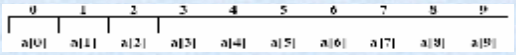
CHƯƠNG IV:MẢNG

KHÁI NIỆM

- Mảng là một tập hợp các phần tử cố định có cùng một kiểu được đặt liên tiếp trong bộ nhớ gọi là kiểu phần tử.
- Kiểu phần tử có thể là: số, ký tự….
- Mỗi phần tử được xác định bởi một **chỉ số** biểu thị vị trí của phần tử trong mảng.



KHÁI NIỆM

- Nếu mảng có **n** phần tử thì chỉ số của các phần tử có giá trị từ **0** đến **n-1**.
- Số phần tử 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.

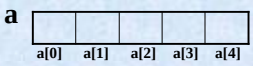
MẢNG 1 CHIỀU

Khai báo mảng

- Khai báo mảng với số phần tử tường minh

<Kiểu> <Tên mảng ><[số phần tử]>

- Ví dụ: int **a[5]**



Khai báo mảng

- Khai báo mảng với số phần tử không xác định (khai báo không tường minh)

<Kiểu> <Tên mảng> <[ ]>

Ví dụ: int **a[]**

- Kiểu khai báo này thường được áp dụng trong các trường hợp:
  - Vừa khai báo vừa gán giá trị
  - Khai báo mảng là tham số hình thức của hàm.

Vừa khai báo vừa gán giá trị

<Kiểu> <Tên mảng> [ ]= {gtri1, gtri2,…}

Ví dụ:
char str[] = "HELLO";
int nums[] = {5, 10, 15};

Vừa khai báo vừa gán giá trị

- Nếu vừa khai báo vừa gán giá trị thì số phần tử của mảng là số giá trị mà ta gán cho mảng trong cặp dấu {}.
- Sử dụng hàm **sizeof()** để lấy số phần tử của mảng như sau:

Số phần tử=***sizeof(tên mảng)/ sizeof(kiểu)***

Khai báo mảng là tham số hình thức của hàm

- Trong một số trường hợp ta cần phải truyền một mảng tới một hàm như là một tham số
- Trong C++, việc truyền theo tham số giá trị một khối nhớ là không hợp lệ
- Để có thể nhận mảng là tham số thì khai báo hàm theo cú pháp:

<kiểu dữ liệu> ***<tên mảng>*** []

Ví dụ: void procedure(int arg[ ])

Ví dụ

```
#include <iostream.h>
void printarray (int arg[], int length) {
   for (int n=0; n<length; n++)
        cout <<setw(3)<< arg[n] ;
}
int main (){
   int x[] = {5, 10, 15};
   int y[] = {2, 4, 6, 8, 10};
   printarray (x,3);
   printarray (y,5);
   return 0;
}
```


Truy xuất từng phần tử của mảng

- ***Cú pháp*:**

<array_Name>[<index1>][<index2>][…][<indexN>]

- **Ví dụ:**

M[0], M[2], M[1][5],…

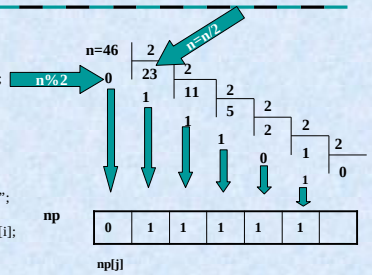
- Chỉ số của phần tử mảng là một biểu thức có giá trị là kiểu số nguyên.
- Với cách truy xuất này thì *có thể coi như là một biến* có kiểu dữ liệu là **kiểu** được chỉ ra trong khai báo biến mảng.

Ví dụ:

```
#include <iostream.h>
#include <conio.h>
int main()
{
   int n,i,j,tam;
   int dayso[ ]={66, 65, 69, 68, 67, 70};
   clrscr();
   n=sizeof(dayso); //Lấy số phần tử
   cout<< " Noi dung cua mang ";
   for (i=0;i<n;i++)
      cout<<setw(3)<<dayso[i];
   return 0;
}
```

Ví dụ: Đổi một số nguyên dương thập phân thành số nhị phân

```
void main()
{
    int i,j=0,n,np[20];
    cout<<"n=";cin>>n;
    do
    {
        np[j]=n%2;
        j++;
        n=n/2;
    }while(n>0);
cout<<"dang nhi phan: ";
for(i=j-1 ; i>0 ; i--)
    cout<<setw(3)<<np[i];
getch();
}
```



Ví dụ : Nhập vào một mảng số nguyên sau đó sắp xếp theo thứ tự tăng dần

```
#include <iostream.h>
#define n 5
main ( )
{
   int a [ n ] ; int i , j, t ;
   for ( i = 0 ; i < n ; i ++)//nhập mảng
   {
        cout<<"a ["<<i<<"] = ";cin>>a[i];  cout<<endl;
   }
   for ( i = 0 ; i < n - 1 ; i ++)//sắp xếp
       for ( j = i + 1 ; j < n ; j ++ )
          if ( a [ i ] < a [ j ] )
            {
              t = a [ i ] ; a [ i ] = a [ j ]; a [j ] = t ;
            }
   for ( i = 0 ; i < n ; i ++ )//xuất mảng
        cout<<setw(3)<<a[i];
   getch ( )
}
```


Sử dụng hàm tạo số ngẫu nhiên

- C++ cung cấp hàm random để tạo ra các số ngẫu nhiên.
- Cú pháp:

int random(int n)

- Kết quả của hàm là các số nguyên từ 0 đến n-1
- Khi sử dụng random ta phải gọi ***randomize*** để khởi tạo chế độ tạo số ngẫu nhiên.
- Muốn sử dụng các hàm trên thì trong chương trình ta phải khai báo

#include <stdlib.h>

Ví dụ : Chương trình sau tạo giá trị ngẫu nhiên cho một mảng n phần tử, thực hiện sắp xếp và in mảng lên màn hình trước và sau khi sắp xếp.

```
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
#include <stdlib.h>
void taomang(int a[], int n)
{
   randomize;
   cout <<"Tao mang ngẫu nhiên :\n";
   for (int i=0; i<n; i++)
        a[i]=random(100);
}
```


Ví dụ : Chương trình sau tạo giá trị ngẫu nhiên cho một mảng n phần tử, thực hiện sắp xếp và in mảng lên màn hình trước và sau khi sắp xếp

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


Ví dụ : Chương trình sau tạo giá trị ngẫu nhiên cho một mảng n phần tử, thực hiện sắp xếp và in mảng lên màn hình trước và sau khi sắp xếp

```
void inmang(int a[], int n)
{
   for(int i=0; i<n; i++)
        cout <<setw(3)<<a[i];
}
```

Ví dụ : Chương trình sau tạo giá trị ngẫu nhiên cho một mảng n phần tử, thực hiện sắp xếp và in mảng lên màn hình trước và sau khi sắp xếp(viết dưới dạng hàm)

```
void main()
{
  int n, a[];
  clrscr();
  cout<<“Nhap so phan tu:”;cin>>n;
  taomang(a, n);
  cout<<“Mang truoc khi sap xep:\n”;
  inmang(a, n);
  sapxepmang(a,n)
   cout<<“Mang sau khi sap xep:\n”;
  inmang(a, n);
  getch();
  return;
}
```


MẢNG NHIỀU CHIỀU

Khai báo mảng nhiều chiều

- Mảng nhiều chiều có thể được coi như mảng của mảng
- Một mảng hai chiều có thể được xem như là một bảng hai chiều gồm các phần tử có kiểu dữ liệu cụ thể và giống nhau.

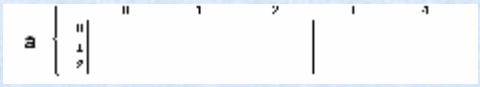
Khai báo mảng 2 chiều tường minh

- **Cú pháp**

<Kiểu> <Tên mảng><[Số dòng]><[Số cột]>

- **Ví dụ**:

int a[3][5];



Khai báo mảng 2 chiều không tường minh

- Để khai báo mảng 2 chiều không tường minh, ta vẫn phải chỉ ra số phần tử của chiều thứ hai (chiều cuối cùng).
- Cú pháp:

<Kiểu> <Tên mảng> <[ ]><[Số phần tử chiều 2]>

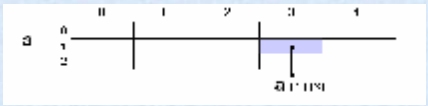
- Cách khai báo này cũng được áp dụng trong trường hợp vừa khai báo, vừa gán trị hay đặt mảng 2 chiều là tham số hình thức của hàm

Truy xuất từng phần tử của mảng 2 chiều

- Cú pháp:

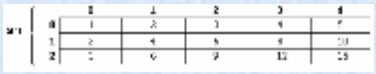
Tên mảng[Chỉ số 1][Chỉ số 2]

- Ví dụ: a[1][3]



Ví dụ :

```
#define cot 5
#define hang 3
int arr [hang][cot];
int n,m;
int main ()
{
    for (i=0 ; i<hang ; i++)
                for (j=0 ; j<cot ; j++)
                {
                        a[i][j]=(i+1)*(j+1);
                }
    return 0;
}
```



Khai báo mảng 2 chiều như một tham số

- Trong C++ không thể truyền toàn bộ một khối bộ nhớ bằng một giá trị như là một tham số đến một hàm, nhưng được phép truyền địa chỉ của nó
- Để chấp nhận những mảng như là những tham số của hàm thì khi khai báo hàm phải chỉ rõ trong các tham số của nó loại phần tử của mảng

<*kiểu dữ liệu*> <*tên mảng*> [ ][Số phần tử chiều 2]


```
#include<iostream.h>
#include<conio.h>
void Nhap(int a[][10],int M,int N)
{
for(int i=0 ; i<M ; i++)
    for(int j=0 ; j<N ; j++)
    {
        cout<<“a[ ”<<i<<”][ “<<j<<“]=”;
        cin>>a[i][j];
    }
}
void InMaTran(int a[ ][10], int M, int N)
{
    for(int i=0;i<M;i++)
    {
        for(int j=0; j< N; j++)
        cout<<setw(3)<<a[i][j];
        cout<<"\n";
    }
```

Các phương pháp tìm kiếm và sắp xếp trên mảng

Tìm kiếm

***Tìm kiếm trên mảng một chiều*:**

- Xây dựng hàm

  int search(int a[ ],int n, int x)

  Tìm một giá trị x trên mảng a có n phần tử. Kết quả của hàm trả về vị trí tìm thấy, hàm trả về giá trị âm khi không tìm thấy.

Tìm kiếm

```
int search1(int a[ ],int n, int x)
{
      for (int i=0; i<n; i++)
      if (a[i] == x )
            return (i);
      return (-1);
 }
```

Tìm kiếm

***Tìm kiếm trên mảng hai chiều*:**

- Xây dựng hàm

  int search2(int a[ ][COLS],int n, int x)
- Tìm một giá trị x trên mảng a có n dòng và COLS cột.
- Kết quả của hàm trả về vị trí tìm thấy, hàm trả về giá trị âm khi không tìm thấy.

Tìm kiếm

```
int search2 (int m[ ][ ], int n, int m, int x, int &r, int &c)
{
   int i=0, j=0;
   r = -1; c = -1;
   for (; i<n; i++)
        for (; j<m; j++)
            if (m[i][j] == x )
             {
                  c= i;
                  r = j;
                  return (1);
             }
            return (-1);
   }
```


Sắp xếp

***Sắp xếp trên mảng một chiều*:**

Viết hàm

sort(a[ ],n)

sắp xếp một mảng n phần tử theo thứ tự tăng.

***Sắp xếp trên mảng một chiều*:**

Nguyên tắc sắp xếp mảng hai chiều là đưa toàn bộ giá trị của mảng hai chiều ra một mảng một chiều, tiếp theo sắp xếp trên mảng một chiều này và cuối cùng là đưa các giá trị từ mảng một chiều vào mảng hai chiều. Viết hàm ***sort2(a[][COLS],n)*** để sắp xếp một mảng n dòng, COLS cột phần tử theo thứ tự tăng.