Polymorphism

Function Overloading and Operator Overloading

Function Overloading and operator overloading helps to achieve polymorphism in C++.Polymorphism is a term – poly means multiple and morphos (form) or meanings

Function overloading is the process of using same name for two or more function. In Function overloading is the name of the function would be same but different types , number of arguments/parameters are passed to it. Compiler will decide which function to execute by looking at the function declaration/signature/prototype.

Example

It a class has a display function which is used to display int number,float numer,character and string then using function overloading we can write a function as shown below

```
void display (int);
void display (float);
void display (char);
void display(char*);
```

NOTE – The minimum requirement for function overloading is that the function musst differ in the type numer or sequence or order of the arguments/parameters.

Example

```
void show(int,int);
void show(float,int);
void show(float,float,float);
void show(int,float,char);
void show(char*,int);
```

are allowed but if two functions are having same argument type and number and if their return type is different then it is not function overloading for e.g int add(int,int) and float add(int,int) is not function overloading.

Any function in c++ can be overloaded.It can be a normal function or it can be a member function

Example :- Write a program using function overloading to add two different numbers of different data types.

```
#include<iostram>
using namespace std;
void  add(int a,int b)
{
```

```
cout<<"\n addition of two int nos is = "<<a+b;
}
void  add(float a,float b)
{
cout<<"\n addition of two float nos is = "<<a+b;
}

void  add(char a,char b)
{
cout<<"\n addition of two characters is = "<<a+b;
}

void  add(int a,float b)
{
cout<<"\n addition of one int and one character is  = "<<a+b;
}

void  add(char a,float b)
{
cout<<"\n addition of two int nos is = "<<a+b;
}

int main()
{

add(100,200);
add(5.23,6.7);
add('Z',3.14);
add('e','L');
add(2,3.24);
return 0;
}
```

Same program can also be written using member function add as shown below:-

```
#include<iostram>
using namespace std;

class test
{
public:

void  add(int a,int b)
{
cout<<"\n addition of two int nos is = "<<a+b;
}
void  add(float a,float b)
```

```
{
cout<<"\n addition of two float nos is = "<<a+b;
}

void  add(char a,char b)
{
cout<<"\n addition of two characters is = "<<a+b;
}

void  add(int a,float b)
{
cout<<"\n addition of one int and one character is  = "<<a+b;
}

void  add(char a,float b)
{
cout<<"\n addition of two int nos is = "<<a+b;
}
};
int main ()
{

test t;

t.add(100,200);
t.add(5.23,6.7);
t.add('Z',3.14);
t.add('e','L');
t.add(2,3.24);
}
```

Points to Remember about Function Overloading

When a function is called compiler looks for the exact match for the function call,if a function is found it is called and executed.

If a match is not found the compiler look for the possibility of type casting and try to find the function which match with the call e.g. if a call is to display('P') and in a program there is a function written as display(int) then a function is executed as every character is associated with a ASCII value.But if a ambiguity is there then a call is not executed e.g. if a there is function as display(int),display(long),display(long) or display(double) and if a function is called as display('P') then a compiler will have a problem as which copy of display function is to execute.

To make a function overloading possible the functions must have different arguments/parameters types or different numbers of arguments/parameters.
Atleast one of the argument must be different.
If the type of arguments is same then there sequence must change (order of arguments / parameters).

Constructors can be overloaded.
Destructors cannot be overloaded.
The return type of the function is not a factor in distinguishing overloaded functions.

Programs for practice

1. Write program to find area of rectangle, square, circle and cylinder, rectangular box. Use a function name area.

2. Using function overloading write a function to find a given number is present in a list of integer numbers or not and a given name is present in a list of names or not.

Operator Overloading

Like a function overloading operators can also be overloaded.The operators can be overloaded to perform the operation on the objects.There are two types of overloading unary operator overloading and binary operator overloading.

Unary operator overloading

Unary operators are the operators which takes only one operand e.g. ++ , -- .
If we want to increment the values of data members,it can be done by overloading operator ++.

Example

```
#include<iostream>
using namespace std;
class test
{
private : int x;
public:
test(int a)
{
x=a;
}
void operator ++ (void)
{
x++;
}
void show()
{
cout<<"\n the value of data member is "<<x;
}
};
```

```
int main()
{
test t(10);
t.show();
++t;
t.show(;
return 0;
}
```

Prefix and Postfix

The operator ++ in the above program cannot be used as a postfix operator. To make it for postfix we need to use as

```
void operator ++(int);
```

The integer above is a dummy argument which is used on to distinguish between the prefix and postfix operators.
Now
++t is as operator ++(void) and t++ as operator ++(int)

Practice programs

Write a program to overload operator --.

Binary Operator

The binary operators are arithmetic operator +,-,*,/ and % ,Conditional operators > ,<,<=,>=,== and != and assignment operators like +=,-=,*=,/= etc.

Example :- operator overloading of +

```
class test
{
int x;
public:
test(int a)
{
X=a;
}
test operator +(test&  t)
{
x=t.x+x;
return *this;
}
void show()
```

```
{
cout<<"\n the value of data member is "<<x;
}
};

int main()
{
test t1(10),t2(20);
t1=t1+t2;
t1.show();
return 0;
}
```

The operator overloading + can also be written as

```
test operator +(test&  t)
{
x=t.x+x;
return x;
}
```

OR

```
test operator +(test  t)
{
test temp;
temp.x=t.x+x;
return temp;
}
```

Operator overloading is also possible using friend function the same program can also be written using friend function as shown below :

```
class test
{
int x;
public:
test(int a)
{
X=a;
}
test()
{  }

void show()
{
cout<<"\n the value of data member is "<<x;
}
```

```
friend test operator +(test,test);
};

test operator +(test t1,test 2)
{
temp t;
temp.x=t1.x+t2.x;
return t;
}

int main()
{
test t1(10),t2(20),t3;
t3=t1+t2;
t3.show();
return 0;
}
```

Points to Remember

The overloaded operator should not change the meaning of the operator.
Overloaded operators must follow the syntax rules of the original operators.
Operators can be overloaded using member function and friend function.
If a unary operator is overloaded using a member function it takes zero arguments.
If a unary operator is overloaded using friend function it takes one argument.
If a binary operator is overloaded using a member function it takes one arguments.
If a binary operator is overloaded using friend function it takes two arguments.
Friend function cannot use to overload certain operators like = , ( ),[ ],- >.
Binary operators such as +,-,*,/ must return value.
Some operators cannot be overloaded as . , ::,.* and ?.
Operator precedence is should not altered.

Programs for practice

1. Write a program to overload the following operators

    a) +, - , /, * and %
    b) >, <,>=, <=,== and !=
    c) +=, -=,*= and /=

(do this using both member functions and friend functions

2. Write a program to add two complex number using operator overloading +