.h	time.h
ctype.h	locale.h	stddef.h	wchar.h
errno.h	math.h	stdio.h	wctype.h
float.h	setjmp.h	stdlib.h	
iso646.h	signal.h	string.h	

Ví dụ. Khai báo việc sử dụng thư viện (header file).

#include <iostream> : khai báo việc sử dụng các hàm vào ra chuẩn của C++.

#include <string> : khai báo việc sử dụng các hàm xử lý ký tự.

Chú ý: khi sử dụng các hàm trong thư viện chuẩn của C, ta dịch chuyển thành thư viện của C++ theo nguyên tắc: bỏ **filename.h** và thay bằng **cfilename**. (bỏ .h ở cuối, thêm c vào đầu tên file).

#include <math.h>      ⇔     #include <cmath>

#include <stdio.h>      ⇔     #include <cstdio>

**Phần 2.** Mô tả các hàm, hoặc các lớp sử dụng trong chương trình. Trong mục này ta trọng tâm vào mô tả các hàm. Phần mô tả lớp sẽ được đề cập đến trong Part II của môn học.

**Hàm (Function, Subroutine, Procedure, Method, Proccess):** một đoạn chương trình thực hiện một nhiệm vụ nào đó phục vụ mục tiêu chủ quan của người lập trình, được xây dựng một lần, sử dụng nhiều lần ở mọi lúc, mọi nơi, mọi thời điểm trong chương trình.

-Hàm còn có nghĩa là đơn vị chương trình (unit program): lập trình có cấu trúc là phương pháp lập trình trên các hàm hoặc thủ tục . Một chương trình theo kiểu của lập trình cấu trúc là dãy các lời gọi hàm hoặc thủ tục.

- Hàm còn có nghĩa là chức năng (Function): một ứng dụng lớn thường được chia thành các chức năng. Mỗi chức năng thực hiện một ứng dụng nhỏ hơn.

- Hàm còn có nghĩa là quá trình xử lý (process): khi nó thực hiện một nhiệm vụ hoặc thuật toán đơn lẻ.

- Khi giải quyết một bài toán hoặc xây dựng ứng dụng thì điều quan trọng nhất là làm thế nào ta phân tích được lời giải bài toán hoặc ứng dụng để có được các hàm.

Cú pháp xây dựng hàm:

```
[Kiểu hàm]  Tên-hàm ( danh sách đối của hàm ) {  
    <Thân-hàm>; //dãy các chỉ thị thực hiện để đạt được mục tiêu  
    return (giá trị); //giá trị trả về của hàm  
}
```

**Ví dụ về hàm.** Ví dụ dưới đây mô tả các hàm tính tổng, hiệu, tích của hai số nguyên a và b.

**//Hàm tính tổng hai số**

```
int      tong ( int  a,  int  b ) {  
    int  c = a + b; // c là tổng của a và b  
    return(c); //trả lại giá trị c  
}
```

**//Hàm tính hiệu hai số**

```
int      hieu ( int  a,  int  b ) {  
    int  c = a - b; // c là hiệu của a và b  
    return(c); //trả lại giá trị c  
}
```

**//Hàm tính tích hai số**

```
int      hieu ( int  a,  int  b ) {  
    int  c = a * b; // c là tích của a và b  
    return(c); //trả lại giá trị c  
}
```

**Phần 3. Chương trình chính.** Là nơi xác định điểm bắt đầu thực hiện chương trình và kết thúc chương trình.

```
int main (void ) { //điểm bắt đầu thực hiện chương trình  
    <thân hàm main>; //tập các lời gọi hàm sẽ diễn ra ở đây  
    return 0; //giá trị trả về của hàm main  
} //điểm kết thúc thực hiện chương trình
```

**Ghi chú:**

- Khi xây dựng chương trình cần viết chú giải đầy đủ. Chú giải theo từng dòng là dòng văn bản được viết sau hai ký hiệu “//”. Chú giải theo từng khối là khối văn bản được bao bì đầu bằng “/\*” và kết thúc bằng “\*/”. Các dòng hoặc khối chú giải ghi lại ý nghĩa câu lệnh hoặc đoạn chương trình và sẽ được bỏ qua trong khi biên dịch chương trình.

Ví dụ : chú giải theo dòng:

//đây là một chú giải theo dòng

Ví dụ chú giải theo khối:

/\* Đây là đoạn văn bản  
được chú giải theo khối  
khối chú giải gồm 3 dòng\*/

**Ví dụ.** Cấu trúc *chương trình*. Viết chương trình tính tổng, hiệu hai số a và b.

**//Phần 1. Khai báo việc sử dụng chương trình**

```
#include <iostream> //khai báo sử dụng các hàm trong iostream  
using namespace std; //khai báo sử dụng không gian tên của iostream
```

**//Phần 2: mô tả các hàm sử dụng trong chương trình**

```
int tong( int a, int b) { //hàm tính tổng hai số
```

```
    int c = a + b; // lấy c = a+b  
    return (c); //trả lại c
```

```
}
```

```
int hieu( int a, int b) { //hàm tính hiệu hai số
```

```
    int c = a - b; // lấy c = a - b  
    return (c); //trả lại c
```

```
}
```

**// Chương trình chính**

```
int main (void ) { // điểm đầu thực hiện chương trình
```

```
    int a, b; //khai báo hai biến nguyên a, b;
```

```
    cout<<“Nhập a =”; cin>>a; //nhập a từ bàn phím
```

```
    cout<<“Nhập b =”; cin>>b; //nhập b từ bàn phím
```

```
    cout<<“Tổng a + b =”<<tong(a,b)<<endl; //đưa ra tổng a+b và xuống dòng
```

```
    cout<<“Hiệu a - b =”<<hieu(a,b)<<endl; //đưa ra tổng a-b và xuống dòng
```

```
    system(“PAUSE”); //dừng lại xem kết quả
```

```
}
```

## 1.4. Các kiểu dữ liệu trong C++

Kiểu dữ liệu (data type) : là một tên (name) dùng để chỉ tập các đối tượng thuộc miền xác định cùng với các phép toán trên nó. Các kiểu dữ liệu nguyên thủy trong C++ bao gồm:

Kiểu (type)	Tùy khóa (Keyword)
Kiểu logic (Boolean)	bool
Ký tự (character)	char
Số nguyên (integer)	int
Kiểu số thực (floating point)	float
Kiểu số thực có độ chính xác kép (double floating point)	double
Vô kiểu (valueless)	void
Kiểu số nguyên dài (long integer)	long
Kiểu ký tự rộng (wide character)	wchar_t

Từ các kiểu dữ liệu cơ bản trên, một số kiểu dữ liệu sửa đổi được kết hợp với các từ khóa signed (có dấu), unsigned (không dấu), short (ngắn), long (dài) để tạo nên các kiểu dữ liệu khác nhau. Dưới đây là các kiểu dữ liệu cùng với kích cỡ của kiểu.

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	Range	0 to 65,535
signed short int	Range	-32768 to 32767
long int	4bytes	-2,147,483,647 to 2,147,483,647
signed long int	4bytes	same as long int
unsigned long int	4bytes	0 to 4,294,967,295
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

## Ví dụ. Kiểm tra kích cỡ các kiểu dữ liệu.

```
#include <iostream> //Khai bao viec su dung thu vien
using namespace std; //Khai bao su dung khong gian ten

//Kiem tra kich co cac kieu du lieu co ban
int main(void ) { //Noi bat dau thuc hien chuong trinh
    cout<<"Kich co kieu bool   : "<<sizeof(bool)<<endl;
    cout<<"Kich co kieu char   : "<<sizeof(char)<<endl;
    cout<<"Kich co kieu wchar_t : "<<sizeof(wchar_t)<<endl;
    cout<<"Kich co kieu short  : "<<sizeof(int)<<endl;
    cout<<"Kich co kieu int   : "<<sizeof(short)<<endl;
    cout<<"Kich co kieu long   : "<<sizeof(long)<<endl;
    cout<<"Kich co kieu float  : "<<sizeof(float)<<endl;
    cout<<"Kich co kieu double : "<<sizeof(double)<<endl;
    cout<<"Kich co kieu long long: "<<sizeof(long long)<<endl;
    cout<<"Kich co kieu double long: "<<sizeof(double long)<<endl;
    system("PAUSE");//Dung lai cho bam mot ky tu
    return 0;
}
```

## Ví dụ. Kết quả thực hiện chương trình.

```
Kich co kieu bool   : 1
Kich co kieu char   : 1
Kich co kieu wchar_t : 2
Kich co kieu short   : 4
Kich co kieu int     : 2
Kich co kieu long    : 4
Kich co kieu float   : 4
Kich co kieu double  : 8
Kich co kieu long long: 8
Kich co kieu double long: 12
Press any key to continue . . .
```

## 1.5. Biến trong C++

**Biến (variable)** : là một tên (name) thuộc kiểu. Mỗi tên chứa đựng bộ ba: tên biến, địa chỉ của biến và giá trị của biến. Mọi biến đều phải được khai báo trước khi sử dụng.

### Khai báo biến:

[Tên kiểu ] biến1, biến2;

#### Ví dụ:

int a, b; // khai báo biến a, b có kiểu int

char c; // Khai báo biến c có kiểu char

float x = 0.5, y = 3.14; //vừa khai báo vừa khởi đầu cho biến

**Biến toàn cục (global):** Một biến là biến toàn cục nếu nó được khai báo ở ngoài tất cả các hàm kể cả hàm main(). Biến toàn cục có một số tính chất sau:

- Không gian nhớ dành cho biến toàn cục được cấp phát từ khi bắt đầu thực hiện chương trình cho đến khi kết thúc thực hiện chương trình.
- Phạm vi hoạt động của biến toàn cục: Được phép truy nhập và thay đổi nội dung của biến toàn cục ở bất kỳ vị trí nào trong chương trình.

**Biến cục bộ (local, private):** Một biến được gọi là biến cục bộ nếu nó được khai báo trong thân của hàm kể cả hàm main(), hoặc được khai báo trong thân của các chu trình (lặp, tuyển chọn). Biến cục bộ có các tính chất:

- Không gian nhớ dành cho biến toàn cục chỉ được cấp phát mỗi khi có lời gọi hàm hoặc thủ tục.
- Phạm vi hoạt động của biến toàn cục chỉ được giới hạn trong nội bộ hàm hoặc chu trình.

**Nguyên tắc:** Ưu tiên sử dụng biến cục bộ, hạn chế sử dụng biến toàn cục.

**Ví dụ về biến toàn cục và biến cục bộ:** đổi nội dung của hai biến.

```
#include <iostream>
using namespace std;
int a = 8, b = 12; // khai báo hai biến toàn cục a và b
void Swap1(void ) { int temp =a; a=b; b=temp; }
void Swap2 ( int a, int b) {
    int temp; // Khai báo biến cục bộ temp
    temp =a; a = b; b = temp;
}
void Swap3 ( int &a, int & b) { int temp; // Khai báo biến cục bộ temp
    temp =a; a = b; b = temp;
}
int main(void ){
    Swap1();//gọi hàm Swap1()
    cout<<"Giá trị a ="<<a<<" b ="<<b<<endl;
    int a=5 , b=7; // Khai báo biến cục bộ temp
    Swap1();// gọi hàm Swap1()
    cout<<"Giá trị a ="<<a<<" b ="<<b<<endl;
    Swap2(a, b);// gọi hàm Swap2()
    cout<<"Giá trị a ="<<a<<" b ="<<b<<endl;
    Swap3(a, b);// gọi hàm Swap3()
    cout<<"Giá trị a ="<<a<<" b ="<<b<<endl;
    system("PAUSE");return 0;
}
```

## 1.6. Các phép toán trong C++

Các phép toán trong C++ bao gồm:

- Các phép toán số học (Arithmetic Operators)
- Các phép toán quan hệ (Relational Operators)
- Các phép toán logic (Logical Operators)
- Các phép toán cắp bit (Bitwise Operators)
- Các phép toán số học mở rộng (Assignment Operators)

Các phép toán số học (Arithmetic Operators):

Phép toán	Mô tả	Ví dụ a = 10, b = 20
+	Phép cộng hai số	$a + b = 30$
-	Phép trừ hai số	$a - b = -10$
*	Phép nhân hai số	$a * b = 200$
/	Phép chia hai số	$b/a = 2$
%	Phép lấy phần dư hai số	$a \% b = 10$
++	Tăng số lên 1 đơn vị	$a++ \Leftrightarrow a = a + 1 = 11$
--	Giảm số đi 1 đơn vị	$a-- \Leftrightarrow a = a - 1 = 9$

**Ví dụ :** Kiểm tra các phép toán số học.

```
#include <iostream>
using namespace std;
int main(void ){
    int a = 10, b = 20; //khai báo và khởi đầu cho a và b
    cout<<"Tổng a + b ="<<a+b<<endl; //đưa ra a + b
    cout<<"Hiệu a + b ="<<a-b<<endl; //đưa ra a - b
    cout<<"Tích a * b ="<<a*b<<endl; //đưa ra a * b
    cout<<"Thương a / b ="<<a/b<<endl; //đưa ra a / b
    cout<<"Phần dư a % b ="<<a%b<<endl; //đưa ra a % b
    cout<<"Tăng a a ++ ="<<++a<<endl; //đưa ra a =a+1
    cout<<"Giảm a -- ="<<--a<<endl; //đưa ra a =a-1
    cout<<"Tang b ="<<b++<<endl; //đưa ra b sau đó tăng b =b+1
    cout<<"Giam b ="<<b--<<endl; //đưa ra b sau đó giảm b =b-1
    system("PAUSE");return 0;
}
```

## Các phép toán quan hệ (Relation Operators):

Phép toán	Mô tả	Ví dụ a = 10, b = 20
<code>==</code>	Đúng bằng	(a == b) là sai (0)
<code>!=</code>	Khác nhau	(a != b) là đúng (1)
<code>&gt;</code>	Lớn hơn	(a > b) là sai (0)
<code>&lt;</code>	Nhỏ hơn	(a < b) là đúng (1)
<code>&gt;=</code>	Lớn hơn hoặc bằng	(a >= b) là sai (0)
<code>&lt;=</code>	Nhỏ hơn hoặc bằng	(a <= b) là đúng (1)

## Các phép toán logic (Logical Operators):

Phép toán	Mô tả	Ví dụ a = 10, b = 20
<code>&amp;&amp;</code>	Phép và logic	(a == b) là sai (0)
<code>  </code>	Phép hoặc logic	(a != b) là đúng (1)
<code>!</code>	Phép phủ định	(a > b) là sai (0)

**Ví dụ :** Kiểm tra các phép toán quan hệ và logic.

```
#include <iostream>
using namespace std;
int main( void ) { int a = 10, b = 20;
    cout << " Giá trị (a == b):" << (a==b) << endl;
    cout << " Giá trị (a != b):" << (a!=b) << endl;
    cout << " Giá trị (a > b):" << (a>b) << endl;
    cout << " Giá trị (a < b):" << (a<b) << endl;
    cout << " Giá trị (a >= b):" << (a>=b) << endl;
    cout << " Giá trị (a <= b):" << (a<=b) << endl;
    cout << " Giá trị ( a && b):" << (a&&b) << endl;
    cout << " Giá trị (a || b):" << (a||b) << endl;
    cout << " Giá trị !(a) :" << !(a) << endl;
    system("PAUSE");
    return 0;
}
```

## Các phép thao tác bít (Bitwise Operators):

Phép toán	Mô tả	Ví dụ a = 60 = 0011 1100, b = 13 = 0000 1101
$\wedge\wedge$	Phép và cấp bít	$a \wedge\wedge b = 0000.1100 = 12$
	Phép hoặc cấp bít	$(a   b) = 0011.1101 = 61$
$\wedge$	Phép tuyển cấp bít	$(a \wedge b) = 0011.0001 = 49$
$\sim$	Phép phủ định cấp bít	$\sim(a) = 1100.0011 = -61$
<<	Phép dịch trái cấp bít	$(a <<2) = 1111.0000=240$
>>	Phép dịch phải cấp bít	$(a >> b) = 0000. 1111 =15$

**Ví dụ :** Kiểm tra các phép toán cấp bít.

```
#include <iostream>
using namespace std;
int main( void ) {
    int a = 60, b = 13;
    cout << " a & b = "<<(a&&b) <<endl;
    cout << " a | b = "<<(a|b) <<endl;
    cout << " a ^ b = "<<(a^b) <<endl;
    cout << " a ~ b = "<<~(a) <<endl;
    cout << " a <<2 = "<<(a<<2) <<endl;
    cout << " a >>2 = "<<(a>>2) <<endl;
    system("PAUSE");
    return 0;
}
```

## 1.7. Các cấu trúc lệnh trong C++

Các cấu trúc lệnh trong C++ bao gồm:

- **Cấu trúc lệnh tuần tự** (Sequential): dãy các chỉ thị hoặc lời gọi hàm thực hiện một cách tuần tự. Các lệnh có thể là lệnh đơn, lệnh có cấu trúc hoặc lệnh phức hợp. Các ví dụ được nêu ở trên đều được thể hiện bằng cấu trúc lệnh tuần tự.

- **Cấu trúc lệnh tuyển chọn**: bao gồm cấu trúc if..else và switch(). Điểm khác biệt duy nhất giữa hai cấu trúc lệnh này là if..else lựa chọn một trong hai khả năng đúng, sai của biểu thức điều kiện để thực hiện, trái lại switch() lựa chọn khả năng đúng của biểu thức điều kiện trong nhiều khả năng để thực hiện.

### Cấu trúc lệnh if..else.

```
if (biểu-thức-điều-kiện) {  
    <Câu lệnh 1>;  
}  
else {  
    <Câu lệnh 2>;  
}
```

### Cấu trúc lệnh if..else khuyết.

```
if (biểu-thức-điều-kiện) {  
    <Câu lệnh 1>;  
}
```

### Cấu trúc lệnh switch().

```
switch (biểu-thức) {  
    case H1: <câu lệnh 1>; break;  
    case H2: <câu lệnh 2>; break;  
    .....;  
    case Hn: <câu lệnh n>; break;  
    default: <câu lệnh n+1>; break;  
}
```

**Ví dụ :** Kiểm tra số nhập vào là số chẵn hay số lẻ sử dụng if.else.

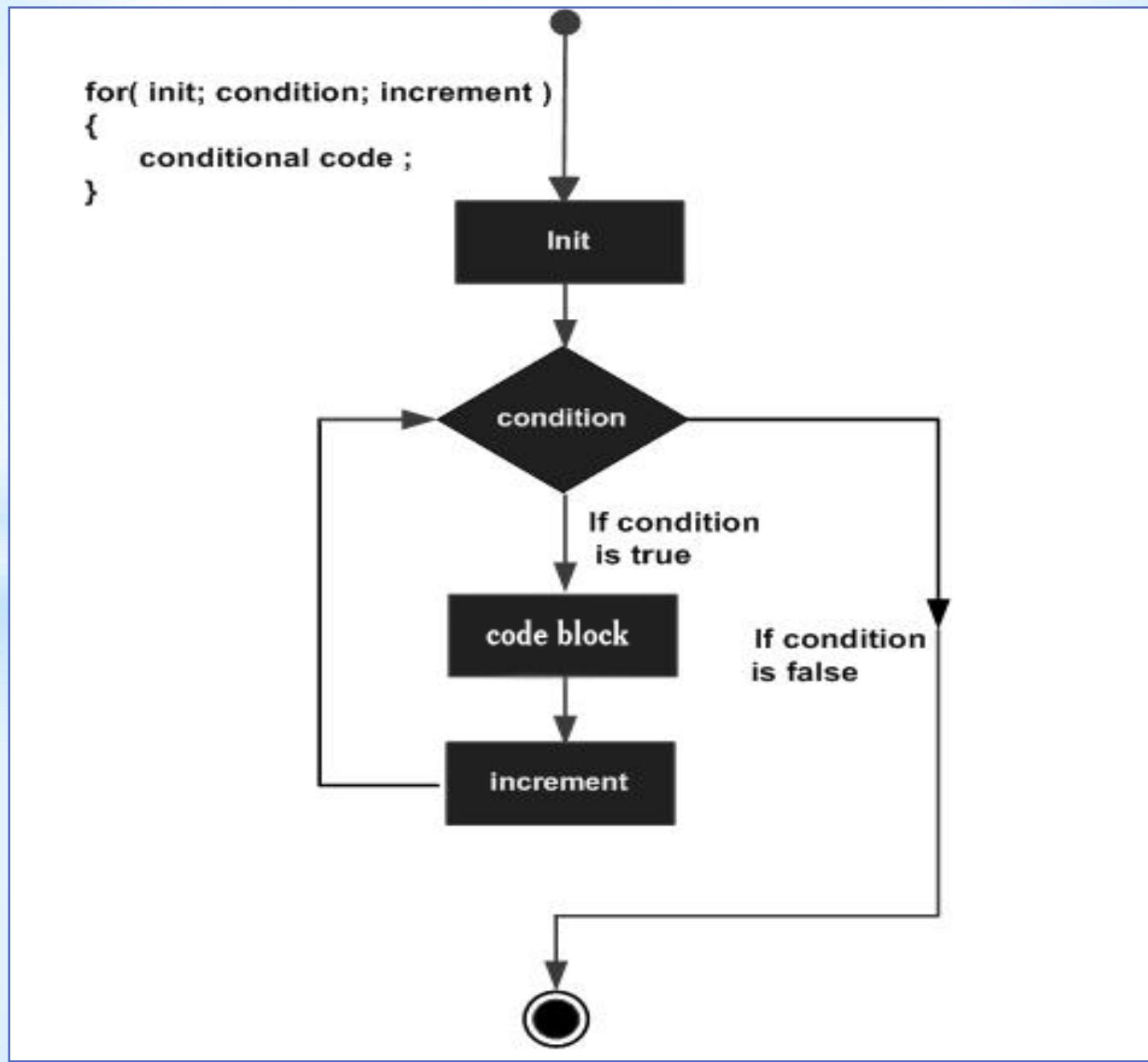
```
#include <iostream>
using namespace std;
int main( void ) {
    int number;
    cout<<"Nhập một số:"; cin>>number;
    if (number % 2 ) {
        cout<<"Số lẻ"<<endl;
    }
    else {
        cout<<"Số chẵn"<<endl;
    }
    system("PAUSE");
    return 0;
}
```

**Ví dụ :** Dịch tên tháng sang tiếng anh sử dụng cấu trúc switch().

```
#include <iostream>
using namespace std;
int main( void ) {
    int thang;
    cout<<"Nhập tháng:"; cin>>thang;
    switch(thang) {
        case 1: cout<<"January"<<endl; break;
        case 2: cout<<"February"<<endl; break;
        case 3: cout<<"March"<<endl; break;
        case 4: cout<<"April"<<endl; break;
        case 5: cout<<"May"<<endl; break;
        case 6: cout<<"June"<<endl; break;
        case 7: cout<<"July"<<endl; break;
        case 8: cout<<"August"<<endl; break;
        case 9: cout<<"September"<<endl; break;
        case 10: cout<<"October"<<endl; break;
        case 11: cout<<"November"<<endl; break;
        case 12: cout<<"December"<<endl; break;
    }
    system("PAUSE");
    return 0;
}
```

- Cấu trúc lệnh lặp (loop): for, while, do..while.

Cấu trúc lệnh lặp for:

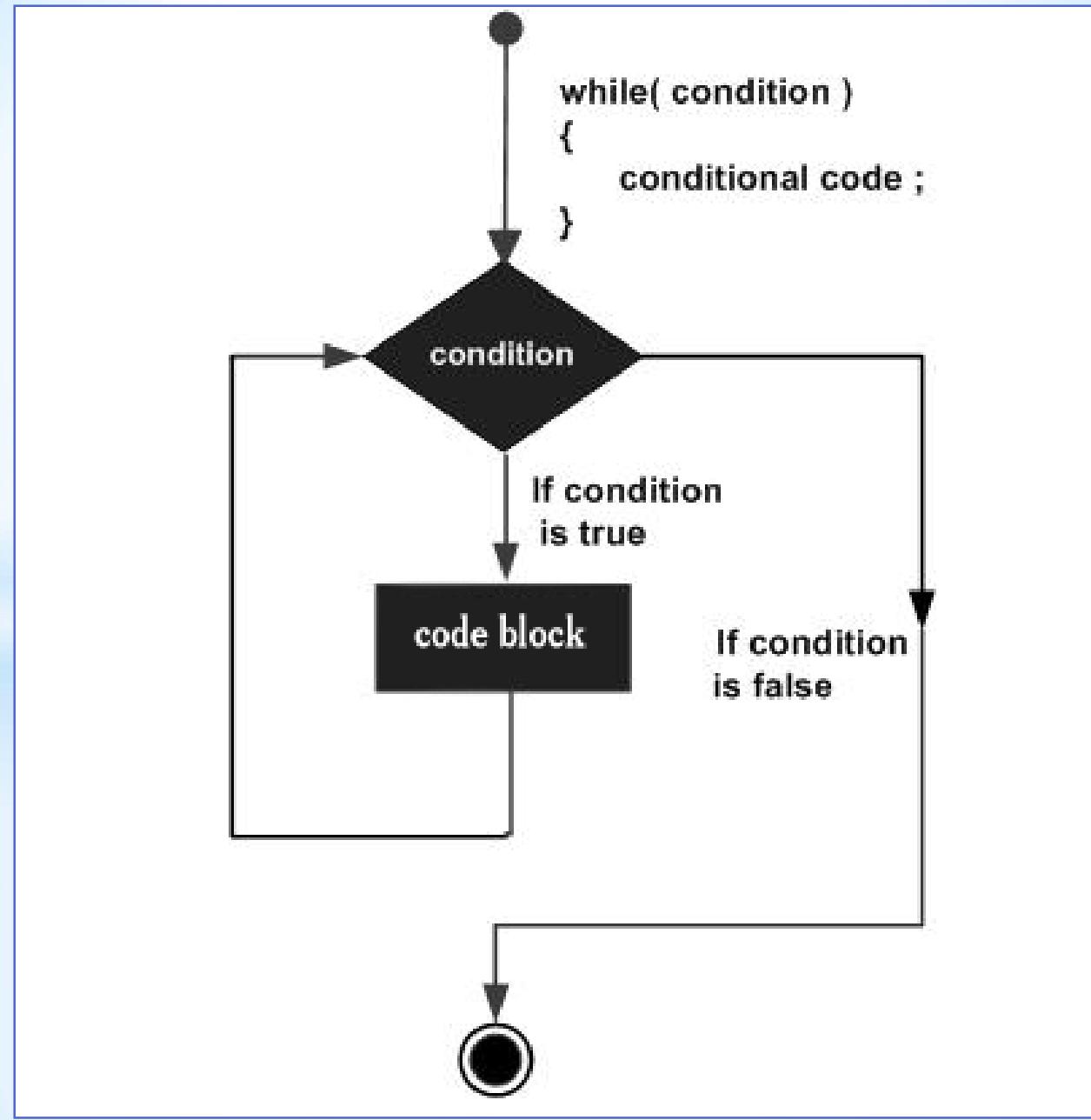


**Ví dụ :** Giải bài toán cỗ trăm trâu trăm cỏ sử dụng cấu trúc for

```
#include <iostream>
using namespace std;
int main( void ) {
    int x, y, z;
    for (x=0; x<=20; x++) {
        for(y=0; y<=33; y++){
            for (z=0; z<=99; z+=3){
                if ((5*x + 3*y + z/3)==100 && (x+y+z==100))
                    cout<<"x ="<<x <<" y = "<<y << "z="<<z<<endl;

            }
        }
    }
    system("PAUSE");
    return 0;
}
```

**Cấu trúc lệnh lặp while:** trong khi biểu thức điều kiện còn đúng thì thực hiện câu lệnhj.

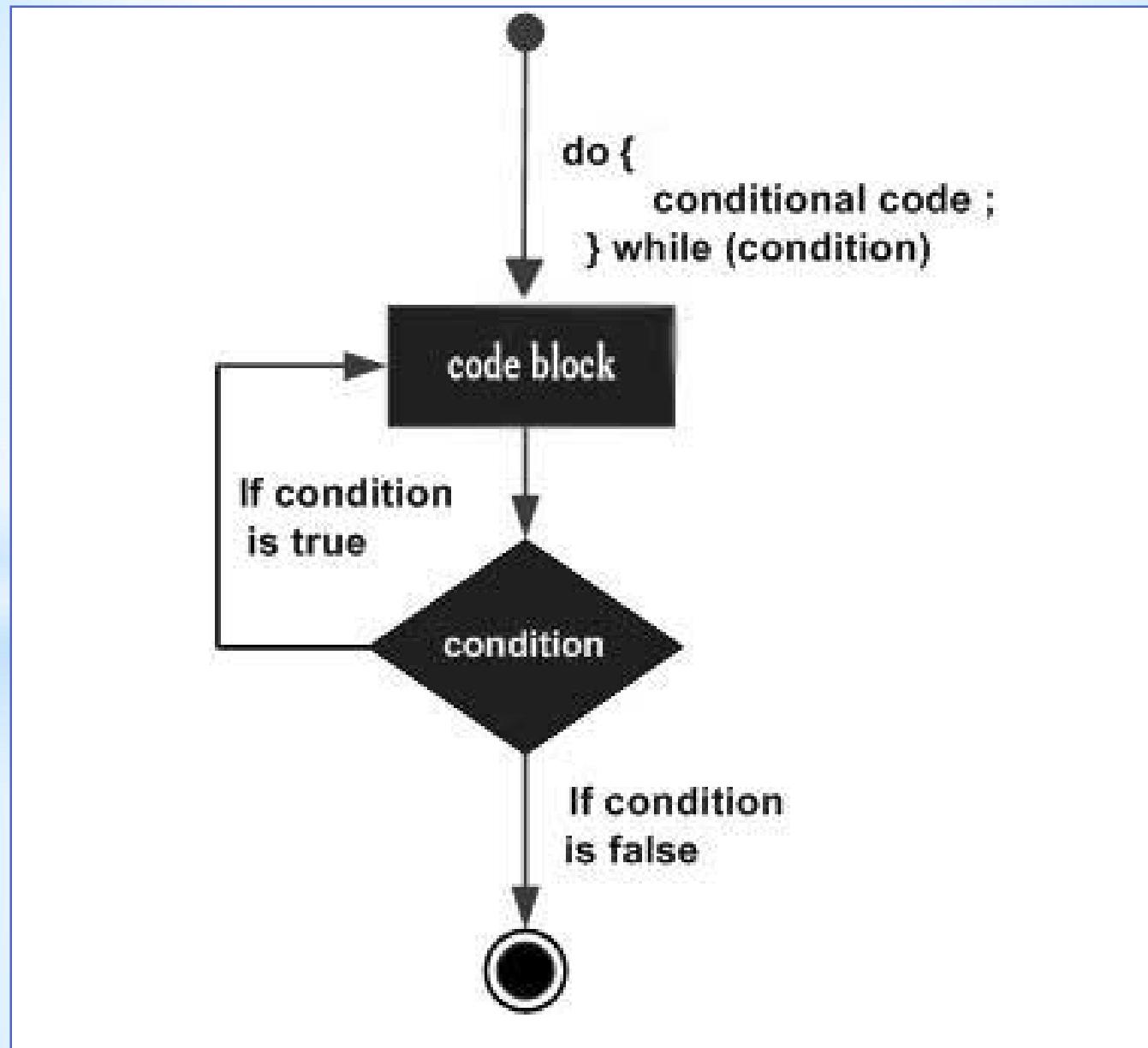


**Ví dụ :** Tìm N và s để  $(1/N) \leq \text{Epxilon}$  cho trước.

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{N}$$

```
#include <iostream>
using namespace std;
int main(void) {
    float S = 0, Epxilon; int N=1;
    cout<<"Nhập Epxilon="; cin>>Epxilon;
    while( (Epxilon<=(float)1/ (float)N)) {
        S = S + ((float)1) / ((float) N);
        N = N + 1;
    }
    cout<<" N = "<<N <<" S ="<<S;
    system("PAUSE");
}
```

**Cấu trúc lệnh lặp do..while:** thực hiện câu lệnhj trong khi biến thức điều kiện còn đúng.



**Ví dụ :** Nhập đúng password là một số sử dụng cấu trúc lặp do..while.

```
#include <iostream>
using namespace std;
int main( void ) {
    int number=1001, pass;
    do {
        system("cls");//xóa màn hình
        cout<<"Nhập Password:"; cin>>pass;
    } while(pass!=number);
    system("PAUSE");
    return 0;
}
```

**Bài tập.** Giải các bài tập dưới đây sử dụng các kiểu dữ liệu cơ bản và các cấu trúc lệnh.

1. Cho số tự nhiên N. Hãy tìm tổng các chữ số của N. Ví dụ  $N = 12345$  thì  $S = 1 + 2 + 3 + 4 + 5 = 15$ .
2. Cho số tự nhiên N. Hãy liệt kê tất cả các số nguyên tố nhỏ hơn N.
3. Hãy liệt kê tất cả các số nguyên tố có N chữ số sao cho tổng các chữ số của số đó đúng bằng S cho trước.
4. Cho số tự nhiên N, hãy phân tích N thành tích các thừa số nguyên tố.
5. Cho số tự nhiên a và b, hãy tìm ước số chung lớn nhất của hai số.
6. Một số tự nhiên được gọi là số hoàn hảo nếu tổng các ước số thực sự của nó kể cả 1 bằng chính nó. Hãy liệt kê các số hoàn hảo nhơn N.
7. Cặp số a, b được gọi là hữu nghị nếu tổng các ước số thực sự của a là b và tổng các ước số thực sự của b là a. Hãy liệt kê các số hoàn hảo có N chữ số.
8. Hãy liệt kê mã của các ký tự từ A đến Z cùng với mã nhị phân của ký tự.
9. Viết chương trình giải phương trình bậc 2.
10. Hãy liệt kê tất cả các số đối xứng có N chữ số và tổng các chữ số đúng bằng S cho trước.

**Bài tập.** Giải các bài tập dưới đây sử dụng các kiểu dữ liệu cơ bản và các cấu trúc lệnh.

11. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:

- a) K là số có N chữ số;
- b) K là số nguyên tố;
- c) Đảo ngược các chữ số trong K là một số nguyên tố;
- d) Tổng các chữ số trong K cũng là một số nguyên tố;
- e) Mỗi chữ số trong K cũng là các số nguyên tố.

12. Hãy viết chương trình liệt kê tất cả các số tự nhiên K thỏa mãn đồng thời những điều kiện dưới đây:

- a) K là số có N chữ số;
- b) K là số đối xứng;
- c) Tổng các chữ số của K là số chia hết cho 10;
- d) Các chữ số của K đều khác 0.

12. Nhân dịp phát hành các số điện thoại 0919xxx.xxx. Công ty Vinaphone dự định phát phát hành N số điện thoại loại 1, M số điện thoại loại 2, K số điện thoại loại 3. Trong đó, số điện thoại loại 1, 2, 3 được định nghĩa như sau:

**Loại 1:** là những số điện thoại có sáu số cuối cùng của nó tạo thành một số đối xứng (thuận nghịch) có sáu chữ số.

**Loại 2:** là những số điện thoại loại 1 có sáu số cuối cùng của nó là các chữ số khác 0.

**Loại 3:** là những số điện thoại loại 2 có tổng của sáu chữ số cuối cùng là một số chia hết cho 10.

Bài toán được đặt ra là: cho một phương án phát hành N, M, K. Hãy cho biết công ty Vinaphone có thể thực hiện được phương án phát hành kể trên hay không? Đưa ra kết quả “YES” nếu phương án phát hành thực hiện được, đưa ra kết quả “NO” nếu phương án phát hành không thể thực hiện được.

## 1.8. Dữ liệu kiểu mảng (Array)

**Định nghĩa.** Mảng là dãy các phần tử (biến) có cùng chung một kiểu dữ liệu được tổ chức liên tục nhau trong bộ nhớ.

**Khai báo mảng một chiều:**

*Tên-kiểu    Tên-mảng [số lượng phần tử];*

**Ví dụ:** Khai báo mảng một chiều

int A[10]; //khai báo mảng A kiểu int gồm 10 phần tử A[0], A[1], .., A[9]

float X[10]; //khai báo mảng X kiểu float gồm 10 phần tử X[0], X[1], .., X[9]

int A[] = { 9, 7, 12, 8, 6, 5 }; // vừa khai báo vừa khởi đầu cho mảng

**Khai báo mảng nhiều chiều:**

*Tên-kiểu    Tên-mảng [chiều 1][chiều 2]...[chiều k];*

**Ví dụ:** Khai báo mảng nhiều chiều (2 chiều):

int A[3][3]; //Khai báo ma trận vuông cấp 3x3 gồm 9 phần tử

// Các phần tử A[0][0],.., A[0,2], ..., A[2][0], A[2][1], A[2][2]

**Ví dụ:** Khai báo và khởi đầu ch0 mảng nhiều chiều (2 chiều):

```
int A[3][3] = { //Khai báo mảng gồm 9 phần tử
    { 1, 2, 3}, // khởi đầu cho hàng 0
    { 4, 5, 6}, // khởi đầu cho hàng 1
    { 7, 8, 9} // khởi đầu cho hàng 2. Chú ý không có dấu ','.
};
```

Hoặc ta có thể vừa khai báo và khởi đầu thế này cũng được:

```
int A[3][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
```

**Truy cập phần tử của mảng:** thông qua chỉ số phần tử trong mảng.

**Ví dụ:**

```
int A[10]; //Khai báo mảng gồm 10 phần tử.  
A[5] = 12; //Khởi tạo cho A[5] giá trị là 12.  
int B[3][3]; // Khai báo mảng 2 chiều gồm 9 phần tử  
B[1][2] = 9; //Khởi tạo cho B[1][2] giá trị là 9.
```

**Ví dụ:** Khởi tạo giá trị và kiểm tra tổ chức lưu trữ của mảng một chiều.

```
#include <iostream>
#include <iomanip>
using namespace std;
void Init ( int A[], int n ){
    for (int i=0; i<n; i++) { cout<<"Nhập A["<<i<<"]="; cin>>A[i]; }
}
void Address_Array ( int A[], int n) {
    cout<<"\n Địa chỉ các phần tử:"<<endl;
    for(int i=0; i<n; i++) cout<<"Địa chỉ A["<<i<<"] ="<<&A[i]<<endl;
}
void Result( int A[], int n) { cout<<"Nội dung mảng:";
    for (int i=0; i<n; i++) cout<<A[i]<<setw(4);
}
int main(void ) {
    int A[10], n; //Khai báo mảng A[10] và số phần tử của mảng là n.
    cout<<"Nhập n="; cin>>n; //nhập n
    Init( A, n); //Khởi tạo giá trị cho mảng A gồm n phần tử
    Result( A, n); // Giá trị các phần tử của mảng A
    Address_Array( A, n); //Địa chỉ các phần tử của mảng A
    system("PAUSE");
    return 0;
}
```

**Ví dụ:** Khởi tạo giá trị và kiểm tra tổ chức lưu trữ của mảng hai chiều.

```
#include <iostream>
using namespace std;
void Init(int A[][10], int n, int m){
    for(int i=0; i<n; i++)
        for (int j=0; j<m; j++){
            cout<<"Nhập A["<<i<<"]["<<j<<"]="; cin>>A[i][j];
        }
}
void Result(int A[][10], int n, int m){
    cout<<"Địa chỉ và giá trị các phần tử"<<endl;
    for (int i=0; i<n; i++)
        for (int j=0; j<m; j++){
            cout<<"Giá trị A["<<i<<"]["<<j<<"]="<<A[i][j]<<endl;
            cout<<"Địa chỉ A["<<i<<"]["<<j<<"]="<<&A[i][j]<<endl;
        }
}
int main(void ) {  int A[10][10], n, m;
    cout<<"Nhập n, m:"; cin>>n>>m;
    Init(A, n, m); Result(A,n,m); system("PAUSE");
    return 0;
}
```

## Bài tập: Cấu trúc dữ liệu mảng.

- 1) Viết chương trình đổi số tự nhiên N thành số ở hệ cơ số b ( $1 < b \leq 512$ ).
- 2) Liệt kê các xâu nhị phân có độ dài N. Ví dụ với  $N = 4$  ta có kết quả sau:

0 0 0 0	0 1 0 0	1 0 0 0	1 1 0 0
0 0 0 1	0 1 0 1	1 0 0 1	1 1 0 1
0 0 1 0	0 1 1 0	1 0 1 0	1 1 1 0
0 0 1 1	0 1 1 1	1 0 1 1	1 1 1 1

- 3) Liệt kê các tổ hợp chập K của  $1, 2, \dots, N$ . Ví dụ với  $N=5, K=3$  ta có kết quả sau:

1 2 3	1 3 5	2 4 5
1 2 4	1 4 5	3 4 5
1 2 5	2 3 4	
1 3 4	2 3 5	

- 4) Liệt kê các hoán vị của  $1, 2, \dots, N$ . Ví dụ với  $N=3$  ta có các hoán vị sau:

1 2 3	2 3 1
1 3 2	3 1 2
2 1 3	3 2 1

## Bài tập: Cấu trúc dữ liệu mảng.

5) Viết chương trình xây dựng các thao tác trên đa thức.

- a) Khởi tạo đa thức  $P_n(x)$ ,  $Q_m(x)$ ;
- b) Tìm  $P_n(x_0)$  ;
- c) Tìm đạo hàm cấp I của đa thức;
- d) Tìm  $R(x) = P_n(x) + Q_m(x)$ ;
- e) Tìm  $R(x) = P_n(x) - Q_m(x)$ ;
- f) Tìm  $R(x) = P_n(x) * Q_m(x)$ ;
- g) Tìm  $R(x) = P_n(x) / Q_m(x)$  và đa thức dư.

6) Viết chương trình xây dựng các thao tác trên ma trận.

- a) Tạo lập ma trận A cấp N, B cấp M;
- b) Nhân hai ma trận.
- c) Tìm hạng của ma trận.
- d) Tìm vector riêng và giá trị riêng.
- e) Tính định thức.
- f) Tính nghịch đảo;
- g) Giải hệ PTTT thuần nhất bằng phương pháp Grame..

## 1.9. Xâu ký tự (String)

**Định nghĩa.** Xâu ký tự là một mảng, mỗi phần tử của nó là một ký tự, ký tự cuối cùng là ký tự *null* ('\0') chỉ rõ điểm kết thúc của xâu ký tự. Đây là định nghĩa được mô tả trong thư viện chuẩn của C. Muốn sử dụng các hàm này, ta chỉ cần khai báo sử dụng thư viện <cstring> thay cho <string.h>.

**Khai báo:**

char str[20]; //Khai báo xâu ký tự độ dài không quá 20.

char str[] = "Hello"; //Vừa khai báo vừa khởi đầu cho xâu ký tự.

Tổ chức lưu trữ xâu ký tự được thể hiện như bảng dưới đây:

Phần tử:	str[0]	str[1]	Str[2]	str[3]	str[4]	str[0]
Giá trị:	H	e	I	I	o	'\0'
Địa chỉ:	0x2341	0x2342	0x2343	0x2344	0x2345	0x2346

**Ví dụ:** Kiểm tra giá trị phần tử trong xâu ký tự.

```
#include <iostream>
#include <cstring>
using namespace std;
int main(void) {
    char str[] = "Hello";
    cout<<"Xau str:"<<str<<endl;
    int n = strlen(str); //tinh do dai xau str[]
    for (int i=0; i<=n; i++){
        cout<<"Ky tu str["<<i<<"] ="<<str[i];
    }
    system("PAUSE");
    return(0);
}
```

## Một số hàm thông dụng trong cstring:

STT	Tên hàm và ý nghĩa của hàm
1	strcpy(s1, s2): copy xâu s2 vào s1.
2	strcat(s1, s2): nối xâu s2 vào sau xâu s1
3	strlen(s) : tính độ dài xâu s.
4	strchr( s, c): tìm vị trí đầu tiên của ký tự c trong s
5	strstr(s1, s2): tìm vị trí đầu tiên của s2 trong s1.
6	strupr(s) : đảo ngược xâu s.
7	strcmp (s1, s2) : so sánh hai xâu s1 và s2 theo thứ tự từ điển. Hàm trả lại giá trị lớn hơn 0 khi s1>s2, nhỏ hơn 0 khi s1<s2, bằng 0 khi s1=s2.
	.....

**Ví dụ:** Thực hiện các hàm xử lý xâu ký tự.

```
#include <iostream>
#include <cstring>/Nhớ chõ này
using namespace std;
int main(void ) {
    char s1[] ="Hello";//xau s1 khoi dau la Hello
    char s2[] ="World";//xau s1 khoi dau la Hello
    char s3[10]; //khai bao s3 do dai khong qua 10
    strcpy(s3, s1);//copy s1 vao s3
    cout<<"s3="<<s3<<endl;//s3 chinh la "Hello"
    strrev(s3);//Dao nguoc s3
    cout<<"s3="<<s3<<endl;//s3 chinh la "olleH"
    int k = strcmp(s1, s2); //so sanh s1 va s2
    cout<<"k ="<<k<<endl;
    strcat(s1,s2); //noi s2 vao sau s1
    cout<<"s2="<<s2<<endl;//s3 chinh la "HelloWorld"
    k = strlen(s1); //tinh do dai xau s1
    cout<<"Do dai s1:"<<k<<endl;
    system("PAUSE");
    return 0;
}
```

## 1.9. Xâu ký tự (String)

Thư viện chuẩn của C++ cung cấp một tập các phép toán và thao tác (phương thức) với xâu ký tự được khai báo trong lớp **string**. Từ bây giờ ta hiểu một mảng các ký tự, một string đều là xâu ký tự và chỉ khác biệt về cú pháp khai báo.

**Khai báo:**

```
char      str[20] ; // Khai báo kiểu char ta phải đưa vào số lượng phần tử  
string    str; //Khai báo kiểu string không cần đưa vào số lượng phần tử  
string str = "Hell World"; // Vừa khai báo vừa khởi đầu  
string str ("Hell World"); // Vừa khai báo vừa khởi đầu thế này cũng được
```

**Các phép toán đối với string:** giả sử ta có các string s1, s2. Khi đó:

STT	Phép toán	Ý nghĩa
1	s1 = s2	Copy s2 vào s1
2	s1 = s1 + s2	Nối s2 vào sau s1
3	(s1==s2)	Hỏi s1 và s2 có đúng bằng nhau hay không?
4	(s1>s2)	Hỏi s1 có lớn hơn s2 hay không?
5	(s1<s2)	Hỏi s1 có bé hơn s2 hay không?
6	(s1<=s2)	Hỏi s1 có bé hơn hoặc bằng s2 hay không?
7	(s1!=s2)	Hỏi s1 và s2 có giống nhau hay không?

**Ví dụ:** Kiểm tra các phép toán với tring.

```
#include <iostream>
#include <string>
using namespace std;
int main(void ){
    string s1("Hello"), s2 ="World", s3;
    cout<<"Gia tri s1="<<s1<<endl;
    cout<<"Gia tri s2="<<s2<<endl;
    s3 = s1; //copy hai string
    cout<<"Gia tri s3="<<s3<<endl;
    s3 = s1+s2; //nối hai xau
    cout<<"Gia tri s3="<<s3<<endl;
    cout<<"Kiem tra s1==s2:"<<(s1==s2)<<endl;
    cout<<"Kiem tra s1>s2:"<<(s1>s2)<<endl;
    cout<<"Kiem tra s1<s2:"<<(s1<s2)<<endl;
    cout<<"Kiem tra s1>=s2:"<<(s1>=s2)<<endl;
    cout<<"Kiem tra s1<=s2:"<<(s1<=s2)<<endl;
    cout<<"Kiem tra s1!=s2:"<<(s1!=s2)<<endl;
    system("PAUSE");
}
```

**Một số hàm (phương thức) trong lớp string của C++:** Giả sử s, s1, s2 là các string. Khi đó:

STT	Tên hàm (Phương thức)	Ý nghĩa
1	s.size()	Trả lại độ dài string s
2	s.length()	Trả lại độ dài string s
3	getline(cin, s)	Nhập một dòng từ bàn phím cho string s
4	s.erase(n, k)	Xóa k ký tự trong s kể từ vị trí thứ n
5	s.insert(n, s1)	Chèn s1 vào s kể từ vị trí thứ n.
6	s.insert(n, s1, k, m)	Chèn m ký tự kể từ ký tự thứ k trong s1 vào s kể từ vị trí thứ n.
7	s.replace(n, k, s1)	Thay thế k ký tự trong s kể từ vị trí thứ k bằng xâu s1.
8	s.find(s1)	Trả lại vị trí xuất hiện đầu tiên của s1 trong s.
9	s.rfind(s1)	Trả lại vị trí xuất hiện tiếp theo của s1 trong s.
10	s.at(int i)	Truy nhập đến phần tử thứ I trong string

**Ví dụ:** Kiểm tra các phép toán với tring.

```
#include <iostream>
#include <string>
using namespace std;
int main(void) {
    string s("ABCDEFGHIJK"), s1("EF");
    cout<<"Do dai s:"<<s.length()<<"Do dai s1:"<<s1.size()<<endl;
    s.erase(4, 2);//xoá "EF"
    cout<<"Gia tri moi cua s:"<<s<<endl;
    s.insert(4,s1); //chen s1 vao s ke tu vi tri 4
    cout<<"Gia tri moi cua s:"<<s<<endl;
    s.replace(s.length(),0,s1);//thay the s1 vao cuoi
    cout<<"Gia tri moi cua s:"<<s<<endl;
    int k = s.find(s1); //tim vi tri dau tien cua s2 trong s
    cout <<"Vi tri dau tien "<<s1 <<" :" <<k<<endl;
    k = s.rfind(s1);
    cout <<"Vi tri ke tiep "<<s1 <<" :" <<k<<endl;
    system("PAUSE");return(0);
}
```

## 1.10. Con trỏ (pointer)

**Định nghĩa.** Con trỏ là một biến mà giá trị của nó là địa chỉ của một biến khác.

### Tính chất:

- Một biến con trỏ được phép trỏ đến bất kỳ đối tượng nào có cùng kiểu với nó.
- Một biến con trỏ được phép biến đổi gián tiếp giá trị của biến mà con trỏ trỏ đến.
- Một biến con trỏ có thể trỏ đến bất kỳ một miền nhớ nào, thiết lập giá trị của miền nhớ đó, thay đổi nội dung của miền nhớ.
- Nếu A là một biến thì địa chỉ của A trong bộ nhớ là &A .

### Khai báo:

```
int *A; //Khai báo A là một biến con trỏ kiểu int.  
char *str; //Khai báo str là một con trỏ kiểu char.  
float *X; //Khai báo X là một con trỏ kiểu float  
double *Y; // Khai báo X là một con trỏ kiểu double
```

**Các phép toán trên con trỏ:** Giả sử p là một con trỏ và x là biến có cùng kiểu. Khi đó:

STT	Phép toán	Ý nghĩa
1	p = &x	p trỏ đến miền nhớ dành cho x hay ngắn gọn là p trỏ đến x
2	*p	Lấy nội dung của biến được con trỏ trỏ đến
3	++p	P trỏ đến miền nhớ tiếp theo
4	--p	P trỏ đến miền nhớ sau vị trí hiện tại
5	p += k	P trỏ đến k miền nhớ tiếp theo
6	p -= k	P trỏ đến k miền nhớ sau vị trí hiện tại

**Ví dụ:** Thay đổi nội dung của biến thông qua con trỏ.

```
#include <iostream>
using namespace std;
int main(void) {
    int a = 20, b=10;// biến a có giá trị là 20, b là 10
    int *p; //khai báo p là con trỏ kiểu int
    p = &a; //p trỏ đến địa chỉ ô nhớ dành cho a.
    cout<<"Dia chi p="<<p<<" Dia chi a:"<<&a<<endl;
    cout<<"Gia tri p="<<*p<<" Gia tri a:"<<a<<endl;
    *p = *p + b; //thay đổi giá trị tiếp nối dung của a
    cout<<"Gia tri p="<<*p<<" Gia tri a:"<<a<<endl;
    p = &b; //bây giờ p lại trỏ đến b
    cout<<"Dia chi p="<<p<<" Dia chi b:"<<&b<<endl;
    cout<<"Gia tri p="<<*p<<" Gia tri b:"<<b<<endl;
    *p = *p + 10; //thay đổi giá trị tiếp nối dung của b
    cout<<"Gia tri p="<<*p<<" Gia tri b:"<<b<<endl;
    p = new int; //lúc này p lại trỏ đến ô nhớ mới
    *p = 100; //thiết lập nội dung ô nhớ là 100
    cout<<"Dia chi p="<<p<<" Gia tri p:"<<*p<<endl;
    delete p; //giải phóng ô nhớ p trỏ đến
    system("PAUSE");return 0;
}
```

## Con trỏ và mảng:

- Một mảng được xem là một hằng con trỏ. Tên của mảng là địa chỉ của mảng trong bộ nhớ. Ví dụ ta có mảng `int A[10]`  $\Leftrightarrow$  Hệ thống cấp phát một miền nhớ là  $10 * \text{sizeof(int)}$ . Địa chỉ phần tử đầu tiên được qui ước là tên mảng A và cũng là địa chỉ của phần tử đầu tiên `&A[0]`. Địa chỉ phần tử thứ i là `(A+i) = &A[i]`. Giá trị của phần tử thứ i là `*(A+i) = A[i]`.
- Một con trỏ p được phép trỏ đến một mảng A theo chỉ thị `p = A`. Sau đó p được thực hiện các thao tác như một mảng : `p[0] = 5, p[3] = 7`. Nhưng mảng A thì không được phép trỏ đi đâu cả vì nó là một hằng con trỏ.
- Khi sử dụng một mảng trong lập trình, ta không biết khai báo số phần tử của mảng là bao nhiêu cho đủ. Nếu số lượng phần tử nhỏ thì sợ thiếu không gian nhớ, nếu số lượng phần tử lớn lại gây lãng phí bộ nhớ. Trong tình huống này ta nên sử dụng con trỏ thay thế cho mảng thông qua hai chỉ thị:
  - Chỉ thị **new** : cấp phát miền nhớ cho con trỏ.
  - Chỉ thị **delete**: giải phóng miền nhớ cho con trỏ.

## Ví dụ: Khai báo và cấp phát bộ nhớ cho con trỏ.

`int *A; //Khai báo A là một biến con trỏ kiểu int.`

`A = new int[100]; //cấp phát miền nhớ gồm một 100 biến nguyên cho con trỏ A`

`delete(A); // Giải phóng miền nhớ đã cấp phát cho con trỏ A trước đó.`

`char *str = new char [20] ; //Khai báo và cấp phát bộ nhớ cho con trỏ str.`

**Ví dụ:** Sử dụng mảng giống như con trỏ và sử dụng con trỏ giống như mảng.

```
#include <iostream>
using namespace std;
int main (void ) {
    int A[] = {9, 7, 12, 8, 6, 5}, n= 6;
    int *P; //Khai báo P là con trỏ kiểu int
    for (int i=0; i<n; i++) {//Xử lý mảng như con trỏ
        cout<<"Dia chi A["<<i<<"]="<<(A+i);
        cout<<" Gia tri A["<<i<<"]="<<*(A+i)<<endl;
    }
    P = A; // P trỏ đến A và xử lý P như mảng
    for (int i=0; i<n; i++) {
        cout<<"Dia chi A["<<i<<"]="<<&P[i];
        cout<<" Gia tri A["<<i<<"]="<<P[i]<<endl;
    }
    system("PAUSE");return 0;
}
```

**Ví dụ:** Thay thế mảng bằng con trỏ.

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(void ){
    int *A, n; //Khai báo A là con trỏ kiểu int
    cout<<"Nhập n ="; cin>>n; //Nhập giá trị cho n
    A = new int[n];
    for (int i=0; i<n; i++){ //Thao tác con trỏ giống như mảng
        cout<<"Nhập A["<<i<<"] ="; cin>>A[i];
    }
    cout<<"Giá trị mang A:";
    for(int i = 0; i<n; i++)
        cout<<A[i]<<setw(5);
    delete(A); //Giải phóng bộ nhớ đã cấp cho A khi không còn dùng đến
    system("PAUSE");
}
```

## Mảng các con trỏ:

**Định nghĩa:** Mảng các con trỏ là một mảng mà mỗi phần tử của nó là một con trỏ.

### Khai báo:

Kiểu \* Tên-con-trỏ[số-phần-tử];

**Ví dụ:** Truy nhập mảng một chiều bằng bảng con trỏ.

```
#include <iostream>
const int MAX = 6;
using namespace std;
int main (){
    int A[MAX] = {9, 7, 12, 8, 6, 5};
    int *ptr[MAX]; // Khai báo mảng gồm MAX con trỏ int
    for (int i = 0; i < MAX; i++){
        ptr[i] = &A[i]; // Trỏ đến phần tử thứ i.
    }
    for (int i = 0; i < MAX; i++){
        cout << "Gia tri cua A[" << i << "] = ";
        cout << *ptr[i] << endl;
    }
    system("PAUSE");
    return 0;
}
```

**Ví dụ:** Truy nhập mảng các xâu ký tự bằng mảng con trỏ.

```
#include <iostream>
using namespace std;
const int MAX = 4;
int main (){
    char *names[MAX] = {
        "Tran Anh Tuan",
        "Nguyen Tien Hung",
        "Trinh Xuan Tuan",
        "Tran Xuan Bach",
    };
    for (int i = 0; i < MAX; i++){
        cout << "Gia tri names[" << i << "] = ";
        cout << names[i] << endl;
    }
    system("PAUSE"); return 0;
}
```

**Ví dụ:** Thay thế mảng hai chiều bằng mảng con trỏ.

```
#include <iostream>
#include <iomanip>
using namespace std;
void Init(int *A[], int n, int m){
    for (int i=0; i<n; i++)
        for (int j=0; j<m; j++){
            cout<<"Nhập A["<<i<<""]["<<j<<"]="; cin>>A[i][j];
        }
}
void Result(int *A[], int n, int m) { cout<<"Giá trị ma trận:";
    for (int i=0; i<n; i++) { cout<<endl;
        for (int j=0; j<m; j++) cout<<A[i][j]<<setw(4);
    }
}
int main(void){  int n, m;
    cout<<"Nhập n="; cin>>n; cout<<"Nhập m="; cin>>m;
    int *P[n]; //Khai báo mảng gồm n con trỏ
    for (int i=0; i<m; i++) P[i] = new int[m]; //Mỗi con trỏ trỏ đến mảng m phần tử
    Init(P, n, m); //Thiết lập giá trị cho ma trận
    Result(P,n,m); //Đưa ra kết quả ma trận
    delete(A); system("PAUSE"); return 0;
}
```

## 1.11. Con trỏ và đổi của hàm

**Hàm** (Function, Subroutine, Procedure, Method, Process): Một đoạn chương trình xây dựng một lần, thực hiện nhiều lần ở mọi lúc, mọi nơi trong chương trình và phục vụ mục tiêu chủ quan của người lập trình.

**Thực hiện hàm**: thông qua lời gọi hàm. Hàm được thực hiện khi có lời gọi hàm và được truyền đầy đủ các tham biến cho hàm. Hàm được thực hiện theo hai cơ chế:

- **Cơ chế truyền tham trị**: biến được truyền cho hàm dưới dạng giá trị. Hàm thực hiện theo cơ chế truyền theo tham trị không làm thay đổi nội dung của biến.
- **Cơ chế truyền tham biến**: biến được truyền cho hàm dưới dạng địa chỉ của biến hoặc con trỏ. Hàm thực hiện theo cơ chế truyền theo tham biến sẽ thay đổi nội dung của biến truyền cho hàm.
- Tên của mảng, con trỏ, hoặc địa chỉ được truyền cho hàm đều thực hiện theo cơ chế truyền theo tham biến.

**Hàm đệ qui**: một hàm được viết dưới dạng đệ qui nếu nó gọi đến chính nó với tham biến tiến dần về điểm hội tụ của hàm.

**Ví dụ:** Hàm truyền theo tham biến, tham trị.

```
#include <iostream>
using namespace std;
void Swap (int a, int b) { //Hàm truyền theo tham trị
    int temp = a; a = b; b = temp;
}
void Swap1 (int &a, int &b) {/Hàm truyền theo tham biến
    int temp = a; a = b; b = temp;
}

int main(void){  int n, m;
    int a = 10, b= 20;
    Swap(a, b); //Hoặc Swap (10, 20) là giống nhau
    cout <<"Giá trị a =" << a <<" b=" << b; // a =10, b=20;
    Swap1(a, b); //Không thể gọi theo kiểu Swap1 (10, 20)
    cout <<"Giá trị a =" << a <<" b=" << b; // a =10, b=20;
    system("PAUSE"); return 0;
}
```

**Ví dụ:** Hàm đệ qui tìm ước số chung lớn nhất của hai số.

```
#include <iostream>
using namespace std;
int USCLN (int a, int b ) {
    if (b != 0 ) {
        int r = a %b;
        a = b; b=r;
        return (USCLN(a,b));
    }
    return (a);
}
int main(void ) {
    int a = 9, b =11;
    cout<<"USCLN ="<<USCLN(a,b)<<endl;
    system("PAUSE");
    return(0);
}
```

## 1.12. Cấu trúc (struct)

**Định nghĩa**. Cấu trúc là một kiểu dữ liệu tập thể gồm nhiều thành viên có quan hệ với nhau và được khai báo chung với nhau bằng từ khóa struct. Các thành viên của cấu trúc có thể có kiểu cơ bản, hoặc có kiểu do người dùng định nghĩa, hoặc là chính nó.

**Khai báo:**

```
struct [tên-cấu-trúc] {  
    [kiểu 1] thành-viên 1;  
    [kiểu 2] thành-viên 2;  
    .....;  
    [kiểu n] thành-viên n;  
} danh-sách-biến-cấu-trúc;
```

**Ví dụ.** Định nghĩa cấu trúc Book.

```
struct Book {  
    int     book_id;          //Mã cuốn sách  
    char    title[50];         //Tiêu đề cuốn sách  
    char    author[50];        //Tác giả cuốn sách  
    char    subject[100];       //chủ đề cuốn sách  
} Book1, Book2;
```

**Chú ý:** Book là tên cấu trúc, Book1 và Book2 là biến cấu trúc có kiểu Book. Do vậy, nếu ta khai báo:

```
struct Book Book3;
```

Thì từ khóa struct phải được nhắc lại. Để thuận tiện, ta sử dụng định nghĩa bằng từ khóa typedef.

**Ví dụ.** Định nghĩa cấu trúc Book bằng typedef.

```
typedef struct Book { //Book bây giờ là tên cấu trúc
    int      book_id;          //Mã cuốn sách
    char    title[50];         //Tiêu đề cuốn sách
    char    author[50];        //Tác giả cuốn sách
    char    subject[100];      //chủ đề cuốn sách
};
```

Khi đó khai báo các biến cấu trúc ta không phải nhắc lại từ khóa struct:

Book X, Y; //X và Y là hai biến kiểu Book

**Các phép toán trên cấu trúc:**

- Phép truy nhập vào thành viên của cấu trúc:

Tên-biến-cấu-trúc.tên-thành-viên;

Ví dụ: X và Y là hai biến cấu trúc kiểu Book, thì ta có thể thực hiện:

X.book\_id = 1; Y.book\_id = 2;

X.book\_title = "Lap Trinh C++"; Y.title = "Cong nghe phan mem";

- Phép gán hai cấu trúc: nếu X và Y là hai cấu trúc có cùng kiểu thì ta được phép gán hai cấu trúc. Ví dụ X và Y là hai cấu trúc kiểu Book thì phép toán sau là hợp lệ:

X = Y;

**Ví dụ.** Các phép toán truy nhập thành viên và gán hai cấu trúc.

```
#include <iostream>
#include <string>
using namespace std;
typedef struct Book {
    int Book_id;
    string Title;
    string Author;
    string Subject;
};
int main(void) {    Book X, Y; //Khai báo X, Y là biến cấu trúc kiểu Book
    cout<<"Nhập Book_id:"; cin>> X.Book_id;
    fflush(stdin); //Đón phím Enter tu bổ đệm
    cout<<"Nhập Book_Title:"; getline(cin, X.Title);
    cout<<"Nhập Book_Author:"; getline(cin, X.Author);
    cout<<"Nhập Book_Subject:"; getline(cin, X.Subject);
    Y = X; //Phép gán hai cấu trúc được thực hiện
    cout<<"\n Book_id : "<<Y.Book_id<<endl;
    cout<<" Book_Title : "<<Y.Title<<endl;
    cout<<" Book_Author :"<<Y.Author<<endl;
    cout<<" Book_Subject:"<<Y.Subject<<endl;
    system("PAUSE"); return 0;
}
```

**Mảng các cấu trúc.** Ta cũng có thể định nghĩa một mảng mà mỗi phần tử của nó là một cấu trúc. Ví dụ dưới đây sẽ minh họa cho mảng các cấu trúc Book đã được mô tả ở trên.

```
#include <iostream>
#include <string>
using namespace std;
const int MAX = 100;
typedef struct Book {
    int Book_id;
    string Title, Author, Subject;
};
void Init ( Book X[], int n ) {
    for (int i=0; i<n; i++){
        cout<<"Phan tu thu:"<<i<<endl;
        cout<<"Nhập Book_id:"; cin>> X[i].Book_id; fflush(stdin);
        cout<<"Nhập Book_Title:"; getline(cin, X[i].Title);
        cout<<"Nhập Book_Author:"; getline(cin, X[i].Author);
        cout<<"Nhập Book_Subject:"; getline(cin, X[i].Subject);
        system("cls"); //Xóa màn hình
    }
}
int main(void) { int n; Book X[MAX]; //Khai báo X là mảng cấu trúc
    cout<<"Số lượng sách cần nhập:"; cin>>n;
    Init(X, n); system("PAUSE"); return 0;
}
```

**Con trỏ đến cấu trúc.** Ta cũng có thể định nghĩa một con trỏ đến cấu trúc, cũng có thể cấp phát miền nhớ cho con trỏ bằng new, giải phóng miền nhớ cho con trỏ bằng delete.

### Khai báo con trỏ cấu trúc:

```
Book *X;           // Khai báo con trỏ đến cấu trúc  
X = new Book ;    // Cấp phát miền nhớ cho con trỏ X  
delete(X);        // Giải phóng miền nhớ cho con trỏ X
```

### Các phép toán trên con trỏ cấu trúc:

- Nếu X là một biến cấu trúc thì &X là địa chỉ của miền nhớ dành cho X.
- Nếu X là một biến cấu trúc thì độ lớn không gian nhớ dành cho X là sizeof(tên-cấu-trúc). Ví dụ sizeof(Book) cho lại độ lớn tính theo Byte của cấu trúc Book.
- Nếu X là một con trỏ cấu trúc thì X->thành-viên là phép truy nhập đến thành viên của cấu trúc.
- Nếu X là một con trỏ cấu trúc thì \*X là nội dung của cấu trúc đó.

**Chú ý:** khi sử dụng con trỏ đến cấu trúc, ta cũng có thể khai báo, cấp phát và giải phóng bộ nhớ cho con trỏ. Điểm khác biệt duy nhất giữa biến cấu trúc và con trỏ cấu trúc là phép truy nhập thành viên “->”.

### Ví dụ :

```
Book *X;           // Khai báo con trỏ đến cấu trúc  
X = new Book[10] ;// Cấp phát miền nhớ 10 phần tử Book cho con trỏ X  
delete(X);        // Giải phóng miền nhớ cho con trỏ X  
X[0] .Book_id = 1; // Đây là phép toán được phép
```

:

## Ví dụ về con trỏ cấu trúc.

```
#include <iostream>
#include <string>
using namespace std;
typedef struct Book {
    int Book_id;
    string Title, Author, Subject;
};
void Init(Book *X){ cout<<"Nhập Book_id:"; cin>> X->Book_id;
    fflush(stdin);//Đón phím Enter tu bổ đệm
    cout<<"Nhập Book_Title:";getline(cin, X->Title);
    cout<<"Nhập Book_Author:";getline(cin, X->Author);
    cout<<"Nhập Book_Subject:";getline(cin, X->Subject);
}
void Display( Book X ) { cout<<"\n Book_id : "<<X.Book_id<<endl;
    cout<<" Book_Title : "<<X.Title<<endl;
    cout<<" Book_Author : "<<X.Author<<endl;
    cout<<" Book_Subject:<<X.Subject<<endl;
}
int main(void) {
    Book *X; X = new Book;//cấp phát miền nhớ cho con trỏ:
    Init(X); Display(*X); delete (X);
    system("PAUSE"); return 0;
}
```

## 1.12. Các thao tác trên File

**Định nghĩa .** File là một tổ chức thông tin trên các thiết bị nhớ ngoài được truy nhập gián tiếp thông qua tên file. Nội dung trong file được xác định bởi người tạo ra file. Hệ điều hành chỉ quan tâm đến những thông tin cơ bản về file: tên file, phần mở rộng của file, độ lớn file, ngày giờ tạo file, thời gian gần nhất truy nhập file.

**Các thao tác trên file:** luôn phụ thuộc vào thiết bị lưu trữ file. Bao gồm:

- Mở file. Đóng file.
- Đọc file; Ghi file
- Di chuyển vị trí trong file.
- Và một số thao tác cơ bản khác: đổi tên file, loại bỏ file.

**Khai báo file:** ta có thể sử dụng các hàm trong C được khai báo trong `<cstdio>` hoặc các phương thức trong C++ được khai báo trong `<fstream>`. Các hàm trong lớp `fstream` bao gồm các lớp cơ sở sau:

- Lớp `ifstream` có nguồn gốc từ lớp `istream` cung cấp các thao tác đọc file.
- Lớp `ofstream` có nguồn gốc từ lớp `ostream` cung cấp các thao tác ghi file.
- Lớp `fstream` cung cấp các phương thức đọc và ghi file.

**Khai báo file :**

```
ifstream [tên-biến-kiểu-file]; //Khai báo biến kiểu file chỉ để đọc.  
ofstream [tên-biến-kiểu-file]; //Khai báo biến kiểu file chỉ để ghi.
```

**Ví dụ.**

```
ifstream fp; //khai báo fp là biến kiểu file chỉ để đọc  
ofstream fp1; //khai báo fp1 là biến kiểu file chỉ để ghi  
fstream fp2; //khai báo fp2 là biến kiểu file để đọc và ghi
```

**Một số thao tác đọc file:** Giả sử `fp` là một biến kiểu file. Khi đó, một số phương thức sau dùng để đọc file, ghi file, vừa đọc vừa ghi file.

STT	Tên phương thức	Ý nghĩa
1	<code>fp.open("tên file")</code>	Mở file đã tồn tại để đọc.
2	<code>fp.open("tên file")</code>	Tạo file mới để ghi.
3	<code>fp.open("tên file", ios::in, ios::out)</code>	Mở file đã tồn tại hoặc tạo mới file để đọc hoặc ghi.
4	<code>fp.getline(line, n);</code>	Đọc một dòng từ file
5	<code>fp&gt;&gt;tên-biến;</code>	Nhận giá trị cho biến từ file
6	<code>fp.read (buff, n)</code>	Đọc n byte vào buff từ file
7	<code>fp1&gt;&gt;tên-biến</code>	Ghi nội dung của biến vào file
8	<code>fp.write(buff, n)</code>	Ghi n byte từ buff vào file
9	<code>fp.close()</code>	Đóng file
10	<code>fp.eof()</code>	Kiểm tra cuối file
11	<code>fp.fail()</code>	Kiểm tra lỗi đọc hoặc ghi file
12	<code>fp.seekg(n, ios::&lt;const&gt;)</code>	Di chuyển vị trí thứ n trong file để đọc
13	<code>fp.seekp(n, ios::&lt;const&gt;)</code>	Di chuyển vị trí thứ n trong file để ghi

## Ví dụ. Copy hai file văn bản.

```
#include <iostream>
#include <fstream>
using namespace std;
void Copy_File (char *name1, char *name2){
    ifstream fp1;// khai bao bien file de doc
    fp1.open(name1);//Mo file de doc
    ofstream fp2;// khai bao bien file de doc
    fp2.open(name2);//Mo file de ghi
    char line[255];
    while (!fp1.eof()) { //Lap den cuoi file
        fp1.getline(line, 80);//doc mot dong
        cout<<line<<endl;//dua ra noi dung fp1
        fp2<<line<<endl; //ghi dong line vao fp2;
    }
    fp1.close();//dong file fp1
    fp2.close();//dong file fp2
}
int main(void){
    Copy_File("Copy-File.cpp","New-Copy-File.cpp");
    system("PAUSE"); return(0);
}
```

## BÀI TẬP

1. Cho số tự nhiên N. Hãy viết chương trình chuyển đổi số tự nhiên n thành số ở hệ cơ số b ( $2 \leq b \leq 32$ ).

Dữ liệu vào (Input) cho bởi file data.in theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên K là số lượng các test ( $k \leq 100$ ).
- K dòng kế tiếp ghi lại mỗi dòng một test. Mỗi test bao gồm một cặp số  $N, b$ . Hai số được viết cách nhau một vài khoảng trắng.

Kết quả ra (Output): ghi lại K dòng trong file ketqua.out, mỗi dòng ghi lại bộ ba số  $n, k, x$ . Trong đó  $x$  là số ở hệ cơ số  $b$  được chuyển đổi từ  $n$ . Ví dụ dưới đây minh họa cho file input và output của bài toán.

Input.in

5	
8	2
32	16
255	16
100	10
64	32

Output.out

8	2	1000
32	16	20
255	16	FF
100	10	100
64	32	20

## BÀI TẬP

2. Hãy viết chương trình tìm số các số tự nhiên  $N$  thỏa mãn đồng thời những điều kiện dưới đây ( $N \leq 2^{31}$ ):

- $N$  là số có  $K$  chữ số ( $K \leq 15$ ).
- $N$  là số thuận nghịch (số đối xứng).
- Chuyển đổi  $N$  thành số hệ cơ số  $B$  cũng là một số thuận nghịch ( $2 \leq N \leq 512$ ).
- Thời gian thực hiện chương trình không quá 1sec.

Dữ liệu vào (Input) cho bởi file data.in theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên  $M$  là số lượng các test ( $M \leq 100$ ).
- $M$  dòng kế tiếp ghi lại mỗi dòng một test. Mỗi test bao gồm một cặp số  $N, B$ . Hai số được viết cách nhau một vài khoảng trắng.

Kết quả ra (Output): ghi lại  $M$  dòng trong file ketqua.out, mỗi dòng ghi lại bộ ba số  $N, B, X$ . Trong đó  $X$  là số các số có  $N$  chữ số ở hệ cơ số  $B$  thỏa mãn yêu cầu của bài toán. Ví dụ dưới đây minh họa cho file input và output của bài toán.

<u>Input.in</u>		<u>Output.out</u>		
3		8	2	10
8	2	32	16	20
32	16	255	16	35
255	16			

## BÀI TẬP

3. Cho hai file Data1.in và Data2.in. Hãy viết chương trình tập từ và số lần xuất hiện mỗi từ của cả hai file. Ví dụ dưới đây sẽ minh họa cho Inpu và Output của bài toán.

<b>Data1.in</b>	<b>Data2.in</b>	<b>Ketqua.out</b>
AB AC AD AE AF	AB AC AD AG AH	7
AB AC AD AE AF	AB AC AD AG AH	AB 4
		AC 4
		AD 4
		AE 2
		AF 2
		AG 2
		AH 2

4. Cho hai file Data1.in và Data2.in. Hãy viết chương trình tập từ và số lần xuất hiện mỗi từ xuất hiện trong cả hai file. Ví dụ dưới đây sẽ minh họa cho Inpu và Output của bài toán.

<b>Data1.in</b>	<b>Data2.in</b>	<b>Ketqua.out</b>
AB AC AD AE AF	AB AC AD AG AH	3
AB AC AD AE AF	AB AC AD AG AH	AB 4
		AC 4
		AD 4

## BÀI TẬP

5. Cho hai file Data1.in và Data2.in. Hãy viết chương trình tập từ và số lần xuất hiện mỗi từ xuất hiện trong data1.in nhưng không xuất hiện trong file data2.in. Ví dụ dưới đây sẽ minh họa cho Input và Output của bài toán.

Data1.in	Data2.in	Ketqua.out
AB AC AD AE AF AB AC AD AE AF	AB AC AD AG AH AB AC AD AG AH	2 AE 2 AF 2

6. Cho file dữ liệu data.in ghi lại các số tự nhiên. Biết mỗi số trong file hoặc là các số nguyên tố, hoặc là các số thuận nghịch. Hãy viết chương trình tách tập data.in thành ba tập ketqua1.out, ketqua2.out và ketqua3.out. Trong đó, tập ketqua1.out là tập các số nguyên tố nhưng không là số thuận nghịch và số lần xuất hiện mỗi số trong data.in; ketqua2.out là tập các số thuận nghịch nhưng không là nguyên tố và số lần xuất hiện mỗi số trong data.in; ketqua3.out là tập các số vừa là số nguyên tố vừa là số thuận nghịch và số lần xuất hiện mỗi số trong data.in. Ví dụ dưới đây sẽ minh họa cho file data.in và các file ketqua.out.

Data.in	ketqua1.out	ketqua2.out	ketqua3.out
13 131 171 393	1	2	1
13 131 171 393	13 2	171 2	131 393 2

## BÀI TẬP

7. Cho dãy A[] gồm N số tự nhiên khác nhau và số tự nhiên K. Hãy sử dụng *thuật toán nhánh cận* viết chương trình liệt kê tất cả các dãy con của dãy số A[] sao cho tổng các phần tử trong dãy con đó đúng bằng K. Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N là số các số của dãy số A[] và số tự nhiên K, hai số được viết cách nhau bởi một vài khoảng trắng;
- Dòng kế tiếp ghi lại N số của dãy số A[], hai số được viết cách nhau một vài khoảng trắng.

Các dãy con thỏa mãn điều kiện tìm được ghi lại trong file ketqua.out theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số các dãy con có tổng các phần tử đúng bằng K tìm được;
- Những dòng kế tiếp mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trắng.

Ví dụ dưới đây sẽ minh họa cho file dayso.in và ketqua.out của bài toán.

### Dayso.in

5 50

5 10 15 20 25

### Ketqua.out

3

10 15 25  
5 20 25  
5 10 15 20

## BÀI TẬP

8. Hãy đếm số các xâu nhị phân có độ dài N ( $1 < N < 32$ ) thỏa mãn đồng thời những điều kiện dưới đây:

- Chứa duy nhất một dãy K số 1 liên tiếp ( $1 < K < 32$ );
- Chứa duy nhất một dãy M số 0 liên tiếp ( $1 < M < 32$ );
- Thời gian thực hiện không quá 5sec.

Ví dụ với  $N=5$ ,  $K=3$ ,  $M = 2$  ta đếm được 2 xâu 00111 và xâu 11100 thỏa mãn điều kiện. là các xâu:

Cho tập Input.in ghi lại X bộ test ( $X < 100$ ). Mỗi bộ test là bộ ba số  $N$ ,  $M$ ,  $K$  được ghi trên một dòng. Hãy tìm số các xâu nhị phân thỏa mãn yêu cầu của bài toán ứng với mỗi bộ test. Ghi kết quả mỗi bộ test vào file ketqua.out trên một dòng. Ví dụ dưới đây sẽ minh họa cho file Input và Output của bài toán.

Input.in

5		
5	3	2
5	2	2
5	3	3
6	2	2
6	3	3

Ketqua.out

2
4
0
8
2

## BÀI TẬP

9. **Bài toán cái túi.** Một nhà thám hiểm cần đem theo một cái túi trọng lượng không quá B. Có N đồ vật cần đem theo. Đồ vật thứ  $i$  có trọng lượng  $A_i$  có giá trị sử dụng  $C_i$ . Hãy tìm cách đưa đồ vật vào túi cho nhà thám hiểm sao cho tổng giá trị sử dụng các đồ vật trong túi là lớn nhất.

Dữ liệu vào cho bởi file caitui.in theo khuôn dạng:

- Dòng đầu tiên ghi lại hai số N, B tương ứng với số lượng đồ vật và trọng lượng túi.
- N dòng kế tiếp mỗi dòng ghi lại bộ đôi  $A_i, C_i$  tương ứng với trọng lượng và giá trị của đồ vật thứ  $i$ .

Giá trị tối ưu và phương án tối ưu của bài toán ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại giá trị tối ưu của bài toán.
- Dòng kế tiếp ghi lại phương án tối ưu của bài toán. Hai phần tử khác nhau của phương án tối ưu được viết cách nhau một vài khoảng trắng.

Ví dụ dưới đây sẽ minh họa cho Input và Output của bài toán.

**Data.in**

4	10
5	5
1	3
9	6
3	4

**Ketqua.out**

12			
0	0	1	1

9. Cho dãy gồm  $N$  số nguyên phân biệt  $A[] = \{a_1, a_2, \dots, a_N\}$  và số tự nhiên  $K$  ( $K \leq N \leq 100$ ). Hãy viết chương trình liệt kê tất cả các dãy con  $K$  phần tử giảm dần tự nhiên của dãy số  $A[]$ . Dữ liệu vào cho bởi file dayso.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại hai số tự nhiên  $N, K$ . Hai số được viết cách nhau một vài khoảng trắng.
- Những dòng kế tiếp ghi lại  $N$  số nguyên của dãy số  $A[]$ , hai số khác nhau được viết cách nhau một vài khoảng trắng.

Các dãy con  $K$  phần tử giảm dần của dãy số  $A[]$  tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số tự nhiên  $M$  là số các dãy con  $K$  phần tử giảm dần của dãy số  $A[]$  tìm được;
- $M$  dòng kế tiếp, mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một vài khoảng trắng.

Ví dụ với file dayso.in dưới đây sẽ cho ta file ketqua.out tương ứng.

dayso.in

5 3

20 10

15

5

3

ketqua.out

7

20 10 5

20 10 3

20 15 5

20 15 3

20 5 3

10 5 3

15 5 3

10. Một từ được hiểu là dãy các ký tự không chứa khoảng trắng, dấu tab, dấu xuống dòng và dấu về đầu dòng. Ta nói, từ X có thể nối được với từ Y nếu ký tự đầu tiên của từ X trùng với ký tự cuối cùng của từ Y (Ví dụ từ X = "ABC" có thể nối được với từ Y ="EFA").

**Yêu cầu:** Cho xâu ký tự S có không quá 80 từ (các từ trong S có thể lặp lại). Hãy cho biết có thể nối tất cả các từ trong S để tạo nên một từ mới theo nguyên tắc nêu trên hay không?

**Input:** Dòng đầu tiên ghi số bộ test, không lớn hơn 100. Mỗi bộ test là một xâu ký tự S được viết trên một dòng..

**Output:** Viết ra trên mỗi dòng giá trị “YES” hoặc “NO” nếu ta có thể thực hiện được hoặc không thực hiện được yêu cầu đặt ra của bài toán.

### Ví dụ cho Input và Output:

**INPUT**

2

ABC DEA ABC EAD DEA EAD

ABC ABD ABF ABG ADH ADA

**OUTPUT**

YES

NO

11. Cho ma trận vuông  $C_{i,j}$  cấp N ( $1 \leq i, j \leq N \leq 100$ ) gồm  $N^2$  số tự nhiên và số tự nhiên K (Các số không nhất thiết phải khác nhau) ghi lại trong file matran.in theo khuôn dạng sau:

- Dòng đầu tiên ghi lại số tự nhiên N và K. Hai số được viết cách nhau một vài khoảng trống;
- N dòng kế tiếp ghi lại ma trận vuông  $C_{i,j}$ ; Hai phần tử khác nhau của ma trận được ghi cách nhau bởi một vài khoảng trống.

Hãy sử dụng thuật toán *sinh* (*quay lui*, *nhánh cận*, *qui hoạch động*) viết chương trình lấy mỗi hàng, mỗi cột duy nhất một phần tử của ma trận C sao cho tổng các phần tử này đúng bằng K. Kết quả tìm được ghi lại trong file ketqua.out theo khuôn dạng:

- Dòng đầu tiên ghi lại số các nghiệm tìm được của bài toán.
- Những dòng kế tiếp, mỗi dòng ghi lại N số là một phương án của bài toán, số thứ i ghi lại giá trị j tương ứng với chỉ số cột của phần tử được lựa chọn. Các số được viết cách nhau một vài khoảng trống.

Ví dụ về file viec.in và ketqua.out:

matran.in

6	180				
10	64	57	29	18	15
34	20	19	30	16	12
57	49	40	16	11	19
29	21	46	26	21	18
28	16	11	21	21	37
15	12	15	48	37	30

ketqua.out

6					
2	1	4	6	3	5
3	6	1	5	4	2
3	6	2	4	5	1
4	3	2	6	1	5
5	3	2	6	1	4
6	3	2	5	1	4

12. Một dãy số tự nhiên bất kỳ  $A_N = \{a_1, a_2, \dots, a_N\}$  được gọi là một *đường nguyên tố bậc K* nếu tổng K phần tử liên tiếp bất kỳ của dãy số  $A_N$  là một số nguyên tố ( $K \leq N$ ). Ví dụ dãy số  $A_N = \{3, 27, 7, 9, 15\}$  là một *đường nguyên tố bậc 3*. Cho dãy số  $A_N$ . Hãy liệt kê tất cả các *đường nguyên tố* bậc K có thể có được tạo ra bằng cách tráo đổi các phần tử khác nhau của dãy số  $A_N$ .

Ví dụ với dãy  $A = (3, 7, 9, 15, 27)$  ta sẽ thành lập được 4 dãy nguyên tố thuần nhất bậc 3 như dưới đây:

3	27	7	9	15
15	9	7	3	27
15	9	7	27	3
27	3	7	9	15

13. Cho ma trận vuông  $C_{i,j}$  cấp  $N$  ( $1 \leq i, j \leq N \leq 100$ ) gồm  $N^2$  số tự nhiên và số tự nhiên K (Các số không nhất thiết phải khác nhau). Hãy lấy trên mỗi hàng, mỗi cột duy nhất một phần tử sao cho tổng các phần tử này đúng bằng K. Ví dụ với K = 180 dưới đây sẽ cho ta kết quả tương ứng.

**Ma trận C**

10	64	57	26	18	15
34	20	19	30	16	12
57	49	40	16	11	19
29	21	46	26	21	18
28	16	11	21	21	37
15	12	15	48	37	30

**Kết quả**

2	1	4	6	3	5
3	6	1	5	4	2
3	6	2	4	5	1
4	3	2	6	1	5
5	3	2	6	1	4
6	3	2	5	1	4

## 14. CASE STUDY 1

Hãy viết chương trình thực hiện những công việc dưới đây:

### I. Xây dựng tập thao tác với số nguyên

- Biểu diễn N ở hệ cơ số b.
- Phân tích N thành tích các thừa số nguyên tố.
- Duyệt các số nguyên tố có N chữ số.
- Duyệt các cặp số hữu nghị a, b nhỏ hơn N.
- Duyệt các số hoàn hảo nhỏ hơn N.
- Duyệt các cặp số P, 4P + 1 là nguyên tố nhỏ hơn N.
- Duyệt các số nguyên tố nhỏ hơn N có tổng các chữ số là S.
- Duyệt các số thuận nghịch có N chữ số có tổng các chữ số là S.
- Duyệt các số thuận nghịch có N chữ số sao cho biểu diễn số đó ở hệ cơ số b cũng là số thuận nghịch.
- Xây dựng phép cộng, trừ, nhân chia giữa hai số lớn (512 chữ số).
- Tìm số nguyên tố lớn (512 chữ số).
- Đưa ra 5 thuật toán tìm số nguyên tố khác nhau, so sánh độ phức tạp tính toán của các thuật toán.

## II. Xây dựng tập thao tác với xâu ký tự:

- Tìm  $X = \{x \in S_1 \text{ hoặc } x \in S_2\}$ .
- Tìm  $X = \{x \in S_1 \text{ và } x \in S_2\}$ .
- Tìm  $X = \{x \in S_1 \text{ và } x \text{ không thuộc } S_2\}$ .
- Tìm tập ký tự và số lần xuất hiện mỗi ký tự trong cả  $S_1, S_2$  (Không kể ký tự trùng).
- Tìm tập ký tự và số lần xuất hiện mỗi ký tự thuộc cả  $S_1$  và  $S_2$  (Không kể ký tự trùng).
- Tìm tập ký tự và số lần xuất hiện mỗi ký tự thuộc  $S_1$  nhưng không thuộc  $S_2$  (Không kể ký tự trùng).
- Mã hóa  $X$  bằng kỹ thuật chẵn lẻ.
- Giải mã  $X$  bằng kỹ thuật chẵn lẻ.
- Tìm tập từ và số lần xuất hiện mỗi từ trong  $S_1$  hoặc  $S_2$ .
- Tìm tập từ và số lần xuất hiện mỗi từ trong  $S_1$  và  $S_2$ .
- Tìm tập từ và số lần xuất hiện mỗi từ trong  $S_1$  nhưng không xuất hiện trong  $S_2$ .

### **III. Xây dựng tập thao tác với đa thức:**

- Tạo lập hai đa thức  $P_n(X)$ ,  $Q_m(X)$ .
- Tìm  $P_n(X_0)$ ,  $Q_m(X_0)$ .
- Tìm đạo hàm cấp  $L$  của  $P_n(x)$ ,  $Q_m(x)$ .
- Tìm  $R = P + Q$ .
- Tìm  $R = P - Q$ .
- Tìm  $R = P^*Q$ .
- Tìm  $R = P/Q$  và đa thức dư.
- Xây dựng các thao tác cộng, trừ nhân, chia hai số nguyên bằng đa thức.

### **IV. Xây dựng tập thao tác trên ma trận**

- Tạo lập ma trận.
- Nhân hai ma trận
- Tìm phần tử lớn nhất của ma trận.
- Tìm hạng của ma trận.
- Tìm các vector riêng & giá trị riêng.
- Tìm chuyển vị của ma trận.
- Tìm định thức của ma trận.
- Tìm nghịch đảo của ma trận.
- Giải hệ phương trình tuyến tính thuận nhất  $AX=B$ .

## V. Xây dựng tập thao tác với đa thức bằng cấu trúc

- Tạo lập hai đa thức  $P_n(X)$ ,  $Q_m(X)$ .
- Tìm  $P_n(X_0)$ ,  $Q_m(X_0)$ .
- Tìm đạo hàm cấp  $L$  của  $P_n(x)$ ,  $Q_m(x)$ .
- Tìm  $R = P + Q$ .
- Tìm  $R = P - Q$ .
- Tìm  $R = P^*Q$ .
- Tìm  $R = P/Q$  và đa thức dư.

## VI. Xây dựng tập thao tác trên số phức bằng cấu trúc

- Tạo lập hai số phức.
- Cộng hai số phức.
- Nhân hai số phức.
- Chia hai số phức.
- Lũy thừa số phức.
- Căn bậc hai của số phức.

## VII. Xây dựng hệ quản lý sách bao gồm những thao tác sau

- Nhập sách.
- Hiển thị thông tin về sách
- Sắp xếp theo chủ đề sách.
- Kiểm theo chủ đề sách.

## VIII. Xây dựng tập thao tác trên FILE

- Đếm số dòng trong file.
- Đếm số từ trong file.
- Tìm tập từ và số lần xuất hiện mỗi từ trong Data1.in.
- Tìm tập từ và số lần xuất hiện từ trong Data1.in hoặc data2.in.
- Tìm tập từ và số lần xuất hiện từ trong Data1.in và data2.in.
- Tìm tập từ và số lần xuất hiện từ trong Data1.in nhưng không xuất hiện trong data2.in.
- Mã hóa file bằng kỹ thuật chẵn lẻ.
- Giải mã file bằng kỹ thuật chẵn lẻ.
- Đổi tên file.
- Loại bỏ file . . .

## **VIII. Xây dựng tập thao tác với tập hợp**

- Duyệt các xâu nhị phân có độ dài n.
- Duyệt các tập con K phần tử của 1, 2,..,n.
- Duyệt các hoán vị của 1, 2, ..,n.
- Duyệt các cách chi số N thành tổng các số tự nhiên nhỏ hơn N.
- Duyệt các xâu nhị phân độ dài N có đúng 1 dãy K số 0 và 1 dãy m M số 1 liên tiếp.
- Duyệt các dãy con K phần tử tăng dần tự nhiên của dãy số An.
- Duyệt dãy số gồm N phần tử có tổng số K phần tử bất kỳ là nguyên tố
- Giải bài toán N quân hậu.
- Giải bài toán mã đi tuần.
- Giải bài toán cái túi.
- Giải bài toán người du lịch.
- Giải bài toán cho thuê máy.