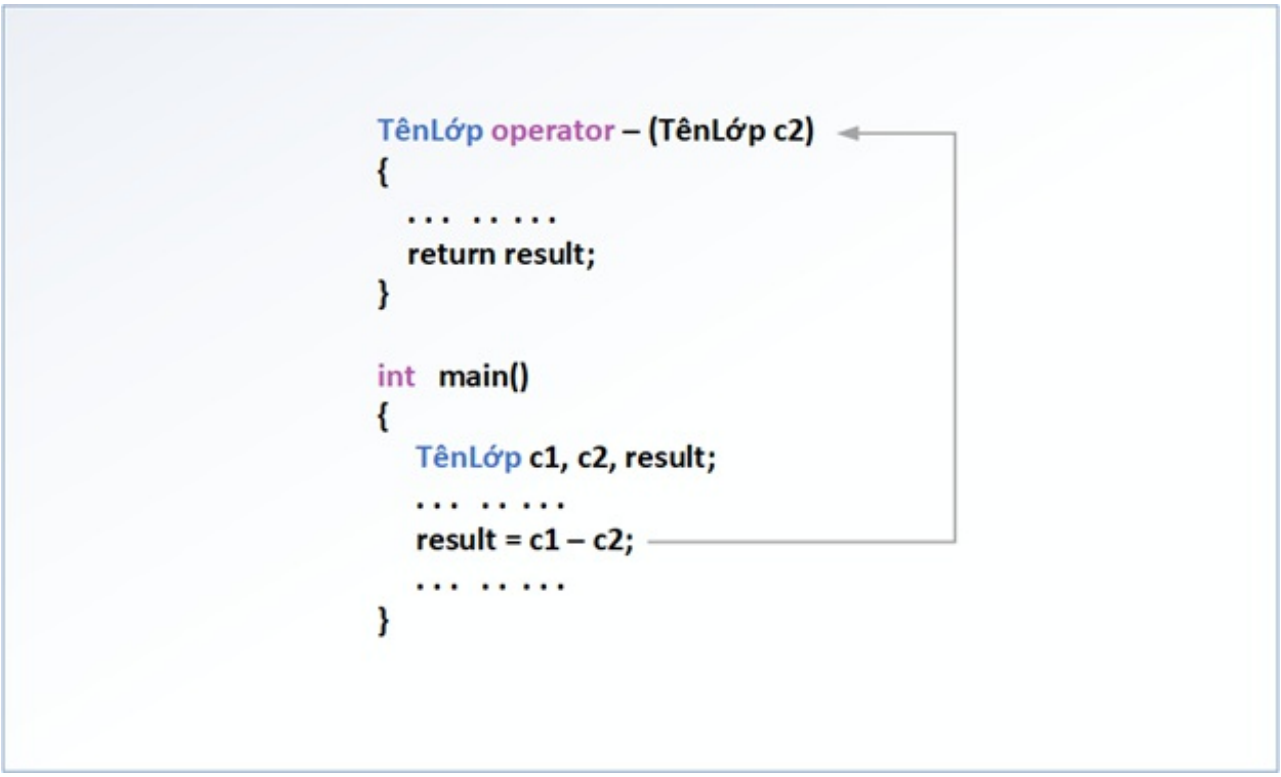
Nạp chồng toán tử trong C++

Trong C++, hoàn toàn có thể thay đổi cách toán tử hoạt động (đối với các kiểu người dùng định nghĩa). Trong bài này, bạn sẽ được học cách lập trình tính năng nạp chồng toán tử.



Ý nghĩa của một toán tử luôn luôn là như nhau đối với các biến thuộc kiểu cơ bản như int, float, double…Ví dụ khi cộng hai số nguyên, toán tử + được sử dụng.

Tuy nhiên, đối với kiểu người dùng định nghĩa (như đối tượng), bạn có thể định nghĩa lại cách toán tử hoạt động. Ví dụ:

Nếu có hai đối tượng của một lớp chứa chuỗi là dữ liệu thành viên. Bạn có thể định nghĩa lại ý nghĩa của toán tử + và dùng nó để nối các chuỗi đó.

Tính năng trong lập trình C++ cho phép lập trình viên định nghĩa lại ý nghĩa của một toán tử (khi họ thao tác trên các đối tượng lớp) được gọi là nạp chồng toán tử.

Tại sao nạp chồng toán tử được sử dụng?

Bạn có thể viết bất kỳ chương trình C++ nào mà không cần biết nạp chồng toán tử. Tuy nhiên, nạp chồng toán tử được các lập trình viên rất hay sử dụng để khiến chương trình trở nên có nghĩa hơn. Ví dụ:

Bạn có thể thay thế đoạn mã nguồn dưới đây:

```
calculation = add(multiply(a, b),divide(a, b));
```

thành

```
calculation = (a*b)+(a/b);
```

Làm thế nào để nạp chồng toán tử trong lập trình C++?

Để nạp chồng một toán tử, một hàm toán tử đặc biệt cần được định nghĩa trong lớp như sau:

```
class tên_lớp
{
   ... .. ...
   public
     kiểu_trả_về operator ký_hiệu (đối_số)
     {
         ... .. ...
     }
   ... .. ...
};
```

- Ở đây, kiểu_trả_về là kiểu giá trị hàm sẽ trả về
- kiểu_trả_về của hàm được theo sau bởi từ khóa operator
- ký_hiệu là ký hiệu của toán tử bạn muốn nạp chồng. Ví dụ: +, <, -, ++
- Bạn có thể truyền vào tham số cho hàm toán tử tương tự như với các hàm khác.

Ví dụ: Nạp chồng toán tử trong lập trình C++

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

class Test
{
   private:
      int count;

   public:
      Test(): count(5){}

      void operator ++()
      {
         count = count+1;
      }
      void Display() { cout<<"Count: "<<count; }
};

int main()
{
    Test t;
    // câu lệnh này sẽ gọi hàm "void operator ++()"
    ++t;
    t.Display();
    return 0;
}
```

Đầu ra

```
Count: 6
```

Hàm này được gọi khi toán tử `++` thao tác trên các đối tượng của lớp `Test` (đối tượng `t` trong trường hợp này).

Trong chương trình, hàm toán tử `void operator ++ ()` được định nghĩa (trong lớp Test).

Hàm này sẽ tăng giá trị của `count` lên 1 cho đối tượng `t`.

Những điều cần ghi nhớ

1. Nạp chồng toán tử cho phép bạn định nghĩa lại cách mà toán tử hoạt động chỉ đối với các kiểu do người dùng định nghĩa (đối tượng, cấu trúc). Nó không thể được sử dụng cho các kiểu dựng sẵn (int, float, char…).
2. Hai toán tử = và & đã mặc định được nạp chồng trong C++. Ví dụ: để sao chép đối tượng thuộc cùng lớp, bạn có thể trực tiếp sử dụng toán tử =. Bạn không cần tạo một hàm toán tử.
3. Nạp chồng toán tử không thể thay đổi thứ tự thực hiện và khả năng kết hợp của các toán tử. Tuy nhiên, nếu bạn muốn thay đổi thứ tự của việc đánh giá, bạn có thể sử dụng dấu ngoặc.
4. Có 4 toán tử không thể bị nạp chồng trong C++. Chúng là:: (phân giải phạm vi), . (lựa chọn thành viên), .* (lựa chọn thành viên thông qua con trỏ tới hàm) và?: (toán tử tam nguyên).

Những trường hợp khuyến khích sử dụng khi dùng nạp chồng toán tử

Nạp chồng toán tử cho phép bạn định nghĩa cách toán tử hoạt động (theo cách bạn muốn).

Trong ví dụ trên, toán tử++ sẽ thao tác trên đối tượng để tăng giá trị của dữ liệu thành viên count lên 1.

```
void operator ++()
   {
      count = count+1;
   }
```

Tuy nhiên, nếu bạn sử dụng đoạn mã sau. Nó sẽ giảm giá trị của count đi 100 khi toán tử ++ được sử dụng.

```
void operator ++()
   {
      count = count-100;
   }
```

Về mặt kỹ thuật, đây là câu lệnh hợp cách. Nhưng đoạn mã này sẽ khiến việc hiểu và gỡ lỗi trở nên rắc rối và phức tạp hơn.

Công việc của lập trình viên đó là sử dụng nạp chồng toán tử một cách hợp lý và đồng nhất.

Trong ví dụ trên, giá trị củacount được tăng lên 1 khi toán tử++ được sử dụng. Tuy nhiên, chương trình này lại không hoàn thiện vì bạn sẽ không thể sử dụng đoạn mã như:

```
t1 = ++t
```

Đó là do kiểu trả về của hàm toán tử làvoid.