Course Name: A Level (2nd Sem) **Subject:** Data Struture using C++
Topic: Types of Operator Overloading **Date:** 07-04-2020

Types of Operator overloading

Operator Overloading can be done by using three approaches, they are

- Overloading unary operator.
- Overloading binary operator.
- Overloading binary operator using a friend function.

Overloading Unary Operator

Let us consider the unary ' – ' operator. A minus operator when used as a unary it requires only one operand. We know that this operator changes the sign of an operand when applied to a basic data variable. Let us see how to overload this operator so that it can be applied to an object in much the same way as it is applied to an int or float variable. The unary minus, when applied to an object, should decrement each of its data items.

Example:

```
#include <iostream>
using namespace std;
class Height {
public:
// Member Objects
int feet, inch;
// Constructor to initialize the object's value
Height(int f, int i)
{
feet = f;
inch = i;
}
// Overloading(-) operator to perform decrement
// operation of Height object
void operator-()
{
feet--;
inch--;
cout << "Feet & Inches after decrement: " << feet << " ' " << inch <<endl;
}
};
int main()
{
//Declare and Initialize the constructor of class Height
Height h1(6, 2);
//Use (-) unary operator by single operand
-h1;
return
}
```

Output:

```
Feet & Inches after decrement: 5'1
[Finished in 1.5s]
```

Explanation:

In the above example, we overload ' – ' unary operator to perform decrement in the two variables of Height class. We pass two parameters to the constructor and save their values in feet and inch variable. Then we define the operator overloading function (void operator-()) in which the two variables are decremented by one position. When we write -h1 it calls the operator overloading function and decrements the values passed to the constructor.

Overloading Binary Operator

It is an overloading of an operator operating on two operands. Let's take the same example of class Height, but this time, add two Height objects h1 and h2.

Example:

```
#include <iostream>
using namespace std;
class Height
{
public:
int feet, inch;
Height()
{
feet = 0;
inch = 0;
}
Height(int f, int i)
{
feet = f;
inch = i;
}
// Overloading (+) operator to perform addition of
// two distance object using binary operator
Height operator+(Height& d2) // Call by reference
{
// Create an object to return
Height h3;
// Perform addition of feet and inches
h3.feet = feet + d2.feet;
h3.inch = inch + d2.inch;
// Return the resulting object
return h3;
}
};
int main()
{
Height h1(3, 7);
Height h2(6, 1);
Height h3;
//Use overloaded operator
```

```
h3 = h1 + h2;
cout << "Sum of  Feet & Inches: " << h3.feet << "'" << h3.inch << endl;
return 0;
}
```

Output:

```
Sum of  Feet & Inches: 9'8
[Finished in 1.5s]
```

Explanation:

Height operator+(Height &h2), here returns_type of function is class Height thus it returns an object h3 of class Height. In the line h3 = h1 + h2, h1 calls the operator function of its classes objects and takes h2 as a parameter, then we apply h3.feet = feet + d2.feet; and h3.inch = inch + d2.inch; which stores the sum of values of the variables feet and inch in variables associated with the h3 object.

When the statement 'h3 = h1 + h2' invokes the operator overloaded function the object h1 took the responsibility of invoking the function and h2 plays the role of argument that is passed to the function. The above invocation statement is equivalent to h3= h1.operator+(h2); therefore the data member of h2 are accessed directly and the data member of h2 (that is passed as an argument) are accessed using the dot operator.

Rules for Operator Overloading

Only the existing operators can be overloaded and new operators cannot be overloaded. The overloaded operator must contain at least one operand of the user-defined data type.

We do not use a friend function to overload certain operators. However, the member functions can be used to overload those operators.

When unary operators are overloaded through a member function they take no explicit arguments, but, if they are overloaded by a friend function they take one argument.

When binary operators are overloaded through a member function they take one explicit argument, and if they are overloaded through a friend function they take two explicit arguments.