BÀI 12: KIỂU MẢNG

1. Mảng một chiều

a. Khai báo

```
<kiểu> <tên mảng> [<số phần tử>];
```

Trong đó:

+ `<kiểu>`: là tên kiểu dữ liệu của các phần tử trong mảng;

+ `<tên mảng>`: do người lập trình tự đặt (phù hợp quy tắc đặt tên);

+ `<số phần tử>`: là hằng hay biểu thức nguyên chỉ số lượng các phần tử tối đa.

b. Truy xuất phần tử của mảng

```
<tên mảng>[chỉ số]
```

Trong đó chỉ số có thể là hằng, biến, hàm, biểu thức nguyên.

Chú ý: Chỉ số của mảng trong C/C++ được đánh số bắt đầu từ 0 nên một mảng có N phần tử được đánh số từ 0 đến N-1.

c. Khởi gán giá trị các phần tử cho mảng

Trong khai báo có thể khởi gán các giá trị ban đầu cho các phần tử của mảng.

VD:
```
int  a[4]={1,2,3,4};
int  b[5]={6,7,8};
```

Những phần tử không được gán gián trị thì giá trị mặc định của nó bằng 0.

Nếu không khai báo số phần tử của mảng thì C sẽ lấy số lượng các giá trị khởi gán và kiểu của mảng để cấp phát bộ nhớ cho mảng.

Vd:
```
char c[] = {'a','b', 'c', 'd'};
```

Sẽ tạo mảng kiểu char có 4 phần tử.

d. Truyền giá trị từ một mảng cho một mảng

Trong C không cho phép gán một mảng cho một mảng. Để thực hiện điều đó ta thực hiện một trong hai cách sai:

+ Gán từng phần tử một.

+ Dùng hàm *memmove*.

```
void memmove(void * dest, char * scr, n)
```

Hàm sẽ sao chép một khối n byte từ scr sang dest.

Để sử dụng hàm memmove cần khai báo thư viện ***#include <mem.h>***

VD:

```
#include <conio.h>
#include <mem.h>
int main(){
  int a[]={1,2,3,4,5,6};
  int b[6];
  memmove(b,a,sizeof(a));
  for (int i=0; i<6; i++)
       printf("b[%d]=%d",i,b[i]);
  return 0;
}
```

2. Mảng nhiều chiều

Để khai báo mảng nhiều chiều trong C ta sử dụng mỗi cặp dấu [] cho một chiều.

Ví dụ:

```
int  x[3][5];
```

Mảng x có thể xem là mảng một chiều gồm 3 phần tử, mỗi phần tử của mảng lại là mảng một chiều có 5 phần tử. Cũng có thể xem x như một mảng 2 chiều có 3 dòng và 5 cột.

Để truy cập đến phần tử mảng nhiều chiều ta cũng sử dụng các cặp dấu [] cho mỗi chiều

VD: `x[i][j]`

3. Một số ví dụ

Vd 1: Tính trung bình cộng của phần tử trong mảng có N phần tử số nguyên được nhập từ bàn phím.

```
#include <stdio.h>
int A[1001],n;
float s=0;

int main(){
```

```
  printf("Nhap so phan tu cua mang: ");
  scanf("%d",&n);
  for (int i=0;i<n;i++){
       printf("A[%d]=",i);
       scanf("%d",&A[i]);
  }
  for (int i=0;i<n;i++) s+=A[i];
  printf("Trung binh cong cua mang la: %0.5f",s/n);
  return 0;
}
```

Vd 2: sử dụng mảng một chiều để tính số fibonacy thứ n ($0<n\leq1000$).

Với $f_0=1$, $f_1=1$, $f_i=f_{i-1}+f_{i-2}$ với $i\geq2$

```
#include <stdio.h>
int F[1001],n;
int main(){
  printf("Nhap so phan tu cua mang: ");
  scanf("%d",&n);
  F[0]=1;
  F[1]=1;
  for (int i=2;i<=n;i++) F[i]=F[i-1]+F[i-2];
  printf("So Fibonacy thu %d la: %d",n,F[n]);
  return 0;
}
```

Vd 3: viết chương trình sắp xếp mảng một chiều có n phần tử số nguyên bằng giải thuật sắp xếp đơn giản.

```
#include <stdio.h>
#include <algorithm>
using namespace std;
int A[1001],n;
float s=0;
void sapxep(){
  for (int i=0;i<n-1;i++){
       for (int j=i+1;j<n;j++){
            if (A[i]>A[j]) swap(A[i],A[j]);
       }
  }
}

int main(){
  printf("Nhap so phan tu cua mang: ");
  scanf("%d",&n);
  for (int i=0;i<n;i++){
```

```
        printf("A[%d]=",i);
        scanf("%d",&A[i]);
    }
    sapxep();
    printf("\nMang sau khi sap xep tang dan\n");
    for (int i=0;i<n;i++) printf("%d ",A[i]);
    return 0;
}
```

Vd 4: Viết chương trình nhập mảng hai chiều số nguyên có m hàng, n cột và tìm số lớn nhất trên mỗi hàng.

```
#include <stdio.h>
int A[1001][1001],m,n;
int MaxHang(int hang){
    int maxh = A[hang][0];
    for (int j=1;j<n;j++){
        if (A[hang][j]>maxh) maxh = A[hang][j];
    }
    return maxh;
}

int main(){
    printf("Nhap so hang va so cot cua mang: ");
    scanf("%d%d",&m,&n);
    for (int i=0;i<m;i++){
        for (int j=0;j<n;j++){
            printf("A[%d][%d]=",i,j);
            scanf("%d",&A[i][j]);
        }
    }
    for (int i=0;i<m;i++)
        printf("max tai hang %d la %d\n",i+1,MaxHang(i));
    return 0;
}
```

Vd 5: Viết chương trình nhập mảng hai chiều số nguyên có m hàng, n cột và tìm số nhỏ nhất trên mỗi cột.

```
#include <stdio.h>
int A[1001][1001],m,n;
int MaxCot(int cot){
    int maxc = A[0][cot];
    for (int i=1;i<m;i++){
        if (A[i][cot]>maxc) maxc = A[i][cot];
    }
```

```
  return maxc;
}

int main(){
  printf("Nhap so hang va so cot cua mang: ");
  scanf("%d%d",&m,&n);
  for (int i=0;i<m;i++){
       for (int j=0;j<n;j++){
            printf("A[%d][%d]=",i,j);
            scanf("%d",&A[i][j]);
       }
  }
  for (int j=0;j<n;j++)
       printf("Max o cot %d la %d\n",j+1,MaxCot(j));
  return 0;
}
```