**Objectives**:

* To understand what is operator overloading.
* To understand how to redefine (overload) operators to work with new types.
* Learn the different types of operator overloading.
* To learn when to, and when not to, overload operators.

**Operator overloading**:

* In C++, we can make operators 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.
* Operator overloading is used to overload or redefines most of the operators available in C++. It is used to perform the operation on the user-defined data type.
* It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it.
* Almost any operator can be overloaded in C++. However, there are few operators which can not be overloaded. Operator that are not overloaded are follows
* scope operator (::)
* sizeof
* member selector (.)
* member pointer selector (\*)
* ternary operator (?:)

**syntax**:

The basic syntax of operator overloading is:

class className {

... .. ...

public

returnType operator symbol (arguments) {

... .. ...

}

... .. ...

};

Where

* “returnType” is the type of value returned by the function.
* “operator” is the keyword to use operatior overloading.
* “className” is the name of the class.

**Rules for operator overloading**:

* Existing operators can only be overloaded, but the new operators cannot be overloaded.
* The overloaded operator contains at least one operand of the user-defined data type.
* We cannot use friend function to overload certain operators. However, the member function can be used to overload those operators.
* When unary operators are overloaded through a member function take no explicit arguments, but, if they are overloaded by a friend function, takes one argument.
* When binary operators are overloaded through a member function takes one explicit argument, and if they are overloaded through a friend function takes two explicit arguments.

**Types of operator overloading**:

There are 2 types of operator overloading.

* Unary operator overloading
* Binary operator overloading

**Unary operator overloading**:

In unary operator function, no arguments should be passed. It works only with one class objects. It is a overloading of an operator operating on a single operand.

Example:

|  |  |
| --- | --- |
| *#include<iostream>*  *using namespace std;*  *class Test{*  *private:*  *int num;*  *public:*  *Test(): num(8){};*  *void operator ++() //unary operator*  *{*  *num += 2; //increment num by 2*  *}*  *void print()*  *{*  *cout<< "The count is : "<<num;*  *}*  *};* | *int main(){*  *Test obj;*  *++obj;*  *obj.print();*  *return 0;*  *}* |

Output:

*The count is : 10*

Here no arguments are passed in the operator overloading function and the overloading is done to the increment (++) operator which is operating only one operand.

**Binary operator overloading**:

In binary operator overloading function, there should be one argument to be passed. It is overloading of an operator operating on two operands.

Example:

|  |  |
| --- | --- |
| *#include<iostream>*  *using namespace std;*  *class Complex{*  *private:*  *int real, imag;*  *public:*  *Complex() //default constructor*  *{*  *real = 0;*  *imag = 0;*  *}*  *Complex(int r, int i)*  *{*  *real = r;*  *imag = i;*  *}*  *void print()*  *{*  *cout<<real<< "+" <<imag<< "i" <<endl;*  *}*  *//operator overloading*  *Complex operator +(Complex c)*  *{*  *Complex temp;*  *temp.real = real + c.real;*  *temp.imag = imag + c.imag;*  *return temp;*  *}*  *};* | *int main(){*  *Complex c1(5,4);*  *Complex c2(2,5);*  *Complex c3;*  *c3 = c1+c2; //c3 = c1.add(c2)*  *c3.print();*  *return 0;*  *}* |

Output: *7+9i*

Here only one argument is passed in the operator overloading function. The overloading is done to the operator “+” which operates on two operands c1 and c2.

**Operator Functions as Class Members vs. as friend Functions**

* **Member vs non-member**
* Operator functions can be member or non-member functions
* When overloading ( ), [ ], -> or any of the assignment operators, must use a member function
* **Operator functions as member functions**
* Leftmost operand must be an object (or reference to an object) of the class. If left operand of a different type, operator function must be a non-member function
* **Operator functions as non-member functions**
* Must be friends if needs to access private or protected members
* Enable the operator to be commutative

**Overloading Stream-Insertion and Stream-Extraction Operators**

* overloaded << and >> operators
* Overloaded to perform input/output for user-defined types
* Left operand of types ostream & and istream &
* Must be a non-member function because left operand is not an object of the class
* Must be a friend function to access private data members