BỘ CÔNG THƯƠNG
ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH

Bài giảng

NGÔN NGỮ LẬP TRÌNH C

THE C PROGRAMMING LANGUAGE

Lecturer : Le Ngoc Tran, PhD
Email : lengoctran@iuh.edu.vn

INDUSTRIAL UNIVERSITY OF HOCHIMINH CITY

CHƯƠNG 6

MẢNG (ARRAY): MẢNG MỘT CHIỀU

NỘI DUNG

- Tìm hiểu về mảng
- Cách khai báo mảng
- Nhập dữ liệu vào mảng
- Truy xuất dữ liệu trong mảng

6.1. 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?

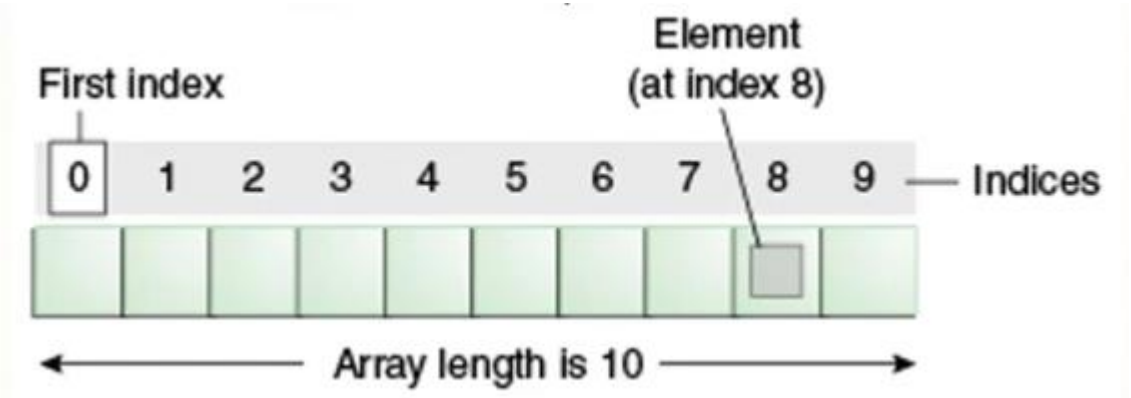
    =>khai báo **100** biến kiểu số nguyên!
  - Người dùng muốn nhập **n** số nguyên?

    =>không thực hiện được!
- Giải pháp
  - Kiểu dữ liệu mới cho phép **lưu trữ một dãy** các số nguyên và **dễ dàng truy xuất**

6.2. Giới thiệu mảng (Array)

- Mảng là tập hợp các phần tử cùng kiểu
- Mảng có số lượng phần tử cố định và được cấp phát vùng nhớ liên tục.
- Truy xuất các phần tử mảng bằng chỉ số, bắt đầu là 0.
- Ví dụ:

```
Int a[10] = {5, 8, 22, 1, 7, 6, 11, 25, 33, 9};
```

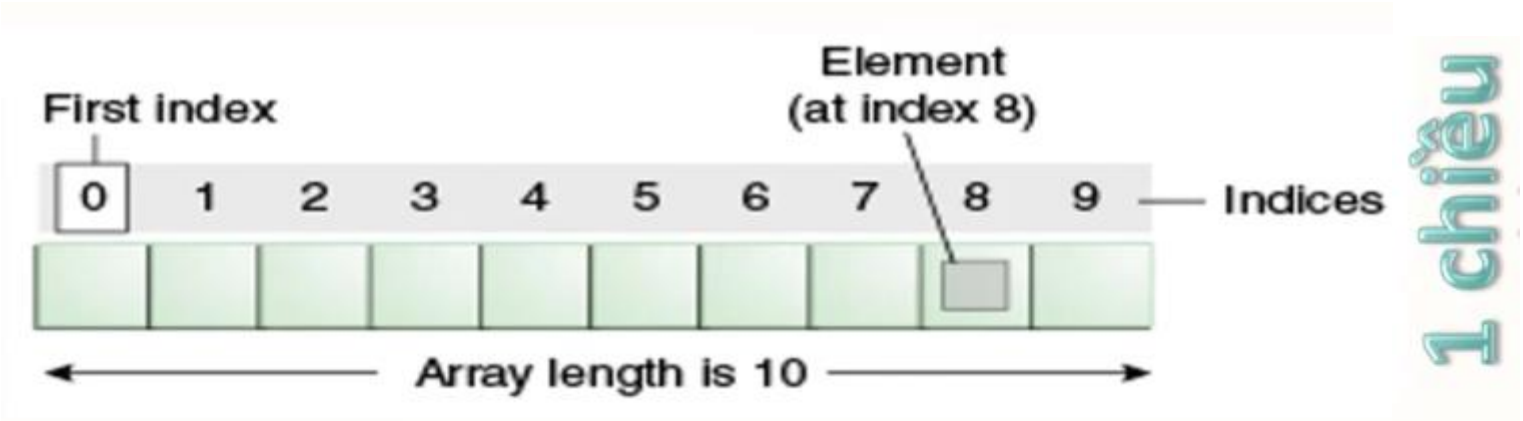


6.2. Giới thiệu mảng (Array)

- Lợi ích của Mảng
  - Sử dụng mảng để nắm giữ nhiều giá trị thay vì phải khai báo nhiều biến.
  - Truy xuất nhanh
  - Dễ dàng đọc dữ liệu từ các phần tử và sắp xếp
- Hạn chế của mảng
  - Số phần tử cố định nên không thể bổ sung hoặc bớt
  - Cấp phát liên tục nên cần phải có vùng nhớ liên tục đủ lớn để cấp phát.

6.3. Phân loại mảng

- Mảng một chiều
- Mảng nhiều chiều



6.4. Khai báo mảng

- Cú pháp:

*<**kiểu DL**> <tên biến mảng> [<số phần tử>];*

- Ví dụ:

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

6.5. Khai báo mảng 1 chiều (không tường minh)

- Cú pháp:
  - không tường minh (thông qua định nghĩa kiểu)

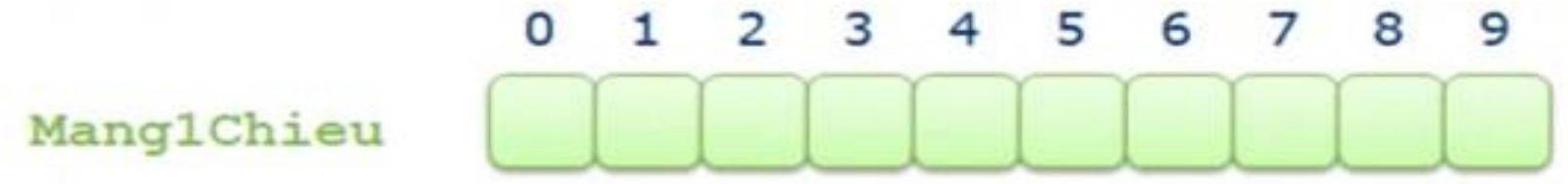
typedef ***<kiểu DL> <tên kiểu mảng>*** **[*<số phần tử>*];**

***< tên kiểu mảng > <tên biến mảng>*;**

- Ví dụ:

typedef int Mang1Chieu **[**10**]**;

Mang1Chieu m1, m2, m3;



CT Demo thực hiện mảng (slide6): **arr-decleration.c**

6.5. Khai báo mảng 1 chiều (không tường minh)

arr-declaration.c > ...

```c
#include<stdio.h>

int main(int argc, char const *argv[])
{
    int ars[10];

    int ars3[5];

    typedef int MyArray[20];

    MyArray brs, brs2;


    return 0;
}

```

6.5. Khai báo mảng 1 chiều (không tường minh)

❖ 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.

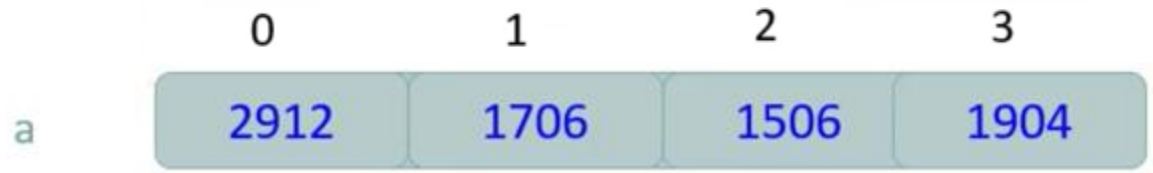
```
#define n1 10;
#define n2 20;
int a [n1];          //  ⇔ int a[10];
int b [n1] [n2];  //  ⇔ int b[10] [20] ;//mảng 2 chiều
```

6.5. Khai báo mảng 1 chiều (không tường minh)

❖ Khởi tạo giá trị mảng

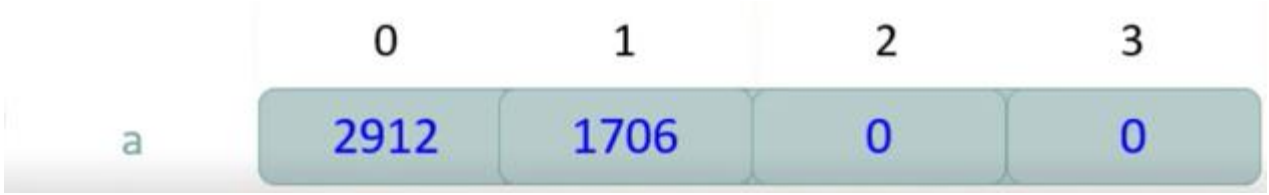
❑ Khởi tạo giá trị tất cả phần tử mảng.

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



❑ Khởi tạo giá trị cho một số phần tử đầu mảng.

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

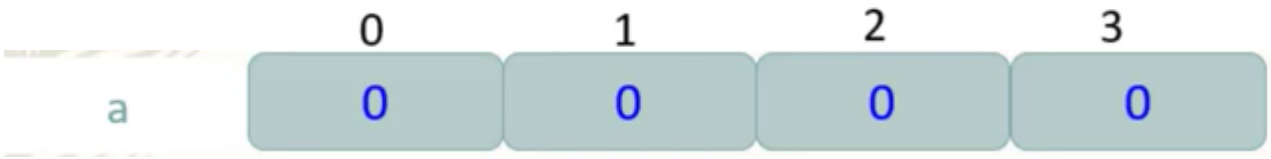


6.5. Khai báo mảng 1 chiều (không tường minh)

❖ Khởi tạo giá trị mảng

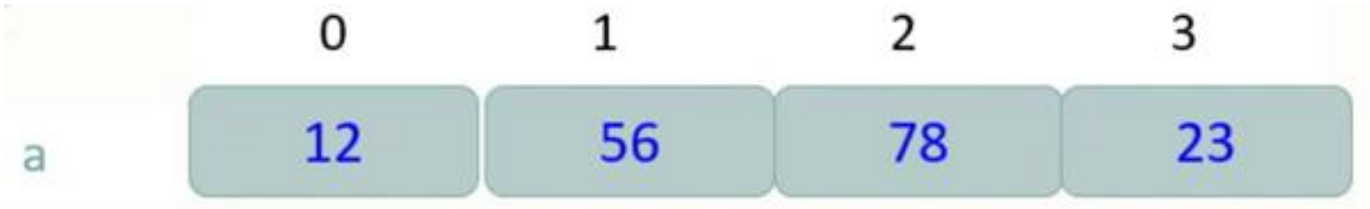
❑ Khởi tạo giá trị 0 cho mọi phần tử mảng.

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



❑ Tự động xác định số lượng phần tử.

```
int a[ ] = {12, 56, 78, 23};
```



CT Demo thực hiện mảng (slide6): **arr-initialization.c**

6.5. Khai báo mảng 1 chiều (không tường minh)

arr-initialization.c > main(int, char const * [])

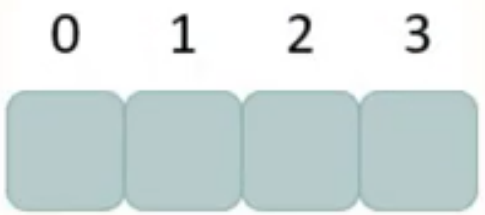
```c
#include<stdio.h>

int main(int argc, char const *argv[])
{
    int a[5] = {1, 5, 8, 9, 6};

    int b[] = {8, 7, 9};

    int c[5] = {7, 8};

    int d[5] = {0};


    return 0;
}
```

6.6. Truy xuất các phần tử của mảng

❑ Thông qua các chỉ số:

```
<tên biến mảng>[<gt cs1>][<gt cs2>]…[<gt csn>]
```

```
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], …

6.7. 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];     // Đúng
```

6.8. Duyệt mảng một chiều

- Sử dụng vòng lặp để duyệt mảng
- Vòng lặp for thường được dùng

```c
int i = 0;
int marks[5];  // khai báo mảng
marks[0] = 80; // khởi tạo mảng
marks[1] = 60;
marks[2] = 70;
marks[3] = 85;
marks[4] = 75;
//duyệt mảng
for (i = 0; i < 5; i++) {
    printf("%d \n", marks[i]);
}
```

```c
int i = 0;
int marks[5]={20, 30, 40, 50, 60};
//duyệt mảng
for (i = 0; i < 5; i++) {
    printf("%d \n", marks[i]);
}
```

6.8. Duyệt mảng

❑ Ví dụ:

```c
#include<stdio.h>

int main(int argc, char const *argv[])
{
    int ars[5] = {1, 5, 9, 3, 7};

    for (int i = 0; i < 5; i++)
    {
        printf("%d  ", ars[i]);
    }
    printf("\n");

    ars[0] = 8;
    ars[4] = 11;

    for (int i = 0; i < 5; i++)
    {
        printf("ars[%d] = %d \n", i, ars[i]);
    }


    return 0;
}
```

```
PS C:\MyData\Demonstration\PolyCLang2021\Slide6> cd
1  5  9  3  7
ars[0] = 8
ars[1] = 5
ars[2] = 9
ars[3] = 3
ars[4] = 11
PS C:\MyData\Demonstration\PolyCLang2021\Slide6>
```

6.9. Thao tác với mảng

- **Nhập và xuất giá trị của mảng, sau đó tìm 1 giá trị trong mảng**

```
int a[10],x;
for(int i = 0; i < 10; i++) {    //nhập giá trị vào mảng
    scanf("%d", &a[i]);
}
for (int i = 0; i < 10; i++) { //hiển thị giá trị mảng
    printf("%d", a[i]);
}
printf("Nhap phan tu can tim x = ");
scanf("%d", &x);
for(int i = 0; i < 10; i++){
    if(x == a[i]) {
        printf("%d co trong mang", x);
        break;
    }
}
```

6.9. Thao tác với mảng

```c
#include<stdio.h>

int main(int argc, char const *argv[])
{
    int a[5], x;

    printf("Enter array elements: \n");
    for (int i = 0; i < 5; i++)
    {
        printf("a[%d]: ", i);
        scanf("%d", &a[i]);
    }
    printf("\n");
    for (int i = 0; i < 5; i++)
    {
        printf("%d  ", a[i]);
    }
    printf("\n");
```

```
Enter array elements:
a[0]: 5
a[1]: 7
a[2]: 2
a[3]: 3
a[4]: 5

5  7  2  3  5
Enter a number: 5
5 is existed
```

```c
    printf("Enter a number: ");
    scanf("%d", &x);

    char flag = 0; //not existed

    for (int i = 0; i < 5; i++)
    {
        if (a[i] == x){
            flag = 1;
            break;
        }
    }
    if (flag == 1){
        printf("%d is existed \n", x);
    }else{
        printf("%d is not existed \n",x);
    }

    return 0;
}
```

6.9. Thao tác với mảng

```c
#include<stdio.h>

void display(int a[], int length){
    printf("\n");
    for (int i = 0; i < length; i++)
    {
        printf("%d  ", a[i]);
    }
    printf("\n");
}
void input(int a[], int length){
    printf("Enter array elements: \n");
    for (int i = 0; i < length; i++)
    {
        printf("a[%d]: ", i);
        scanf("%d", &a[i]);
    }
}
```

```c
int main(int argc, char const *argv[])
{
    int a[5];

    input(a, 5);

    display(a, 5);

    find(a, 5);

    display(a, 5);

    return 0;
}
```

```c
void find(int a[], int length){
    int x;
    printf("Enter a number: ");
    scanf("%d", &x);

    char flag = 0; //not existed

    for (int i = 0; i < length; i++)
    {
        if (a[i] == x){
            flag = 1;
            break;
        }
    }
    if (flag == 1){
        printf("%d is existed \n", x);
    }else{
        printf("%d is not existed \n",x);
    }
}
```

```
PS C:\MyData\Demonstration\PolyCl
Enter array elements:
a[0]: 5
a[1]: 6
a[2]: 2
a[3]: 3
a[4]: 9

5  6  2  3  9
Enter a number: 7
7 is not existed

5  6  2  3  9
PS C:\MyData\Demonstration\PolyCl
```

6.10. Lưu ý về mảng

- 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]; => int a[10];
  - const int n2=10; int a[n2]; => int a[10];
- 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;

6.11. Bài tập mảng 1 chiều

- Viết chương trình thực hiện:
  - Nhập vào 100 số ngẫu nhiên từ 1 đến 200
  - Tìm số lớn nhất và số nhỏ nhất trong mảng 100 phần tử
  - Đếm xem có bao nhiêu số chia hết cho 5 và 9.

6.12. Sắp xếp mảng 1 chiều (bubble sort)

- **Đặt vấn đề:**
  - Sắp xếp các sản phẩm theo giá từ thấp đến cao
  - Sắp xếp sinh viên theo điểm từ cao tới thấp
  - Sắp xếp các bộ phim theo số lượt xem

6.12. Sắp xếp mảng 1 chiều (bubble sort)

- Sắp xếp là sắp đặt các phần tử của một cấu trúc theo thứ tự tăng dần (hay giảm dần).
- Với một cấu trúc đã được sắp xếp giúp
  - Thực hiện dễ dàng và nhanh chóng tìm kiếm, trích lọc, duyệt cấu trúc...
- Có nhiều thuật toán sắp xếp

6.12. Sắp xếp mảng 1 chiều (bubble sort)

- **Thuật toán sắp xếp:**
  - Bubble sort
  - Quick sort
  - Simple selection sort
  - Head sort
  - Simple insertion sort
  - Shell sort
  - Merge sort
  - Radix sort

6.12. Sắp xếp mảng 1 chiều (bubble sort)

❖ Thuật toán sắp xếp nổi bọt (bubble sort):

- Thuật toán Bubble sort sẽ duyệt danh sách nhiều lần, trong mỗi lần duyệt sẽ lần lượt so sánh cặp nút thứ i và thứ i+1 và đổi chỗ hai nút này nếu chúng không đúng thứ tự.

Ví dụ: Sắp xếp danh sách các số theo thứ tự tăng dần

Giải pháp cho 1 vòng lặp:

5 6 3 1 8 7 2 4

=> Bước 1: 5 6 3 1 8 7 2 4

=> Bước 2: 5 3 6 1 8 7 2 4

=> Bước 3: 5 3 1 6 8 7 2 4

6.12. Sắp xếp mảng 1 chiều (bubble sort)

=> Bước 4: 5 3 1 6 **8 7** 2 4

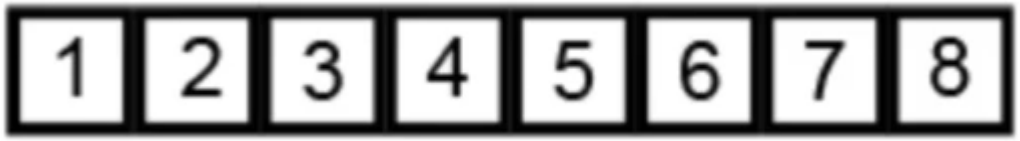
=> Bước 5: 5 3 1 6 7 **8 2** 4

=> Bước 6: 5 3 1 6 7 2 **8 4**

=> Bước 7: 5 3 1 6 7 2 4 **8**

=> Bước 8: **5 3** 1 6 7 2 4 **8**

Tiếp tục so sánh để chọn các số có giá trị lớn để dồn phía bên phải. Kết quả ta sẽ đạt được



6.12. Sắp xếp mảng 1 chiều (bubble sort)

- Chúng ta có thể sắp xếp tăng dần hoặc giảm dần các phần tử trong mảng

```
int a[7]={8,2,6,2,9,1,5};
int i , j, temp;
    for(i=0;i<6;i++){
        for(j=i+1;j<7;j++){
        if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
    }
    }
```

Nếu thay đổi toán tử so sánh thành < thì thuật toán trở thành sắp xếp giảm dần.

6.12. Sắp xếp mảng 1 chiều (bubble sort)

- **Ví dụ:** cho 1 mảng a[5] = {2 5 7 1 3} sắp xếp mảng theo thứ tự tăng dần.

```c
#include<stdio.h>

int main(int argc, char const *argv[])
{
    int a[5] = {2, 5, 7, 1, 3};
    printf("\n");
    for (int i =0; i < 5; i++)
    {
        printf("%d", a[i]);
    }
    printf("\n");
    for(int i = 0; i < 4; i++)
    {
        for(int j =i+1; j <5; j++)
        {
            if (a[i] > a[j]){
                int temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }
    for (int i =0; i < 5; i++)
    {
        printf("%d", a[i]);
    }
    printf("\n");

    return 0;
}
```

6.13. Khai báo mảng 2 chiều

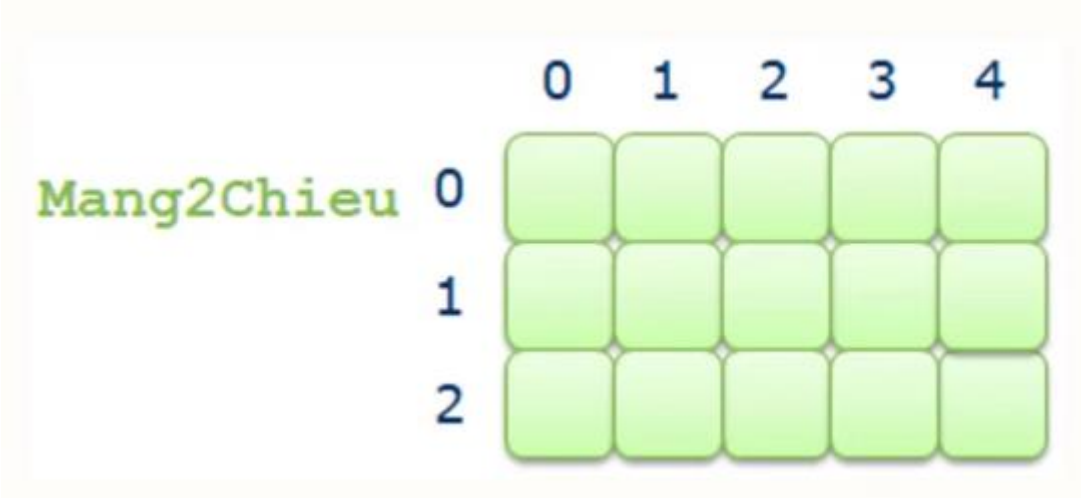
Cú pháp:

<*kiểu DL*> <*tên biến mảng*> [<*N1*>] [<*N2*>]...[<*Nn*>];

[<*N1*>] [<*N2*>]...[<*Nn*>]: số lượng phần tử mỗi chiều.

Ví dụ:

```
int Mang2Chieu [3] [5];
```



6.13. Khai báo mảng 2 chiều

- **Khai báo mảng 2 chiều không tường minh (thông qua khai báo kiểu):**

  typedef <*kiểu DL*> <*tên kiểu mảng*> [<*N1*>]...[<*Nn*>];

  <*tên kiểu mảng*> <*tên biến mảng*>;

- **Ví dụ:**

```
typedef int Mang2Chieu [3] [4];
Mang2Chieu m1, m2;
```

6.13. Khai báo mảng 2 chiều

two-dim-array.c > ...

```c
#include<stdio.h>

int main(int argc, char const *argv[])
{
    int a[5];

    int b[3][5];

    int bs[3][5] = {
        {1, 2, 4, 7, 3},
        {3, 7, 9, 4, 4},
        {0, 2, 0, 1, 8},
    };
    typedef int TwoDimArray[3][4];
    TwoDimArray c;

    return 0;
}
```

6.14. Thao tác với mảng 2 chiều

❖ Khởi tạo giá trị cho mảng 2 chiều:

```
//Khởi tạo mảng 2 chiều bằng cách gán giá trị.
int i = 0;
int j = 0;
int ma_tran[4][3] = {{1, 2, 3}, {2, 3, 4}, {3, 4, 5}, {4, 5, 6}};
for (i = 0; i < 4; i++)
{ //duyệt mảng
    for (j = 0; j < 3; j++)
    {
        printf("%d ", ma_tran[i][j]);
    }
    printf("\n");
}
```

6.14. Thao tác với mảng 2 chiều

❖ Khởi tạo giá trị cho mảng 2 chiều:

```
//Khởi tạo mảng 2 chiều bằng cách nhập giá trị từ bàn phím.
int i, j;
int ma_tran[4][3];
printf("Nhap mang: \n");
for (int i = 0; i < 4; i++)
{   //nhập mảng
    for (int j = 0; j < 3; j++)
    {
        printf("Nhap a[%d][%d] = ", i, j);
        scanf("%d", &ma_tran[i][j]);
    }
    printf("\n");
}
```

6.14. Thao tác với mảng 2 chiều

CT Demo thực hiện nhập matran (slide5): **matrix.c**

matrix.c > ...

```c
#include<stdio.h>

int main(int argc, char const *argv[])
{
    int matrix[3][4] ={
        {0, 8, 7, 9},
        {0, 18, 5, 7},
        {0, 28, 17, 1},
    };
    printf("Enter matrix values: \n");
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            printf("matrix[%d][%d] = ", i, j);
            scanf("%d", &matrix[i][j]);
        }

    }

    return 0;
}
```

6.14. Thao tác với mảng 2 chiều

❖ Duyệt mảng:

- ❑ Dùng 2 vòng lặp for lồng nhau để duyệt mảng 2 chiều.

```
int i = 0;
int j = 0;
int ma_tran[4][3] = {{1, 2, 3}, {2, 3, 4}, {3, 4, 5}, {4, 5, 6}};
for (i = 0; i < 4; i++)
{
    for (j = 0; j < 3; j++)
    {
        printf("%d ", ma_tran[i][j]);
    }
    printf("\n");
}
```

6.14. Thao tác với mảng 2 chiều

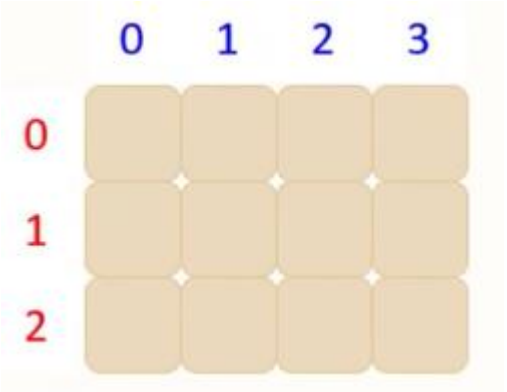
❖ Truy xuất mảng 2 chiều:

❑ Thông qua chỉ số

*<**tên biến mảng**>* [*<**giá trị cs1**>*] [*<**giá trị cs2**>*]

❑ 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],

CT Demo thực hiện nhập matran (slide5): **matrix.c**

6.14. Thao tác với mảng 2 chiều

❖ Gán dữ liệu kiểm mảng 2 chiều:

- ❑ 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**> [<**giá trị cs1**>] [<**giá trị cs2**>] = <**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];
```

6.14. Thao tác với mảng 2 chiều

CT Demo thực hiện copy PT mảng (slide5): **copy-array.c**

```c
#include<stdio.h>
#include<string.h>

int main(int argc, char const *argv[])
{
    int source[3][4]= {
        {1, 2, 4, 5},
        {6, 7, 6, 9},
        {8, 4, 5, 7},
    };
    int destination[3][4];
    memcpy(destination, source, sizeof(source));
    // for (int i = 0; i < 3; i++)
    // {
    //     for (int j = 0; j < 4; j++)
    //     {
    //         destination[i][j] = source[i][j];

    //     }
    // }
    printf("Elements of destination:");
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            printf("%d ", destination[j][j]);

        }
        printf("\n");
    }
    return 0;
}
```

6.14. Thao tác với mảng 2 chiều

❖ Nhập và tìm giá trị trong mảng 2 chiều:

```
int i, j,x;
int ma_tran[2][2];
printf("Nhap mang: \n");
    for (int i = 0; i < 2; i++) {    //nhập mảng
        for (int j = 0; j < 2; j++) {
            printf("Nhap a[%d][%d] = ", i, j);
            scanf("%d", &ma_tran[i][j]);
        }
        printf("\n");
    }
```

```
printf("Nhap phan tu can tim x = ");
scanf("%d", &x);
for (i = 0; i < 2; i++) {        //duyệt mảng
        for (j = 0; j < 2; j++) {
            if(x == ma_tran[i][j]){
                printf("%d co trong mang", x);
                break;
            }
        }
}
```

6.14. Thao tác với mảng 2 chiều

```c
#include<stdio.h>

int main(int argc, char const *argv[])
{
    int const ROW = 3;
    int const COLUMN = 4;

    int matrix[ROW][COLUMN];
    printf("Enter elements of matrix: \n");
    for (int i = 0; i < ROW; i++)
    {
        for (int j = 0; j < COLUMN; j++)
        {
            printf("matrix[%d][%d] = ", i, j);
            scanf("%d", &matrix[i][j]);
        }
    }
```

```c
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    char flag = 0;
    for (int i = 0; i < ROW && flag == 0; i++)
    {
        for (int j = 0; j < COLUMN; j++)
        {
            if (num == matrix[i][j]){
                flag = 1;
                break;
            }
        }
    }
    if (flag == 1){
        printf("%d is existed \n", num);
    }else{
        printf("%d is not existed \n", num);
    }

    return 0;
}
```


THANK
YOU