

[Logging in From a Linux System or Localhost](#)
4 days 45 min ago

[Auto response](#)
4 days 6 hours ago

[Re: final issue.](#)
4 days 22 hours ago

[Re: Rocky, thank you for this](#)
4 days 23 hours ago

Newsletter

Subscribe to HowtoForge Newsletter and stay informed about our latest HOWTOs and projects.

enter email address:

Submit

(To unsubscribe from our newsletter, visit this [link](#).)

You are here: [Home](#) » [Learning C/C++ Step-By-Step](#) » [Learning C/C++ Step-By-Step - Page 15](#)

Learning C/C++ Step-By-Step - Page 15

Want to support HowtoForge? Become a [subscriber](#)!

Submitted by [ganesh35](#) ([Contact Author](#)) ([Forums](#)) on Wed, 2009-01-07 18:50. ::

0

Like

2

Send

Tweet

0

15. Step-by-Step C/C++ --- C++ Programming - Operator Overloading

Operator Overloading

1. Introduction
2. Operator
 - Rules of Operator Overloading
 - Restrictions on Operator Overloading
3. Overloading Unary Operators
4. Overloading Binary Operators
5. Operator Overloading with Strings

1. Introduction

```
// Assign a variable to another
#include <iostream>
using namespace std;
int main()
{
    int a = 10, b;
    b = a;           // valid
    cout << b;
    return 0;
}
```

```
// Assign an object to another
#include <iostream>
using namespace std;
class emp
{
public:
    int eno;
    float sal;
};
int main()
{
    emp e1= { 1001, 2300.45 },e2 ;
    cout << endl << e1.eno << e1.sal;
    e2 = e1;          // valid
    cout << endl << e2.eno << e2.sal;
    return 0;
}
```

Expressions are common in every language; an expression is a collection of operands and operators. Where as an operation is a collection of expressions. The above two programs demonstrate how variables/objects were assigned together.

Both programs are valid, they demonstrates the use of equalto (=) operator.

```
// using operator + to perform an arithmetic operation with variables
#include <iostream>
using namespace std;
int main()
{
    int a = 10, b = 15, c;
    c = a + b; // valid expression
    cout << c;
    return 0;
}

// using operator + to perform an arithmetic operation with objects
#include <iostream>
using namespace std;
class emp
{
    public:
    int eno;
    float sal;
};
int main()
{
    emp e1 = { 1001, 2300.45 }, e2 = e1, e3;
    cout << endl << e1.eno << e1.sal;
    e3 = e1 + e2; // Illegal structure operation
    cout << endl << e3.eno << e3.sal;
}
```

Operator overloading is one of the most exciting feature of object-oriented programming. It is used to overcome the situation like the above **illegal structure operation**. It can transform complex, obscure program listing into intuitively obvious ones.

Through Operator overloading we can see how the normal C++ operators can be given new meanings when applied to user-defined data types. The keyword `operator` is used to overload an operator, and the resulting operator will adopt the meaning supplied by the programmer.

For example using object we can perform direct string assignment operation.

```
// Program to assign a string to other
#include <string.h>
#include <stdio.h>
#include <iostream>
using namespace std;
class string
{
    char *str;
    public:
    string() { }
    string(char *s) { str = s; }
    void putstring()
    {
        cout << str;
    }
};
int main()
{
    string s1("Computer");
    string s2;
    s2 = s1;
    s2.putstring();
    return 0;
}
```

2. Operator

type **operator** *operator-symbol* (*parameter-list*)

The **operator** keyword declares a function specifying what *operator-symbol* means when applied to instances of a class. This gives the operator more than one meaning, or "overloads" it. The compiler distinguishes between the different meanings of an operator by examining the types of its operands.

Rules of Operator Overloading

- You can overload the following operators:


+	-	*	/	%	^
!	=	<	>	+=	-=
^=	&=	=	<<	>>	<<=
<=	>=	&&		++	--
()	[]	new	delete	&	
~	*=	/=	%=	>>=	==
!=	,	->	->*		

- If an operator can be used as either a unary or a binary operator, you can overload each use separately.
- You can overload an operator using either a non-static member function or a global function that's a friend of a class. A global function must have at least one parameter that is of class type or a reference to class type.
- If a unary operator is overloaded using a member function, it takes no arguments. If it is overloaded using a global function, it takes one argument.
- If a binary operator is overloaded using a member function, it takes one argument. If it is overloaded using a global function, it takes two arguments.

Restrictions on Operator Overloading

- You cannot define new operators, such as `**`.
- You cannot change the precedence or grouping of an operator, nor can you change the numbers of operands it accepts.
- You cannot redefine the meaning of an operator when applied to built-in data types.
- Overloaded operators cannot take default arguments.
- You cannot overload any preprocessor symbol, nor can you overload the following operators:

`.` `.*` `::`



“Monitor your Network with PRTG. Quick & Easy!”

1st class monitoring for

Freewa

?:

The assignment operator has some additional restrictions. It can be overloaded only as a non-static member function, not as a friend function. It is the only operator that cannot be inherited; a derived class cannot use a base class's assignment operator.

3. Overloading Unary Operators

Let's start off by overloading a **unary operator**. Unary operators act on only one operand. (An operand is simply a variable acted on by an operator). Examples of unary operators are the increment and decrement operators **++** and **--**, and the unary minus.

Example:

The following example demonstrates the use of increment operator **++**.

```
#include <iostream>
using namespace std;
class counter
{
    private:
        unsigned int count;
    public:
        counter(){ count = 0; }
        int get_count() { return count; }
        counter operator ++()
        {
            count++;
            counter temp;
            temp.count = count;
            return temp;
        }
};

int main()
{
    counter c1, c2;
    cout << "\nC1 = " << c1.get_count(); // c1 = 0, c2 = 0
    cout << "\nC2 = " << c2.get_count(); // display

    ++c1;
    c2 = ++c1; // c1 = 1, c2 = 2
               // c1 = 2, c2 = 2

    cout << "\nC1 = " << c1.get_count(); // display again
    cout << "\nC2 = " << c2.get_count(); // c2 = 3
    return 0;
}
```

One more example to overloading unary minus.

```
#include <iostream>
using namespace std;
class subtract
{
    int a;
    int b;
    public:
        void getdata(int x, int y)
        {
            a = x; b = y;
        }
        void putdata()
        {
            cout << endl << "A = " << a << "B = " << b;
        }
        void operator -()
        {
            a = -a; b = -b;
        }
};

int main()
{
    subtract s;
    s.getdata(34, -6);
    cout << endl << "S : ";
    s.putdata();
    -s;
    cout << endl << "S : ";
    s.putdata();
    return 0;
}
```

4. Overloading Binary Operators

But operators can be overloaded just as easily as unary operators. We will look at examples that overload arithmetic operators, comparison operators, and arithmetic assignment operators.

We have just seen how to overload a unary operator. The same mechanism can be used to overload a binary operator.

```
// Overloading + operator
#include <iostream>
using namespace std;
class time
{
    int hh; int mm; int ss;
    public:
        time() { }
        time(int h, int m, int s)
        {
            hh = h; mm = m; ss = s;
        }
        void disp_time()
        {
            cout << endl << hh << " : "
            << mm << " : " << ss;
        }
}
```

```

        time operator+(time);
};

time time::operator+(time t)
{
    time temp;
    temp.hh = hh + t.hh;
    temp.mm = mm + t.mm;
    temp.ss = ss + t.ss;
    return temp;
}

int main()
{
    time t1(12,1,24) , t2(5, 23, 45), t3;
    t3 = t1 + t2;
    t3.display_time();
    return 0;
}

```

5. Operator Overloading with Strings

C/C++ deals with strings quite differently; we never copy, concatenate, or compare strings using operators like other languages. C/C++ has built functions to perform the above operations. But C++ provides the facility to do every thing on strings using operators. That means we have to provide extra responsibility to operators to perform such things.

The following example demonstrates the comparison between two strings using comparison operator ==.

```

// Program to compare two strings using operator overloading
#include <string.h>
#include <stdio.h>
#include <iostream>
using namespace std;

enum boolean{ false, true };

class string
{
    char *str;
public:
    string() { *str = NULL; }
    string(char *s) { str = s; }
    int operator ==(string ts)
    {
        if (strcmp(str, ts.str) >= 0)
            return true;
        else
            return false;
    }
};

int main()
{
    string s1("Computer");
    string s2("Computers");

    if(s1 == s2)
        cout << "Equal";
    else
        cout << "Not Equal";

    return 0;
}

```

```

// concatenation of two strings
#include <string.h>
#include <stdio.h>
#include <iostream>
using namespace std;

class string
{
    char *str;
public:
    string()
    {
        str = new char[30] ;
        *str = NULL;
    }
    string(char *s) { str = s; }
    string operator +(string ts)
    {
        string t;
        strcat(t.str, str);
        strcat(t.str, ts.str);
        return t;
    }
    void putstring()
    {
        cout << endl << str;
    }
};

int main()
{
    string s1("Computer"); string s2("Institute");
    s1.putstring(); s2.putstring();
    string s3;

    s3 = s1 + s2;

    s3.putstring();
    return 0;
}

```

[previous](#)[up](#)[next](#)

Learning C/C++ Step-By-Step - Page 14

Learning C/C++ Step-By-Step - Page 16

Copyright © 2009 Ganesh Kumar Butcha
All Rights Reserved.

0

Like

2

Send

Tweet

0

[add comment](#) |  [view as pdf](#) |  print: [this](#) | [all](#) page(s)

Related Tutorials

- [Beginner's Guide To c++](#)
- [An Explanation of Pointers \(C++\)](#)



Please do not use the comment function to ask for help! If you need help, please use our [forum](#).
Comments will be published after administrator approval.

[Howtos](#) | [Mini-Howtos](#) | [Forums](#) | [News](#) | [Search](#) | [Contribute](#) | [Subscription](#)
[Site Map/RSS Feeds](#) | [Advertise](#) | [Contact](#) | [Disclaimer](#) | [Imprint](#)



Copyright © 2013 HowtoForge - Linux Howtos and Tutorials
All Rights Reserved.