Unit IV

Object Oriented Programming with C++



Polymorphism



Learning Objectives

After learning this chapter, the students will be able to

- Understand the purpose of overloading
- Construct C++ programs using function, constructor and operator overloading
- Execute and debug programs which contains the concept of polymorphism

15.1 IntroductionThe word polymorphism means many forms (poly – many, morph – shapes) Polymorphism is the ability of a message or function to be displayed in more than one form. In C++, polymorphism is achieved through function overloading and operator overloading. The term overloading means a name having two or more distinct meanings. Thus an 'overloaded function' refers to a function having more than one distinct meaning.

15.2 Function overloading

The ability of the function to process the message or data in more than one form is called as function overloading. In other words function overloading means two or more functions in the same scope share the same name but their parameters are different. In this situation, the functions that share the same name are said to be overloaded and the process is called function overloading. The number and types of a function's parameters are called the function's signature. When you call an overloaded function, the compiler determines the most appropriate definition to use, by comparing the argument types you have used to call the function with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called **overload resolution**

15.2.1 Need For Function overloading

Sometimes it's hard to find many different meaningful names for a single action.

Consider the situation to find the area of circle, triangle and rectangle the following function prototype is given

float area_circle(float radius)

- // to calculate the area of a circle
- // to calculate the area of a triangle
- float area_rectangle(float length, float breadth)

float area_triangle(float half,floatbase,float height)

// to calculate the area of a rectangle

This can be rewritten using a single function header in the following manner float area (float radius); float area (float half, float base, float height); float area (float length , float breadth);

Illustration 15.1 C++ Program to demonstrate function overloading

۲

```
#include <iostream>
using namespace std;
void print(int i)
{cout<< " It is integer " << i <<endl;}
void print(double f)
{ cout<< " It is float " << f <<endl;}
void print(string c)
{ cout<< " It is string " << c <<endl;}
int main() {
    print(10);
    print(10.10);
    print("Ten");
    return 0;
}
Output:</pre>
```

It is integer 10 It is float 10.1 It is string Ten

Tip Notes

Function overloading is not only implementing polymorphism but also reduces the number of comparisons in a program and makes the program to execute faster. It also helps the programmer by reducing the number of function names to be remembered.

15.2.2 Rules for function overloading

- 1. The overloaded function must differ in the number of its arguments or data types
- 2. The return type of overloaded functions are not considered for overloading same data type
- 3. The default arguments of overloaded functions are not considered as part of the parameter list in function overloading.

Illustration 15.2 C++ Program to demonstrate function overloading #include <iostream> using namespace std; long add(long, long); long add(long,long,long); float add(float, float); int main() long a, b, c,d; float e, f, g; cout << "Enter three integers\n";</pre> cin >> a >> b>>c; d=add(a,b,c);//number of arguments different but same data type cout << "Sum of 3 integers: " << d << endl; cout << "Enter two integers\n";</pre> cin >> a >> b;c = add(a, b);//two arguments with same data type cout << "Sum of 2 integers: " << c << endl; cout << "Enter two floating point numbers\n";</pre> $\operatorname{cin} >> \operatorname{e} >> \operatorname{f};$ g = add(e, f);//two arguments with different data type cout << "Sum of floats: " << g << endl; long add(long c, long g) long sum; sum = c + g;return sum; float add(float c, float g) ł float sum; sum = c + g;return sum; long add(long c, long g,long h) long sum; sum = c + g + h;return sum; Output Enter three integers 345 Sum of 3 integers: 12 Enter two integers 46 Sum of 2 integers: 10 Enter two floating point numbers 2.1 3.1 Sum of floats: 5.2

۲

۲

۲

15.3 Constructor overloading

۲

Function overloading can be applied for constructors, as constructors are special functions of classes .A class can have more than one constructor with different signature. Constructor overloading provides flexibility of creating multiple type of objects for a class.

Illustration 15.3 Constructor overloading #include<iostream> using namespace std; Compiler identifies a given member function is class add a constructor by its name and the return type. { int num1, num2, sum; public: add() cout<<"\n Constructor without parameters.. "; { $num1 = 0; num2 = 0; sum = 0; \}$ add (int s1, int s2) { cout<<"\n Parameterized constructor... "; $num1 = s1; num2 = s2; sum=0; \}$ add (add &a) { cout<<"\n Copy Constructor ... "; num1= a.num1; num2=a.num2; $sum = 0; \}$ void getdata() { cout<<"\nEnter data ... "; cin>>num1>>num2; } void addition() { sum=num1+num2; } void putdata() { cout<<"\n The numbers are.."; cout<<num1<<'\t'<<num2; cout<<"\n The sum of the numbers are.. "<< sum; } };</pre> int main() { add a, b (10, 20), c(b); a.getdata(); a.addition(); b.addition(); c.addition(); cout<<"\n Object a : ";</pre> a.putdata(); cout<<"\n Object b : ";</pre> b.putdata(); cout<<"\n Object c.. ";</pre> c.putdata(); return 0; }

 \bigcirc

Output

Constructor without parameters.. Parameterized constructor... Copy Constructor ... Enter data ... 20 30 Object a : The numbers are..20 30 The sum of the numbers are.. 50 Object b : The numbers are..10 20 The sum of the numbers are.. 30 Object c.. The numbers are..10 20 The sum of the numbers are.. 30

Since, there are multiple constructors present, argument to the constructor should also be passed while creating an object.

Note

15.4 Operator overloading

The term Operator overloading, refers to giving additional functionality to the normal C++ operators like +,++,-,-,+=,-=,*.<,>. It is also a type of polymorphism in which an operator is overloaded to give user defined meaning to it.

For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String(concatenation) etc.

Almost all operators can be overloaded in C++. However there are few operator which can not be overloaded. Operator that are not overloaded are follows

- Scope operator (**::**)
- Sizeof

۲

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

Operator Overloading Syntax

 Keyword
 Operator to be overloaded

 ReturnType classname :: Operator Operator Symbol (argument list)

 { \\ Function body }

15.4.1 Restrictions on Operator Overloading

Following are some restrictions to be kept in mind while implementing operator overloading.

- 1. Precedence and Associativity of an operator cannot be changed.
- 2. No new operators can be created, only existing operators can be overloaded.
- 3. Cannot redefine the meaning of an operator's procedure. You cannot change how integers are added.Only additional functions can be given to an operator
- 4. Overloaded operators cannot have default arguments.
- 5. When binary operators are overloaded, the left hand object must be an object of the relevant class

```
Illustration 15.5 Binary operator overloading using '+'in Complex number addition
#include<iostream>
using namespace std;
class complex
{ int real, img;
  public:
void read()
cout << "\nEnter the REAL PART : ";
cin>>real;
cout<<"\nEnter the IMAGINARY PART : ";
cin>>img;
complex operator +(complex c2)
                                         Output
                                         Enter the First Complex Number
  complex c3;
                                         Enter the REAL PART: 3
  c3.real=real+c2.real;
                                         Enter the IMAGINARY PART: 4
  c3.img=img+c2.img;
                                         Enter the Second Complex Number
  return c3;
                                         Enter the REAL PART : 5
                                         Enter the IMAGINARY PART : 8
  void display()
                                         SUM = 8 + 12i
{ cout<<real<<"+"<<img<<"i"; }
};
int main()
ł
  complex c1,c2,c3;
  int choice, cont;
  cout<<"\n\nEnter the First Complex Number";
  c1.read();
  cout<<"\n\nEnter the Second Complex Number";</pre>
  c2.read();
  c3=c1+c2; // binary + overloaded
  cout<<"\n\nSUM = ";</pre>
  c3.display();
  return 0;
}
```

۲

۲

#include<iostream>
using namespace std;
class strings
{
 public:
 char s[20];
 void getstring(char str[])
 {
 strcpy(s,str);
 }
 void operator+(strings);
};

Illustration 15.6 Concatenation of string using operator overloading

۲

```
void strings::operator+(strings ob)
```

```
strcat(s,ob.s);
cout<<"\nConcatnated String is:"<<s;</pre>
```

```
int main()
```

#include<string.h>

```
{
```

}

۲

{

```
strings ob1, ob2;
char string1[10], string2[10];
cout<<"\nEnter First String:";
cin>>string1;
ob1.getstring(string1);
cout<<"\nEnter Second String:";
cin>>string2;
ob2.getstring(string2);
//Calling + operator to Join/Concatnate strings
ob1+ob2;
return 0;
```

}

Output

Enter First String:COMPUTER Enter Second String:SCIENCE Concatnated String is:COMPUTERSCIENCE

Points to Remember

- In C++, polymorphism is achieved through function overloading and operator overloading.
- The term overloading means a name having two or more distinct meanings.
- Overloaded function' refers to a function having more than one distinct meaning.
- Overloaded functions have same name but different signatures (Number of argument and type of argument)
- A function's argument list is known as a function signature
- Two function cannot be overloaded when the only difference is that one takes a reference parameter and the other takes a normal, call-by-value parameter.
- Ordinary functions as well member functions can be overloaded

- A class can have overloaded constructors where as destructor function cannot be overloaded.
- The mechanism of giving special meaning to an operator is known as operator overloading.
- Operator overloading provides new definitions for most of the C++ operators.
- Even user defined types (objects) can be overloaded.
- The definition of the overloaded operator is given using the keyword 'operator' followed by an operator symbol.
- We can overload all the C++ operators except the following:
- Scope resolution operator (::), sizeof (), Conditional operator (?:), Member selection(.) and Member pointer selector (*) operator

Evaluation



SECTION – A

256

۲

Choose the correct answer

- 1. Which of the following refers to a function having more than one distinct meaning?
 - (A) Function Overloading
 - (C) Operator overloading
 - (D) Operations overloading

(B) Member overloading

- 2. Which of the following reduces the number of comparisons in a program ?
 - (A) Operator overloading
 - (C) Function Overloading
- 3. void dispchar(char ch='\$',int size=10)

for(int i=1;i<=size;i++)
cout<<ch;</pre>

(B) Operations overloading

(D) Member overloading

{

}

+

.

How will you invoke the function dispchar() for the following input?

To print \$ for 10 times

(A) dispchar();	<pre>(B) dispchar(ch,size);</pre>
(C) dispchar(\$,10);	(D)dispchar('\$',10 times);

4. Which of the following is not true with respect to function overloading?

- (A) The overloaded functions must differ in their signature.
- (B) The return type is also considered for overloading a function.
- (C) The default arguments of overloaded functions are not considered for Overloading.
- (D) Destructor function cannot be overloaded.
- 5. Which of the following is invalid prototype for function overloading

(A) void fun (intx);	(B) void fun (intx);
void fun (char ch);	void fun (inty);
(C) void fun (double d);	(D) void fun (double d);
void fun (char ch);	void fun (inty);
	SECTION-B

Very Short Answers

- 1. What is function overloading?
- 2. List the operators that cannot be overloaded.
- 3. class add{int x; public: add(int)}; Write an outline definition for the constructor.
- 4. Does the return type of a function help in overloading a function?
- 5. What is the use of overloading a function?

SECTION-C

Short Answers

- 1. What are the rules for function overloading?
- 2. How does a compiler decide as to which function should be invoked when there are many functions? Give an example.
- 3. What is operator overloading? Give some examples of operators which can be overloaded.
- 4. Discuss the benefits of constructor overloading ?
- 5. class sale (int cost, discount ;public: sale(sale &); Write a non inline definition for constructor specified;

SECTION - D

Explain in detail

- 1. What are the rules for operator overloading?
- 2. Answer the question (i) to (v) after going through the following class.

```
class Book {
int BookCode; char Bookname[20];float fees;
public:
Book()
                         //Function 1
{ fees=1000;
 BookCode=1;
 strcpy(Bookname,"C++"); }
void display(float C)
                         //Function 2
{ cout<<BookCode<<":"<<Bookname<<":"<<fees<<endl; }
~Book()
                         //Function 3
{ cout<<"End of Book Object"<<endl; }
     Book (intSC, char S[], float F);
                                         //Function 4
};
```

- (i) In the above program, what are Function 1 and Function 4 combined together referred as?
- (ii) Which concept is illustrated by Function3? When is this function called/ invoked?
- (iii) What is the use of Function3?
- (iv) Write the statements in main to invoke function1 and function2
- (v) Write the definition for Function4.
- 3. Write the output of the following program

include<iostream>

using namespace std;

class Seminar

{ int Time;

public:

Seminar()

{ Time=30;cout<<"Seminar starts now"<<endl; }

void Lecture()

{ cout<<"Lectures in the seminar on"<<endl; }

Seminar(int Duration)

{ Time=Duration;cout<<"Welcome to Seminar "<<endl; }

Seminar(Seminar &D)

{ Time=D.Time;cout<<"Recap of Previous Seminar Content "<<endl;} ~Seminar()

```
{cout<<"Vote of thanks"<<endl; } };
int main()
{
    Sominor of o2(2) o2(o2);
}</pre>
```

```
{ Seminar s1,s2(2),s3(s2);
    s1.Lecture();
    return 0;
```

}

4. Answer the questions based on the following program

```
#include<iostream>
#include<string.h>
using namespace std;
class comp {
public:
char s[10];
void getstring(char str[10])
  { strcpy(s,str);
void operator==(comp);
};
void comp::operator==(comp ob)
{ if(strcmp(s,ob.s)==0)
 cout<<"\nStrings are Equal";</pre>
else
 cout<<"\nStrings are not Equal"; }</pre>
int main()
{ comp ob, ob1;
char string1[10], string2[10];
cout<<"Enter First String:";</pre>
cin>>string1;
ob.getstring(string1);
cout<<"\nEnter Second String:";</pre>
cin>>string2;
ob1.getstring(string2);
```

```
ob==ob1;
```

```
return 0; }
```

- (i) Mention the objects which will have the scope till the end of the program.
- (ii) Name the object which gets destroyed in between the program
- (iii) Name the operator which is over loaded and write the statement that invokes it.
- (iv) Write out the prototype of the overloaded

member function

- (v) What types of operands are used for the overloaded operator?
- (vi) Which constructor will get executed in the above program? Write the output of the program

CASE STUDY

۲

Suppose you have a Kitty Bank with an initial amount of Rs500 and you have to add some more amount to it. Create a class 'Deposit' with a data member named 'amount' with an initial value of Rs500. Now make three constructors of this class as follows:

- without any parameter no amount will be added to the Kitty Bank
- 2. having a parameter which is the amount that will be added to the Kitty Bank
- whenever amount is added an additional equal amount will be deposited automatically

Create an object of the 'Deposit' and display the final amount in the Kitty Bank.

Reference:

- Object Oriented Programming with C++ (4th Edition), Dr. E. Balagurusamy, Mc.Graw Hills.
- The Complete Reference C++ (Forth Edition), Herbert Schildt. Mc.Graw Hills.
- 3. The C++ Programming Language,BjarneStroustrup

۲