UNIT III

More about Functions

Function overloading, friend function, a function friendly to two classes, objects as function arguments.

Constructors & Destructors

Constructors, parameterized constructors, multiple constructors in a class, constructors with default arguments, copy constructors, dynamic constructors, destructors.

3.1 More About Functions

3.1.1 Function Overloading

Q1. What is Function Overloading? Explain about it.

Ans :

If any class have multiple functions with same names but different parameters then they are said to be overloaded. Function overloading allows you to use the same name for different functions, to perform, either same or different functions in the same class.

Function overloading is usually used to enhance the readability of the program. If you have to perform one single operation but with different number or types of arguments, then you can simply overload the function.

Ways to overload a function

1. By changing number of Arguments.
2. By having different types of argument.

Number of Arguments different

In this type of function overloading we define two functions with same names but different number of parameters of the same type. For example, in the below mentioned program we have made two sum() functions to return sum of two and three integers.

```
int sum (int x, int y)
{
    cout << x+y;
}
int sum(int x, int y, int z)
{
    cout << x+y+z;
}
```

Here sum() function is overloaded, to have two and three arguments. Which sum() function will be called, depends on the number of arguments.

```
int main()
{
    sum (10,20); // sum() with 2 parameter will be called
sum(10,20,30); //sum() with 3 parameter will be called
}
```

Different Datatype of Arguments

In this type of overloading we define two or more functions with same name and same number of parameters, but the type of parameter is different. For example in this program, we have two sum() function, first one gets two integer arguments and second one gets two double arguments.

```
int sum(int x,int y)
{
    cout<< x+y;
}
double sum(double x,double y)
{
    cout << x+y;
}
int main()
{
 sum (10,20);
 sum(10.5,20.5);
}
```

Q2. Write a program to demonstrate Function Overloading.

Ans :

```
#include<iostream>
usingnamespace std;
void display(int);
void display(float);
void display(int,float);
int main(){
int a =5;
float b =5.5;
    display(a);
    display(b);
    display(a, b);
return0;
}
    void display(intvar){
    cout <<"Integer number: "<<var<< endl;
}
    void display(floatvar){
    cout <<"Float number: "<<var<< endl;
}
    void display(int var1,float var2){
    cout <<"Integer number: "<< var1;
    cout <<" and float number:"<< var2;
}
```

Output

Integer number: 5

Float number: 5.5

Integer number: 5 and float number: 5.5

Here, the display() function is called three times with different type or number of arguments.

The return type of all these functions are same but it's not necessary.

Q3. Write a program to compute absolute value using function overloading.

Ans :

```
// Program to compute absolute value
// Works both for integer and float
#include<iostream>
usingnamespace std;
int absolute(int);
float absolute(float);
int main(){
int a =-5;
float b =5.5;
cout <<"Absolute value of "<< a <<" = "<< absolute(a)<< endl;
cout <<"Absolute value of "<< b <<" = "<< absolute(b);
return0;
}
    int absolute(intvar){
    if(var<0)
    var=-var;
    returnvar;
}
    float absolute(floatvar){
    if(var<0.0)
    var=-var;
    returnvar;
}
```

Output

Absolute value of -5 = 5

Absolute value of 5.5 = 5.5

3.1.2 Friend Function

Q4. How to declare the function as a friend function?

(OR)

Write the syntax to define friend function

Ans :

Declaration of fr
class class_name

{
... .
friend return_ty
(argument/s);
... .

}

Now, you can
normal function to
No friend keyword
class className

{
... .
friend return_
(argument/s);
... .

}

return_type f

{
... .

// Private and p
be accessed from
// this function
className.

... .

}

Q5. Write a
members
friend Fun

Ans :

```
#include<i
usingnames
// forward
class B;
class A {
private:
int numA;
public:
```

Ans :

Declaration of friend function in C++

```
class class_name
{
    ... .. ...
    friend return_type function_name
    (argument/s);
    ... .. ...
}
```

Now, you can define the friend function as a normal function to access the data of the class. No friend keyword is used in the definition.

```
class className
{
    ... .. ...
    friend return_type functionName
    (argument/s);
    ... .. ...
}
    return_type functionName(argument/s)
{
    ... .. ...
  // Private and protected data of className can be accessed from
  // this function because it is a friend function of className.
    ... .. ...
 }
```

Q5. Write a program for Addition of members of two different classes using friend Function

Ans :

```
#include<iostream>
usingnamespace std;
// forward declaration
class B;
class A {
private:
int numA;
public:
    A(): numA(12){}
    // friend function declaration
    friendint add(A, B);
};
    class B {
    private:
    int numB;
    public:
   B(): numB(1){}
    // friend function declaration
    friendint add(A , B);
};
    // Function add() is the friend function of classes A and B
    // that accesses the member variables numA and numB
    int add(A objectA, B objectB)
{
    return(objectA.numA + objectB.numB);
}
    int main()
{
    A objectA;
    B objectB;
    cout<<"Sum: "<< add(objectA, objectB);
    return0;
}
```

Output

Sum: 13

3.1.3 A Function Friendly to Two Classes

Q6. What is friend class? Write a syntax to define friend class.

Ans :

Friend class

Similarly like, friend function. A class can be made a friend of another class using keyword friend.

Syntax: .

```
..... ..... .....
class A{
    friend class B;     // class B is a friend class
    ..... ..... .....
```

```
}
class B{
        ..... ..... .....
}
```

When a class is made a friend class, all the member functions of that class becomes friend function. In this program, all member functions of class B will be friend function of class A. Thus, any member function of class B can access the private and protected data of class A.

If B is declared friend class of A then, all member functions of class B can access private data and protected data of class A but, member functions of class A cannot private and protected data of class B. Remember, friendship relation in C++ is granted not taken.

Q7. Write a program to find the height and width of square and rectangle using friend class

Ans :

```
# include <iostream>
using namespace std;
class Square;
class Rectangle {
    int width, height;
  public:
  Rectangle(int w = 1, int h = 1): width(w),
  height(h){}
  void display() {
  cout << "Rectangle: " << width * height
  << endl;
};
  void morph(Square &);
};
  class Square {
  int side;
  public:
  Square(int s = 1):side(s){}
  void display() {
      cout << "Square: " << side * side << endl;
  };
  friend class Rectangle;
};
  void Rectangle::morph(Square &s) {
  width = s.side;
  height = s.side;
}
  int main () {
  Rectangle rec(5,10);
  Square sq(5);
  cout << "Before:" << endl;
  rec.display();
  sq.display();
  rec.morph(sq);
  cout << "\nAfter:" << endl;
  rec.display();
  sq.display();
  return 0;
}
```

3.1.4 Objects as Function Arguments

Q8. How to pass objects as references in C++. Explain with an example.

Ans :

Objects are passed to a function as argument like variable of any primitive data type. There are 2 ways of passing object as argument to a function :

a) Pass by value

b) Pass by reference

Pass object by value to a function

In this, a copy of object is sent to the function, hence any changes made to the object inside the function do not affect the object used to call the function.

Pass object by reference to the function

In this, address of the object is passed to the function, hence the any changes made to the object inside the called function is reflected to the actual object.

Consider below program to understand use of object as function argument.

```
#include<iostream.h>
class time
{
    int hr,min;
    public:
    void gettime(int h,int m)
    {
        hr=h;
        min=m;
    }
    void puttime(void)
    {
        cout<<hr<<" hrs and ";
        cout<<min<<" mins "<<endl;
    }
    void sum(time, time); //declaration of
function with object as argument
};
void time :: sum(time t1, time t2)
{
    min=t1.min+t2.min;
    hr=min/60;
    min=min%60;
    hr=hr+t1.hr+t2.hr;
}
int main()
{
    time obj1,obj2, obje3;
    obj1.gettime(2, 45);
    obj2.gettime(3, 30);
    obj3.sum(obj1,obj2);  //obj3=obj1+obj2;
    cout<<" obj1= ";
    obj1.puttime();
    cout<<"obj2= ";
    obj2.puttime();
    cout<<"obj3= ";
    obj3.puttime();
    return 0;
}
```

Output

Obj1= 2 hrs and 45 mins

Obj2= 3 hrs and 30 mins

Obj3= 6 hrs and 15 mins

An object can also be passed to the non member functions, however this non member function cannot access the private data members of the class and can access only public member functions through the object passed as argument.

3.2 Constructors & Destructors

3.2.1 Constructors

Q9. Write a short note on constructors and destructors.

Ans :

C++ provides a special member function called the constructor which enables an object to initialize itself at the time of its creation. This is known as automatic initialization of objects. This concept of C++ also provides another member function called destructor which is used to destroy the objects when they are no longer required.

- A constructor is a method that has the same name as its class.
- A destructor is a method that has as its name the class name prefixed by a tilde, ~.
- Neither constructors nor destructors return values. They have no return type specified.
- Constructors can have arguments.
- Constructors can be overloaded.

- If any constructor is written for the class, the compiler will not generate a default constructor.
- The default constructor is a constructor with no arguments, or a constructor that provides defaults for all arguments.
- The container classes such as vector require default constructors to be available for the classes they hold. Dynamically allocated class arrays also require a default constructor. If any constructors are defined, you should always define a default constructor as well.
- Destructors have no arguments and thus cannot be overloaded.

Q10. What are constructors ? Explain them with syntax and example.

Ans :

Constructors

Constructors are special class functions which performs initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors iitialize values to object members after storage is allocated to the object.

Syntax:

```
class A
{
    int x;
    public:
    A();  //Constructor
};
```

While defining a contructor you must remeber that the name of constructor will be same as the name of the class, and contructors never have return type.

Constructors can be defined either inside the class definition or outside class definition using class name and scope resolution :: operator.

Example of constructor

```
class A
{
    int i;
    public:
    A(); //Constructor declared
};
    A::A()  // Constructor definition
{
    i=1;
}
```

Q11. Explain the types of constructors

Ans :

Types of Constructors

Constructors are of three types :

1. Default Constructor
2. Parametrized Constructor
3. Copy COnstructor

1. Default Constructor

Default constructor is the constructor which doesn't take any argument. It has no parameter.

2. Parameterised Constructors

These are the constructors with parameter. Using this Constructor you can provide different values to data members of different objects, by passing the appropriate values as argument.

3. COPY Constructor

Copy Constructor is a type of constructor which is used to create a copy of an already existing object of a class type. It is usually of the form X (X&), where X is the class name.he compiler provides a default Copy Constructor to all the classes.

3.2.2 Parameterized Constructors

Q12. How to pass arguments to constructors? Explain with an example

(OR)

Explain briefly about parameterised Constructors

Ans :

Arguments can be passed to constructors by defining the parameterised constructors.

Parameterised Constructors

These are the constructors with parameter. Using this Constructor you can provide different

values to data members of different objects, by passing the appropriate values as argument.

```
// parameterised constructor :
class Cube
{
int side;
public:
Cube(int x)
{
side=x;
}
};
int main()
{
Cube c1(10);
Cube c2(20);
Cube c3(30);
cout << c1.side;
cout << c2.side;
cout << c3.side;
}
```

OUTPUT : 10 20 30

By using parameterized construcor in above case, we have initialized 3 objects with user defined values. We can have any number of parameters in a constructor.

3.2.3 Multiple Constructors in a Class

Q13. Explain multiple constructors in a class.

(OR)

What is Constructor Overloading?

Ans :

Just like other member functions, constructors can also be overloaded. Infact when you have both default and parameterized constructors defined in your class you are having Overloaded Constructors, one with no parameter and other with parameter.

You can have any number of Constructors in a class that differ in parameter list.

```
// demonstration of overloading construcor
class Student
{
int rollno;
string name;
public:
Student(int x)
{
rollno=x;
name="None";
}
Student(int x, string str)
{
rollno=x ;
name=str ;
}
};
int main()
{
Student A(10);
Student B(11,"Ram");
}
```

In above case we have defined two constructors with different parameters, hence overloading the constructors.

In the above case if we write Student S; in main(), it will lead to a compile time error, because we haven't defined default constructor, and compiler will not provide its default constructor because we have defined other parameterized constructors.

Q14. Write a C++ Program to Overload the Constructor.

Ans :

```
#include <iostream.h>
#include<conio.h>
class MyClass
{
public:
int x;
```

```
    int y;
    // Overload the default constructor.
            MyClass()
    {
            x = y = 0;
    }
    // Constructor with one parameter.
            MyClass(int i)
    {
            x=y= i;
    }
// Constructor with two parameters.
            MyClass(int i, int j)
    {
            x=i;
            y=j;
    }
};
    void main()
{
            clrscr();
    My Class t; // invoke default constructor
     MyClass t1 (5); // use MyClass(int)
    MyClass t2(9, 10); // use MyClass(int, int)
     cout<< "t.x: " << t.x<< ",
     t.y: " <<t.y<< "\n";
    cout << "t1.x: " << t1.x << ",
    t1.y: " << t1.y << "\n";
    cout << "t2.x: " << t2.x << ",
    t2.y: " <<t2.y << "\n";
     getch();
}
```

3.2.4 Constructors with Default Arguments

Q15. Explain briefly about default constructor.

Ans :

Similar to functions, it is also possible to pass the arguments to the constructor with default arguments.

Fo ex:

Power(in t 9, int 3)

In the above example the default value of the first argument is 9 and second argument is 3

Default Constructor

Default constructor is the constructor which doesn't take any argument. It has no parameter.

Syntax :

class_name ()

{ Constructor Definition }

Example :

Write a program to demonstrate constructor with default arguments

```
# include <iostream.h>
# include<conio.h>
# include <iomanip.h>
class MyClass
{
    public:
     My Class (int a =1, int b=2, int c=3)
{
    My Class :: a=a;
    My Class :: b= b;
    My Class :: c=c ;
};
    void show_numbers(void)
{
 cout << a << ","<< b << " "<< c << "\n";
};
    private:
    int a, b, c;
};
    void main(void)
{
```

```
clrs cr ();
My Class one (1, 1, 1);
My Class defaults;
My Class happy (101, 101,101);
one.show_numbers();
defaults.show_numbers();
happy. show_numbers();
getch();
}
```

3.2.5 Copy Constructors

Q16. What is copy constructor? Explain with the help of syntax.

Ans :

Copy Constructor is a type of constructor which is used to create a copy of an already existing object of a class type. It is usually of the form X (X&), where X is the class name.he compiler provides a default Copy Constructor to all the classes.

Syntax of Copy Constructor

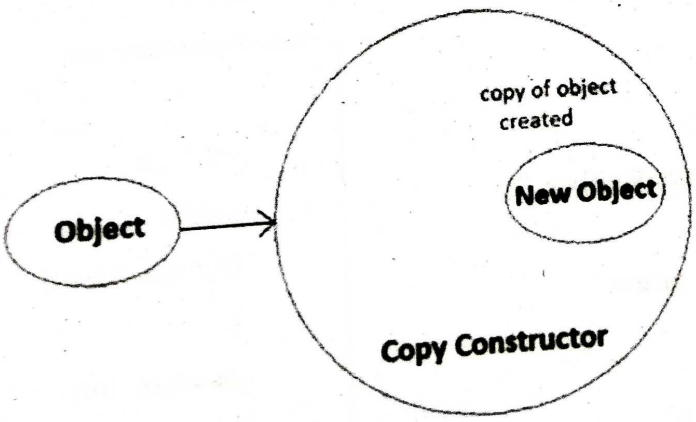
class-name (class-name &)

{

....

}

As it is used to create an object, hence it is called a constructor. And, it creates a new object, which is exact copy of the existing copy, hence it is called copy constructor.



Q17. Write a Program to Calculate Prime Number Using Constructor

Ans :

```
# include<iostream.h>
# include<conio.h>
class prime
{
int a,k,i;
public:
prime(int x)
        {
                a=x;
        }
        void calculate()
        {
          k=1;
          {
                for(i=2;i<=a/2;i++)
    if(a%i==0)
                {
                        k=0;
                        break;
                }
                else
                {
                        k=1;
                }
        }
      }
void show()
        {
          if(k==1)
  cout<< "\n\tA is prime Number. ";
          else
  cout<<"\n\tA is Not prime.";
        }
};
    void main()
{
    clrscr();
    int a;
    cout<<"\n\tEnter the Number:";
    cin>>a;
    prime obj(a);
    obj.calculate();
    obj.show();
    getch();
}
```

Sample Output:

Enter the number: 7

Given number is Prime Number

3.2.6 Dynamic Constructors

Q18. What is a dynamic constructor? Explain with suitable example.

Ans :

Allocation of memory during the creation of objects can be done by the constructors too. The memory is saved as it allocates the right amount of memory for each object.

Allocation of memory to object at the time of their construction is known as Dynamic Constructor of objects. In the dynamic constructor, new operator is used for allocation of memory.

By using this constructor, we can dynamically initialize the objects.

Example :

```
class DynamicCon
{
    int * ptr;
    public
    DynamicCon()
    {
    ptr=new int;
    *ptr=100;
```

```
    }
    DynamicCon(int v)
    {
    ptr=new int;
    *ptr=v;
    }
    int display()
    {
    return(*ptr);
    }
};
void main()
{
    DynamicCon obj1, obj2(90);
    cout<<"\n The value of obj's ptr is;
    cout<<obj1.display();
  cout<<"\nThe value of obj2's ptr is:"<<endl;
    cout<<obj2.display();
}
```

Output:

The value of obj1's ptr is:100

The value of obj2's ptr is: 90

3.2.7 Destructors

Q19. What is the use of Destructors? Write its syntax and explain

Ans :

Destructor is a special class function which destroys the object as soon as the scope of object ends. The destructor is called automatically by the compiler when the object goes out of scope.

The syntax for destructor is same as that for the constructor, the class name is used for the name of destructor, with a tilde ~ sign as prefix to it.

Syntax:

```
class A
{
    public:
    ~A();
};
```

Destructors will never have any arguments.

Example to see how Constructor and Destructor is called

```
    class A
{
    A()
 {
    cout << "Constructor called";
 }
    ~A()
 {
    cout << "Destructor called";
 }
};
    int main()
{
    A obj1; // Constructor Called
    int x=1
    if(x)
  {
    A obj2;  // Constructor Called
  }  // Destructor Called for obj2
}    // Destructor called for obj1
```

Short Question and Answers

1. What is Function Overloading?

Ans :

If any class have multiple functions with same names but different parameters then they are said to be overloaded. Function overloading allows you to use the same name for different functions, to perform, either same or different functions in the same class.

Function overloading is usually used to enhance the readability of the program. If you have to perform one single operation but with different number or types of arguments, then you can simply overload the function.

2. Friend function.

Ans :

Declaration of friend function in C++

```
class class_name
{
    ... .. ...
    friend return_type function_name
    (argument/s);
    ... .. ...
}
```

Now, you can define the friend function as a normal function to access the data of the class. No friend keyword is used in the definition.

```
class className
{
    ... .. ...
    friend return_type functionName
    (argument/s);
    ... .. ...
}
return_type functionName(argument/s)
{
    ... .. ...
// Private and protected data of className can be accessed from
// this function because it is a friend function of
className.
    ... .. ...
}
```

3. What is friend class?

Ans :

Friend class

Similarly like, friend function. A class can be made a friend of another class using keyword friend.

Syntax: .

```
..... ..... .....
class A{
    friend class B;     // class B is a friend class
    ..... ..... .....
}
class B{
    ..... ..... .....
}
```

When a class is made a friend class, all the member functions of that class becomes friend function. In this program, all member functions of class B will be friend function of class A. Thus, any member function of class B can access the private and protected data of class A.

If B is declared friend class of A then, all member functions of class B can access private data and protected data of class A but, member functions of class A cannot private and protected data of class B. Remember, friendship relation in C++ is granted not taken.

4. Constructors.

Ans :

C++ provides a special member function called the constructor which enables an object to initialize itself at the time of its creation. This is known as automatic initialization of objects. This concept

of C++ also provides another member function called destructor which is used to destroy the objects when they are no longer required.

- A constructor is a method that has the same name as its class.
- A destructor is a method that has as its name the class name prefixed by a tilde, ~.
- Neither constructors nor destructors return values. They have no return type specified.
- Constructors can have arguments.
- Constructors can be overloaded.

5. What is Constructor Overloading?

Ans :

Just like other member functions, constructors can also be overloaded. Infact when you have both default and parameterized constructors defined in your class you are having Overloaded Constructors, one with no parameter and other with parameter.

You can have any number of Constructors in a class that differ in parameter list.

```
// demonstration of overloading construcor
class Student
{
    int rollno;
    string name;
    public:
    Student(int x)
{
    rollno=x;
    name="None";
}
    Student(int x, string str)
{
    rollno=x ;
    name=str ;
}
};
    int main()
{
    Student A(10);
    Student B(11,"Ram");
}
```

In above case we have defined two constructors with different parameters, hence overloading the constructors.

6. Default constructor.

Ans :

Similar to functions, it is also possible to pass the arguments to the constructor with default arguments.

Fo ex:

Power(in t 9, int 3)

In the above example the default value of the first argument is 9 and second argument is 3

Default Constructor

Default constructor is the constructor which doesn't take any argument. It has no parameter.

Syntax :

class_name ()
{ Constructor Definition }

7. Copy constructor.

Ans :

Copy Constructor is a type of constructor which is used to create a copy of an already existing object of a class type. It is usually of the form X (X&), where X is the class name.he compiler provides a default Copy Constructor to all the classes.

Syntax of Copy Constructor

class-name (class-name &)
{
....
}

As it is used to create an object, hence it is called a constructor. And, it creates a new object, which is exact copy of the existing copy, hence it is called copy constructor.

8. Dynamic constructors.

Ans :

Allocation of memory during the creation of objects can be done by the constructors too. The memory is saved as it allocates the right amount of memory for each object.

Allocation of memory to object at the time of their construction is known as Dynamic Constructor of objects. In the dynamic constructor, new operator is used for allocation of memory.

By using this constructor, we can dynamically initialize the objects.

Example :

```
class DynamicCon
{
      int * ptr;
      public
      DynamicCon()
      {
      ptr=new int;
      *ptr=100;
      }
      DynamicCon(int v)
      {
      ptr=new int;
      *ptr=v;
      }
      int display()
      {
      return(*ptr);
      }
};
void main()
{
      DynamicCon obj1, obj2(90);
      cout<<"\n The value of obj's ptr is;
      cout<<obj1.display();
   cout<<"\nThe value of obj2's ptr is:"<<endl;
      cout<<obj2.display();
}
```

Output:

```
The value of obj1's ptr is:100
The value of obj2's ptr is: 90
```

9. Destructors.

Ans :

Destructor is a special class function which destroys the object as soon as the scope of object ends. The destructor is called automatically by the compiler when the object goes out of scope.

The syntax for destructor is same as that for the constructor, the class name is used for the name of destructor, with a tilde ~ sign as prefix to it.

Syntax:

```
class A
{
      public:
      ~A();
};
```

Destructors will never have any arguments.

Example to see how Constructor and Destructor is called

```
class A
{
      A()
{
      cout << "Constructor called";
}
      ~A()
{
      cout << "Destructor called";
}
};
      int main()
{
      A obj1; // Constructor Called
      int x=1
      if(x)
{
      A obj2;  // Constructor Called
}     // Destructor Called for obj2
}     // Destructor called for obj1
```


1. In which
   (a) retur
   (c) calle
2. What sho
   (a) void
   (c) both
3. Assigning
   (a) Fun
   (c) Bot
4. If an arg
   (a) fun
   (b) Fur
   (c) cor
   (d) No
5. Which
   (a) fri
   (c) fri
6. How mar
   (a) 1
   (c) 0
7. What i
   (a) A
   (c) A
8. A cons
   (a) o
   (c) n
9. For au
   (a) e
   (c) a
10. Which
   (a) C
   (c) D

Choose the Correct Answers

1. In which of the following we cannot overload the function? [a]
   (a) return function (b) caller
   (c) called function (d) main function

2. What should be passed in parameters when function does not require any parameters? [b]
   (a) void (b) blank space
   (c) both void & blank space (d) tab space

3. Assigning one or more function body to the same name is called _____. [b]
   (a) Function Overriding (b) Function Overloading
   (c) Both a and b (d) None of the above

4. If an argument to a function is declared as const, then _____. [a]
   (a) function can modify the argument.
   (b) Function can't modify the argument.
   (c) const argument to a function is not possible.
   (d) None of these

5. Which keyword is used to represent a friend function? [a]
   (a) friend (b) Friend
   (c) friend_func (d) Friend_func

6. How many parameters does a default constructor require? [c]
   (a) 1 (b) 2
   (c) 0 (d) 3

7. What is syntax of defining a destructor of class A? [b]
   (a) A(){} (b) ~A(){}
   (c) A::A(){} (d) ~A(){};

8. A constructor that accepts _____ parameters is called the default constructor. [c]
   (a) one (b) two
   (c) no (d) three

9. For automatic objects, constructors and destructors are called each time the objects. [a]
   (a) enter and leave scope (b) inherit parent class
   (c) are constructed (d are destroyed

10. Which of the following cannot be declared as virtual? [d]
    (a) Constructor (b) Destructor
    (c) Data Members (d) Both A and C

Fill in the Blanks

1. A friend class can access _____ members of other class in which it is declared as friend.
2. Friend function uses _____ operator to access members of a class using class object.
3. If we start our function call with default arguments means, _____ will be proceeding arguments.
4. _____ allows the user to initialize an object with the values of another object instead of supplying the same set of values again to initialize the object.
5. _____ are used in Classes to destroy an object after its lifetime is over i.e. to free resources occupied by that object.
6. Copy constructor must receive its arguments by _____.
7. _____ implicitly creates a default constructor when the programmer does not explicitly define at least one constructor for a class?
8. If the copy constructor receives its arguments by value, the copy constructor would _____.
9. A destructor takes _____ arguments.
10. How many times a constructor is called in the life-time of an object?

ANSWERS

1. private and protected members
2. dot operator
3. default arguments
4. Copy constructor
5. Destructors
6. only pass-by-reference
7. compiler
8. call itself recursively
9. no
10. Only once