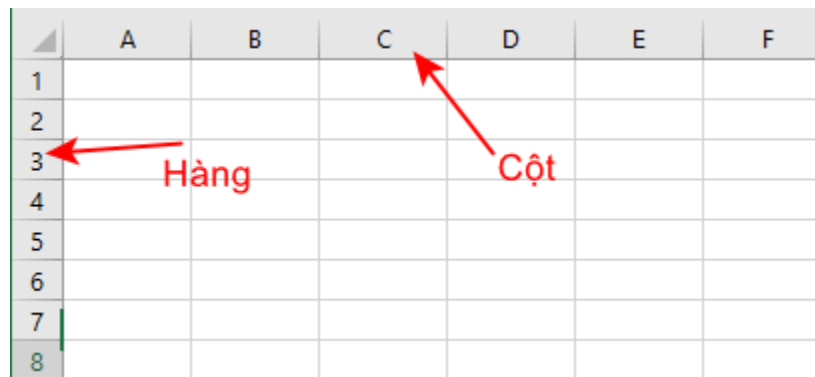


Bài 2.3: Mảng hai chiều

- ✓ Mục đích sử dụng
- ✓ Cú pháp tổng quát
- ✓ Khởi tạo và truy cập mảng
- ✓ Cấp phát động mảng 2 chiều
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Mục đích sử dụng

- C++ hỗ trợ mảng nhiều chiều. Trong đó mảng hai chiều là điển hình trong số này.
- Sử dụng mảng hai chiều chủ yếu để lưu trữ các thông tin dạng bảng như hình ảnh, video, tọa độ trên bản đồ...
- Ví dụ điển hình của mảng hai chiều là áp dụng trong xử lý ảnh, bảng tính xcell, bảng cơ sở dữ liệu, biểu diễn các ma trận kề, ma trận liên thuộc,...



	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						

Cú pháp tổng quát

Cú pháp tổng quát: **type** **arrayName**[**row**][**col**];

- **Type**: là kiểu của mảng. Có thể là bất kì kiểu hợp lệ nào trong C++ như int, float, bool...
- **arrayName**: là tên của tập hợp mà mảng đại diện, thường sử dụng danh từ số nhiều đại diện cho một tập hợp.
- Hai cặp móc vuông là dấu hiệu nhận biết biến kiểu mảng.
- **row, col**: số hàng tối đa của mảng, số cột tối đa của mảng, luôn là số nguyên dương.
- Kết thúc khai báo biến kiểu mảng bằng dấu chấm phẩy ;

Cú pháp tổng quát

	col0	col1	col2	col3
row0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
row1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
row2	a[2][0]	a[2][1]	a[2][2]	a[2][3]
row3	a[3][0]	a[3][1]	a[3][2]	a[3][3]

- Cú pháp để tạo mảng nhiều chiều tổng quát:
type arrayName[size1][size2][sizek...][sizeN];
- Trong đó size_i là số phần tử tối đa ở chiều thứ i của mảng.

Khởi tạo mảng

➤ Có 2 cách khởi tạo mảng hai chiều:

➤ Cách 1:

```
type arr[M][N] = {  
  {a[0][0], a[0][1], a[0][...], a[0][N - 1]},  
  {a[1][0], a[1][1], a[1][...], a[1][N - 1]},  
  ...  
  {a[M - 1][0], a[M - 1][1], a[M - 1][...], a[M - 1][N - 1] }  
};
```

➤ Cách 2:

```
type arr[M][N] = { a[0][0], a[0][1], a[0][...], a[M-1][N - 1] };
```

➤ Trong đó M, N là số hàng, số cột tối đa của mảng.

➤ Cách 1 tường minh hơn cách 2 ở chỗ gom cụm rõ ràng từng hàng của mảng với cặp {} bên trong, các hàng, các phần tử cách nhau bởi dấu phẩy

➤ Sau phần tử cuối và hàng cuối không cần dấu phẩy

➤ Cách thứ 2 cho ta khởi tạo mảng 2 chiều như khởi tạo mảng một chiều

Khởi tạo mảng

- Nếu mảng có kích thước M hàng N cột thì tổng số phần tử của mảng là $M \times N$ phần tử.
- Kết thúc khởi tạo là dấu chấm phẩy;

Cách 1:

```
int arr[3][4] = {  
    {0, 3, 5, 9},  
    {4, 1, 5, 2},  
    {6, 3, 8, 7}  
}
```

Cách 2:

```
int arr[3][4] = { 0, 3, 5, 9, 4, 1, 5, 2, 6, 3, 8, 7 };
```


Truy cập mảng

- Chỉ số phần tử mảng 2 chiều bắt đầu từ hàng 0 cột 0 đến row-1 và col-1 tương ứng
- Cú pháp truy cập phần tử tại vị trí rowIndex, colIndex: `arrayName[rowIndex][colIndex]`
- Nếu bạn cung cấp chỉ số rowIndex và colIndex ngoài đoạn giới hạn ở trên thì chương trình có thể bị lỗi hoặc kết quả không đúng
- Để truy xuất mảng hai chiều ta sử dụng vòng for lồng nhau.

```
int arr[M][N] = { 0, 3, 5, 9, 4, 1, 5, 2, 6, 3, 8, 7 };
int myArray[N][N];
// gán giá trị cho phần tử đầu tiên:
myArray[0][0] = 20;
// gán giá trị cho phần tử hàng chỉ số 2, cột chỉ số 3:
myArray[2][3] = 50;
// gán giá trị cho phần tử cuối cùng
myArray[3][3] = 70;

// hiện kết quả:
cout << "myArr[0][0] = " << myArray[0][0] << endl;
cout << "myArr[2][3] = " << myArray[2][3] << endl;
```

Ví dụ

- Duyệt mảng với vòng lặp for lồng nhau:

```
// hiển thị danh sách phần tử của mảng arr:
cout << "Cac phan tu mang arr: " << endl;
for (int i = 0; i < M; i++)
{
    for (int j = 0; j < N; j++)
    {
        cout << arr[i][j] << " ";
    }
    cout << endl;
}
```


Cấp phát động

- Giả sử ta muốn cấp phát mảng 2 chiều kích thước row x col
- Cách 1: dùng mảng 1 chiều.
- Cách 2: dùng con trỏ trỏ đến con trỏ.
- Cách 3: dùng lớp array.
- Cách 4: dùng unique_ptr.
- Với cách 1, ta tạo ra mảng có row x col phần tử

Cách 1

```
#include <iostream>
#include <memory>

using namespace std;

int main() {
    // cấp phát động mảng 2 chiều như mảng 1 chiều
    int row = 3;
    int col = 4;
    unique_ptr<int[]> matrix(new int[row * col]
                             {1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 5, 7});

    // sử dụng mảng 1 chiều như mảng 2 chiều
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            cout << matrix[i * col + j] << " ";
        }
        cout << endl;
    }
    // đối tượng của unique_ptr sẽ tự delete mảng cho ta
    return 0;
}
```

Cách 2

- Với cách 2, cú pháp cấp phát gồm 2 bước:
 - Bước 1: cấp phát hàng(cấp phát mảng các con trỏ trỏ tới từng hàng).
 - Bước 2: cấp phát cột(các mảng trong từng hàng).
- Ví dụ với con trỏ trỏ tới con trỏ: `int** matrix;`

```
int row = 3;
int col = 4;

int** matrix = new int* [row]; // cấp phát hàng
for (int i = 0; i < row; i++) { // cấp phát cột
    matrix[i] = new int[col]();
}
```

Cách 2

- Khi thu hồi bộ nhớ ta làm theo các bước ngược lại khi cấp phát:
 - Xóa từng mảng đơn trước
 - Sau đó xóa mảng con trở sau cùng

- Ví dụ:

```
// thu hồi bộ nhớ
for (int i = 0; i < row; i++) {
    delete [] matrix[i]; // thu hồi từng hàng trước
}
delete[] matrix; // sau đó thu hồi mảng chứa
                  // con trỏ trỏ đến từng hàng
```

Cách 3

- Ta include thư viện `<array>` vào trước khi sử dụng.
- Cú pháp: `array<array<type, col>, row> arrayName;`
- Trong đó:
 - **type**: là kiểu của mảng. Có thể là các kiểu hợp lệ trong C++.
 - **row, col**: là số hàng, cột tối đa của mảng.
 - **array**: tên lớp, bắt buộc.
 - **arrayName**: tên mảng hai chiều.

Cách 3

➤ Ví dụ sử dụng array để tạo mảng hai chiều:

```
void showElements(array<array<int, SIZE>, SIZE> arr) {  
    for (size_t i = 0; i < arr.size(); i++)  
    {  
        for (size_t j = 0; j < arr.at(0).size(); j++)  
        {  
            cout << arr[i][j] << " ";  
        }  
        cout << endl;  
    }  
}
```

```
int main() {  
    array<array<int, SIZE>, SIZE> arr;  
    for (int i = 0; i < SIZE; i++)  
    {  
        arr[i] = array<int, SIZE>();  
    }  
    addElements(arr);  
    showElements(arr);  
    return 0;  
}
```


Cách 4

- Ta include thư viện `<memory>` trước khi sử dụng `unique_ptr`.
- Cú pháp: `unique_ptr<unique_ptr<type[]>[]> arrayName(new unique_ptr<type[]>[SIZE]);`
 - Từ khóa `unique_ptr` là bắt buộc.
 - Type là kiểu dữ liệu của mảng.
 - `arrayName` là tên mảng.

```
unique_ptr<unique_ptr<int[]>[]> matrix(new unique_ptr<int[]>[SIZE]);
for (int i = 0; i < SIZE; i++)
{
    matrix[i] = unique_ptr<int[]>(new int[SIZE] {1, 2, 3, 5});
}
for (size_t i = 0; i < SIZE; i++)
{
    for (size_t j = 0; j < SIZE; j++)
    {
        cout << matrix[i][j] << " ";
    }
    cout << endl;
}
```

Nội dung tiếp theo

Độ phức tạp thuật toán