Chương 2
CÁC LỆNH VÀO RA
VÀ CÁC LỆNH ĐIỀU KHIỂN

Nội dung

1. Các lệnh vào ra
2. Các lệnh điều khiển
   - 2.1. Lệnh điều kiện,
   - 2.2. Lệnh điều kiện rẽ nhánh,
   - 2.3. Lệnh lặp với số lần xác định,
   - 2.4. Lệnh lặp với số lần không xác định,
   - 2.5. Lệnh ngắt,
   - 2.6. Lệnh tiếp tục

Lệnh vào/ra

Thư viện hàm: iostream.h

Hàm nhập giá trị từ bàn phím: cin>>tên biến;

Ví dụ: int a;

cin>>a;//Lưu giá trị nhập từ phím vào biến a

Hàm xuất giá trị ra màn hình: cout<<tên biến hoặc chuỗi ký tự;

Ví dụ: int a = 5;

cout<<“Giá trị của a = “<<a;

Các biến và chuỗi cách nhau bởi dấu <<

(chuỗi nằm trong cặp dấu nháy kép “”)

Lệnh vào/ra

Thư viện hàm: cout<<setw(n)<<tên biến;

→Chừa một khoảng n ký tự để xuất giá trị

Ví dụ:

```
int a=7, b=9;
cout<<a<<setw(5)<<b;//dùng 5 vị trí để xuất giá trị b.
```

Kết quả: 7   9

Lệnh vào/ra

cout<<setprecision(n);

→Xuất số gồm n-1 chữ số thập phân đã làm tròn.

Ví dụ:

```
float  a=7.56745, b=5.339;
   cout<<a<<endl;
   cout<<setprecision(3)<<a<<endl;
   cout<<setprecision(2)<<b<<endl;
   cout<<setprecision(5)<<b;
```

Kết quả:

```
7.56745
7.57
5.3
5.339
```

Lệnh vào/ra

Xuất ký tự đặc biệt

Ký tự	Ý nghĩa	Ví dụ
\'	Xuất dấu nháy đơn	cout<<" \' "; Kết quả: '
\"	Xuất dấu nháy đôi	cout<<" \" "; Kết quả: "
\\	Xuất dấu chéo ngược "\"	cout<<" \\ "; Kết quả: \
\0	Ký tự Null	Dùng để gán ký tự kết thúc của chuỗi
\a	Alert : Tiếng bip	

Lệnh vào/ra

Xuất ký tự đặc biệt

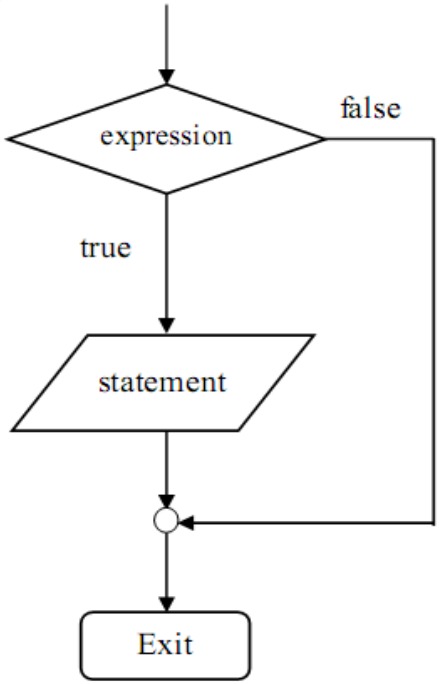
Ký tự	Ý nghĩa	Ví dụ
\t	Tab vào một đoạn ký tự trắng	cout<<"xyz\tzyx”; Kết quả: xyz zyx
\b	Xuất lùi về sau	cout<<"xyz\t\bzyx”; Kết quả: xyzzyx
\n hoặc endl	Xuống dòng	cout<<"xyz\nzyx”; Kết quả: xyz zyx
\r	Về đầu dòng	cout<<"xyz\rzyx”; Kết quả: zyx

33

Lệnh điều kiện

- Dạng 1:
  - Cú pháp:

```
if(expression)
    statement;
```



- Ý nghĩa:

Expression được định trị. Nếu kết quả là true thì statement được thực thi, ngược lại, không làm gì cả.

Lệnh điều kiện

Ví dụ: Viết chương trình nhập vào một số nguyên a. In ra màn hình kết quả a có phải là số dương không.

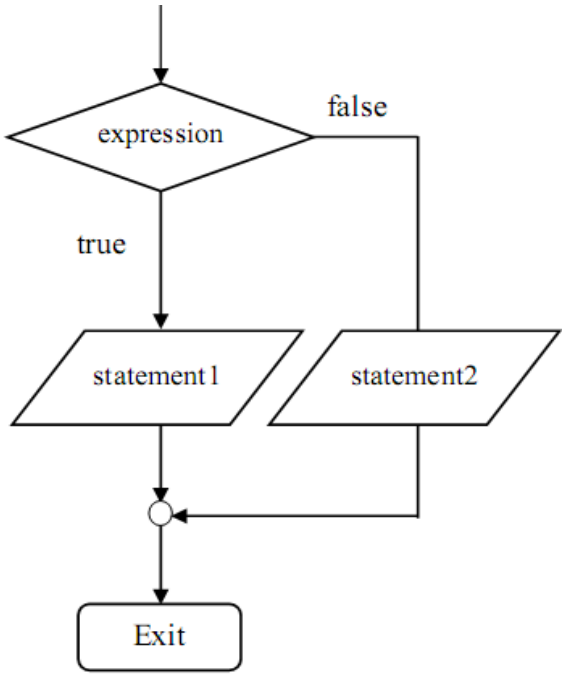
```
#include <iostream.h>
#include <conio.h>
int main()
{
	int a;
	cout << "Input a = ";  cin>>a;
	if(a>=0)
		cout << a << " is a positive.”;
	getch();
	return 0;
}
```

Lệnh điều kiện

- Dạng 2:
  - Cú pháp:

```
if (expression)
    statement1;
else
    statement2;
```

Lưu đồ cú pháp



- Ý nghĩa:
  - Nếu Expression được định là true thì statement1 được thực thi.
  - Ngược lại, thì statement2 được thực thi.

Lệnh điều kiện

Ví dụ: Viết chương trình nhập vào một số nguyên a. In ra màn hình kết quả kiểm tra a là số âm hay dương.

```
#include <iostream.h>
#include <conio.h>
int main()
{
    int a;
    cout << "Input a = ";  cin >> a;
    if(a>=0)
      cout << a << " is a positive.”;
    else
      cout << a << " is a negative.”;
    getch(); return 0;
}
```

Lệnh điều kiện

Lưu ý:

- ✓ Ta có thể sử dụng các câu lệnh if…else lồng nhau. Khi dùng if…else lồng nhau thì else sẽ kết hợp với if gần nhất chưa có else.
- ✓ Nếu câu lệnh if "bên trong" không có else thì phải đặt trong cặp dấu {}

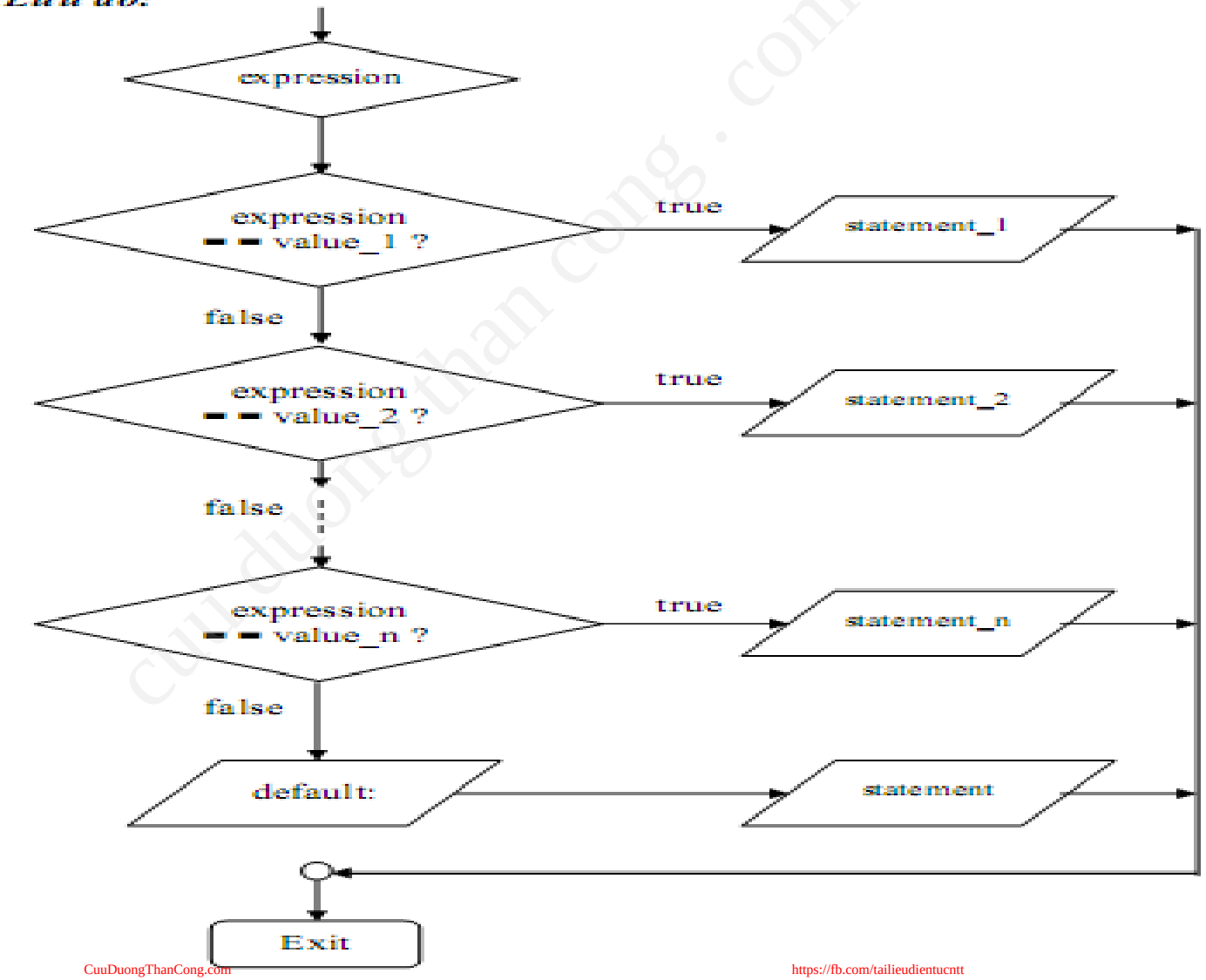
Cấu trúc switch

- Cấu trúc switch là một cấu trúc lựa chọn có nhiều nhánh, được sử dụng khi có nhiều lựa chọn.
- Cú pháp:

```
switch(expression)
 {
        case value_1: statement_1; [break;]
        …
        case value_n: statement_n; [break;]
        [default : statement;]
  }
```

Cấu trúc switch

Lưu đồ:



Cấu trúc switch

- Giải thích:
  - Expression sẽ được định trị.
  - Nếu giá trị của expression bằng value_1 thì thực hiện statement_1 và thoát.
  - Nếu giá trị của expression khác value_1 thì so sánh với value_2, nếu bằng value_2 thì thực hiện statement_2 và thoát…., so sánh tới value_n.
  - Nếu tất cả các phép so sánh đều sai thì thực hiện statement của default.

Cấu trúc switch

✓ Lưu ý:

- Expression trong switch() phải có kết quả là giá trị kiểu số nguyên (int, char, long).
- Các giá trị sau case phải là hằng nguyên.
- Không bắt buộc phải có default.
- Khi thực hiện lệnh tương ứng của case có giá trị bằng expression, chương trình thực hiện lệnh break để thoát khỏi cấu trúc switch.

Cấu trúc switch

Ví dụ: Nhập vào một số nguyên, chia số nguyên này cho 2 lấy phần dư. Kiểm tra nếu phần dư bằng 0 thì in ra thông báo “là số chẵn”, nếu số dư bằng 1 thì in thông báo “là số lẽ”.

```
#include <iostream.h>
#include <conio.h>
void main () {
    int n, remainder;
    cout<<”Input an number: "; cin>>n; remainder = (n % 2);
    switch(remainder)
    {
        case 0: cout << n << ” is an even."; break;
        case 1: cout << n << ” is an odd."; break;
    }
    getch(); }
```

Cấu trúc for

- Cú pháp:

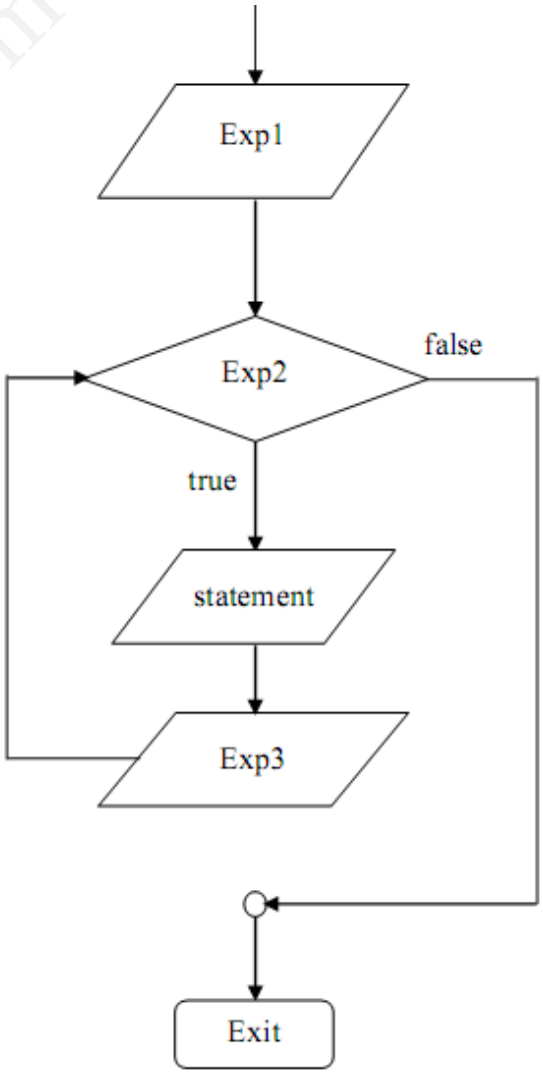
```
for (Exp1; Exp2; Exp3)
    statement;
```

- Ý nghĩa:

– Exp1: là biểu thức khởi tạo được thực hiện.

– Exp2: là biểu thức điều kiện

– Exp3: biểu thức điều khiển lặp



Cấu trúc for

Ví dụ: Viết chương trình tính tổng các số nguyên từ 1 đến n.

```
#include <iostream.h>
#include <conio.h>
void main()
{
      int i, n, sum;
      cout<<”Input a number:”;  cin >> n;
      sum = 0;
      for (i=1 ; i<=n ; i++)
            sum += i;
      cout<<”Sum from 1 to “ << n << ” is: ” << sum;
      getch();
}
```

Cấu trúc for

- C/C++ cho phép Exp1 là một định nghĩa biến

  Ví dụ: for(int i=1; i<=n; ++i)

- Bất kỳ biểu thức nào trong 3 biểu thức của vòng lặp for đều có thể rỗng

  Ví dụ: for(; i != 0;) statement;

- Xóa tất cả các biểu thức trong vòng lặp for sẽ cho một vòng lặp vô tận.

  Ví dụ:

  for (;;) statement;

Cấu trúc while

➢ Cú pháp:

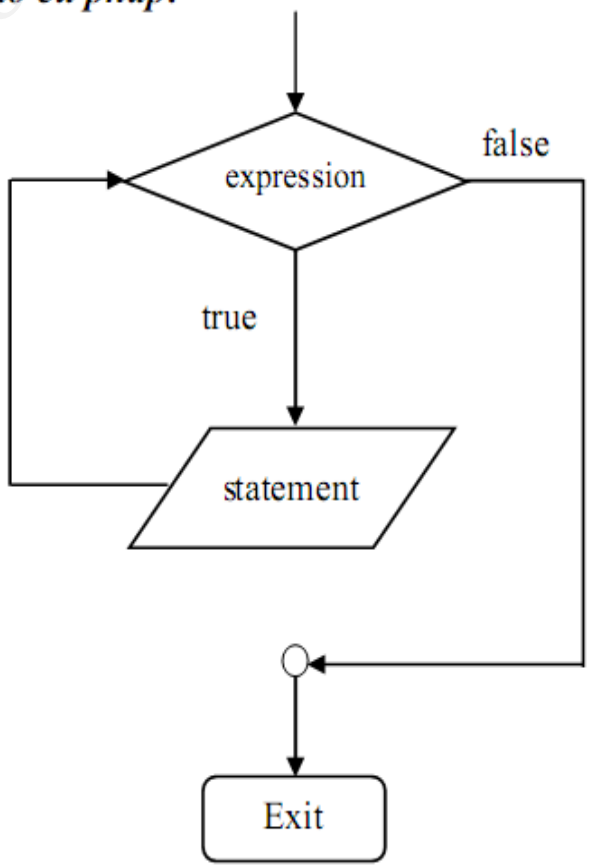
while(expression)

statement;

➢ Ý nghĩa:

- B1: Expression được định trị
- B2: Nếu kết quả là **true** thì statement thực thi và quay lại B1
- B3: Nếu kết quả là **false** thì thoát khỏi vòng lặp while.

Lưu đồ cú pháp:



Cấu trúc while

Ví dụ: Viết chương trình tính tổng các số nguyên từ 1 tới n.

```
#include<iostream.h>
#include<conio.h>
void main ()
{
    int i, n, sum;
    cout<<"Input n= ";  cin >> n;
    i = 1; sum = 0;
    while(i<=n) {
       sum += i;  i++; }
    getch();
}
```

Cấu trúc do … while

➢ Cú pháp:

```
do {
        statement;
}while(expression);
```

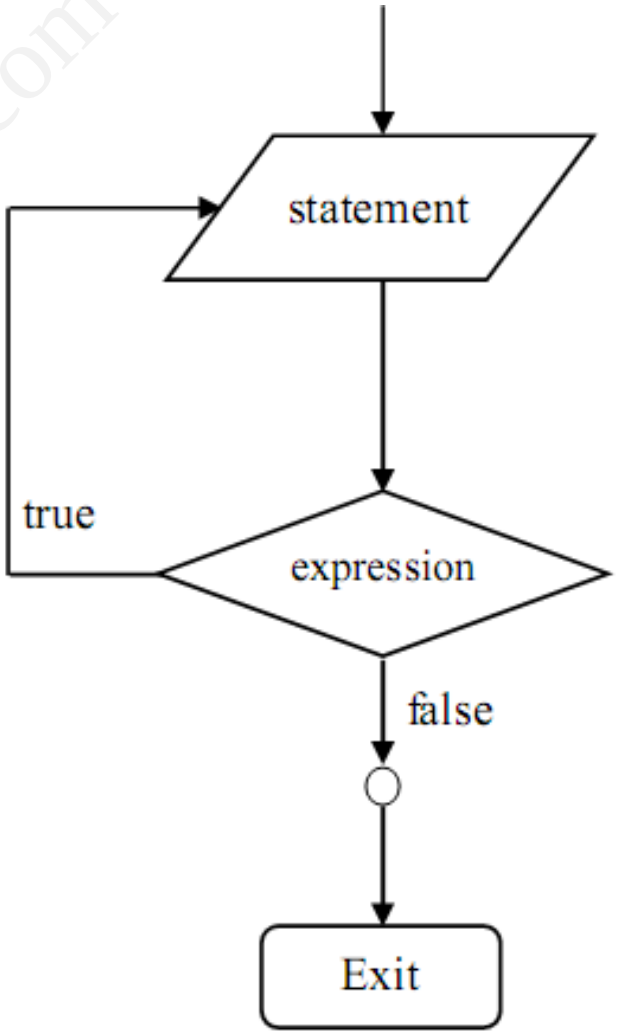
- Ý nghĩa:

– Statement được thực hiện

– Expression được định trị.

– Nếu expression là true thì quay lại bước 1

– Nếu expression là false thì thoát khỏi vòng lặp.



Cấu trúc do … while

Ví dụ 1: Viết chương trình in dãy số nguyên từ 1 đến 10.

```
#include <iostream.h>
#include <conio.h>
void main ()
{
      int i=1;
      clrscr();
      cout<<"Display one to ten: ";
      do
      {
            cout << setw(3) << i;
            i+=1;
       } while(i<=10);
      getch();
}
```

Vòng lặp while và do …. While

```
while(điều kiện )
{
      //thân vòng lặp
}
```

- Nếu điều kiện đúng thì thân vòng lặp đc thực hiện, nếu đk sai thì kết thúc vòng lặp
- Trong hàm while thì thân vòng lặp có thể KHÔNG ĐƯỢC thực thi lần nào cả

```
do
{
   //thân vòng lặp
}while(điều kiện);
```

- Thân vòng ít nhất được thực thi 1 lần
- Hàm do while sẽ thực thi xong rồi mới kiểm tra điều kiện

Thông thường ta sử dụng while, do … while để thiết lập vòng lặp vĩnh cửu để thực các dòng lệnh liên tục nào đó.
Ta sẽ kết hợp với từ khóa break để thoát khỏi vòng lặp vĩnh cửu.

Lệnh break

- Lệnh break dùng để thoát khỏi một cấu trúc điều khiển mà không chờ đến biểu thức điều kiện được định trị.
- Khi break được thực hiện bên trong 1 cấu trúc lặp, điều khiển (control flow) tự động nhảy đến lệnh đầu tiên ngay sau cấu trúc lặp đó.
- Không sử dụng lệnh break bên ngoài các cấu trúc lặp như while, do...while, for hay cấu trúc switch.

Lệnh break

Ví dụ: Đọc vào một mật khẩu người dùng tối đa attempts lần

```
for (i=0; i<attempts ; ++i)
{
    cout<<“Input a password:”;
    cin >> passWord;
    if (check(passWord)) //kiểm tra mật khẩu
        break;     // thoát khỏi vòng lặp
    cout <<“Password is wrong!\n”;
}
```

Lệnh continue

- Lệnh continue dùng để kết thúc vòng lặp hiện tại và bắt đầu vòng lặp tiếp theo.
- Lệnh continue chỉ được dùng trong thân các cấu trúc lặp như for, while, do...while.
- Câu lệnh continue thường đi kèm với câu lệnh if.

Lệnh continue

Ví dụ: Một vòng lặp thực hiện đọc một số, bỏ qua những số âm, và dừng khi số nhập vào là số 0.

```
do
{
    cin >> num;
    if (num < 0)
        continue;  // process num here
} while(num != 0);
```

Mảng (array)

- Mảng là một tập hợp các giá trị có cùng kiểu dữ liệu nằm liên tiếp nhau trong bộ nhớ và được tham chiếu bởi một tên chung chính là tên mảng.
- Mỗi phần tử của mảng được tham chiếu thông qua chỉ mục (index).

Mảng (array)

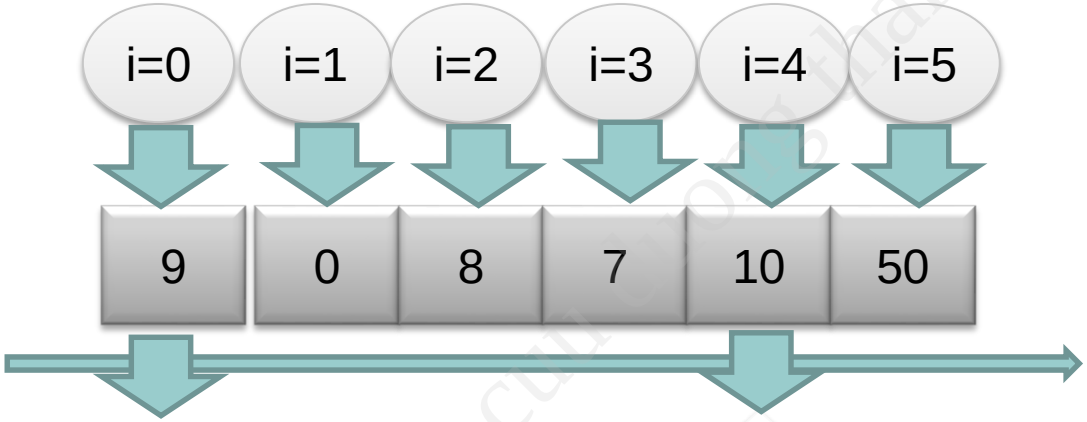
- Nếu mảng có n phần tử thì phần tử đầu tiên có chỉ mục là 0 và phần tử cuối có chỉ mục là n-1.
- Để tham chiếu đến một phần tử ta dùng tên mảng và chỉ mục của phần tử được đặt trong cặp dấu [].

  Ví dụ: a[0]
- Số lượng phần tử có trong mảng được gọi là *kích thước* của mảng, luôn *cố định*, phải được *xác định trước* và không đổi trong suốt quá trình thực hiện chương trình.

Mảng (array)

Mảng là 1 đối tượng dùng để lưu trữ các phần tử có cùng một kiểu dữ liệu

Chỉ số mảng bắt đầu được tính từ 0



Mảng M có 6 phần tử

N=6

M[0]

M[4]

M[i]

Là phần tử
tại vị trí thứ i

Đối với duyệt mảng, người ta thường dùng vòng for, ít khi dùng vòng while, do.. while

Mảng một chiều

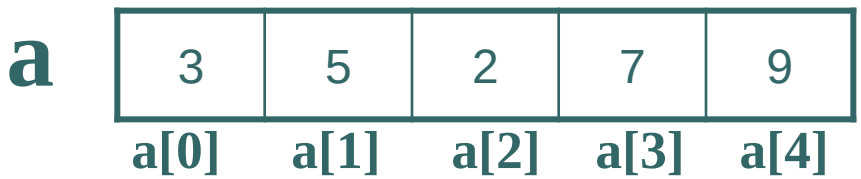
Khai báo một mảng một chiều

✓ Cú pháp:

type arrayName[elements];

- ***type***: kiểu dữ liệu của mỗi phần tử mảng.
- ***arrayName***: tên mảng
- ***elements***: số phần tử có trong mảng

Ví dụ: int a[5]



Mảng một chiều

Ví dụ:

```
int a[100]; //Khai báo mảng số nguyên a gồm 100 phần tử
float b[50]; //Khai báo mảng số thực b gồm 50 phần tử
char str[30]; //Khai báo mảng ký tự str gồm 30 ký tự
```

Mảng một chiều

- Mảng phải được khai báo tường minh
- Kích thước (tính bằng byte) của mảng được tính theo công thức:

Total_size = sizeof(type) * elements

Ví dụ:

int num[100];

Mảng num có kích thước là:

2bytes * 100 = 200bytes (giả sử int chiếm 2 bytes)

Mảng một chiều

- Mỗi phần tử mảng là một biến thông thường.

Ví dụ:

```
int num[3];
num[0] = 2; //gán 2 cho phần tử num[0]
num[1] = num[0] + 3 //num[1] có giá trị 5
num[2] = num[0] + num[1]; //num[2] có giá trị 7
cout << num[1]; //In ra giá trị 5
```

Mảng một chiều

Khai báo và khởi tạo mảng một chiều

- Cú pháp:

```
type arrayName[]= {value1, value2, ..., valuen};
```

- Lưu *ý*:
  - Không khai báo kích thước mảng.
  - Số lượng phần tử trong mảng là số các giá trị được cung cấp trong cặp dấu ngoặc {}, được phân cách nhau bởi dấu phẩy.

Mảng một chiều

Ví dụ:

```
int soChan[] = {2,4,6,8,10};
```

Mảng soChan có 5 phần tử lần lượt là:

soChan[0] có giá trị là 2

soChan[1] có giá trị là 4

...

soChan[4] có giá trị là 10

Mảng một chiều

Khai báo và gán giá trị ban đầu cho mảng một chiều:

Gán từng phần tử

int a[5] = {3, 6, 8, 1, 12};

Giá trị	3	6	8	1	12
Vị trí	0	1	2	3	4

Gán số lượng phần tử khởi tạo ít hơn số phần tử của mảng:

int a[8] = {3,5,2}

Giá trị	3	5	2	0	0	0	0	0
Vị trí	0	1	2	3	4	5	6	7

Mảng một chiều

Truy xuất giá trị

Tên mảng [vị trí cần truy xuất]

```
void main()
{
    int a[5] = {3, 6, 8, 11, 12};
    cout<<“Giá trị mảng tại vị trí 3 = “<<a[3];
}
```

Kết quả: Giá trị mảng tại vị trí 3 = 11

Mảng một chiều

Các thao tác trên mảng:

- ✓ Nhập dữ liệu
- ✓ Xuất dữ liệu
- ✓ Tìm kiếm
- ✓ Đếm
- ✓ Sắp xếp
- ✓ Kiểm tra mảng thỏa điều kiện cho trước
- ✓ Tách/ ghép mảng
- ✓ Chèn / xóa

Mảng một chiều

Nhập dữ liệu

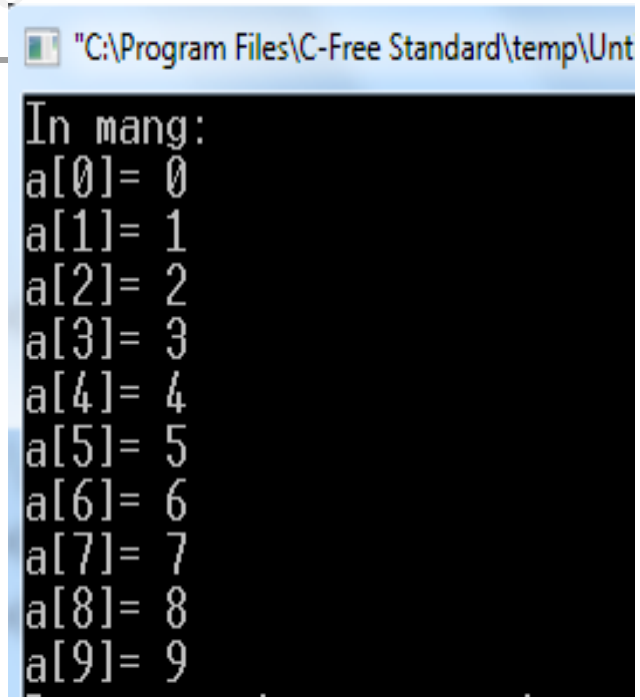
Có 2 cách nhập dữ liệu cho từng phần tử trong mảng:

- Cách 1: nhập bằng tay từ bàn phím
- Cách 2: cho máy tự động nhập

Mảng một chiều

Ví dụ: Tạo một mảng nguyên a có N phần tử. Mỗi phần tử có giá trị là chỉ mục của nó. In mảng ra màn hình.

```
#include<iostream.h>
#define N 10
void main()
{
      int a[N];
      for(int i=0;i<10;i++)
            a[i]=i;
      cout<<"In mang\n";
      for(int i=0;i<N;i++)
            cout<<"a["<<i<<"] = "<<a[i]<<"\n";
}
```



Mảng một chiều

```
void Nhapmang (int a[], int n)
{
    for (int i = 0; i < n; i ++)
    {
        cout<<"Nhap phan tu thu "<<i<<": ";
        cin>>a[i];
    }
}

void Xuatmang (int a[], int n)
{
    for (int i = 0; i < n; i ++)
        cout<<" "<<a[i]<<"\t";
}
```

Mảng một chiều

Nhập mảng tự động bằng máy:

```
#include<iostream.h>
#include<stdlib.h>
#include<time.h>
```

Để khởi tạo xuất ra các số ngẫu nhiên, ta phải viết hàm này trước:

srand(time(NULL));

Hàm trên có ý nghĩa là yêu cầu máy tính khởi động để tạo các số ngẫu nhiên : Tức là mỗi lần chạy chương trình thì nó sẽ tự động phát sinh ra các con số có giá trị khác nhau.

**** nếu như không có dòng này thì mỗi lần chạy chương trình sẽ cho ra có số ngẫu nhiên GIỐNG NHAU.

Mảng một chiều

Để lấy giá trị ngẫu nhiên , ta dùng công thức bên dưới:

```
M[i]=rand()%100;
```

rand()%n: Tức lấy lấy các giá trị ngẫu nhiên từ 0 cho tới n-1

Như vậy nếu ta gọi rand()%100: tức là lấy giá trị ngẫu nhiên từ 0 đến 99

Đề bài : Hãy xuất các số ngẫu nhiên từ -55 cho tới 150

(-77 →88)➔ Rand()%n -77=88 ➔ n=166

(30→95) ➔ rand()%n +30=95 ➔ n=66

(-55→150) ➔ Rand()%n -55 =150 ➔ n=206

(-50 →-100) ➔?

Mảng một chiều

```
#include<iostream.h>
#include<stdlib.h>
#include<time.h>
#define Max 100
void Nhapmang(int a[],int n);
void Xuatmang(int a[],int n);
void main()
{
     int M[Max];
     int n;
     cout<<"Nhap so phan tu mang: ";
     cin>>n;
     Nhapmang(M,n);
     cout<<"\n Xuat mang: \n";
     Xuatmang(M,n);

}
```

```
void Nhapmang(int a[],int n)
{
     srand(time(NULL));
     for(int i=0;i<n;i++)
     {
          a[i]=rand()%100;
     }
}
void Xuatmang(int a[],int n)
{
     for(int i=0;i<n;i++)
     {
          cout<<" "<<a[i]<<"\t";
     }
}
```

Mảng một chiều

Ví dụ: Chương trình nhập vào mảng một chiều số nguyên a, kích thước n. In ra các phần tử có giá trị lớn hơn x có trong mảng

```
#include<iostream.h>
#define MAX 100
void NhapMang(int a[], int n);
void XuatMang(int a[], int n);
void LietKeLonHonX(int a[], int n, int x);
void main()
{
    int a[MAX], n, x;
    cout<<"Nhap vao kich thuoc mang: ";
    cin>>n;
    NhapMang(a,n);
    cout<<"Cac phan tu cua mang:\n";
    XuatMang(a,n);
    cout<<"Nhap gia tri x: ";
    cin>>x;
    cout<<"Cac phan tu co gia tri lon hon \n";
    LietKeLonHonX(a,n,x);
}
```

Mảng một chiều

```
void NhapMang(int a[], int n)
{
    for(int i=0; i<n; i++)
    {
        cout<<"Nhap phan tu tai vi tri "<<i<<": ";
        cin>>a[i];
    }
}
void XuatMang(int a[], int n)
{
    for(int i=0; i<n; i++)
        cout<<a[i]<<"\t";
}
void LietKeLonHonX(int a[], int n, int x)
{
    for (int i = 0; i<n; i++)
        if (a[i] > x)
        cout<<a[i]<<"\t";
}
```

Mảng một chiều

- Sắp xếp:

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

Mảng nhiều chiều

- C/C++ hỗ trợ mảng nhiều chiều. Dạng đơn giản nhất của mảng nhiều chiều là mảng hai chiều.
- Mảng hai chiều thực chất là mảng của những mảng một chiều. Ta có thể xem mảng hai chiều là một ma trận gồm các hàng và các cột
- Khai báo mảng hai chiều:

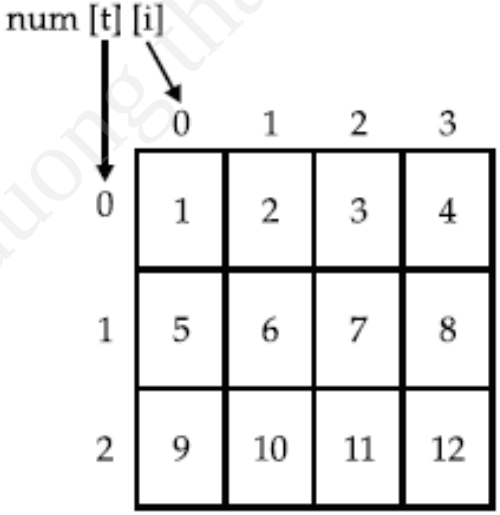
```
type arrayName[rows][columns];
```

  - *rows*: số hàng
  - *columns*: số cột

Mảng nhiều chiều

Ví dụ: Khai báo mảng số nguyên 3 hàng 4 cột

int a[3][4]



Mảng nhiều chiều

Ví dụ: **int M[5][3];**

- Khai báo 1 mảng 2 chiều tên là M, có 5 dòng, 3 cột.
- Tổng số phần tử trong mảng = 5*3 =15 phần tử

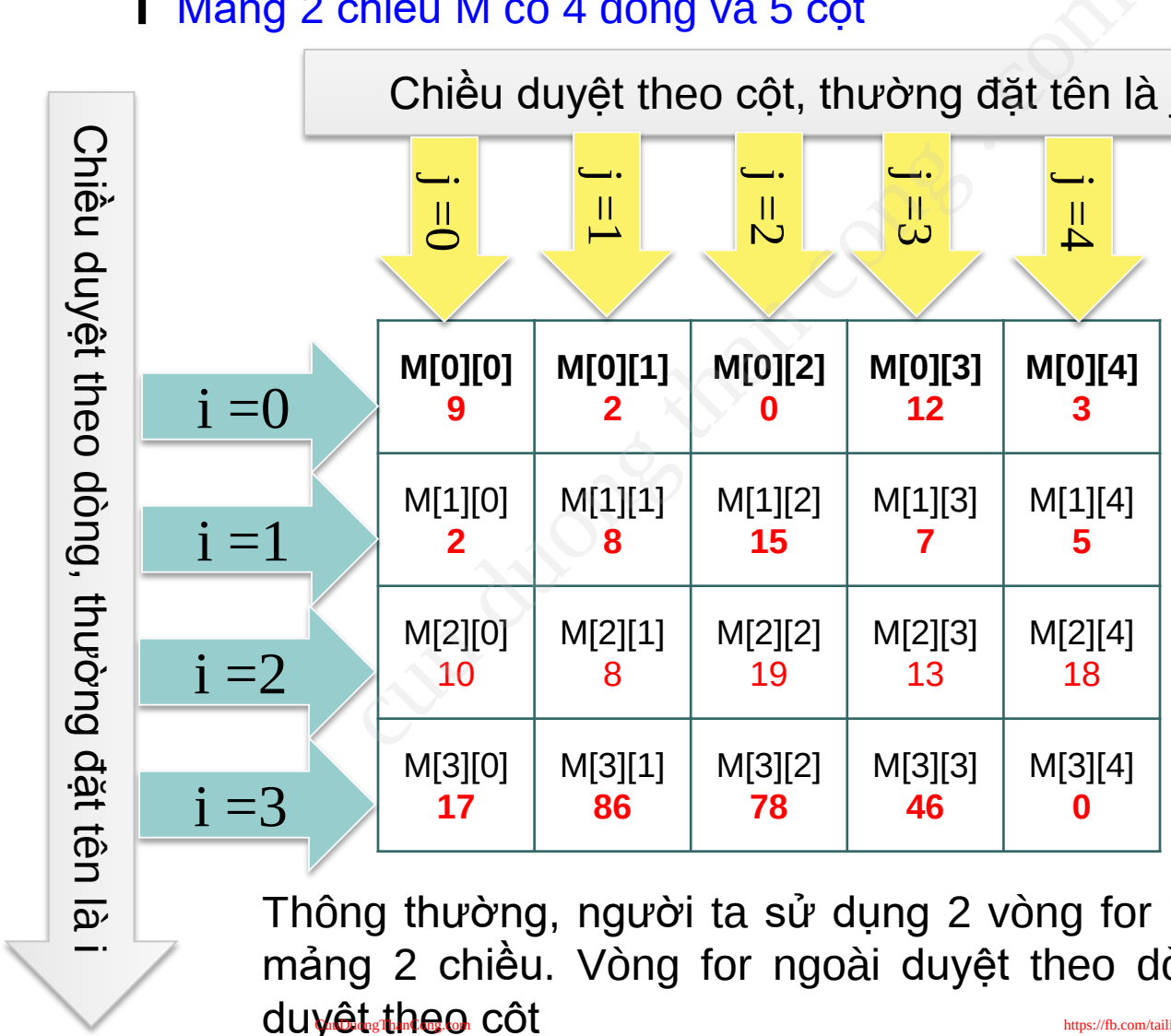
Ví dụ: **float K[3][4];**

- Khai báo 1 mảng 2 chiều tên là K có kiểu số thực, số dòng là 3, số cột là 4 ➔tổng số phần tử là 4*3=12;

❖ Tương tự như mảng 1 chiều, chỉ số của các phần tử được đánh dấu từ số 0 cho tới số dòng -1 hoặc số cột -1

Mảng nhiều chiều

Mảng 2 chiều M có 4 dòng và 5 cột



Tức là để truy suất 1 phần tử tại dòng thứ ivà tại cột thứ j thì ta viết:
M[i][j]
- Ví dụ, muốn lấy phần tử tại dòng 2, cột 3 thì ta viết M[2][3]

Thông thường, người ta sử dụng 2 vòng for lồng nhau để duyệt mảng 2 chiều. Vòng for ngoài duyệt theo dòng, vòng for trong duyệt theo cột

Mảng nhiều chiều

int M[dòng][cột];

Ví dụ:

int M[3][2];

- Dòng =3, cột =2 ➔ tổng số phần tử = 2*3 =6 phần tử
- Để duyệt mảng, ta kết hợp 2 vòng for lồng nhau, vòng for ngoài chạy theo chỉ số dòng (giả sử đặt là i), vòng for trong chạy theo chỉ số cột (giả sử đặt là j).

Mảng nhiều chiều

```
#include<iostream.h>
#define MAX_ROW 100
#define MAX_COL 100
void nhapmang(int M[MAX_ROW][MAX_COL],int d,int c);
void xuatmang(int M[MAX_ROW][MAX_COL],int d,int c);
void main()
{
    int M[MAX_ROW][MAX_COL];
    int d,c;
    cout<<"Nhap vao so dong :";
    cin>>d;
    cout<<"Nhap vao so cot :";
    cin>>c;
    if(d>MAX_ROW || c>MAX_COL)
    {
        cout<<"Vuot qua gioi han";
        exit(0);
    }
    cout<<"Nhap gia tri cho tung phan tu : \n";
    nhapmang(M,d,c);
    cout<<"Mang sau khi nhap:\n";
    xuatmang(M,d,c);
    cout<<"\n";
}
```

Mảng nhiều chiều

```
void nhapmang(int M[MAX_ROW][MAX_COL],int d,int c)
{
    for(int i=0;i<d;i++)
    {
        for(int j=0;j<c;j++)
        {
            cout<<"Nhap M["<<i<<"]["<<j<<"]=";
            cin>>M[i][j];
        }
    }
}
void xuatmang(int M[MAX_ROW][MAX_COL],int d,int c)
{
    for(int i=0;i<d;i++)
    {
        for(int j=0;j<c;j++)
        {
            cout<<M[i][j]<<"\t";
        }
        cout<<"\n";
    }
}
```

Mảng nhiều chiều

```
#include<iostream.h>
#include<stdlib.h>
#include<time.h>
#define Max_col 100
#define Max_row 100
void Nhapmatran(int M[Max_row][Max_col],int m,int n);
void Xuatmatran(int M[Max_row][Max_col],int m,int n);
void main()
{
     int M[Max_row][Max_col];
     int m,n;
     cout<<"Nhap vao so cot cua mang: ";
     cin>>m;
     cout<<"\nNhap vao so hang cua mang: ";
     cin>>n;
     Nhapmatran(M,m,n);
     cout<<"\n Mang vua nhap: \n";
     Xuatmatran(M,m,n);
}
```

Mảng nhiều chiều

```
void Nhapmatran(int M[Max_row][Max_col],int m,int n)
{
     srand(time(NULL));
     for(int i=0;i<=m;i++)
     {
          for(int j=0;j<=n;j++)
          {
               M[i][j]=rand()%100;
          }
     }
}
void Xuatmatran(int M[Max_row][Max_col],int m,int n)
{
     for(int i=0;i<=m;i++)
     {
          for(int j=0;j<=n;j++)
          {
               cout<<M[i][j]<<"\t";
          }
          cout<<"\n";
     }
}
```