

Chapter 14 – Operator Overloading

Function overloading allows programmers to let the compiler choose between functions of the same name based on return type, the number of arguments, and argument type. Operator overloading allows the compiler to call a function to perform an operation when an operator is asked to perform an operation on objects of classes that it was not originally defined to perform. In effect, operator overloading extends the definition of an operator to include specific classes.

The simplest way to *overload an operator* is by declaring a *free non-void function* with the keyword `operator` as part of the function header. This is as follows:

```
return-type operator operator-symbol ( parameters )
function-body
```

In the following example, operator overloading is implemented on the output stream operator (which is of class *ostream*) so objects of class *student* (from the previous chapter) may be output in a simpler manner. Note that all parameters and the return value are preceded by the *ampersand* and are *pass by reference* objects.

```
// place this function in the file student2.h
// after the student class declaration.

ostream & operator << (ostream & out, student & stu)
{
    out << "Student Name: " << stu.getName( )
        << ", ID: " << stu.getId( );
    return out;
}
```

With this function, programs needing to output an object of *class student* need only `cout` the object. For example:

```
#include <iostream>
#include "student2.h" // the full path may be needed here
using namespace std;

int main( ) {
    student student1("James Morrison", 23);
    student student2("Ringo Star", 99);
    cout << student1 << endl;
    cout << student2 << endl;
    return 0;
}
```

When run, this program will produce the following output:

Student name: James Morrison, ID: 23
Student name: Ringo Starr, ID: 99

Exercises

1. What is operator overloading?
2. What is the purpose of the ampersand (&) in the function
ostream & operator << (ostream & out, student & stu)

Programming Assignment 14.1

Implement and use the overloading function for *class student*.

Programming Assignment 14.2

Create an overloading function on the >> (stream input operator) that will allow an object of class student to be created directly from user input. *HINT: aStudent = student(parameters of the constructor); is a valid statement when aStudent is an object of class student.*