NGÔN NGỮ LẬP TRÌNH C++

Ngô Quốc Dũng

Email : quocdung.ngo@gmail.com

Facebook nhóm C++ : https://www.facebook.com/groups/CPP4PTIT13

Tài liệu tham khảo : http://www.tutorialspoint.com/cplusplus/index.htm

Học Viện Bưu Chính Viễn Thông

NGÔN NGỮ LẬP TRÌNH C++

Google

Tổng quan môn học

Part 1: Giới thiệu tổng quan về khác phương pháp lập trình với C++

(trang 15 - 127)

Part 2: Lập trình hướng đối tượng với C++

(trang 128 - 197)

Part 3: Lập trình nâng cao với C++

Cách tính điểm :

Điểm chuyên cần (10%)
Điểm kiểm tra trên lớp (20%)
Điểm đồ án (20% - 5sv/nhóm)
Điểm thi (50%)

Chương 1: Giới thiệu tổng quan về các phương pháp lập trình

- Các phương pháp lập trình
- Phương pháp lập trình hướng cấu trúc
- Các đặc trưng cơ bản của phương pháp hướng đối tượng
- Giới thiệu ngôn ngữ C++
- Bài tập làm quen với C++

Lịch sử và các phương pháp lập trình

phần mềm máy tính là gì?

Một phần mềm máy tính thực hiện một nhiệm vụ cụ thể, và có thể tương tác với người sử dụng và phần cứng của máy tính.

Hướng dẫn - Quy trình

↓

Yêu cầu công việc → Người lao động → Sản phẩm

phầm mềm máy tính

↓

Yêu cầu công việc → máy tính → Sản phẩm

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình

- **Lập trình tuyến tính : mang tính tuần tự**

  vd : Assembly, basic

- **Lập trình hướng cấu trúc : cấu trúc dữ liệu + giải thuật**

  vd : Pascal, Foxpro, C

- **Lập trình hướng đối tượng : Dữ liệu + Hành vi của dữ liệu**

  vd : Java, C++, C#

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Yêu cầu công việc:

Chương trình hello world với C++

```
// Chương trình hello World C++
#include <iostream> // thư viện cho phép hiện thông tin trên màn hình máy tính
using namespace std;

int main() // hàm chính của chương trình
	{
		cout << "Hello World!"; // lệnh hiện thị lên màn hình
		return 0; // chỉ định chương trình kết thúc thành công
	}// kết thúc chương trình
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Một số kiểu dữ liệu đơn giản

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

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Yêu cầu công việc:

Tính tổng 2 số tự nhiên :

```
#include <iostream>
using namespace std;

int main ()
{
   int a,b;
   cout << "nhap so dau tien : ";
   cin >> a;
   cout << "nhap so thu hai : ";
   cin >> b;
   cout << "tong hai so la : "<< a+b;
   return 0;
}
```

```
nhap so dau tien : 3.4
nhap so thu hai : tong hai so la : 3
```

```
#include <iostream>
using namespace std;

int main ()
{
   float a,b;
   cout << "nhap so dau tien : ";
   cin >> a;
   cout << "nhap so thu hai : ";
   cin >> b;
   cout << "tong hai so la : "<< a+b;
   return 0;
}
```

```
nhap so dau tien : 3.4
nhap so thu hai : 4.5
tong hai so la : 7.9
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

```
#include <iostream>
using namespace std;

int main()
{
	int a;
	cout << "nhap gia tri mot so bat ki : ";
	cin >> a;

	if (a > 10){
	cout << "so ban vua nhap lon hon 10 "<< endl;
	}
	else{
		cout << "so ban vua nhap nho hon hoac bang 10 " << endl;
	}
   return 0;
}
```

```
nhap gia tri mot so bat ki : 9
so ban vua nhap nho hon 10
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Yêu cầu công việc

Nhập 3 số tự nhiên bất kì.
Tính tổng 2 số lớn nhất
Hiệu (dương) của 2 số bé nhất

?

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

```
for(lệnh ban đầu ; điều kiện ; lệnh tiếp theo){
    Lệnh trong vòng lặp
}
```

```
#include <iostream>
using namespace std;

int main()
{
    int i;

   for(i=0 ; i<10 ; i++)
        cout << "Toi di hoc day du va dung gio" << endl;
    return 0;

}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Tổng mười số nguyên đầu tiên

```
#include <iostream>
using namespace std;

int main()
{
    int i;
    int a = 0;
    for(i=0 ; i<10 ; i++){
        a = a+i;
    }
    cout << "Tong 10 so nguyen dau tien là : " << a << endl;
    return 0;
}
```

```
Tong 10 so nguyen dau tien là : 45
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Yêu cầu công việc

Nhập một số nguyên bằng bàn phím : n
Tính giai thừa n!
Tính tổng 4 số cuối cùng của dãy số.

?

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

```
#include <iostream>
using namespace std;

int main()
{
    // 10 số bất kì
    int elements[10] = { 3, 14, 15, 9, 26, 5, 35, 8, 7, 10 };

    int i, j;

    for (i=0, j=9; i <= j; i++, j--)
        cout << " Thứ tự "
             << " [" << i << "]=" << So[i]
             << " [" << j << "]=" << So[j]
             << endl;

    return 0;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Yêu cầu công việc:
Lập một chương trình tính so sánh giữa 2 số tự nhiên (a,b).
Tính diện tích một hình vuông có cạnh = a.
Tính tiền điện tiêu thụ với số điện tiêu thụ là a.

STT	Mức sử dụng của một hộ trong một tháng	Giá bán điện (đồng/kWh)
1	Bậc 1: Cho kWh từ 0-50	1.388
2	Bậc 2: Cho kWh từ 51-100	1.433
3	Bậc 3: Cho kWh từ 101-200	1.660
4	Bậc 4: Cho kWh từ 201-300	2.082
5	Bậc 5: Cho kWh từ 301-400	2.324
6	Bậc 6: Cho kWh từ 401 trở lên	2.399

Theo Quyết định số 4887/QĐ-BCT ngày 30/5/2014

Yêu cầu công việc:

Lập một chương trình tính điểm trung bình sinh viên :

- Nhập điểm lý thuyết và điểm thực hành
- Cách tính điểm trung bình = (điểm lý thuyết + điểm thực hành)/2.
- Sinh viên qua nếu điểm trung bình trên 5
- Hiện tên sinh viên, lớp theo học và kết quả : Qua hay không

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Chương trình tính điểm trung bình

```
#include <iostream>
using namespace std;

int main() {
        string ho_ten, ten_lop;
        float diemLyThuyet,diemThucHanh,diemTrungBinh;
        cout << "Nhap ho ten : ";
        cin >> ho_ten;
        cout << "Nhap ten lop : « ;
        cin >> ten_lop;
        cout << "Nhap diem ly thuyet : ";
        cin >> diemLyThuyet;
        cout << "Nhap diem thuc hanh : ";
        cin >> diemThucHanh;

        diemTrungBinh = (diemLyThuyet+diemThucHanh)/2;

        cout << "\nHo ten: " << ho_ten;
        cout << "\ndiem ly thuyet: " << diemLyThuyet;
        cout << "\ndiem thuc hanh: " << diemThucHanh;
        cout << "\ndiem trung binh: " << diemTrungBinh;

        if(diemTrungBinh>=5)
            cout << " => " <<ho_ten << "lop "<< ten_lop <<" Duoc Qua";
        else
            cout << " => " <<ho_ten << "lop "<< ten_lop <<" Bi Truot";
    return 0;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Chương trình tính điểm trung bình P1 (5 điểm)

Chương trình tính điểm trung bình và xét kết quả P2 (6 điểm)

```
#include <iostream>
using namespace std;

int main() {
	string ho_ten, ten_lop;
	float diemLyThuyet,diemThucHanh,diemTrungBinh;
	cout << "Nhap ho ten : ";
	cin >> ho_ten;
	cout << "Nhap ten lop : « ;
	cin >> ten_lop;
	cout << "Nhap diem ly thuyet : ";
	cin >> diemLyThuyet;
	cout << "Nhap diem thuc hanh : ";
	cin >> diemThucHanh;

	diemTrungBinh = (diemLyThuyet+diemThucHanh)/2;

	cout << "\nHo ten: " << ho_ten;
	cout << "\ndiem ly thuyet: " << diemLyThuyet;
	cout << "\ndiem thuc hanh: " << diemThucHanh;
	cout << "\ndiem trung binh: " << diemTrungBinh;

	if(diemTrungBinh>=5)
		cout << " => " <<ho_ten <<" Duoc Qua";
	else
		cout << " => " <<ho_ten <<" Bi Truot";

    return 0;
}
```

```
#include <iostream>
using namespace std;

int main() {
	string ho_ten, ten_lop;
	float diemLyThuyet,diemThucHanh,diemTrungBinh;
	cout << "Nhap ho ten : ";
	cin >> ho_ten;
	cout << "Nhap ten lop : « ;
	cin >> ten_lop;
	cout << "Nhap diem ly thuyet : ";
	cin >> diemLyThuyet;
	cout << "Nhap diem thuc hanh : ";
	cin >> diemThucHanh;

	diemTrungBinh = (diemLyThuyet+diemThucHanh)/2;

	cout << "\nHo ten: " << ho_ten;
	cout << "\ndiem ly thuyet: " << diemLyThuyet;
	cout << "\ndiem thuc hanh: " << diemThucHanh;
	cout << "\ndiem trung binh: " << diemTrungBinh;

	if(diemTrungBinh>=6)
		cout << " => " <<ho_ten <<" Duoc Qua";
	else
		cout << " => " <<ho_ten <<" Bi Truot";

    return 0;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Làm sao để phát triển một chương trình tổng quan ?

```
#include <iostream>
using namespace std;

int main() {
        string ho_ten, ten_lop;
        float diemLyThuyet,diemThucHanh,diemTrungBinh;
        cout << "Nhap ho ten : ";
        cin >> ho_ten;
        cout << "Nhap ten lop : « ;
        cin >> ten_lop;
        cout << "Nhap diem ly thuyet : ";
        cin >> diemLyThuyet;
        cout << "Nhap diem thuc hanh : ";
        cin >> diemThucHanh;

        diemTrungBinh = (diemLyThuyet+diemThucHanh)/2;

        cout << "\nHo ten: " << ho_ten;
        cout << "\ndiem ly thuyet: " << diemLyThuyet;
        cout << "\ndiem thuc hanh: " << diemThucHanh;
        cout << "\ndiem trung binh: " << diemTrungBinh;

        if(diemTrungBinh>=5)
             cout << " => " <<ho_ten <<" Duoc Qua";
        else
             cout << " => " <<ho_ten <<" Bi Truot";
    return 0;
}
```

Lấy thông tin điểm và sinh viên

Xét điểm qua hay không

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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 (proccess): 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
    }
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Ví dụ. Cấu trúc c*hương 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 () { // 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)<<endln; //đưa ra tổng a+b và xuống dòng
      cout<<“Hiệu a - b =“<<hieu(a,b)<<endln; //đưa ra tổng a-b và xuống dòng
      system(“PAUSE”); //dừng lại xem kết quả
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Làm sao để phát triển một chương trình tổng quan ?

Hàm lấy thông tin điểm và sinh viên

`layThongTinDiem()` `layTenLop()` `layTenHocSinh()`

?

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Làm sao để phát triển một chương trình tổng quan ?

Hàm Xét Qua Hay Không

```
void xetQuaHayKhong(float diemTrungBinh, string ho_ten,int diemQua){
    if(diemTrungBinh>=diemQua)
                cout << " => " <<ho_ten <<" Duoc Qua";
    else
                cout << " => " <<ho_ten <<" Bi Truot";
}

int main() {
    string ho_ten, ten_lop;
    float diemTrungBinh;
    cout << "Nhap ho ten : ";
    cin >> ho_ten;
    cout << "Nhap ten lop : ";
    cin >> ten_lop;
    diemTrungBinh = layThongTinDiem();
    xetQuaHayKhong(diemTrungBinh,ho_ten,6);
    return 0;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

```
#include <iostream>
using namespace std;

float layThongTinDiem(){
              float diemLyThuyet,diemThucHanh,diemTrungBinh;
              cout << "Nhap diem ly thuyet : ";
              cin >> diemLyThuyet;
              cout << "Nhap diem thuc hanh : ";
              cin >> diemThucHanh;
              diemTrungBinh = (diemLyThuyet+diemThucHanh)/2;

     return diemTrungBinh;
}

void xetQuaHayKhong(float diemTrungBinh, string ho_ten,int diemQua){
     if(diemTrungBinh>=diemQua)
                    cout << " => " <<ho_ten <<" Duoc Qua";
               else
                    cout << " => " <<ho_ten <<" Bi Truot";
}

int main() {
     string ho_ten, ten_lop;
     float diemTrungBinh;
     cout << "Nhap ho ten : ";
     cin >> ho_ten;
     cout << "Nhap ten lop : ";
     cin >> ten_lop;
     diemTrungBinh = layThongTinDiem();
     xetQuaHayKhong(diemTrungBinh,ho_ten,6);
     return 0;
}
```

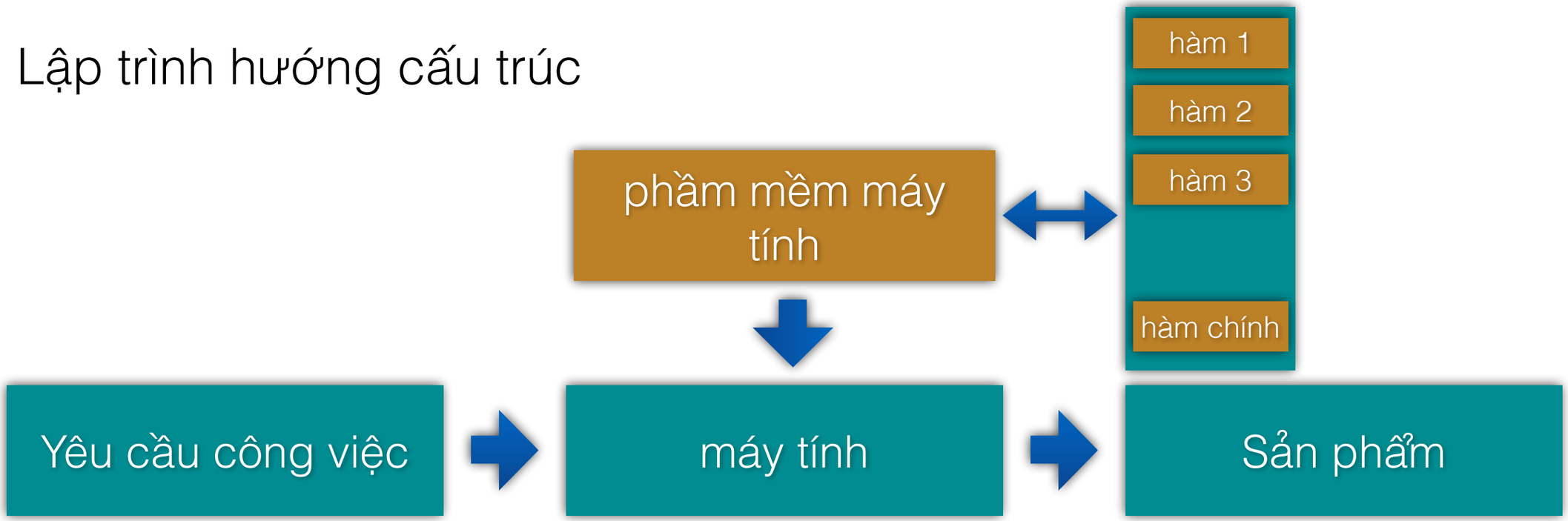
```
int main() {
    function 1()
    function 2()
    function…n()
    return 0;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

các phương pháp hiện lập trình

Lập trình hướng cấu trúc



Trong lập trình hướng cấu trúc, chương trình chính được chia nhỏ thành các chương trình con và mỗi chương trình con thực hiện một công việc xác định. Chương trình chính sẽ gọi đến chương trình con theo một giải thuật, hoặc một cấu trúc được xác định trong chương trình chính

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Đặc trưng cơ bản nhất của lập trình cấu trúc thể hiện ở mối quan hệ:

Chương trình = Cấu trúc dữ liệu + Giải thuật

Trong đó:

Cấu trúc dữ liệu là cách tổ chức dữ liệu, cách mô tả bài toán dưới dạng ngôn ngữ lập trình

Giải thuật là một quy trình để thực hiện một công việc xác định Trong chương trình, giải thuật có quan hệ phụ thuộc vào cấu trúc dữ liệu:

Một cấu trúc dữ liệu chỉ phù hợp với một số hạn chế các giải thuật.

Nếu thay đổi cấu trúc dữ liệu thì phải thay đổi giải thuật cho phù hợp.

Một giải thuật thường phải đi kèm với một cấu trúc dữ liệu nhất định.

Tính chất

Mỗi chương trình con có thể được gọi thực hiện nhiều lần trong một chương trình chính.

Các chương trình con có thể được gọi đến để thực hiện theo một thứ tự bất kì, tuỳ thuộc vào giải thuật trong chương trình chính mà không phụ thuộc vào thứ tự khai báo của các chương trình con.

Các ngôn ngữ lập trình cấu trúc cung cấp một số cấu trúc lệnh điều khiển chương trình.

Ưu điểm

Chương trình sáng sủa, dễ hiểu, dễ theo dõi.

Tư duy giải thuật rõ ràng.

Nhược điểm

Lập trình cấu trúc không hỗ trợ việc sử dụng lại mã nguồn: Giải thuật luôn phụ thuộc chặt chẽ vào cấu trúc dữ liệu, do đó, khi thay đổi cấu trúc dữ liệu, phải thay đổi giải thuật, nghĩa là phải viết lại chương trình.

Không phù hợp với các phần mềm lớn: tư duy cấu trúc với các giải thuật chỉ phù hợp với các bài toán nhỏ, nằm trong phạm vi một modul của chương trình. Với dự án phần mềm lớn, lập trình cấu trúc tỏ ra không hiệu quả trong việc giải quyết mối quan hệ vĩ mô giữa các modul của phần mềm.

Vấn đề

Vấn đề cơ bản của lập trình cấu trúc là bằng cách nào để phân chia chương trình chính thành các chương trình con cho phù hợp với yêu cầu, chức năng và mục đích của mỗi bài toán.
Thông thường, để phân rã bài toán trong lập trình cấu trúc, người ta sử dụng phương pháp thiết kế trên xuống (top-down).

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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 tuyến 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;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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

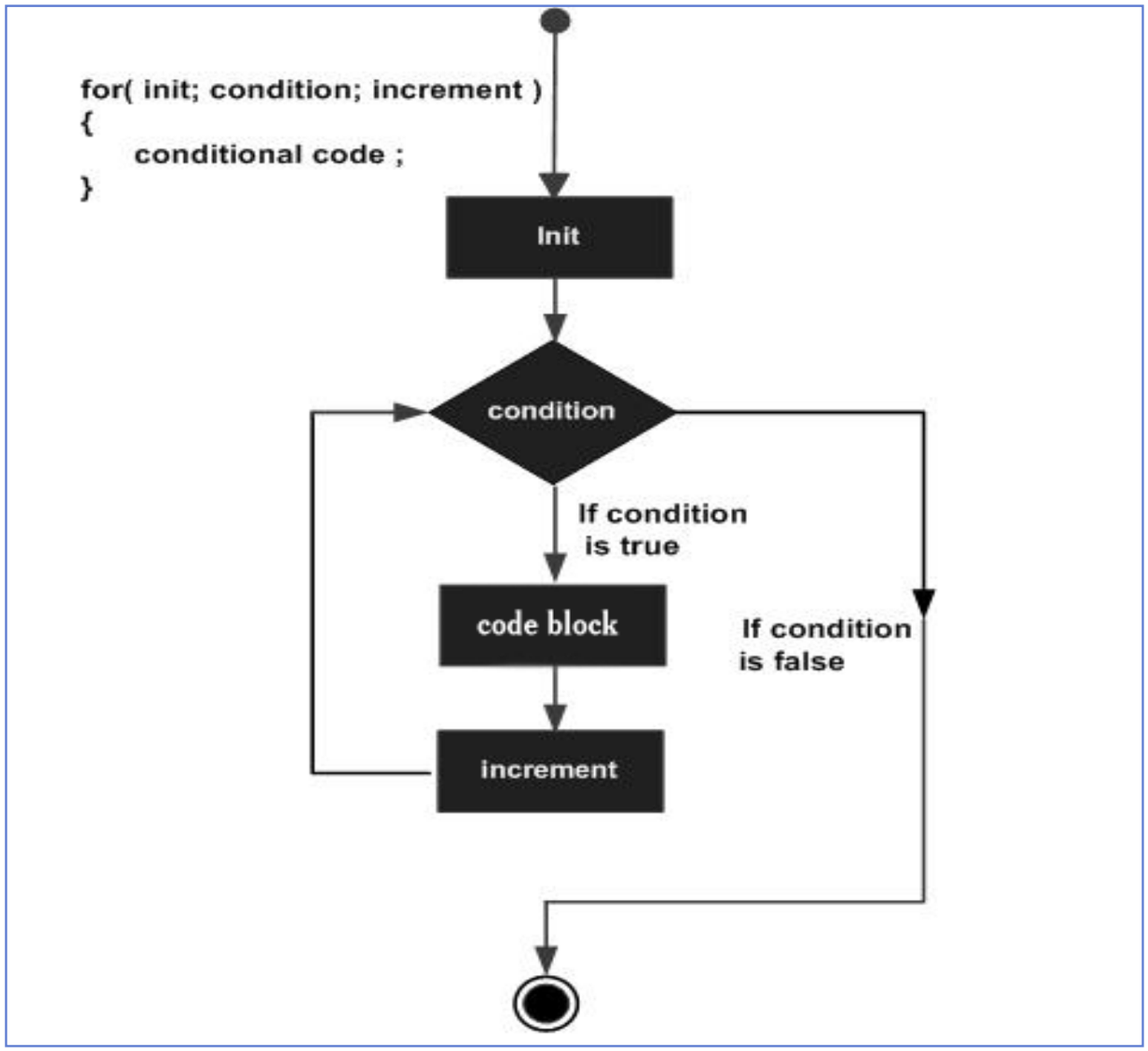
```
#include    <iostream>
using namespace std;
int main( void ) {
    int thang;
    cout<<"Nhap thang:"; cin>>thang;
    switch(thang) {
            case 1: cout<<"January"<<endl; break;
            case 2: cout<<"Febuary"<<endl; break;
            case 3: cout<<"March"<<endl; break;
            case 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;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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

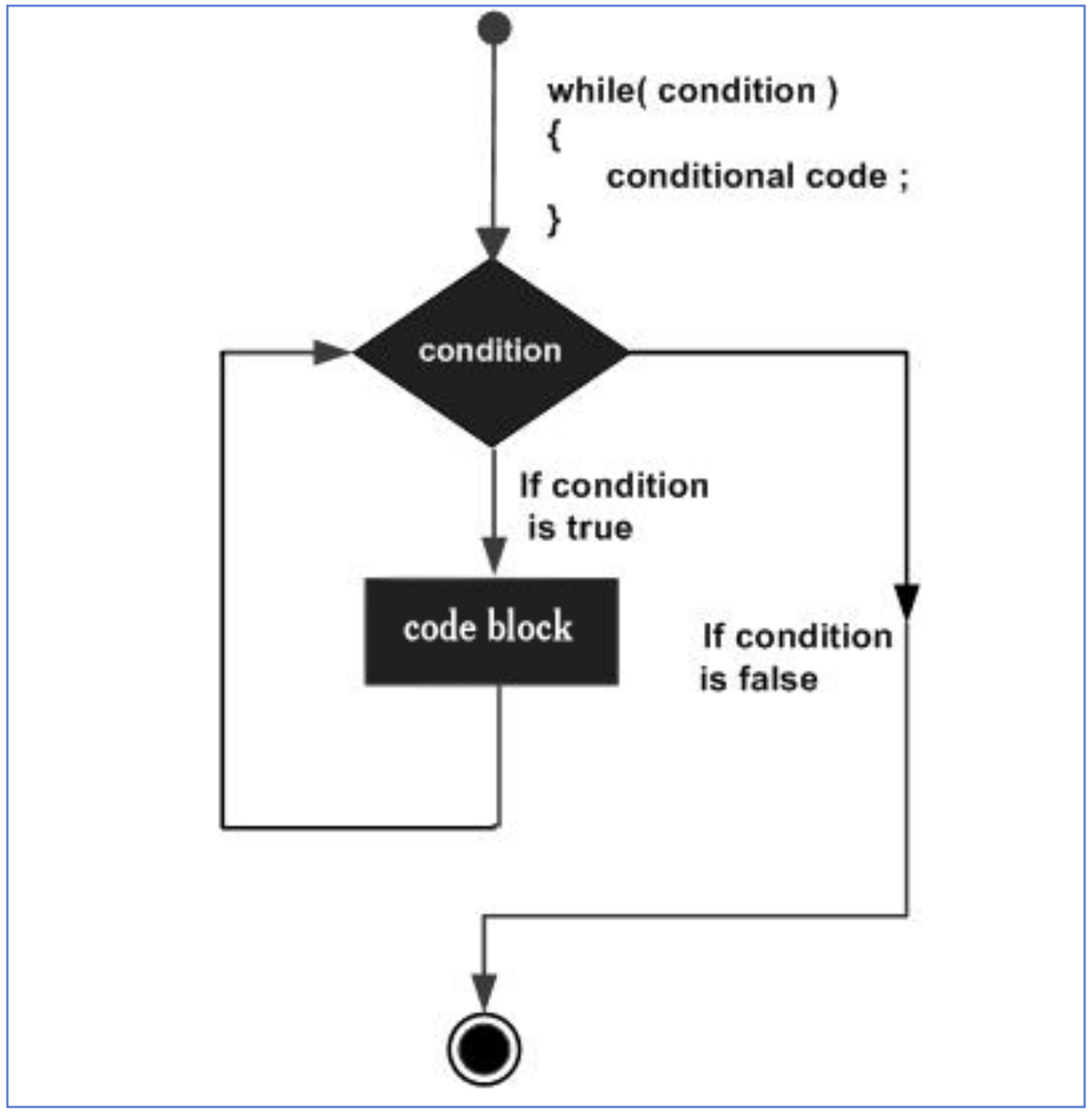
Cấu trúc lệnh lặp for:



Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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



Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Ví dụ : Tìm N và S để (1/N) <=Epsilon cho trước. $S = 1 + 1/2 + 1/3 + \ldots + 1/N$

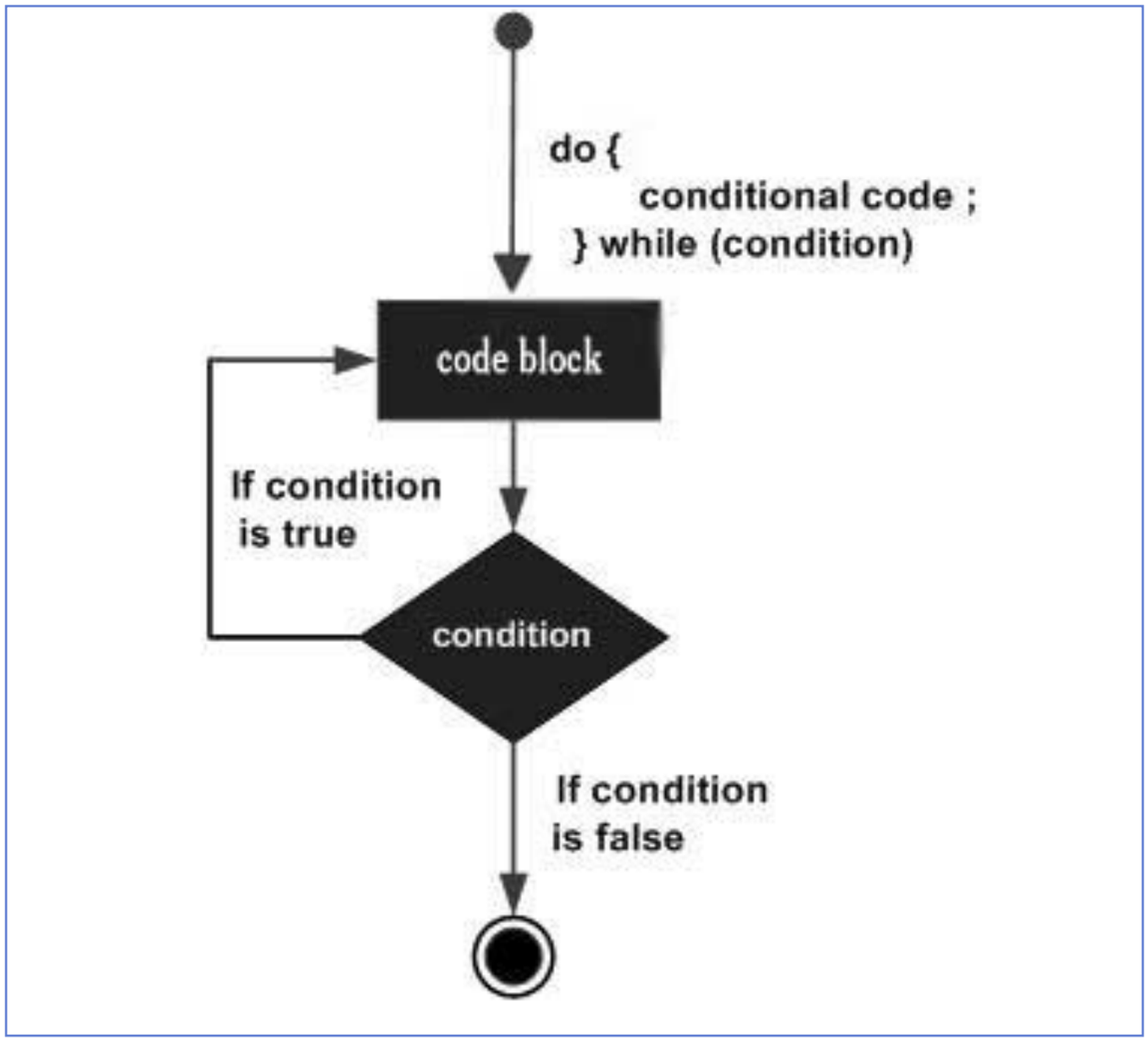
```
#include <iostream>
using namespace std;
int main() {
        float S = 0, Epsilon; int N=1;
        cout<<"Nhap Epsilon="; cin>>Epsilon;
        while( (Epsilon<=(float)1/ (float)N)) {
                S = S + ((float)1) / ((float) N);
                N = N + 1;
        }

        cout<<" N = "<<N <<"  S ="<<S;
        system("PAUSE");
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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



Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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

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

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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

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.

Cho số tự nhiên N. Hãy liệt kê tất cả các số nguyên tố nhỏ hơn N.

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.

Cho số tự nhiên N, hãy phân tích N thành tích các thừa số nguyên tố.

Cho số tự nhiên a và b, hãy tìm ước số chung lớn nhất của hai số.

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 kêt các số hoàn hảo nhoe nhơn N.

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.

Hãy liệt kê mã của các ký tự từ A đến Z cùng với mã nhị phân của ký tự.
Viết chương trình giải phương trình bậc 2.

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:
   K là số có N chữ số;
   K là số nguyên tố;
   Đảo ngược các chữ số trong K là một số nguyên tố;
   Tổng các chữ số trong K cũng là một số nguyên tố;
   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:
   K là số có N chữ số;
   K là số đối xứng;
   Tổng các chữ số của K là số chia hết cho 10;
   Các chữ số của K đều khác 0.

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

13. 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:

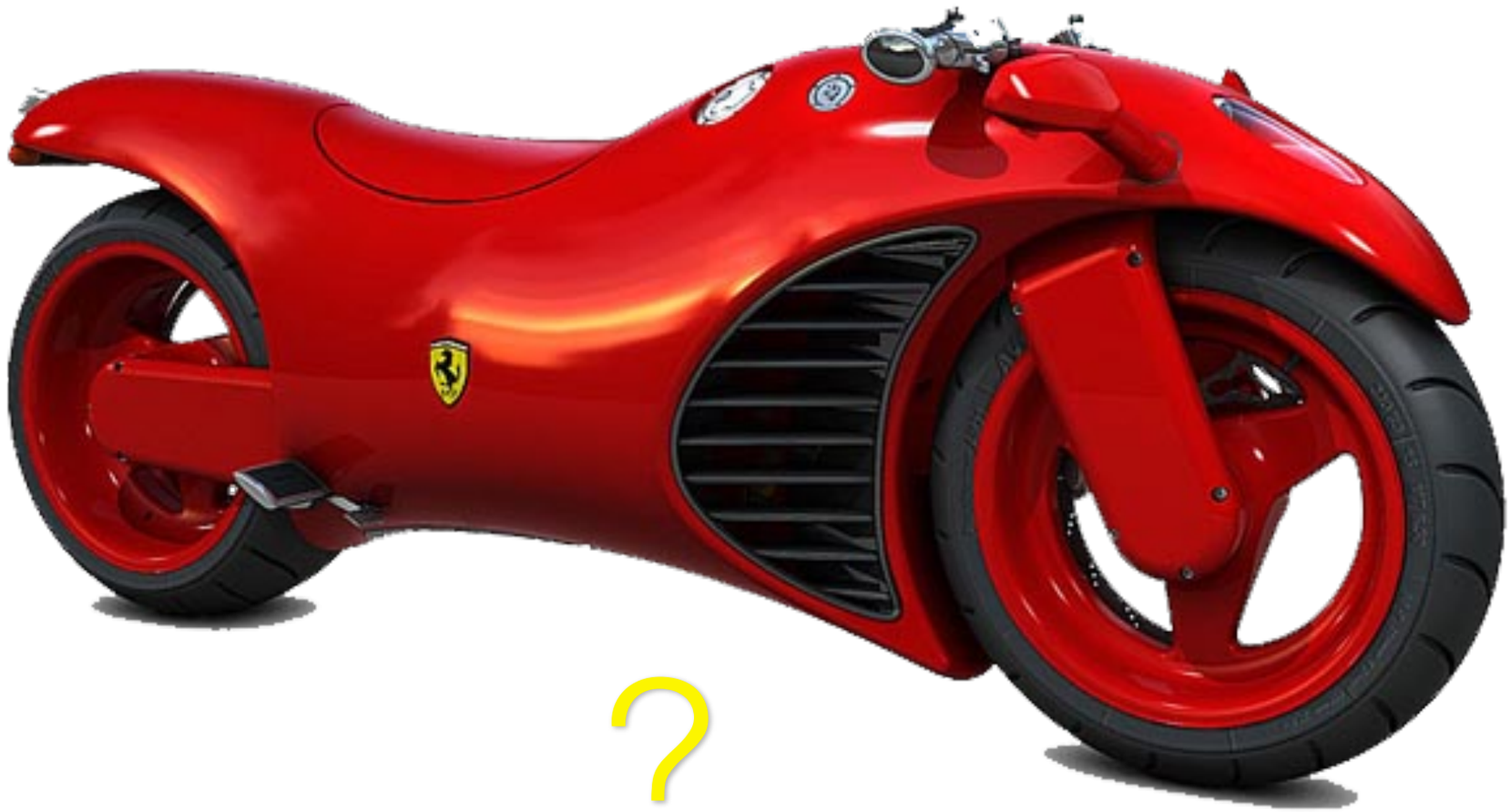
Loại 1: là những số điện thoại có sáu số cuối cùng của nó tạo thành một số đối xứng (thuận nghịch) có sáu chữ số.
Loại 2: là những số điện thoại loại 1 có sáu số cuối cùng của nó là các chữ số khác 0.
Loại 3: là những số điện thoại loại 2 có tổng của sáu chữ số cuối cùng là một số chia hết cho 10.

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.

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Thiết kế trên xuống (Top-Down)



?

Thiết kế mô tô với phương pháp Top - Down

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Yêu cầu công việc:
Lập một chương trình tính so sánh giữa 2 số tự nhiên (a,b).
Tính diện tích một hình vuông có cạnh = a.
Tính tiền điện tiêu thụ với số điện tiêu thụ là a.

STT	Mức sử dụng của một hộ trong một tháng	Giá bán điện (đồng/kWh)
1	Bậc 1: Cho kWh từ 0-50	1.388
2	Bậc 2: Cho kWh từ 51-100	1.433
3	Bậc 3: Cho kWh từ 101-200	1.660
4	Bậc 4: Cho kWh từ 201-300	2.082
5	Bậc 5: Cho kWh từ 301-400	2.324
6	Bậc 6: Cho kWh từ 401 trở lên	2.399

Theo Quyết định số 4887/QĐ-BCT ngày 30/5/2014

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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 foat 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]
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Ví dụ: Khai báo và khởi đầu cho 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.
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Ví dụ: Khởi tạo giá trị và kiểm tra tổ chức lưu trũ 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<<"Nhap A["<<i<<"]="; cin>>A[i]; }
 }
void  Address_Array ( int A[],  int  n) {
     cout<<"\n Dia chi cac phan tu:"<<endl;
     for(int i=0; i<n; i++)  cout<<"Dia chi A["<<i<<"]="<<&A[i]<<endl;
 }
void  Result( int  A[], int  n) {  cout<<"Noi dung mang:";
     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<<"Nhap 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;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Ví dụ: Khởi tạo giá trị và kiểm tra tổ chức lưu trũ 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<<"Nhap A["<<i<<"]["<<j<<"]="; cin>>A[i][j];
        }
}
void Result(int A[][10], int n, int m){
    cout<<"Dia chi va gia tri cac phan tu"<<endl;
    for (int i=0; i<n; i++)
        for (int j=0; j<m; j++){
            cout<<"Gia tri A["<<i<<"]["<<j<<"]="<<A[i][j]<<endl;
            cout<<"Dia chi A["<<i<<"]["<<j<<"]="<<&A[i][j]<<endl;
        }
}
int main(void ) {   int A[10][10], n, m;
    cout<<"Nhap n, m:"; cin>>n>>m;
    Init(A, n, m);  Result(A,n,m);  system("PAUSE");
    return 0;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

```
#include <iostream>
#include <iomanip>
using namespace std;

int main ()
{
   int n[ 10 ]; // n is an array of 10 integers

   // initialize elements of array n to 0
   for ( int i = 0; i < 10; i++ )
   {
      n[ i ] = i + 100; // set element at location i to i + 100
   }
   cout << "Element" << setw( 13 ) << "Value" << endl;

   // output each array element's value
   for ( int j = 0; j < 10; j++ )
   {
      cout << setw( 7 )<< j << setw( 13 ) << n[ j ] << endl;
   }

   return 0;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Bài tập: Cấu trúc dữ liệu mảng.

Viết chương trình sắp xếp mảng theo thứ tự giảm dần/tăng dần
Nhập một số x chèn vào mảng sao cho vẫn đảm bảo tính sắp xếp giảm dần/tăng dần
Tính tổng các số chẵn/lẻ có trong mảng
Tính tổng các số trong các ô có thứ tử lẻ có trong mảng

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

a) Khởi tạo đa thức Pn(x), Qm(x);
b) Tìm Pn(x0) ;
c) Tìm đạo hàm cấp l của đa thức;
d) Tìm R (x) = Pn(x) + Qm(x);
e) Tìm R (x) = Pn(x) - Qm(x);
f) Tìm R (x) = Pn(x) * Qm(x);
g) Tìm R (x) = Pn(x) / Qm(x) và đa thức dư.

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 Cramer..

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 một chiều dưới đây:

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

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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	strrup(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<2, bằng 0 khi s1=s2.
	. . . . .

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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

```
#include <iostream>
#include <cstring>//Nhớ chỗ 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"
    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;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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 =“Hello World”; // Vừa khai báo vừa khởi đầu
string str (“Hello 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?

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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; //noi 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"); return(0);
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Ví dụ: Kiểm tra các phép toán với string.

```
#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);//xoa  "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);
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Viết chương trình đoán 1 chữ đã có sẵn trong một mảng

Viết chương trình đếm số kí tự có trong một mảng (không sử dụng strlen())

Viết chương trình đổi những kí tự thường thành kí tự hoa trong một mảng

t[i] = t[i]+(‘A’-‘a')

Viết chương trình xoá tất cả các nguyên âm có trong mảng

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

```
#include <iostream>

using namespace std;

int main ()
{
   int  var1 = 0;
   int var2[10]={};

   cout << "Address of var1 variable: ";
   cout << &var1 << endl;
   cout << var1 << endl;

   cout << "Address of var2 variable: ";
   cout << &var2 << endl;
   for(int i=0;i<10;i++)
      cout<<var2[i]<< endl;
   return 0;
}
```

Address of var1 variable: 0xbfebd5c0
Address of var2 variable: 0xbfebd5b6

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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 dến dịa 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 gián tiếp nội dung cuả 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 đổii gián tiếp nội dung cuả b
      cout<<"Gia tri p="<<*p<<"  Gia tri b:"<<b<<endl;
      p = new int; ///lúc này p lại trỏ đến ô nhở 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;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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]* Hệ thống cấp phát một miền nhớ là 10*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.
cha *str = new char [20] ; //Khai báo và cấp phát bộ nhớ cho con trỏ str.
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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<<"Nhap 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<<"Nhap A["<<i<<"]="; cin>>A[i];
 }
 cout<<"Gia tri 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");return(0);
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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<<"Nhap A["<<i<<"]["<<j<<"]=“;      cin>>A[i][j];
                }
}
void Result(int *A[], int n, int m) { cout<<"Gia tri ma tran:";
      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<<"Nhap n="; cin>>n; cout<<"Nhap 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
      for(int j=0;j<m;j++)
            delete(P[j]);
      system("PAUSE"); return 0;
}
```

1.11. Con trỏ và đối của hàm

Hàm (Function, Subroutine, Procedure, Method, Proccess): 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 giọ 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.

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Ví dụ: Hàm tryền 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 truyên theo tham biên
     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 =20, b=10;
     system("PAUSE"); return 0;
}
```

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;
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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<<"Nhap Book_id:"; cin>> X.Book_id;
      cout<<"Nhap Book_Title:";getline(cin, X.Title);
      cout<<"Nhap Book_Author:";getline(cin, X.Author);
      cout<<"Nhap 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;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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<<"Nhap Book_id:"; cin>> X[i].Book_id;
                cout<<"Nhap Book_Title:";getline(cin, X[i].Title);
                cout<<"Nhap Book_Author:";getline(cin, X[i].Author);
                cout<<"Nhap Book_Subject:";getline(cin, X[i].Subject);
                system("cls"); //Xoa man hinh
      }
}
int main(void) {  int  n; Book X[MAX]; //Khai báo X là mảng cấu trúc
       cout<<" So luong sach can nhap:"; cin>>n;
      Init(X, n);    system("PAUSE"); return 0;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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
```

:

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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<<"Nhap Book_id:"; cin>> X->Book_id;
      cout<<"Nhap Book_Title:";getline(cin, X->Title);
      cout<<"Nhap Book_Author:";getline(cin, X->Author);
      cout<<"Nhap 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;
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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.h> hoặc các phương thức trong C++ được khai báo trong <ifstream><ofstream><fstream>. Các hàm trong lớp <ifstream><ofstream><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
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

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 ghi
      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);
}
```

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

BÀI TẬP

Tìm tập từ với số lần xuất hiện mỗi từ trong file

Input ;

AB AC AD AE AF

AB AC AD AE AF

Output TanXuat.out

5
AB 2
AC 2
AD 2
AE 2
AF 2

5
AB 0,2
AC 0,2
AD 0,2
AE 0,2
AF 0,2

Input 2 ;

AB AC AD AG AH

AB AC AD AG AH

Output

7
AB 4
AC 4
AD 4
AE 2
AF 2
AG 2
AH 2

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<= b <= 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 <= 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 trống.

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 dự dưới đây minh họa cho file input và output của bài toán.

Input.in		**Output.out**		
5		8	2	1000
8	2	32	16	20
32	16	255	16	FF
255	16	100	10	100
100	10	64	32	20
64	32			

BÀI TẬP

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

N là số có K chữ số (K<=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<=N<=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<=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 trống.

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 dự dưới đây minh họa cho file input và output của bài toán.

Input.in			**Output.out**	
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 Input và Output của bài toán.

Data1.in

AB AC AD AE AF
AB AC AD AE AF

Data2.in

AB AC AD AG AH
AB AC AD AG AH

Ketqua.out

7
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 Input và Output của bài toán.

Data1.in

AB AC AD AE AF
AB AC AD AE AF

Data2.in

AB AC AD AG AH
AB AC AD AG AH

Ketqua.out

3
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 AG AH	2
AB AC AD AE AF	AB AC AD AG AH	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	13 2	171 2	131 2
13 131 171 393		393 2	

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

7. Cho ma trận vuông Ci,j cấp N (*1 i, j N100*) 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

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

8. 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.
   Kiếm theo chủ đề sách.
9. 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 chẫn lẻ.
   Giải mã file bằng kỹ thuật chẫn lẻ.
   Đổi tên file.
   Loại bỏ file . . .

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

Bài tập: Sử dụng Cấu trúc

Viết chương trình sắp xếp mảng theo thứ tự giảm dần/tăng dần
Nhập một số x chèn vào mảng sao cho vẫn đảm bảo tính sắp xếp giảm dần/tăng dần
Tính tổng các số chẵn/lẻ có trong mảng
Tính tổng các số trong các ô có thứ tử lẻ có trong mảng

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

a) Khởi tạo đa thức Pn(x), Qm(x);
b) Tìm Pn(x0) ;
c) Tìm đạo hàm cấp l của đa thức;
d) Tìm R (x) = Pn(x) + Qm(x);
e) Tìm R (x) = Pn(x) - Qm(x);
f) Tìm R (x) = Pn(x) * Qm(x);
g) Tìm R (x) = Pn(x) / Qm(x) và đa thức dư.

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 Cramer..

Lịch sử và các phương pháp lập trình

Các phương pháp lập trình - Lập trình cấu trúc

CASE STUDY

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

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.