UNIT- III U pravallika N. Akhila
CSE dept

Operator Overloading:- operator overloading is one of the Important and useful feature of C++. C++ allows programmers to redefine the meaning of operators when they operate on class objects. This feature is called operator overloading. with this feature the overloading principle is not only applied to function .but also on operators.

Operator overloading allows programmers to extend the meaning of existing operators so that in addition to basic datatypes, they can also applied to user defined datatypes (objects)

with operator overloading a programmer is allowed to ~~provide~~ his own definition for an operator to a class by overloading the built in operator. ~~while~~ This enables the programmer to perform specific computation when operator is applied on class objects and apply standard definition when same operator is applied on built in datatypes.

Therefore while evaluating an expression with operators, C++ looks around operator. if the operands are built in types, C++ calls a built in routine. if the operator is being applied on ~~class objects~~ objects the c++ compiler ~~[illegible]~~ implement

if the programmer has overloaded operator func

that it can call. if such function whose parameter

match the types and number of operands exists

in the program, the function is called, otherwise

a compile time error is generated.

<u>Advantages</u> of operator overloading:

* operator overloading makes the program clear.

In other words it makes code much more

readable. For example

add (c1, c2) can be better written as c1+c2

* with operator overloading a similar level of

Syntactic support is provided to user defined types

(objects) as provided to the built in types.

<u>Syntax</u> of operator overloading:-

```
class <classname>
 {
      public:
        returntype operator op( arguments)
          {
            - . . .
            - . . .
              //Function body;
          }
 };
```

Implementing operator overloading:

operator overloading is usually implemented in two ways as follows

1. Through member function 2. Through friend function

operators can be overloaded using any of these techniques.

* Difference between member-function and friend function for Operator overloading

member function	Friend function
* unary operators take no explicit parameters	* unary operators takes only one parameter
* Binary operator requires one explicit parameter	* Binary operator takes two parameters
* member function takes Argument explicitly	* friend function needs the parameters to be explicitly passed.

op is the operator that has to be overloaded

Operator is the keyword that defines a new

to the operator

<u>Rules</u> for operator overloading:

* operator overloading cannot change the operation performed by an operator

* Operator overloading cannot alter the precedence and the associativity of operator

* overloaded operators cannot have default arguments

* only existing operators are overloaded, no new operator can be created

* Operator overloading cannot change number of operands.

erloading unary operator :

unary operator work only on single operand. Some examples of unary operators are as follows

1. Increment (++)
2. Decrement (--)
3. Unary minus (-)
4. logical not operator (!)

unary operators can be overloaded using friend function or member function. In both cases unary operator operates on object

using a member function to overload a unary operator:-

The syntax for operator overloading using a member function can be given as

```
Returntype. operator op()
{
    - - - -   //Function code.
    - - - -
    - - - -
}
```

Ex:

```
class  sample
{
    private:
        inr x;
    Public:
     void getdata()
      {
          x= 10;
      }
     void show();
     void  operator -();
};
void  sample :: operator -()
   {
          x = -x;
   }
void  sample :: show()
   {
        cout<<"x = "<<x<<endl;
   }
int  main()
 {
       sample  s;
        s.getdata();
        s.show();
        -s;
        s.show();
 }
```

Returning object :-

```
#include <iostream>
using namespace std;
class sample
  {
        private:
          int x ;
        public:
         void getdata()
         {
             cout<<"Enter x value"<<endl;
             cin>>x;
         }
        void  operator -()
            {
               Sample temp ;
                 temp.x = -x ;
            }   return temp;

        void  show()
           {
             cout<<"x = " <<x;
           }
    };
        int main()
           {
               Sample s1, s2;
                  S1.getdata();
                 s2 = -s1;
                 s2.show();
           }
```

Scanned by CamScanner

using friend function to overload a unary operator--

when a unary operand is overloaded using friend function then one must ensure the following

* The function will take one operand as an argument
* This operand will be object of the class
* Function may or may not return any value.
* the friend function doesnot have access to the this pointer
* ~~this~~ function will use private members of a class only with object name

The syntax for operator overloading using member function can be given as

friend returntype operator op (class_name object);

where op is the operator to be overloaded and object is an Instance of the class on which operator has to be applied.

Scanned by CamScanner