KỸ THUẬT LẬP TRÌNH C/C++

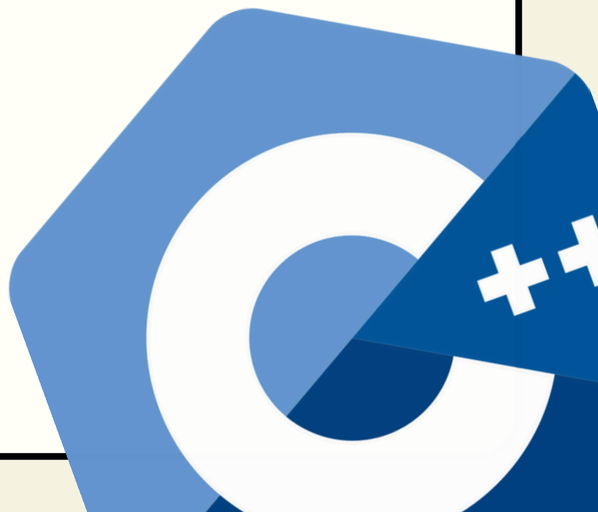
Dạy bởi 1 SVBK

Đại học Bách Khoa Hà Nội
Kỹ thuật Điện tử - Viễn thông

2024-25

NỘI DUNG

- Mảng 1 chiều
- Mảng 2 chiều



GIỚI THIỆU CHUNG

Mảng trong C là một trong những cấu trúc dữ liệu được sử dụng phổ biến nhất trong lập trình C. Đây là cách đơn giản và nhanh chóng để lưu trữ nhiều giá trị dưới một tên duy nhất. Trong bài viết này, chúng ta sẽ nghiên cứu các khía cạnh khác nhau của mảng trong ngôn ngữ C như **khai báo mảng, định nghĩa, khởi tạo, các loại mảng, cú pháp mảng**, cùng nhiều nội dung khác.

Mảng (Array) thường được dùng để lưu trữ **dãy hữu hạn các giá trị cùng kiểu**. Trong bộ nhớ, các giá trị này được đặt tại các vị trí cố định **(gọi là các phần tử)** và chúng được sắp xếp liên tiếp nhau. Các phần tử được đánh số bằng các **chỉ số (index).**

MẢNG 1 CHIỀU

CÚ PHÁP - KHAI BÁO BIẾN MẢNG

```
kiểu tên_biến_mảng[kích_thước];
kiểu tên_biến_mảng[kích_thước] = { các biểu thức khởi tạo };
kiểu tên_biến_mảng[] = { các biểu thức khởi tạo };
```

Mảng được sử dụng để lưu trữ nhiều giá trị **trong một biến duy nhất**, thay vì khai báo các biến riêng lẻ cho từng giá trị.
Để tạo một mảng, hãy **xác định kiểu dữ liệu (như int)** và chỉ định tên của mảng theo sau bởi **dấu ngoặc vuông [].**
Để chèn các giá trị vào mảng, sử dụng **danh sách các giá trị được phân tách bằng dấu phẩy** bên trong **dấu ngoặc nhọn**, và đảm bảo rằng tất cả các giá trị **đều thuộc cùng một kiểu dữ liệu.**

MẢNG 1 CHIỀU

Khởi tạo khi khai báo

```
int arr[5] = {1, 2, 3, 4, 5};
```

Trình biên dịch tự xác định kích thước:

```
int arr[] = {10, 20, 30};
```

Ít phần tử hơn kích thước:

```
int arr[5] = {1, 2};
```

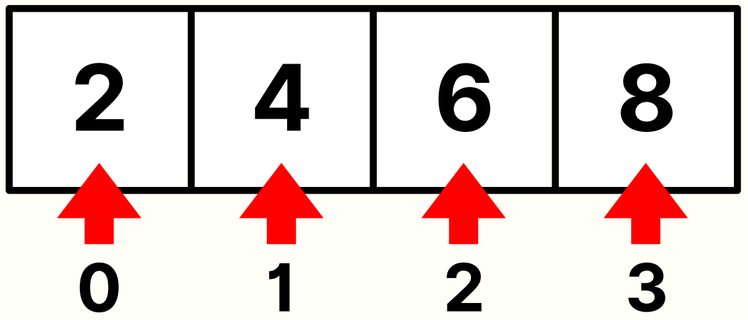
// Các phần tử còn lại mặc định là 0

MẢNG 1 CHIỀU

TRUY CẬP MẢNG

Để truy cập một phần tử trong mảng, hãy tham chiếu đến **số chỉ mục (index)** của nó.
Chỉ mục của mảng bắt đầu từ **[0] là phần tử đầu tiên, [1] là phần tử thứ hai, v.v.**

PHẦN TỬ TRONG MẢNG



INDEX CỦA MẢNG

MẢNG 1 CHIỀU

TRUY CẬP MẢNG

VÍ DỤ

```c
int numbers[] = {1, 2, 3, 4};
printf("Phan tu dau tien: %d", numbers[0]);
```

OUTPUT

```
Phan tu dau tien: 1
```

MẢNG 1 CHIỀU

THAY ĐỔI GIÁ TRỊ PHẦN TỬ

Để thay đổi giá trị của một phần tử trong mảng, ta sẽ **tham chiếu đến index** của phần tử đó và **gán** cho nó giá trị mới, cũng như mọi công việc khác, ta sẽ **luôn dùng cách tham chiếu này** để làm việc với các phần tử của mảng.

VÍ DỤ

```c
int numbers[] = {1, 2, 3, 4};
printf("Phan tu dau tien truoc: %d\n", numbers[0]);
numbers[0] = 10;
printf("Phan tu dau tien sau: %d", numbers[0]);
```

OUTPUT

```
Phan tu dau tien truoc: 1
Phan tu dau tien sau: 10
```

MẢNG 1 CHIỀU

NHẬP XUẤT MẢNG 1 CHIỀU

Khi duyệt một mảng, chúng ta thường sử dụng **vòng lặp for.** Trong vòng lặp này, biến khởi tạo (thường là một biến đếm, ví dụ i = 0) sẽ **đại diện cho chỉ mục (index)** của phần tử trong mảng. Ở mỗi lần lặp, **biến này sẽ tăng dần lên (thường là i++)**, giúp chúng ta lần lượt truy cập từng phần tử của mảng theo thứ tự từ **đầu đến cuối.**

VÍ DỤ

```c
int numbers[] = {1, 2, 3, 4};
for (int i = 0; i < 4; i++)
{
    printf("%d ", numbers[i]);
}
```

In ra các phần tử trong mảng ra màn hình.

OUTPUT

```
1 2 3 4
```

MẢNG 1 CHIỀU

NHẬP XUẤT MẢNG 1 CHIỀU

Ta cũng có thể sử dụng vòng lặp *for* để **nhập vào các phần tử cho mảng:**

```
int n;
scanf("%d", &n);
int arr[n];
for (int i = 0; i < n; i++)
{
    scanf("%d", &arr[i]);
}
```

Nhập vào từ bàn phím mảng có **n phần tử** và các phần tử của mảng. Để nhập mảng, ta sẽ nhập vào kích thước n của mảng trước, rồi sẽ nhập vào các phần tử của mảng từ đầu đến cuối.

Chú ý: ta có thể cho mảng có **kích thước vừa đủ** như trong ví dụ hoặc dư ra như arr[1000], arr[2000],... **(tối đa 10^7 phần tử)**

MẢNG 1 CHIỀU

KÍCH THƯỚC MẢNG

Khi nói về **kích thước của mảng**, ta sẽ thương nghĩ theo 2 hướng:

1

Số lượng phần tử trong mảng: Như đã học trong phần khai báo thì, số lượng phần tử sẽ là **cố định sau khi xác định**, nên việc **khai báo cho dư ra chút** sẽ là 1 thói quen tốt khi làm việc với mảng tĩnh.

2

Kích thước theo byte: Kích thước của mảng trong C được tính bằng số byte chiếm dụng trong bộ nhớ, theo công thức:

Tổng kích thước = Số phần tử x Kích thước mỗi phần tử

MẢNG 1 CHIỀU

KÍCH THƯỚC MẢNG

Ngoài ra kích thước theo byte của mảng cũng có thể tính qua **toán tử sizeof():**

VÍ DỤ

```c
int numbers[] = {1, 2, 3, 4};
printf("%d", sizeof(numbers));
```

OUTPUT

```
16
```

GIẢI THÍCH

4 phần tử int (4 byte mỗi phần tử) → 16 byte.

MẢNG 2 CHIỀU

Trong mục trước, ta đã học về mảng, còn được gọi là **mảng một chiều.** Đây là một khái niệm rất hữu ích và ta sẽ sử dụng nó thường xuyên khi lập trình trong C. Tuy nhiên, nếu bạn muốn lưu trữ dữ liệu dưới dạng bảng, như một bảng có hàng và cột, thì ta cần làm quen với mảng đa chiều.

Mảng đa chiều về cơ bản là **một mảng chứa các mảng bên trong.**

Mảng có thể có bất kỳ số chiều nào. Trong chương này, chúng ta sẽ giới thiệu loại phổ biến nhất**: mảng hai chiều (2D).**

MẢNG 2 CHIỀU

Mảng 2 chiều còn được gọi là **1 ma trận** (1 bảng với **hàng và cột**).
Số phần tử trong mảng là: **số hàng x số cột.**
Sau đây là ví dụ về 1 mảng 2 chiều với **2 hàng và 3 cột:**

```
int matrix[2][3] = { {1, 2, 3}, {4, 5, 6} };
```

Cú pháp: kiểu_dữ_liệu tên_mảng[số_hàng][số_cột];

Chú ý rằng: thứ tự của index các phần tử sẽ luôn là **hàng trước - cột sau.**

MẢNG 2 CHIỀU

```
int matrix[2][3] = { {1, 2, 3}, {4, 5, 6} };
```

	Cột 0	Cột 1	Cột 2
Hàng 0	matrix[0][0] = 1	matrix[0][1] = 2	matrix[0][2] = 3
Hàng 1	matrix[1][0] = 4	matrix[1][1] = 5	matrix[1][2] = 6

MẢNG 2 CHIỀU

NHẬP XUẤT MẢNG 2 CHIỀU

Tương tự với mảng 1 chiều, ta cũng sẽ dùng **vòng lặp for** để thực hiện các công việc với mảng:

Nhập mảng 2 chiều với số lượng hàng và cột theo yêu cầu của đề bài:

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

MẢNG 2 CHIỀU

TRUY CẬP PHẦN TỬ MẢNG

Không chỉ đối với nhập xuất, chúng ta làm mọi công việc khi thao tác với phần tử mảng 2 chiều sẽ tương tự như mảng 1 chiều.

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

Nhập mảng 2 chiều với số lượng hàng và cột theo yêu cầu của đề bài:

OUTPUT

```
21
```