Lecture - 12

Introduction to operator overloading

Object manipulation

- Data members of objects are manipulated by calling the member functions of the object

```
void main()
{  a object1;
object1.set(10);
}
```

Example

```
class a
{ int x;
public: void set(int i)
 {x=i; }
a add(a para1)
{ a ob; a.x=x+para1.x;
return(a);
} };
```

```
void main()
{
   a object1, object2,object3;
   object1.set(10);
   object2.set(20);
   object3=object1.add(object2);
}
```

Can we use built in C++ operators fo manipulating objects?

```
void main()
{
   a object1, object2,object3;
   object1.set(10);
   object2.set(20);
   object3=object1+object2;
}
```

Operator overloading

- It is a process which enable C++ operators like +,-,* to work with object

- For example, C++ language itself overloads the addition operator (+) as these operators perform differently when used with int, float and pointers.

Operator overloading (contd..)

- An operator is overloaded by writing a non-static member function definition or global function definition
- The function name now becomes the keyword **operator** followed by the symbol for the operator being overloaded, eg +

Restrictions on operator overloading

Operators that can be overloaded

+ - / * % ^ & | ~ ! = < > += -= *= /= %= << >> == != <= >= && ||
++ -- , -> ( ) new delete

Operators that cannot be overloaded

.

::

?:

Contd..

- The associativity of an operator (left to right, or right to left) cannot be changed by overloading
- It is not possible to change arity of an operator (how many operands an operator takes)
- The meaning of how an operator works on objects of fundamental types cannot be changed

Contd..

- It is not possible to create new operators
- Only existing can be overloaded

Class members vs. Global functions

- Operator functions can be member functions or global functions
- Global functions are made friends
- Either way operator will be used the same way in expression

Which implementation is best?

- Operator member functions of specific class are called (implicitly by compiler) only when left operand of a binary operator is specifically an object of that class, or when the single operand of a unary operator is an object of that class

Contd..

- If left operand must be an object of a different class or a fundamental type, operator must be implemented as global function

Example (unary operator -)

```
class point
{  int x,y,z;
  public:
  point(int d,int e, int f)
 { x=d; y=e; f=z; }
void display()
{cout<<x<<y<<z; }
void operator –()
{ x=-x; y=-y; z=-z; }
};
```

```
void main()
{
  point a(10,20,15);
  a.display();
  -a ;
  a.display();
}
```