**C++ Operators Overloading**

Operator overloading is a compile-time polymorphism in which the operator is overloaded to provide the special meaning to the user-defined data type. 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. For example, C++ provides the ability to add the variables of the user-defined data type that is applied to the built-in data types.

**Note -** *The advantage of Operators overloading is to perform different operations on the same operand.*

**Operator that cannot be overloaded are as follows -**

* Scope operator (::)
* Sizeof
* member selector(.)
* member pointer selector(\*)
* ternary operator(?:)

**Syntax of Operator Overloading -**

return\_type class\_name : : operator op(argument\_list)

{

// body of the function.

}

**Rules for Operator Overloading -**

* Existing operators can only be overloaded, but the new operator 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.

**C++ Operators Overloading Example –**

#include <iostream.h>

class Test

{

private:

int num;

public:

Test(): num(8){}

void operator ++() {

num = num+2;

}

void Print() {

cout<<"The Count is: "<<num;

}

};

int main()

{

Test tt;

++tt; // calling of a function "void operator ++()"

tt.Print();

return 0;

}

**Program to overload the binary operators -**

#include <iostream.h>

class A

{

int x;

public:

A(){}

A(int i)

{

x=i;

}

void operator+(A);

void display();

};

void A :: operator+(A a)

{

int m = x+a.x;

cout<<"The result of the addition of two objects is : "<<m;

}

void main()

{

A a1(5);

A a2(4);

a1+a2;

}