Object-Oriented Programming using C++(Unit-4)

- this pointer
- Operator overloading

Prepared by: Anil Kumar Tailor, Assistant Prof.,
Engineering College Ajmer

this pointer

- A keyword called "this" to represent an object that invokes a member function.
- It passes to the member function automatically.
- It acts as an implicit argument to all the member functions.
- Used to differentiate between data member of the class and local variable with same name of a member function.

this pointer

- Example-

```
class ABC
{
int a ;
public:
void set_values( )
{
int a =10; //local variable
this->a=20; //data member
}
};
```

this pointer

- Also used to return the object it points to.
- Example-

```
class person
{
char name[20]; int age;
public:
person(char *s , int a)
{
strcpy(name, s);
age = a;
}
```

this pointer

```
person & greater(person & x)
{ if(x.age>=age)
return x;    //argument object
else
return *this; //invoking object
}
void  display()
{  cout<<name<<"\t"<<age; }
};
```

this pointer

```
int main()
{
person p1("Ramesh", 24);
person p2("Suresh", 30);
person p3 =p1.greater(p2);
p3.display();
return 0;
}
```

output:

Suresh 30

Operator overloading

- The mechanism of giving special meanings to an operator for a data type is known as operator overloading.
- All the operators in C++ can be overload except the following:

➢ Class member access operators (., .*)

➢ Scope resolution operator (::)

➢ Size operator (sizeof)

➢ Conditional operator (?:)

Operator overloading

- When we overload an operator, its original meaning remains same.
- For example, if we overload + operator to add two matrices, can still be used to add two numbers.
- To give additional meaning to an operator, we use a special function called operator function.

Operator overloading

- Syntax of the operator function is :

return_type class_name :: **operator** op(arguments list)

{

function body

}

- Where operator is a keyword and op is an operator to be overload.
- Operator function must be either member function or friend function of a class.

Overloading post increment operator (++) to increment an object by one using member function.

```
class point
{
int x,y;
public:
void getdata(int a, int b)
{ x=a; y=b; }
void show(void)
{
cout<<“x=”<<x;
cout<<“y=”<<y<<“\n”;
}
void operator ++(int)
{ x++; y++; }
};
```

Overloading post increment operator (++) to increment an object by one using member function.

```
int main()
{
    point p;
    p.getdata(5,8);
    cout<<“p:”;
    p.show();
    p++; // invoke operator function
    cout<<“p++:”;
    p.show();
    return 0;
}
```

Overloading post increment operator (++) to increment an object by one using member function.

- Output

p: x=5 y=8

p++: x=6 y=9

- The **int** in the operator function is used to differentiate between postfix and prefix increment operators.

Overloading pre decrement operator (--) to decrement an object by one using friend function.

```
class point
{
int x,y;
public:
void getdata(int a, int b)
{  x=a;  y=b; }
void show(void)
{ cout<<“x=”<<x;
cout<<“y=”<<y<<“\n”; }
friend void operator--(point &s)
{  s.x=s.x-1; s.y=s.y-1; }
};
```

Overloading pre decrement operator (--) to decrement an object by one using friend function.

```
int main()
{
   point p;
   p.getdata(7,10);
   p.show();
   --p;
   p.show();
   return 0;
}
```

Overloading pre decrement operator (--) to decrement an object by one using friend function.

- Output

x=7 y=10

x=6 y=9

- If we pass argument by value it will not work because the changes made in operator function will not reflect in main function.

Overloading binary + operator for adding two complex numbers using member function.

```
class complex
{
 float real, img;
public:
complex(float r, float i)
{ real=r;  img=i;  }
void show()
{
 if(img>0)  cout<<real<<"+i"<<img<<"\n";
else cout<<real<<"-i"<<-img<<"\n";
 }
```

Overloading binary + operator for adding two complex numbers using member function.

```
complex operator +(complex &p)
{   complex q;
q.real=real+p.real;
q.img=img+p.img;
return q;  } };
void main()
{complex c1(3,4);
complex c2(4,5);
complex c3
c3=c1+c2;
c1.show();
c2.show();
c3.show(); }
```

Overloading binary + operator for adding two complex numbers using member function.

- Output

3+i4
4+i5
7+i9

- For binary operators, the left-hand side operand is used to invoke the operator function and the right-hand side operand is passed as an argument.

Overloading binary + operator for adding two complex numbers using friend function.

- The operator function will be-

```
friend complex operator+(complex &p, complex &q)
{
complex r;
r.real=p.real+q.real;
r.img=p.img+q.img;
return r;
 }
```

Overloading Binary operators using friend function

- We can't overload the following operators using friend function.

➢ = Assignment operator

➢ () Function call operator

➢ [] Subscripting operator

➢ -> Class member access operator

Thank You

Prepared by: Anil Kumar Tailor, Assistant Prof.,
Engineering College Ajmer