C++ in depth

Operator overloading

Saurabh Shukla (MySirG)

Agenda

1. Recall add method
2. Operator overloading
3. Polymorphism
4. Not all the operators can be overloaded
5. Overloading of binary operator
6. Overloading of unary operator

Recall add method

```
Complex Complex::add(Complex C)
{
        Complex temp;
        temp.a = a + C.a;
        temp.b = b + C.b;
        return temp;
}
```

```
C3 = C1.add(C2);
```

change name from add to +

```
Complex Complex::operator + (Complex C)
{
        Complex temp;
        temp.a = a + C.a;
        temp.b = b + C.b;
        retun temp;
}
```

```
C3 = C1.Operator + (C2);
        or
C3 = C1 + C2;
```

Class

1. variables
2. functions
3. operators

object.variables

object.functions()

object.operator+(arg)

object + arg

cout << x

cout.operator<<(x)

Operator Overloading

When one operator symbol is overloaded with multiple operations, it is known as overloaded operator.

Defining an operator in a class, is providing a new behaviour of operator for specific type operands.

In simple words, operator overloading occurs when you define an operator with respect to the class.

Polymorphism

Operator overloading is another way of implementing Polymorphism.

3 + 4	int + int
5.4 + 6.3	double + double
C1 + C2	Complex + Complex
t1 + t2	Time + Time

Not all the operators can be overlocked

- Only those symbols can be defined as an operator which were valid operators in C language.
- There are few operators in C language which you cannot overload in C++
  - sizeof()
  - . member access operator
  - .* pointer to member operator
  - ?: conditional operator
  - :: scope resolution operator

Overloading of binary operators

When a binary operator is overloaded in a class as a member, only left operand is a caller object and right operand is an argument.

```
C3 = C1 + C2
     ↑
  callerobject
```

Overloading of Unary operators

The way of writing/using unary operator will remain same for overloaded version.

>

==

!

*

[]

()

++ post pre

-- post pre

=

<<

>>