Operator Overloading in C++

In C++, we can make operators to work for user defined classes. This means C++ has the ability to provide the operators with a special meaning for a data type, this ability is known as operator overloading.
For example, we can overload an operator '+' in a class like String so that we can concatenate two strings by just using +.
Other example classes where arithmetic operators may be overloaded are Complex Number, Fractional Number, Big Integer, etc.

A simple and complete example

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

class Complex {
private:
    int real, imag;
public:
    Complex(int r = 0, int i =0)  {real = r;   imag = i;}

    // This is automatically called when '+' is used with
    // between two Complex objects
    Complex operator + (Complex const &obj) {
         Complex res;
         res.real = real + obj.real;
         res.imag = imag + obj.imag;
         return res;
    }
    void print() { cout << real << " + i" << imag << endl; }
};

int main()
{
    Complex c1(10, 5), c2(2, 4);
    Complex c3 = c1 + c2; // An example call to "operator+"
    c3.print();
}
```

Output:

```
12 + i9
```

What is the difference between operator functions and normal functions?
Operator functions are same as normal functions. The only differences are, name of an operator function is always operator keyword followed by symbol of operator and operator functions are called when the corresponding operator is used.
Following is an example of global operator function.

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

class Complex {
private:
    int real, imag;
public:
    Complex(int r = 0, int i =0)  {real = r;   imag = i;}
    void print() { cout << real << " + i" << imag << endl; }

// The global operator function is made friend of this class so
// that it can access private members
friend Complex operator + (Complex const &, Complex const &);
};


Complex operator + (Complex const &c1, Complex const &c2)
{
     return Complex(c1.real + c2.real, c1.imag + c2.imag);
}


int main()
{
    Complex c1(10, 5), c2(2, 4);
    Complex c3 = c1 + c2; // An example call to "operator+"
    c3.print();
    return 0;
}
```

Can we overload all operators?

Almost all operators can be overloaded except few. Following is the list of operators that cannot be overloaded.

```
    . (dot)
    ::
    ?:
    sizeof
```

Why can't . (dot), ::, ?: and sizeof be overloaded?

See this for answers from Stroustrup himself.

Important points about operator overloading

1) For operator overloading to work, at leas one of the operands must be a user defined class object.

2) Assignment Operator: Compiler automatically creates a default assignment operator with every class. The default assignment operator does assign all members of right side to the left side and works fine most of the cases (this behavior is same as copy constructor). See this for more details.

3) Conversion Operator: We can also write conversion operators that can be used to convert one type to another type.

```
#include <iostream>
using namespace std;
class Fraction
{
    int num, den;
public:
    Fraction(int n,  int d) { num = n; den = d; }

    // conversion operator: return float value of fraction
    operator float() const {
        return float(num) / float(den);
    }
};

int main() {
    Fraction f(2, 5);
    float val = f;
    cout << val;
    return 0;
}
```

Output:

```
0.4
```

Overloaded conversion operators must be a member method. Other operators can either be member method or global method.

4) Any constructor that can be called with a single argument works as a conversion constructor, means it can also be used for implicit conversion to the class being constructed.

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

class Point
{
private:
    int x, y;
public:
    Point(int i = 0, int j = 0) {
        x = i;   y = j;
    }
    void print() {
        cout << endl << " x = " << x << ", y = " << y;
    }
};

int main() {
    Point t(20, 20);
    t.print();
    t = 30;   // Member x of t becomes 30
    t.print();
    return 0;
}
```

Output:

```
x = 20, y = 20
x = 30, y = 0
```

We will soon be discussing overloading of some important operators like new, delete, comma, function call, arrow, etc.

Quiz on Operator Overloading

References:
http://en.wikipedia.org/wiki/Operator_overloading

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Recommended Posts:

Overloading New and Delete operator in c++
Types of Operator Overloading in C++
Rules for operator overloading
C++ | Operator Overloading | Question 10
Increment (++) and Decrement (--) operator overloading in C++
C++ Program to concatenate two strings using Operator Overloading
Overloading Subscript or array index operator [] in C++
C++ program to compare two Strings using Operator Overloading
Operator overloading in C++ to print contents of vector, map, pair, ..
Why overriding both the global new operator and the class-specific operator is not ambiguous?
Function Overloading in C++
Constructor Overloading in C++
Overloading in Java

Constructor Overloading in Java

Function Overloading and float in C++

Improved By : routpranab03

Bücher einfach drucken

Hardcover, Fadenheftungen, Klebebindungen, einfach und schnell fertigen lassen

Article Tags : C++ School Programming cpp-operator-overloading cpp-overloading

Practice Tags : CPP

16

2.9

☐ To-do ☐ Done

Based on **37** vote(s)

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments