Agenda

1. Introduction to operator overloading
2. Polymorphism in OO
3. Operators that cannot be overloaded
4. overloading of binary operator
5. overloading of unary operator

Operator Overloading

We can define operators for classes.
We can provide operator definition in the class so that it works in a specific way for the objects of that class

Operator overloading is defining new meaning of the operator. Now one operator symbol has more than one meaning depending on type of data used by the operator.

One operator symbol is overloaded with multiple interpretation is known as operator overloading.

which interpretation of operator has to be considered is resolved at compile time on the basis of operands of the operator.

Therefore it is compile time polymorphism.

```
class Complex
{
    private:
        int a, b;
    public:
        Complex operator+(Complex c)
        {
            Complex temp;
            temp.a = a + c.a;
            temp.b = b + c.b;
            return temp;
        }
        void setData(int x, int y)
        { a = x;
          b = y;
        }
```

```
    void showData()
    {
        cout<<"\na="<<a<<" b="<<b;
    }
};
int main()
{
    Complex c1,c2,c3;
    c1.setData(3,4);
    c2.setData(5,6);
    c3=c1+c2;
    c3.showData();
}
```

Operators that cannot be overloaded

::	Scope resolution
sizeof()	size of
?:	conditional operator
.	member access
.*	pointer to member

Only those operators can be overloaded in c++ which are valid operators in C language.

Overloading of Binary operator

```
C3=C1+C2;
```

Always left operand is caller object in the case of overloading of binary operators.

Overloading of unary operator

- Overload unary - operator for complex.

C3 = -C1 + C2

C1.operator - ()

temp . operator + (C2)

object