CSE-206
Object oriented Programming

Operator Overloading

Lecturer Afia Anjum
Military Institute of Science and technology

What is Operator Overloading?

- This feature in C++ programming that allows programmer to redefine the meaning of an operator (when they operate on class objects) is known as operator overloading.
- The meaning of an operator is always same for variable of basic types like: int, float, double etc. For example: To add two integers, + operator is used.
- However, for user-defined types (like: objects), you can redefine the way operator works.
- For example: If there are two objects of a class that contains string as its data members. You can redefine the meaning of + operator and use it to concatenate those string objects

Structure of Operator Overloading?

To overload an operator, a special operator function is defined inside the class as:

```
class className
{
   ... .. ...
   public:
      returnType operator symbol (arguments)
      {
         ... .. ...
      }
   ... .. ...
};
```

Operator Overloading

We can overload all the operators in C++ except the following:

- Scope operator (::)
- Size operator (Sizeof)
- Member selector(.)
- Member pointer selector(*)
- Ternary operator(?:)

Operator Function vs Normal Function

Operator functions are the same as normal functions. The only difference is, the name of an operator function is always operator keyword followed by the symbol of operator and operator functions are called when the corresponding operator is used.

Rules for Operator Overloading

- Operator function must be either non-static (member function) or friend function to get overloaded.
- When unary operators (for ex: ++a) are overloaded through a member function they take no explicit arguments, but, if they are overloaded by a friend function they take one argument.

  In main function if we write: **op X or X op**

  If operator function is member function: **X.operator()**

  If operator function is friend function: **operator(X)**
- When binary operators (for ex: a+b) 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.

  In main function if we write: **X op Y**

  If operator function is member function: **X.operator(Y)**

  If operator function is friend function: **operator(X, Y)**