Constructor & Destructor

Constructors are special members that automatically initialize class instances or when the object is created. A constructor is a member function that is executed automatically whenever object is created. A member function with the same name as its class is called constructor and it is used to initialize objects with a legal initial value.

RULES ABOUT DECLARING AND USING CONSTRUCTORS

a. Name of constructor must match name of its class.
b. A constructor is declared with no return type.
c. A constructor may not be static or virtual.
d. A constructor should have public access.
e. If a class does not declare a constructor, C++ declares default constructor for that class.
f. A class may have multiple constructors. (Polymorphism – Function Overloading)
g. A constructor with no parameter is called void or default constructor.
h. A constructor with one parameter that has class type is called copy constructor.
i. A constructor with one or more parameters is called parameterized constructor.

Declaration

```
class classname {
                private : int A,B;
                public:  …..;
                classname( ) //constructor
                { A=0; B=0;
                }
                };
```

A constructor is defined like other members of the class. It can be defined either inside class definition or outside class definition. When function is declared outside class definition, prototype must appear in class definition.

Example for above question if constructor is defined outside :

```
classname :: classname ( )
{ A=0;B-0;
}
```

TYPE OF CONSTRUTORS

There are three types of constructors:

i. Default Constructor Constructor that accepts no arguments is called default constructor. If a class has no explicit constructor, defined, compiler will supply a default constructor.
ii. Parameterized constructor- Constructor with arguments is called parameterized constructor. If there is only one parameter is there , it is called one argument constructor. If there is default argument is passed in function definition then the constructor is called constructor with default arguments. The constructor with default argument hides default constructor.
iii. **Copy Constructor -** A copy constructor is a constructor that can be used to initialize one object with value from another object of same class during declaration. Copy constructor takes a reference to an object of same class as itself as an argument.

DESTRUCTOR FUNCTION

It is inverse of constructor as it removes memory of an object which was allocated by constructor during creation of an object. It has same name as class with a ~ (tilde) as prefix. C++ automatically invokes a destructor when a class instance reaches end of its scope or goes out of existence.

RULES ABOUT USING AND DECLARING DESTRUTORS

a. Name of destructor and class must be same.

b. A destructor has no parameter list.
c. Destructor has no return type.
d. A class has only one destructor.
e. Destructor can't be declared static.
f. If class does not declare a destructor. C++ creates a default destructor.
g. Access is public for destructor.

Example

```
class  travel {
                int tid;
                char place[20];
                float fare;
                public:
                travel()        // Function -1 default Constructor
                {          tid=0;
                           Strcpy(place,"AGRA");   }

                travel(int t,float f)  // Funtcion -2 Parameterised Constructor
                {   tid=t; fare=f;
                }
                travel(int t,float f=20)  // Funtcion -3 Default argument Constructor
                {   tid=t; fare=f;
                }
                travel(travel &t)  // Function -4 Copy Construtor
                {
                tid=t.tid;
                strcpy(place,t.place);
                }
                ~travel()  //Funtion 5 - Destrutor
                {  cout<<" Scope Over……..";
                }
      };
//Calling statements
void main()
{
travel t;   //default constructor call statement
travel t1(2,56); // parameterized constructor call statement
travel t2(6); //default argument constructor call statement
travel t3=t1; // copy constructor call statement
travel t4(t2); // copy constructor call statement
} //Destructor invokes automatically when object scope is over
```

Note –

1. **Polymorphism / function overloading is implemented in above code, as constructors can be overloaded.**
2. **Constructor gets automatically invoked at the time of object declaration and destructor gets invoked when object scope is over**
3. **Destructor function cannot be overloaded.**
4. **If a class has an array of objects then it is must to have default constructor in the class with other constructors defined in the class.**

5. **Rules for finding best match in case of overloaded function are also applicable on overloaded constructors.**

Q. What is copy constructor? Discuss two situations when a copy constructor is automatically invoked giving appropriate examples.

Copy constructor is a constructor function (Function with the same name as the class) and used to make deep copy of objects.

There are 3 important places where a copy constructor is called.

i. When an object is created from another object of the same type eg.
ii. When an object is passed by value as a parameter to a function. Eg. Void Funct(travel t); i.e. call by value method is used and the copy of the passed argument is created. Hence copy constructor is invoked.
iii. When an object is returned from a function.

If a copy constructor is not defined in a class, the compiler itself defines one. This will ensure a shallow copy. If the class does not have pointer variables with dynamically allocated memory, then one need not worry about defining a copy constructor. It can be left to the compiler's discretion.

```
class time
{ int hr;
int min;
public:
time ( ) // ******** default constructor function ********
{ hr = min = 0;}
void gettime(int x,int y) {hr=x;min=y;}
void showtime( ) { cout <<" : " <<<"\n";} };
time (time &t) //copy construtor
{
hr=t.hr;
min=t.min;
}
time addtime( time T1, time T2)
{ time T3;
T3.hr = T1.hr + T2.hr;
T3.min = T1.min + T2.min;
if (T3.min >= 60)
{ T3.min =T3.min - 60;
T3.hr = T3.hr+1;
}
return T3;
}

void main( )
{ time X1 (5,45); // Parameterized constructor Invoked
time X2 (7,40); // Parameterized constructor Invoked
Time X4=X2; //copy constructor invoked
time X3=addtime(X1,X2); // Copy Constructor invoked
X3.showtime( ); // This will show the time as 13 : 25
}
```

Q. Why reference of an object is passed as parameter to the copy constructor.
[Hint] To prevent creation of multiple copies of object.
Q. What is temporary instance?
A temporary instance of a class is an object without any name. it leaves in memory during the execution of the only statement in which it is used. It is deleted from memory immediately thereafter. For eg. In case of explicit call the temporary instance is created.
travel T= travel(5,20);
Read Member – Initialization List from Book

SOLVED PROBLEMS

a. Answer the questions (i) and (ii) after going through the following program: 2

```
#include <iostream.h>
#include<string.h>
class bazaar
{ char Type[20] ;
char product [20];
int qty ;
float price ;
bazaar() //function 1
{
strcpy (type , .Electronic.) ;
strcpy (product , .calculator.);
qty=10;
price=225;
}
public :
void Disp() //function 2
{
cout<< type <<.-.<<product<<.:. <<qty<< .@. << price << endl ;
}
};
void main ()
{
Bazaar B ; //statement 1
B. disp() ; //statement 2
}
```

(i)Will statement 1 initialize all the data members for object B with the values given in the function 1 ? (YES OR NO). Justify your answer suggesting the correction(s) to be made in the above code.
Ans: No. The reason is the constructor should be defined under the public visibility label.
(ii) What shall be the possible output when the program gets executed ? (Assuming, if required _ the suggested correction(s) are made in the program).
Ans: Possible Output: Electronic.Calculator:10@225

b. Answer the question (i)and (ii)after going through the following class: 4

```
class Maths
{
char Chapter[20]
int Marks;
```

```
public:
Maths() //Member Function 1
{ strcpy (Chapter, “Geometry”);
Marks=10;
cout <<”Chapter Initialised “;
}
-Maths() //Member Functions 2
{
cout<<”Chapter Over”;
}
Maths(int M,char t[]) // Member Function 3
{
 strcpy (Chapter, t);
Marks=M;
}
Maths(Maths & M); // Member Funtcion 4
};
```

(i)Name the specific features of class shown by member Function 1 and Member Function 2 in the above example.

Ans: Member function 1 is a (non-parameterized or default constructor),which will be executed automatically at the time of creation of an object of class Maths).

Member function 2 is a destructor (,which will be executed automatically at the time of destruction of an object of class Maths).

(ii)How would Member Function 1 and Member Function 2 get executed ?

Ans: They will be executed automatically.

Member function 1 will be executed at the time of creation of an object of class Maths. Member function 2 will be executed at the time of destruction of an object of class Maths.

(iii) Give complete function of function 4.

Ans:
```
Maths::Maths(Maths &M)
    {   Marks = M.Marks;
    Strcpy(Marks,M.Marks);
    }
```

(iv) Give statements that would call Function 1 and Function 3.

Ans.
```
Maths M; // Function 1 call
    Maths M(20,”Computer Sc.”); // Function 3 call
```

Practice Questions

1. Define a class Tour in C++ with the description given below : (4)

Private Members:

TCode	of type string
No of Adults	of type integer
No of Kids	of type integer
Kilometers	of type integer
TotalFare	of type float

Public Members:

- A constructor to assign initial values as follows:
  TCode with the word “NULL”
  No of Adults as 0
  No of Kids as 0

Kilometers as 0
TotalFare as 0

- A function AssignFare() which calculates and assigns the value of the data member Totalfare as follows :

For each Adult

Fare (Rs)	For Kilometers
500	>=1000
300	<1000 & >=500
200	<500

For each Kid the above Fare will be 50% of the Fare mentioned in the above table. For Example:
If Kilometers is 850, Noofadults =2 and NoofKids =3 Then TotalFare should be calculated as
Numof Adults *300+ NoofKids *150 i.e., 2*300+ 3 *150 =1050

- A function EnterTour() to input the values of the data members TCode, NoofAdults, NoofKids and Kilometers ; and invoke the AssignFare() function.
- A function ShowTour() which displays the content of all the data members for a Tour.

2. Answer the questions (i) and (ii) after going through the following class : 2

```
class Science
{ char Topic[20] ;
int Weightage ;
public :
Science () //Function 1
{ strcpy (Topic, "Optics") ;
Weightage =30
cout<<"Topic Activated";
}
~Science() //Function 2
{ cout<<"Topic Deactivated"; }
};
```

(i)Name the specific features of class shown by Function 1 and Function 2 in the above example.
(ii)How would Function 1 and Function 2 get executed ?

3. Answer the following questions (i) and (ii) after going through the following class. 2

```
class Interview
{
int Month;
public:
interview(int y) {Month=y;} //constructor 1
interview(Interview&t); //constructor 2
};
```

(i) create an object, such that it invokes Constructor 1.
(ii) write complete definition for Constructor 2.

4. Define a class Competition in C++ with the following descriptions:

Data Members:

Event_no	integer
Description	char(30)
Score	integer
qualified	char

Member functions:

- A constructor to assign initial values Event_No number as 101, Description as "State level",

Score is 50 and qualified as 'N'.

- Input(), To take the input for event_no, description and score.
- Award(int), To award qualified as 'Y', if score is more than the cutoffscore passed as argument to the function else 'N'.
- Show(), To display all the details.

5. Answer the questions (i) and (ii) after going through the following class : **3**

```
class number
{ float M;                  char str[25];
public:
number( )                   //Function 1
{ M=0;  str='\0';}
number(number &t);              //Function 2
~number() ;// Function 3
};
```

i) Write c++ statement such that it invokes Function 1.
ii) Complete the definition for Function 2.
iii)Give difference between Function 1 and Function 3.