Bài 33: MẢNG 1 CHIỀU TRONG C++ (ARRAYS)

Xem bài học trên website để ủng hộ Kteam: Mảng 1 chiều trong C++ (Arrays)

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Ở bài học trước, mình đã chia sẻ cho các bạn về phương pháp PHÁT SINH SỐ NGẪU NHIÊN TRONG C++ (Random number generation).

Hôm nay, mình sẽ giới thiệu cho các bạn về 1 kiểu dữ liệu có cấu trúc, do lập trình viên tự định nghĩa, đó là **Mảng 1 chiều trong C++ (Arrays)**.

Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về:

- VÒNG LẶP FOR TRONG C++ (For statements)

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

- Tại sao lại sử dụng mảng?
- Tổng quan về mảng 1 chiều
- Khai báo và khởi tạo mảng 1 chiều
- Xuất các phần tử mảng 1 chiều
- Nhập dữ liệu cho mảng 1 chiều
- Phát sinh dữ liệu ngẫu nhiên cho mảng 1 chiều

Tại sao lại sử dụng mảng?

Một công ty có nhu cầu xây dựng phần mềm lưu trữ mức lương của từng nhân viên để tiện cho việc quản lý. Giả sử:

- Công ty có **3** nhân viên => Khai báo **3** biến int salary1, salary2, salary3;
- Công ty có **100** nhân viên => Khai báo **100** biến int salary1, ...;
- Công ty có **1000** nhân viên => **Không** thực hiện được !!!

Để giải quyết những vấn đề đó, C++ cho phép lập trình viên có thể xây dựng kiểu dữ liệu đáp ứng nhu cầu lưu trữ và quản lý nhiều đối tượng cùng kiểu trong một định danh, nó được gọi là kiểu dữ liệu **mảng (arrays)**.

Sử dụng mảng để giải quyết vấn đề trên:

```
// allocate 1000 double variables in a fixed array
int salary[1000];
```

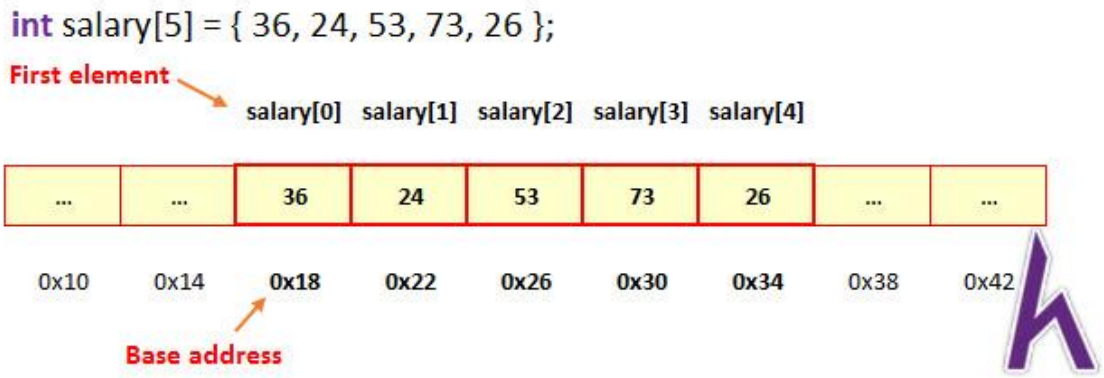
Tổng quan về mảng 1 chiều

Mảng 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 (mảng tĩnh)**.
- C++ luôn chỉ định **một khối nhớ liên tục** cho một biến kiểu mảng.

Ví dụ:

Hình bên dưới mô tả 1 mảng tên là **salary** có kiểu **int** gồm **5 phần tử (đã khởi tạo)** nằm trong vùng nhớ RAM:



Mỗi ô nhớ trong RAM có kích thước 4 byte, **salary** là 1 mảng kiểu **int**, nên mỗi phần sẽ nằm trong 1 ô nhớ, và những ô nhớ đó là liên tiếp nhau.

Các vấn đề về địa chỉ và vùng nhớ của mảng sẽ được chia sẻ chi tiết trong bài Con trỏ và mảng (Pointers and arrays).

Khai báo và khởi tạo mảng 1 chiều

Khai báo mảng 1 chiều

Cú pháp:

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

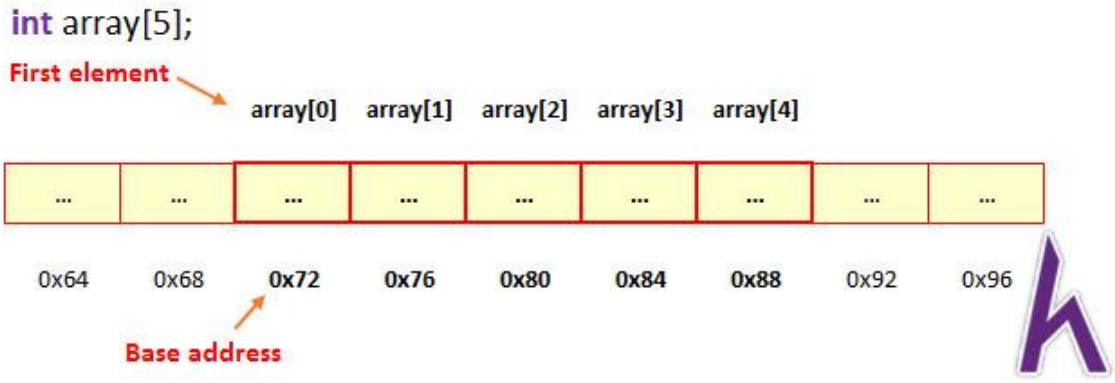
Lưu ý:

- Phải **xác định <số phần tử> cụ thể (hằng số)** khi khai báo.
- Nên sử dụng chỉ thị tiền xử lý **#define** để định nghĩa <số phần tử> mảng.
- Một mảng liên tục có chỉ số từ **0** đến **<tổng số phần tử> - 1.**
- Bộ nhớ sử dụng = <tổng số phần tử> * sizeof(<kiểu cơ sở>).

Ví dụ:

```
int array[5];
```

Hình bên dưới mô tả 1 mảng tên là **array** có kiểu **int** gồm **5 phần tử (chưa khởi tạo)** nằm trong vùng nhớ RAM:

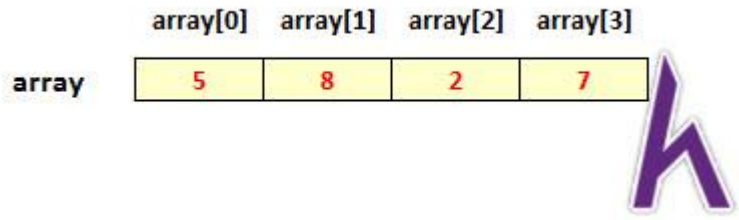


> **Chú ý:** Khi một mảng **chưa khởi tạo**, các phần tử của mảng sẽ mang **giá trị rác**.

Khởi tạo giá trị cho mảng 1 chiều

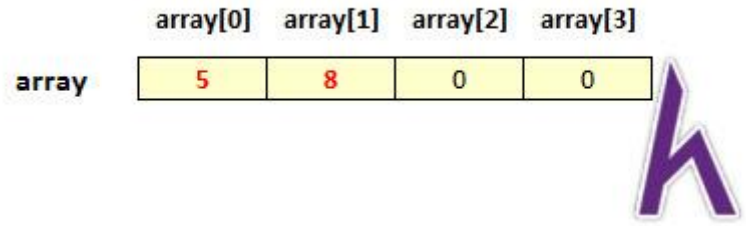
Cách 1: Khởi tạo giá trị cho mọi phần tử của mảng

```
int array[4] = { 5, 8, 2, 7 };
```



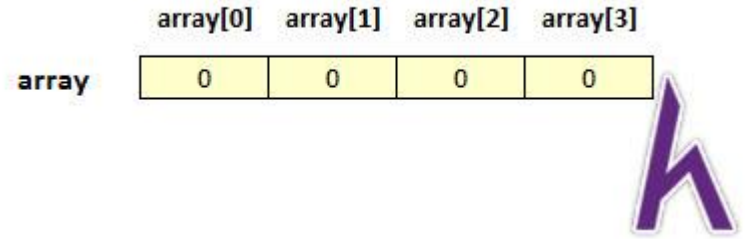
Cách 2: Khởi tạo giá trị cho một số phần tử đầu mảng

```
int array[4] = { 5, 8 };
```



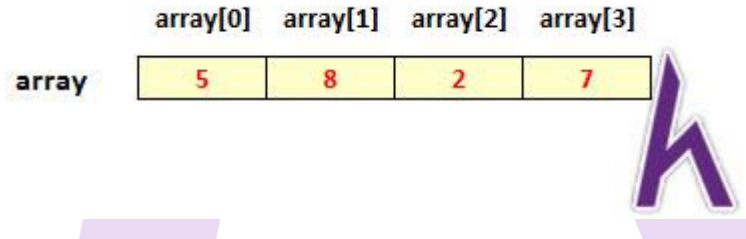
Cách 3: Khởi tạo giá trị 0 cho mọi phần tử của mảng

```
int array[4] = { };
```



Cách 4: Tự động xác định số lượng phần tử

```
int array[] = { 5, 8, 2, 7 };
```



Cách 5: Sử dụng **khởi tạo đồng nhất (uniform initialization)** trong **C++11**

```
int array1[4] { 5, 8, 2, 7 }; // 5 8 2 7
int array2[4] { 5, 8 }; // 5 8 0 0
int array3[4] { }; // 0 0 0 0
int array4[] { 5, 8, 2, 7 }; // 5 8 2 7
```

Xuất các phần tử mảng 1 chiều

Để **truy xuất giá trị của phần tử** trong mảng, ta sử dụng **cú pháp**:

> **<tên biến mảng>**[**<chỉ số thứ i>**];

Trong đó:

- **<chỉ số thứ i> là chỉ số phần tử** trong mảng.
- Nếu mảng có N phần tử, **<chỉ số thứ i>** sẽ nằm trong khoảng từ **0 đến N – 1**.

Ví dụ: Cho mảng như sau:

```
int array[4] { 5, 8, 2, 7 }; // 5 8 2 7
```

Các truy xuất:

- **Hợp lệ:** array[0], array[1], array[2], array[3]
- **Không hợp lệ:** array[-1], array[4], array[5], ... => cho kết quả không như mong muốn (có thể gây chết chương trình).

> **Chú ý**: khi truy xuất một phần tử mảng, luôn đảm bảo chỉ số của phần tử đó là hợp lệ trong phạm vi của mảng.

Ví dụ chương trình khởi tạo và xuất các phần tử mảng:

```
#include <iostream>
#include <string>
using namespace std;

// định nghĩa số phần tử mảng
#define MAX 3

int main()
{
	// khởi tạo mảng string 3 phần tử
```

```
	string arrKteam[MAX]{ "Hello Howkteam.com!", "Free Education", "Share to be better" };

	// xuất giá trị các phần tử mảng
	for (int i = 0; i < MAX; i++)
	{
		cout << arrKteam[i] << endl;
	}

	return 0;
}
```

Output:

Chương trình trên sử dụng vòng lặp for, chạy từ **0** đến **MAX – 1** để truy cập vào từng phần tử trong mảng.

Nhập dữ liệu cho mảng 1 chiều

Để **gán giá trị cho phần tử** trong mảng, ta sử dụng **cú pháp**:

> **<tên biến mảng>**[**<chỉ số thứ i>**] = **<giá trị>**;

Trong đó:

- **<chỉ số thứ i> là chỉ số phần tử** trong mảng.
- Nếu mảng có N phần tử, **<chỉ số thứ i>** sẽ nằm trong khoảng từ **0 đến N – 1**.

Ví dụ các phép gán hợp lệ:

```
string arrKteam[3];
arrKteam[0] = "Hello Howkteam.com!";
arrKteam[1] = "Free Education";
arrKteam[2] = "Share to be better";
```

Ví dụ các phép gán **KHÔNG hợp lệ** (gây chết chương trình):

```
string arrKteam[3];
arrKteam[-1] = "Hello Howkteam.com!";
arrKteam[4] = "Free Education";
arrKteam[5] = "Share to be better";
```

> **Chú ý**: khi truy cập một phần tử mảng, luôn đảm bảo chỉ số của phần tử đó là hợp lệ trong phạm vi của mảng.

Ví dụ chương trình yêu cầu nhập dữ liệu cho 1 mảng, sau đó xuất ra màn hình:

```
#include <iostream>
#include <string>
using namespace std;

// định nghĩa số phần tử mảng
#define MAX 3

int main()
{
	int arr[MAX];

	// nhập mảng
	cout << "Array input:" << endl;
	for (int i = 0; i < MAX; i++)
	{
		cout << "arr[" << i << "] = ";
		cin >> arr[i];
	}

	// xuất mảng
	for (int i = 0; i < MAX; i++)
	{
```

```
                cout << arr[i] << " ";
        }

        return 0;
}
```

Output:

Phát sinh dữ liệu ngẫu nhiên cho mảng 1 chiều

Trong quá trình học tập hoặc làm việc, có thể bạn cần 1 mảng gồm rất nhiều phần tử (ví dụ: mảng số nguyên 1000 phần tử, ...), và bạn không thể nhập giá trị cho từng phần tử được.

Lúc này, bạn có thể áp dụng phương pháp PHÁT SINH SỐ NGẪU NHIÊN (Random number generation) đã được giới thiệu trong bài học trước để tạo ra những phần tử có giá trị ngẫu nhiên.

Ví dụ chương trình phát sinh số ngẫu nhiên cho mảng 1 chiều:

```
#include <iostream>
#include <cstdlib> // for srand() and rand()
#include <ctime>          // for tine()
using namespace std;

// định nghĩa số phần tử mảng
#define MAX 5

int main()
{
```

```
	int arr[MAX];

	// khởi tạo số ngẫu nhiên
	srand(time(NULL));

	// nhập mảng ngẫu nhiên
	for (int i = 0; i < MAX; i++)
	{
		arr[i] = rand();
	}

	// xuất mảng
	for (int i = 0; i < MAX; i++)
	{
		cout << "arr[" << i << "] = " << arr[i] << endl;
	}

	return 0;
}
```

Output:

Kết luận

Qua bài học này, bạn đã biết được khái niệm và cách sử dụng Mảng 1 chiều trong C++ (Arrays). Mảng 1 chiều đã giải quyết được vấn đề về quản lý hàng loạt biến có cùng kiểu dữ liệu. Nó là một cách tổ chức kiểu dữ liệu mới, và là tiền đề để xây dựng lên những kiểu dữ liệu danh sách về sau.

Trong bài tiếp theo, mình sẽ giới thiệu cho các bạn CÁC THAO TÁC TRÊN MẢNG 1 CHIỀU TRONG C++.

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".