**Bài 6: Mảng Tĩnh (Static Arrays) Trong C++**

**I. Giới Thiệu Về Mảng**

* **Khái Niệm:** Mảng là cấu trúc dữ liệu lưu trữ nhiều giá trị cùng loại trong một biến duy nhất. Mảng giúp tổ chức dữ liệu và truy cập dễ dàng, đặc biệt hữu ích khi làm việc với nhiều giá trị giống nhau.

**II. Khai Báo và Khởi Tạo Mảng**

a. **Khai Báo Mảng**

* **Cú pháp:**

type arrayName[arraySize];

* + **type:** Kiểu dữ liệu (int, float, double, char, v.v.).
  + **arrayName:** Tên mảng.
  + **arraySize:** Số lượng phần tử mảng có thể lưu trữ.

b. **Khởi Tạo Mảng**

* **Khai Báo và Khởi Tạo Đồng Thời:**

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

* **Khai Báo và Khởi Tạo Sau:**

int numbers[5];

numbers[0] = 1;

numbers[1] = 2;

numbers[2] = 3;

numbers[3] = 4;

numbers[4] = 5;

**Bài Tập Thực Hành:**

1. Khai báo một mảng số nguyên có kích thước 10 và khởi tạo tất cả các phần tử bằng cách sử dụng vòng lặp for. In ra các phần tử của mảng.

**Bài Tập Thực Hành:** 2. Khai báo một mảng ký tự với 10 phần tử và khởi tạo mảng bằng các ký tự khác nhau (như 'A', 'B', 'C', v.v.). Viết một chương trình để in ra tất cả các ký tự trong mảng.

**III. Truy Cập và Sử Dụng Các Phần Tử Của Mảng**

a. **Truy Cập Phần Tử**

* **Ví dụ:**

int arr[5] = {10, 20, 30, 40, 50};

cout << arr[0]; // In ra 10

b. **Sử Dụng Vòng Lặp Với Mảng**

* **Ví dụ:**

for (int i = 0; i < 5; ++i) {

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

}

**Bài Tập Thực Hành:**

1. Viết một chương trình để truy cập và in ra phần tử ở vị trí giữa của một mảng số nguyên có kích thước lẻ (ví dụ: 7 phần tử).

**Bài Tập Thực Hành:** 2. Viết một chương trình để thay đổi giá trị của một phần tử ở chỉ số 3 của một mảng số thực. In ra toàn bộ mảng sau khi thực hiện thay đổi.

**IV. Mảng Hai Chiều**

* **Khái Niệm:** Mảng hai chiều là một mảng của các mảng, thường dùng để biểu diễn bảng hoặc ma trận.

a. **Khai Báo và Khởi Tạo Mảng Hai Chiều**

* **Cú pháp:**

type arrayName[rowSize][columnSize];

b. **Truy Cập Phần Tử Của Mảng Hai Chiều**

* **Ví dụ:**

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

cout << matrix[1][2]; // In ra 6

**Bài Tập Thực Hành:**

1. Tạo một mảng hai chiều 3x3 và khởi tạo nó với các giá trị từ 1 đến 9. Viết một chương trình để in ra toàn bộ ma trận theo định dạng bảng.

**Bài Tập Thực Hành:** 2. Viết một chương trình để tính tổng của tất cả các phần tử trong một mảng hai chiều 2x4. In ra tổng và ma trận.

**V. Mảng Trong Bộ Nhớ**

a. **Cấu Trúc Bộ Nhớ của Mảng**

* **Địa Chỉ Bộ Nhớ:** Các phần tử của mảng được lưu trữ liên tiếp trong bộ nhớ. Địa chỉ của phần tử đầu tiên gọi là địa chỉ cơ sở của mảng.
* **Truy Cập Phần Tử:** Địa chỉ phần tử thứ i:

Địa chỉ của phần tử thứ i = Địa chỉ cơ sở + (i \* kích thước của mỗi phần tử)

**VI. Mảng Ký Tự và Mảng String**

a. **Mảng Ký Tự (Character Arrays)**

* **Khái Niệm:** Mảng ký tự lưu trữ chuỗi ký tự.
* **Cú Pháp:** Khai báo và khởi tạo mảng ký tự, ví dụ:

char str[] = "Hello";

* **Ví dụ về Mảng Ký Tự Không Có '\0':**

#include <iostream>

using namespace std;

int main() {

char str[6] = {'H', 'e', 'l', 'l', 'o', ' '}; // Không có ký tự '\0' ở cuối

// In ra mảng ký tự

cout << "Mảng ký tự không có '\\0': ";

for (int i = 0; i < 6; ++i) {

cout << str[i];

}

cout << endl;

// Tính chiều dài của chuỗi

// Đây có thể gây ra hành vi không xác định vì không có ký tự '\0'

cout << "Chiều dài của chuỗi: " << strlen(str) << endl;

return 0;

}

* **Ví dụ về Mảng Ký Tự Với '\0':**

#include <iostream>

#include <cstring> // Để sử dụng strlen

using namespace std;

int main() {

char str[6] = {'H', 'e', 'l', 'l', 'o', '\0'}; // Ký tự '\0' ở cuối

// In ra mảng ký tự

cout << "Chuỗi ký tự với '\\0': " << str << endl;

// Tính chiều dài của chuỗi

cout << "Chiều dài của chuỗi: " << strlen(str) << endl;

return 0;

}

**Bài Tập Thực Hành:**

1. Viết một chương trình để khởi tạo một mảng ký tự với một chuỗi ký tự và sau đó in ra chuỗi ký tự đó. Thực hiện điều này cả với ký tự '\0' và không có ký tự '\0' để thấy sự khác biệt.

**Bài Tập Thực Hành:** 2. Tạo một mảng ký tự chứa một chuỗi ký tự và viết một chương trình để tìm số lượng ký tự 'a' trong chuỗi. Sử dụng vòng lặp và điều kiện để thực hiện.

**VII. So Sánh Mảng Ký Tự và std::string**

a. **Quản Lý Bộ Nhớ**

* **Mảng Ký Tự:** Quản lý bộ nhớ thủ công, đảm bảo mảng ký tự kết thúc bằng ký tự null ('\0').

b. **Tại Sao '\0' Quan Trọng?**

* **Đánh Dấu Kết Thúc Chuỗi:** Ký tự '\0' đánh dấu điểm kết thúc của chuỗi ký tự. Các hàm xử lý chuỗi dựa vào '\0' để biết khi nào dừng đọc chuỗi.
* **Tránh Lỗi:** Nếu không có ký tự '\0', các hàm có thể đọc vùng bộ nhớ không được định nghĩa, gây ra hành vi không mong muốn hoặc lỗi bảo mật.

c. **std::string**

* **Khái Niệm:** Lớp trong thư viện chuẩn C++ (STL) dùng để quản lý chuỗi ký tự.
* **Cú Pháp:** Khai báo và khởi tạo chuỗi, ví dụ:

std::string str = "Hello";

* **Đặc Điểm:** Cung cấp nhiều phương thức hữu ích như append, substr, find, replace, v.v.

d. **Lợi Ích và Nhược Điểm**

* **Lợi Ích của std::string:**
  + Dễ sử dụng và quản lý.
  + Cung cấp nhiều phương thức tiện ích.
  + Quản lý bộ nhớ tự động.
* **Nhược Điểm của std::string:**
  + Có thể sử dụng nhiều bộ nhớ hơn so với mảng ký tự nếu không cần tất cả các tính năng.
* **Lợi Ích của Mảng Ký Tự:**
  + Hiệu suất cao trong các ứng dụng yêu cầu xử lý chuỗi cấp thấp.
  + Tinh chỉnh bộ nhớ tốt hơn trong một số trường hợp cụ thể.
* **Nhược Điểm của Mảng Ký Tự:**
  + Quản lý bộ nhớ phức tạp hơn.
  + Thiếu các phương thức tiện ích.