## Bài 8: Tham Chiếu, Phạm Vi Biến, Namespace, và Vòng Lặp Kiểu Phạm Vi trong C++

### **1. Tham Chiếu (References)**

**Khái Niệm:**

* Tham chiếu là một biến đặc biệt trong C++, không chứa giá trị mà chứa địa chỉ của một biến khác.
* Tham chiếu phải được khởi tạo khi khai báo và không thể thay đổi để tham chiếu đến biến khác.

**Cú Pháp Cơ Bản:**

#include <iostream>

int main() {  
    int a = 5;        // Biến a  
    int& ref = a;     // Tham chiếu ref đến biến a

    std::cout << "Giá trị của a: " << a << std::endl;    // In ra 5  
    std::cout << "Giá trị qua tham chiếu ref: " << ref << std::endl; // In ra 5

    ref = 10;         // Thay đổi giá trị của biến qua tham chiếu

    std::cout << "Giá trị của a sau khi thay đổi qua ref: " << a << std::endl; // In ra 10  
    std::cout << "Giá trị qua tham chiếu ref sau khi thay đổi: " << ref << std::endl; // In ra 10

    return 0;  
}

**Bài Tập 1:**

1. Tạo một chương trình C++ với hai biến số nguyên, x và y. Khai báo một tham chiếu ref để tham chiếu đến x.
2. Sử dụng tham chiếu ref để thay đổi giá trị của x. In ra giá trị của x và y trước và sau khi thay đổi thông qua ref.

**Bài Tập 2:**

1. Viết một chương trình C++ có hai biến số thực, num1 và num2. Tạo một tham chiếu refNum để tham chiếu đến num1.
2. Sử dụng refNum để nhân đôi giá trị của num1. In ra giá trị của num1 và num2 trước và sau khi thay đổi thông qua refNum.

### **2. Phạm Vi Biến (Variable Scope)**

**Khái Niệm:**

* Phạm vi của biến chỉ định nơi trong chương trình mà biến có thể được truy cập.
* Biến cục bộ chỉ có thể được truy cập trong phạm vi của nó, trong khi biến toàn cục có thể được truy cập từ bất kỳ đâu trong chương trình.

**Ví Dụ Cơ Bản:**

#include <iostream>

int globalVar = 100;  // Biến toàn cục

int main() {  
    int localVar = 10;  // Biến cục bộ

    std::cout << "Giá trị của biến toàn cục globalVar: " << globalVar << std::endl; // In ra 100  
    std::cout << "Giá trị của biến cục bộ localVar: " << localVar << std::endl;    // In ra 10

    {  
        int blockVar = 20;  // Biến cục bộ trong khối mã  
        std::cout << "Giá trị của biến cục bộ trong khối blockVar: " << blockVar << std::endl; // In ra 20  
    }

    // std::cout << blockVar;  // Lỗi: blockVar không thể truy cập ở đây

    return 0;  
}  
 Lưu ý:

 **Ưu Tiên Phạm Vi**: Biến cục bộ che khuất biến toàn cục trong phạm vi của nó. Biến cục bộ với tên trùng lặp sẽ được ưu tiên sử dụng.

 **Sử Dụng Toán Tử Phạm Vi Toàn Cục**: Để truy cập biến toàn cục trong khi có một biến cục bộ cùng tên, bạn cần sử dụng toán tử phạm vi toàn cục ::.

**Bài Tập 1:**

1. Viết một chương trình C++ có một biến toàn cục global và một biến cục bộ local trong hàm main().
2. Khai báo một biến cục bộ block trong một khối mã bên trong main(). In ra giá trị của cả ba biến (toàn cục, cục bộ trong main(), và cục bộ trong khối mã). Thử truy cập biến block từ ngoài khối mã để quan sát lỗi biên dịch.

**Bài Tập 2:**

1. Viết một chương trình C++ với biến toàn cục total và biến cục bộ amount trong hàm main().
2. Trong một khối mã phụ (như trong một vòng lặp hoặc một khối điều kiện), khai báo một biến cục bộ temp. Tính tổng của amount và temp, sau đó in kết quả ra màn hình. Lưu ý rằng bạn không thể truy cập temp ngoài khối mã đó.

### **3. Namespace (Không Gian Tên)**

**Khái Niệm:**

* Namespace giúp tổ chức mã nguồn và tránh xung đột tên bằng cách nhóm các biến, hàm, lớp liên quan lại với nhau.

**Ví Dụ Cơ Bản:**

#include <iostream>

namespace MyNamespace {  
    int myVar = 42;   // Biến trong namespace  
}

int main() {  
    std::cout << "Giá trị của biến trong namespace MyNamespace: " << MyNamespace::myVar << std::endl; // In ra 42

    return 0;  
}

**Sử Dụng using Để Rút Gọn Cú Pháp:**

#include <iostream>

namespace MyNamespace {  
    int myVar = 42;   // Biến trong namespace  
}

int main() {  
    using namespace MyNamespace;  
      
    std::cout << "Giá trị của biến trong namespace MyNamespace: " << myVar << std::endl; // In ra 42

    return 0;  
}

**Bài Tập 1:**

1. Tạo một namespace có tên Utilities và khai báo một biến số nguyên value trong namespace đó.
2. Trong hàm main(), sử dụng cú pháp Utilities::value để in ra giá trị của biến value. Thay đổi giá trị của biến và in lại để xác nhận rằng bạn đã thay đổi đúng biến.

**Bài Tập 2:**

1. Tạo một namespace có tên MathOps với một hàm có tên add nhận hai tham số và trả về tổng của chúng.
2. Trong hàm main(), sử dụng MathOps::add để tính tổng của hai số và in kết quả ra màn hình.

### **4. Vòng Lặp Kiểu Phạm Vi (Range-Based For Loop)**

**Khái Niệm:**

* Vòng lặp kiểu phạm vi là cách đơn giản và tiện lợi để lặp qua các phần tử của một container (như array, list, v.v.).

**Ví Dụ Cơ Bản Với array:**

#include <iostream>

int main() {

int ages[6] = {15, 18, 20, 22, 19, 17};

int count = 0;

// Vòng lặp kiểu phạm vi để đếm số phần tử

for (int age : ages) {

count++;

}

std::cout << "Số lượng phần tử trong mảng ages: " << count << std::endl; // In ra số lượng

return 0;

}

**Bài Tập 1:**

* Viết một chương trình C++ khai báo một mảng scores chứa 5 số nguyên: 85, 90, 78, 92, 88.
* Sử dụng vòng lặp kiểu phạm vi để in tất cả các giá trị trong mảng scores.

**Bài Tập 2:**

* Viết một chương trình C++ khai báo một mảng temperatures chứa 4 số thực: 23.5, 25.0, 22.8, 24.3.
* Sử dụng vòng lặp kiểu phạm vi để tính tổng của tất cả các giá trị trong mảng temperatures và in tổng ra màn hình.

**Bài Tập 3:**

* Viết một chương trình C++ khai báo một mảng numbers chứa 5 số nguyên: 1, 2, 3, 4, 5.
* Sử dụng vòng lặp kiểu phạm vi để thay đổi giá trị của mỗi phần tử trong mảng numbers thành giá trị gấp đôi giá trị hiện tại của nó.
* In tất cả các giá trị của mảng sau khi thay đổi.

**Bài Tập 4:**

* Viết một chương trình C++ khai báo một mảng values chứa 4 số thực: 10.0, 20.5, 30.8, 40.3.
* Sử dụng vòng lặp kiểu phạm vi để giảm giá trị của mỗi phần tử trong mảng values xuống 5.0.
* In tất cả các giá trị của mảng sau khi giả