CƠ SỞ LẬP TRÌNH

KIỂU DỮ LIỆU MẢNG

Nội dung

- Mảng một chiều
  - Khái niệm
  - Khai báo
  - Thao tác trên mảng
  - Hàm có mảng một chiều là tham số
- Mảng nhiều chiều
  - Khái niệm
  - Khai báo
  - Thao tác trên mảng
  - Hàm có mảng nhiều chiều là tham số

Đặt vấn đề

- Ví dụ
  - Chương trình cần lưu trữ 3 số nguyên?
    => Khai báo 3 biến int a1, a2, a3;
  - Chương trình cần lưu trữ 100 số nguyên?
    => Khai báo 100 biến kiểu số nguyên!
  - Người dùng muốn nhập n số nguyên?
    => Không thực hiện được!
- Giải pháp
  - Kiểu dữ liệu mới cho phép lưu trữ một dãy các số nguyên và dễ dàng truy xuất.

Mảng một chiều

- Khái niệm
- Khai báo
- Thao tác trên mảng
- Hàm có mảng một chiều là tham số

5.1.1 Khái niệm

Khái niệm

- Là một kiểu dữ liệu có cấu trúc do người lập trình định nghĩa.
- Biểu diễn một dãy các biến có cùng kiểu. Ví dụ: dãy các số nguyên, dãy các ký tự…
- Kích thước được xác định ngay khi khai báo và không bao giờ thay đổi.
- C luôn chỉ định một khối nhớ liên tục cho một biến kiểu mảng.

5.1.2 Khai báo

Khai báo tường minh

```
<kiểu cơ sở> <tên biến mảng>[<số phần tử>];
<kiểu cơ sở> <tên biến mảng>[<N1>][<N2>]…[<Nn>];
```

- <N1>, …, <Nn> : số lượng phần tử của mỗi chiều.

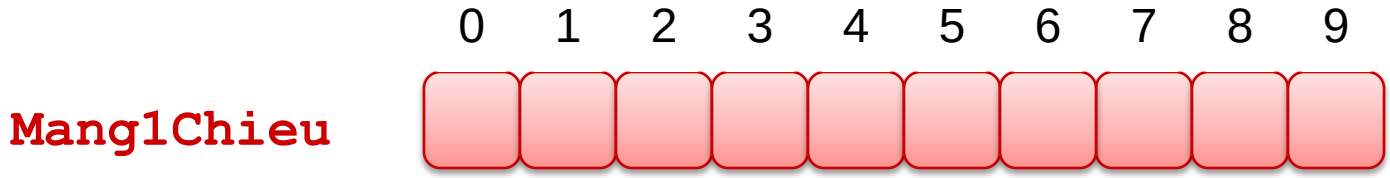
Lưu ý

- Phải xác định <số phần tử> cụ thể (hằng) khi khai báo.
- Mảng nhiều chiều: <tổng số phần tử> = N1*N2*…*Nn
- Bộ nhớ sử dụng = <tổng số phần tử>*sizeof(<kiểu cơ sở>)
- Một dãy liên tục có chỉ số từ 0 đến <tổng số phần tử>-1

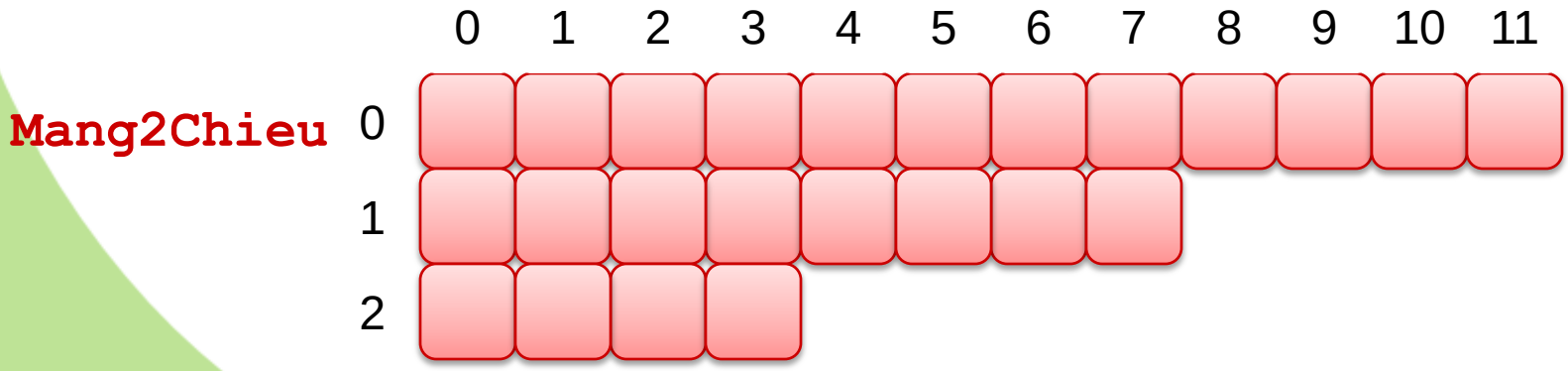
Khai báo tường minh (tt)

Ví dụ

```
int Mang1Chieu[10];
```




```
int Mang2Chieu[3][4];
```



Khai báo không tường minh

Cú pháp

- Không tường minh (thông qua khai báo kiểu)

```
typedef <kiểu cơ sở> <tên kiểu mảng>[<số phần tử>];
typedef <kiểu cơ sở> <tên kiểu mảng>[<N1>]…[<Nn>];
<tên kiểu mảng> <tên biến mảng>;
```

Ví dụ

```
typedef int Mang1Chieu[10];
typedef int Mang2Chieu[3][4];

Mang1Chieu m1, m2, m3;
Mang2Chieu m4, m5;
```

Số phần tử của mảng

- Phải xác định cụ thể số phần tử ngay lúc khai báo, không được sử dụng biến hoặc hằng thường

```
int n1 = 10; int a[n1];
const int n2 = 20; int b[n2];
```

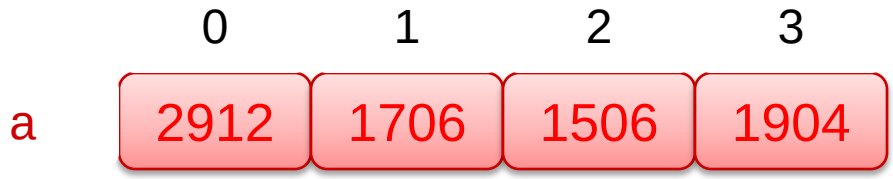
- Nên sử dụng chỉ thị tiền xử lý #define để định nghĩa số phần tử mảng

```
#define n1 10
#define n2 20
int a[n1];          // ⇔ int a[10];
int b[n1][n2];      // ⇔ int b[10][20];
```

Khởi tạo giá trị cho mảng lúc khai báo

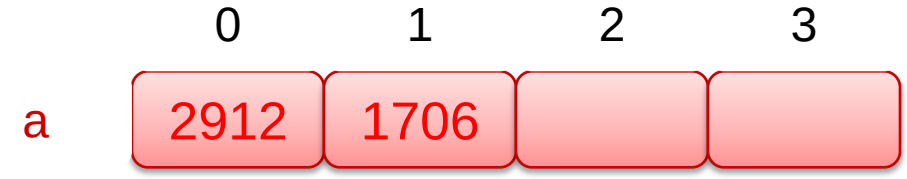
- Gồm các cách sau
  - Khởi tạo giá trị cho mọi phần tử của mảng

```
int a[4] = {2912, 1706, 1506, 1904};
```



  - Khởi tạo giá trị cho một số phần tử đầu mảng

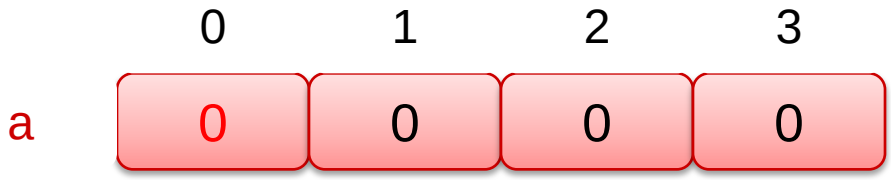
```
int a[4] = {2912, 1706};
```



Khởi tạo giá trị cho mảng lúc khai báo

- Gồm các cách sau
  - Khởi tạo giá trị 0 cho mọi phần tử của mảng

```
int a[4] = {0};
```



  - Tự động xác định số lượng phần tử

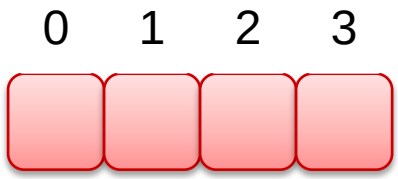
```
int a[] = {2912, 1706, 1506, 1904};
```

5.1.3 Truy xuất đến một phần tử

- Thông qua chỉ số

```
<tên biến mảng>[<chỉ số>]
```

- Ví dụ
  - Cho mảng như sau




```
int a[4];
```

  - Các truy xuất
    - Hợp lệ: a[0], a[1], a[2], a[3]
    - Không hợp lệ: a[-1], a[4], a[5], …

      => Cho kết thường không như mong muốn!

Gán dữ liệu kiểu mảng

- Không được sử dụng phép gán thông thường mà phải gán trực tiếp giữa các phần tử tương ứng

```
<biến mảng đích> = <biến mảng nguồn>; //sai
<biến mảng đích>[<chỉ số thứ i>] = <giá trị>;
```

- Ví dụ

```
#define MAX 3
typedef int MangSo[MAX];
MangSo a = {1, 2, 3}, b;

b = a;        // Sai
for (int i = 0; i < 3; i++) b[i] = a[i];
```

Một số lỗi thường gặp

- Khai báo không chỉ rõ số lượng phần tử
  - int a[]; → int a[100];
- Số lượng phần tử liên quan đến biến hoặc hằng
  - int n1 = 10; int a[n1];
- Khởi tạo cách biệt với khai báo
  - int a[4]; a = {2912, 1706, 1506, 1904};
    → int a[4] = {2912, 1706, 1506, 1904};
- Chỉ số mảng không hợp lệ
  - int a[4];
  - a[-1] = 1; a[10] = 0;

5.1.4 Một số bài toán trên mảng 1 chiều

- Nhập mảng
- Xuất mảng
- Tìm kiếm một phần tử trong mảng
- *Tìm giá trị nhỏ nhất/lớn nhất của mảng*
- *Sắp xếp mảng giảm dần/tăng dần*
- *Thêm/Xóa/Sửa một phần tử vào mảng*

Nhập mảng

- Đoạn chương trình nhập vào từ bàn phím một mảng 1 chiều a gồm n phần tử

```
#define MAXN 100
int a[MAXN];
...
printf("Nhap so luong phan tu n: ");
scanf("%d", &n);
for (int i = 0; i < n; i++)
      {
            printf("Nhap phan tu thu %d: ", i);
            scanf("%d", &a[i]);
      }
```

Xuất mảng

- Đoạn chương trình in ra màn hình một mảng 1 chiều a gồm n phần tử

```
printf("Noi dung cua mang la: ");
for (int i = 0; i < n; i++)
      printf("%d ", a[i]);
printf("\n");  /*Đưa con trỏ màn hình xuống dòng
                  tiếp theo */
```

Hàm có mảng là tham số

- Khai báo
  - Tham số kiểu mảng trong khai báo hàm giống như khai báo biến mảng

```
void NhapMang(int a[]);
```

  - Tham số kiểu mảng truyền cho hàm chính là địa chỉ của phần tử đầu tiên của mảng
    - Có thể bỏ số lượng phần tử hoặc sử dụng con trỏ.
    - Mảng có thể thay đổi nội dung sau khi thực hiện hàm.

```
void NhapMang(int a[]);
void NhapMang(int *a);
```

  - Số lượng phần tử thực sự truyền qua biến khác

```
void NhapMang(int a[100], int n);
void NhapMang(int a[], int n);
void NhapMang(int *a, int n);
```

Hàm nhập mảng

```
void NhapMang(int a[], int &n)
{
        printf("Nhap so luong phan tu n: ");
        scanf("%d", &n);

        for (int i = 0; i < n; i++)
        {
                printf("Nhap phan tu thu %d: ", i);
                scanf("%d", &a[i]);
        }
}
```

Hàm xuất mảng

```
void XuatMang(int a[], int n)
{
        printf("Noi dung cua mang la: ");

        for (int i = 0; i < n; i++)
              printf("%d ", a[i]);

        printf("\n");
}
```

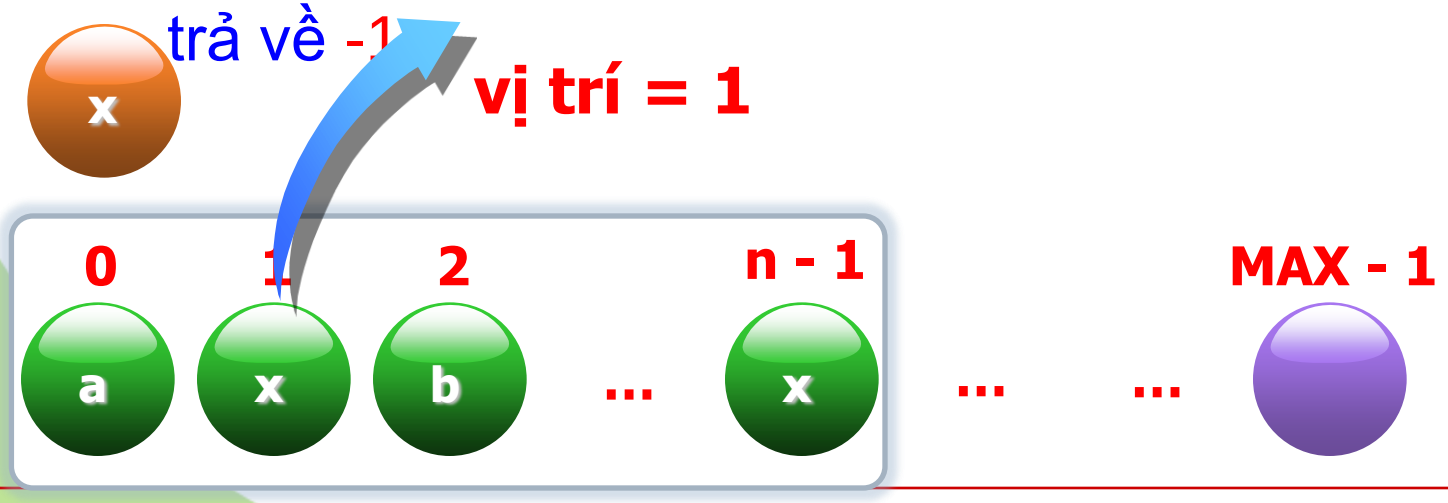
Truyền tham số mảng cho hàm (sử dụng)

- Lời gọi hàm

```
void NhapMang(int a[], int &n);
void XuatMang(int a[], int n);
int main()
{
        int a[100], n;
        NhapMang(a, n);
        XuatMang(a, n);
}
```

Tìm kiếm một phần tử trong mảng

- Yêu cầu
  - Tìm xem phần tử x có nằm trong mảng a kích thước n hay không? Nếu có thì nó nằm ở vị trí đầu tiên nào.
- Ý tưởng (tìm kiếm tuần tự)
  - Xét từng phần của mảng a. Nếu phần tử đang xét bằng x thì trả về vị trí đó. Nếu không tìm được thì trả về -1



Hàm tìm kiếm

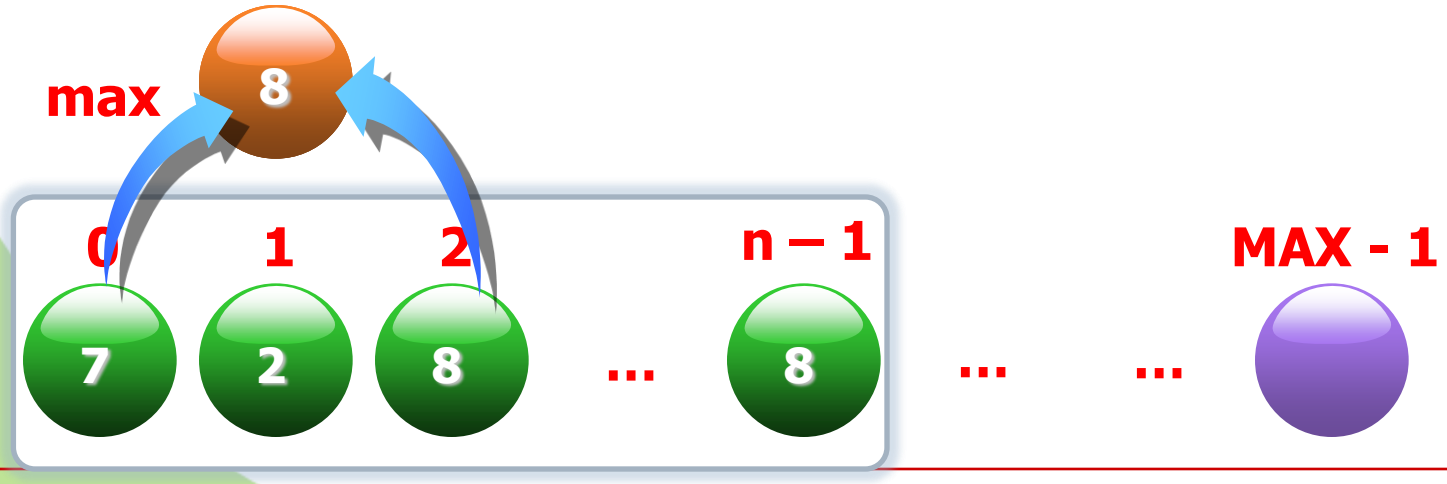
```
int TimKiem(int a[], int n, int x)
{
        for (int vt = 0; vt < n; vt++)
                if (a[vt] == x)
                        return vt;

        return -1;
}
```

- **Bài tập**:
  - Viết lại hàm tìm kiếm sử dụng câu lệnh while

Tìm giá trị lớn nhất của mảng

- Yêu cầu
  - Cho trước mảng a có n phần tử. Tìm giá trị lớn nhất trong a
- Ý tưởng
  - Giả sử giá trị max hiện tại là giá trị phần tử đầu tiên a[0]
  - Với mọi i từ 1 đến n-1, nếu a[i] > max thì max = a[i]



Hàm tìm giá trị lớn nhất của mảng

```
int TimMax(int a[], int n)
{
        int max = a[0];

        for (int i = 1; i < n; i++)
              if (a[i] > max)
                    max = a[i];

        return max;
}
```

Thêm một phần tử vào mảng

- Yêu cầu
  - Thêm phần tử x vào mảng a kích thước n tại vị trí vt.
- Ý tưởng
  - "Đẩy" các phần tử bắt đầu tại vị trí vt sang phải 1 vị trí.
  - Đưa x vào vị trí vt trong mảng.
  - Tăng n lên 1 đơn vị.

Hàm thêm một phần tử vào mảng

```
void Them(int a[], int &n, int vt, int x)
{
        if (vt >= 0 && vt <= n)
        {
                for (int i = n; i > vt; i--)
                        a[i] = a[i - 1];

                a[vt] = x;
                n++;
        }
}
```

Xóa một phần tử trong mảng

- Yêu cầu
  - Xóa một phần tử trong mảng a kích thước n tại vị trí vt
- Ý tưởng
  - "Kéo" các phần tử bên phải vị trí vt sang trái 1 vị trí.
  - Giảm n xuống 1 đơn vị.

Hàm xóa một phần tử trong mảng

```
void Xoa(int a[], int &n, int vt)
{
        if (vt >= 0 && vt < n)
        {
                for (int i = vt; i < n - 1; i++)
                        a[i] = a[i + 1];

                n--;
        }
}
```

Bài tập thực hành

1. Các thao tác nhập xuất
   a. Nhập mảng gồm n số nguyên từ bàn phím
   b. Xuất mảng vừa nhập ra màn hình
2. Các thao tác kiểm tra

Nhập vào một mảng gồm n số nguyên

a. Mảng vừa nhập có phải là mảng toàn số chẵn (số lẻ) hay không?
b. Mảng có phải là mảng toàn số nguyên tố? Số chính phương?
c. Mảng có phải là mảng tăng dần hay giảm dần hay không?

Bài tập thực hành

3. Các thao tác tính toán

Nhập vào mảng gồm n số nguyên, và 2 số m và k

a. Mảng có bao nhiêu số chia hết cho m nhưng không chia hết cho k?
b. Tổng các số nguyên tố, chính phương có trong mảng?

4. Các thao tác tìm kiếm

Nhập vào mảng gồm n số nguyên và số nguyên x

a. Tìm vị trí cuối cùng của phần tử x trong mảng
b. Tìm vị trí số nguyên tố đầu tiên trong mảng nếu có
c. Tìm số nhỏ nhất trong mảng
d. Tìm số dương nhỏ nhất, số âm lớn nhất

Bài tập thực hành

5. Các thao tác xử lý

Nhập mảng a gồm n số nguyên

a. Xây dựng mảng b gồm các số nguyên tố có trong mảng a
b. Xây dựng mảng b (chứa các số nguyên dương) và mảng c (chứa các số còn lại)
c. Sắp xếp mảng theo chiều giảm dần
d. Sắp xếp mảng sao cho các số dương đứng đầu mảng giảm dần, kế đến là các số âm tăng dần, cuối cùng là các số 0.

Bài tập thực hành

6. Các thao tác thêm/xóa/sửa
   a. Sửa các số nguyên tố có trong mảng thành số 0, tất cả các số chính phương thành số -1
   b. Chèn số 0 đằng sau mỗi số nguyên tố trong mảng, chèn số -1 phía trước mỗi số chính phương trong mảng.
   c. Xóa tất cả số nguyên tố trong mảng

7* . Nhập vào từ bàn phím một số nguyên dương n (n<10). Viết chương trình con in ra tất cả các hoán vị của dãy số 1 2 3…n

Ví dụ: n = 3 thì in ra:

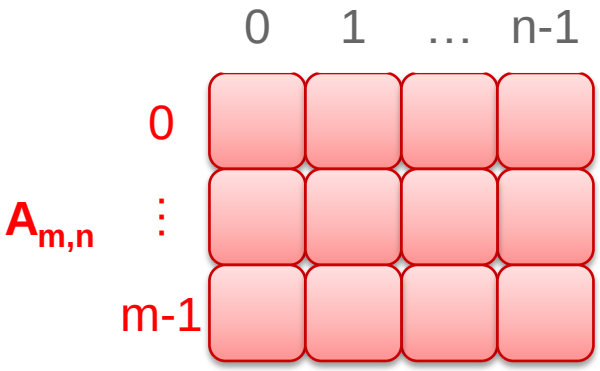
```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

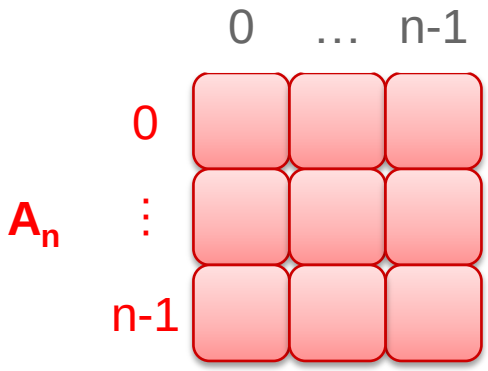
Mảng nhiều chiều

- Khái niệm
- Khai báo
- Thao tác trên mảng
- Hàm có mảng nhiều chiều là tham số

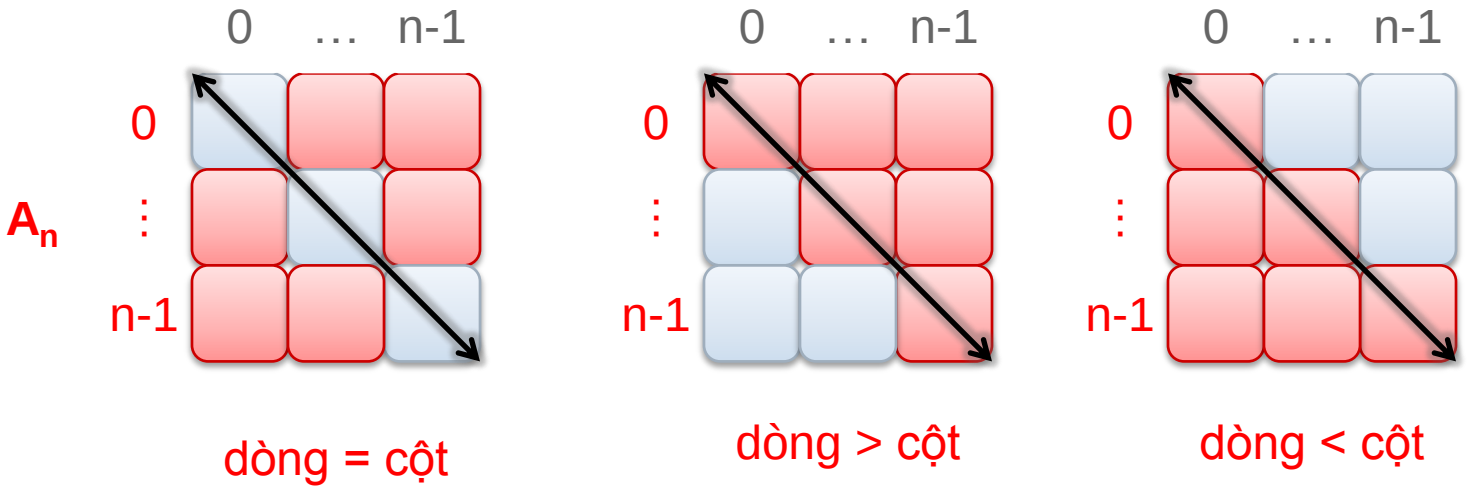
5.2.1 Khái niệm

- Ma trận





Ma trận vuông



5.2.2 Khai báo

Khai báo mảng 2 chiều

- Tường minh

```
<kiểu cơ sở> <tên biến>[<N1>][<N2>];
```

- Không tường minh (thông qua kiểu)

```
typedef <kiểu cơ sở> <tên kiểu>[<N1>][<N2>];

<tên kiểu> <tên biến>;
<tên kiểu> <tên biến 1>, <tên biến 2>;
```

  - N1, N2: số lượng phần tử mỗi chiều

Chú ý:

Ví dụ khai báo mảng 3 chiều: `int m[3][5][8]`

Khai báo mảng 2 chiều (tt)

- Ví dụ
  - Tường minh

```
int a[10][20], b[10][20];
int c[5][10];
int d[10][20];
```

  - Không tường minh (thông qua kiểu)

```
typedef int MaTran10x20[10][20];
typedef int MaTran5x10[5][10];

MaTran10x20 a, b;
MaTran11x11 c;
MaTran10x20 d;
```

Khởi tạo giá trị cho mảng nhiều chiều

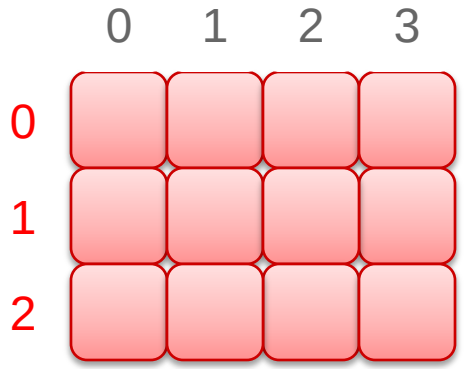
- Tương tự như khởi tạo giá trị ban đầu cho mảng 1 chiều
- Ví dụ:
  - int x[3][2]={{1,2},{3,4},{5,6}};
  - int y[3][2]={1,2,3,4,5,6};
  - int z[5][4]={0}

5.2.3 Truy xuất đến một phần tử

- Thông qua chỉ số

```
<tên biến mảng>[<chỉ số 1>][<chỉ số 2>]
```

- Ví dụ
  - Cho mảng 2 chiều như sau:




```
int a[3][4];
```

  - Các truy xuất
    - Hợp lệ: a[0][0], a[0][1], …, a[2][2], a[2][3]
    - Không hợp lệ: a[-1][0], a[2][4], a[3][3]

Gán giá trị cho mảng

- Không được sử dụng phép gán thông thường mà phải gán trực tiếp giữa các phần tử

```
<biến mảng đích> = <biến mảng nguồn>; //sai
<biến mảng đích>[<chỉ số 1>][chỉ số 2]=<giá trị>;
```

- Ví dụ

```
int a[5][10], b[5][10];

b = a;          // Sai
int i, j;
for (i = 0; i < 5; i++)
      for (j = 0; j < 10; j++)
            b[i][j] = a[i][j];
```

5.2.4 Một số bài toán cơ bản

- Nhập mảng
- Xuất mảng
- Tìm kiếm một phần tử trong mảng
- Kiểm tra tính chất của mảng
- Tìm giá trị nhỏ nhất/lớn nhất của mảng
- …

Nhập mảng 2 chiều

- Yêu cầu
  - Nhập vào từ bàn phím một mảng a gồm m dòng, n cột
- Ý tưởng
  - Khai báo một mảng 2 chiều có dòng tối đa là MAXD, số cột tối đa là MAXC. (dùng #define để định nghĩa)
  - Nhập số dòng m và số cột n thực sự của mảng
  - Nhập từng phần tử từ [0][0] đến [m-1][n-1].

Nhập mảng 2 chiều

- Đoạn chương trình nhập vào từ bàn phím một mảng 2 chiều a gồm m dòng, n cột

```
#define MAXD 100
#define MAXC 100
int a[MAXD][MAXC];
...
int main()
{ printf("Nhap so dong, so cot cua ma tran: ");
  scanf("%d%d", &m, &n);
  int i, j;
  for (i=0; i<m; i++)
       for (j=0; j<n; j++)
              {printf("Nhap a[%d][%d]: ", i, j);
               scanf("%d", &a[i][j]); }
}
```

Xuất mảng 2 chiều

- Yêu cầu
  - In ra màn hình mảng a gồm m dòng, n cột
- Ý tưởng
  - In giá trị từng phần tử của mảng 2 chiều từ dòng có 0 đến dòng m-1;
  - Mỗi dòng giá trị của cột 0 đến cột n-1 trên dòng đó;
  - Kết thúc mỗi dòng chèn thêm dấu xuống dòng "\n"

Xuất mảng 2 chiều (tt)

- Đoạn chương trình in ra màn hình một mảng 2 chiều a gồm m dòng, n cột

```
int main()
{
        /*Các thao tác nhập, xử lý mảng...*/
        /*in mảng ra màn hình */
        int i, j;
        for (i=0; i<m; i++)
        {
              for (j=0; j<n; j++)
                    printf("%d  ", a[i][j]);
              printf("\n");
        }
}
```

Hàm có mảng là tham số

- Tham số kiểu mảng trong khai báo hàm giống như khai báo biến mảng

```
void NhapMaTran(int a[50][100]);
```

- Tham số kiểu mảng truyền cho hàm chính là địa chỉ của phần tử đầu tiên của mảng
  - Có thể bỏ số lượng phần tử chiều thứ 2 hoặc con trỏ.
  - Mảng có thể thay đổi nội dung sau khi thực hiện hàm.

```
void NhapMaTran(int a[][100]);
void NhapMaTran(int (*a)[100]);
```

- Số lượng phần tử thực sự truyền qua biến khác

```
void XuatMaTran(int a[50][100], int m, int n);
void XuatMaTran(int a[][100], int m, int n);
void XuatMaTran(int (*a)[100], int m, int n);
```

Hàm nhập mảng 2 chiều

```
void NhapMaTran(int a[][MAXC], int &m, int &n)
{
        printf("Nhap so dong, so cot cua ma tran: ");
        scanf("%d%d", &m, &n);

        int i, j;
        for (i=0; i<m; i++)
                for (j=0; j<n; j++)
                {
                        printf("Nhap a[%d][%d]: ", i, j);
                        scanf("%d", &a[i][j]);
                }
}
```

Hàm xuất mảng 2 chiều

```
void XuatMaTran(int a[][MAXC], int m, int n)
{
        int i, j;
        for (i=0; i<m; i++)
        {
                for (j=0; j<n; j++)
                        printf("%d ", a[i][j]);

                printf("\n");
        }
}
```

Truyền mảng cho hàm (sử dụng hàm)

- Lời gọi hàm

```
void NhapMaTran(int a[][100], int &m, int &n);
void XuatMaTran(int a[][100], int m, int n);
void main()
{
        int a[50][100], m, n;
        NhapMaTran(a, m, n);
        XuatMaTran(a, m, n);
}
```

Ví dụ

- Viết chương trình nhập 2 ma trận a và b gồm m dòng và n cột, tính ma trận c=a+b theo công thức `c[i][j]=a[i][j]+b[i][j]`.

  In c ra màn hình.
- Ý tưởng
  - Viết các hàm: nhập, cộng, in ma trận
  - Nhập số dòng/số cột cho ma trận
  - Nhập giá trị cho a và b
  - Thực hiện cộng
  - In kết quả ra màn hình

Ví dụ (tt)

Hàm cộng ma trận

```
/* Cong 2 ma tran A & B ket qua la ma tran C*/
void  CongMaTran(int  a[][MAXC],int  b[][MAXC],int  M,int
N,int c[][MAXC])
{
        int i,j;
        for(i=0;i<M;i++)
                for(j=0; j<N; j++)
                        c[i][j]=a[i][j]+b[i][j];
}
int main() //Chương trình chính
{
        /*Nhập m,n */
        NhapMaTran(a, m, n); NhapMaTran(b, m, n);
        CongMaTran(a,b,m,n,c); XuatMaTran(c,m,n);
}
```

Bài tập thực hành

1. Nhập từ bàn phím một mảng 2 chiều a gồm m dòng n cột
   a. In ma trận này ra màn hình.
   b. Nhập vào một số x, tìm x trong ma trận, xuất hiện bao nhiêu lần? những vị trí nào?
   c. Xây dựng b là ma trận chuyển vị của a.
2. Nhập vào ma trận a gồm m dòng, k cột, ma trận b gồm k dòng và n cột từ bàn phím.
   a. In 2 ma trận ra màn hình
   b. Tính c=a*b, với $c_{ij} = \sum_{k=0}^{n-1} a_{ik} * b_{kj}$
   c. In ma trận kết quả ra màn hình

Bài tập thực hành

3. Nhập ma trận vuông a cấp n

   a. Tính tổng các phần tử dương trên đường chéo chính.

   b. Tính tổng các phần tử là số nguyên tố trong ma trận tam giác trên.

   c. Đếm số phần tử là số chính phương trong ma trận tam giác dưới.

4. Nhập vào ma trận a (m dòng, n cột), đưa ra các phần tử yên ngựa trong a.

   Phần tử yên ngựa là phần tử nhỏ nhất trên dòng nhưng lớn nhất trên cột hoặc nhỏ nhất trên cột nhưng lớn nhất trên dòng.