Static member variables and functions

Static member variables:-

As we know that each object has its separate set of data members variables in memory. The member functions are created only once and all objects share the functions. No separate copy of function of each object is created in the memory like data member variables.

 It is possible to create common member variables like functions using the static keyword. Once a data member variable is declared as static only one copy of that member is created for the whole class. The ‘static’ is a keyword used to preserve value of a variable. When a variable is declared as static it is initialized to zero. A static function or data element is only recognized inside the scope of the present class.

 In the earlier version of Turbo C++,it was not necessary to define static data member explicitly. It was linker responsibility to find undefined static data and allocate them required memory without showing error message. In the new version of Turbo C it is necessary to explicitly define static members.

Syntax:-

Static variable declaration;

Static int num;

If a local variable is declared with the static keyword it preserve the last value of the variable. A static data item is helpful when all the objects of the class share the common data. The static variable is accessible within the class but its value remains in the memory throughout the whole program.

 The class and the scope of the static member variable is defined outside the class declaration due to following reasons.

\* The static data members are associated with the class not with any object.

\*The static data members are stored individually rather than an element of an object.

\*The static data member variables must be initialized otherwise the linker will generate an error.

\*The member for static data is allocated only once.

\*Only one copy of static member variable is created for the whole class for any number of objects. All the objects have common static data member.

Static member function

Like member variables function can also be declared as static. When a function is defined as static it can access only static member variable and static member function of the same class. The non static members are not available to these functions. The static member function is declared in public section can be invoked using its class name without using its object. The ‘static’ keyword makes the function free from the individual objects of the class and its scope is global in the class without creating anyside effect for other part of the program. We must follow the following points while declaring the static function.

1>Just one copy of static member is created in the memory for entire class. All objects of class share the same copy of static member.

2>Static member function can access only static data member or function.

3>Static member function can be invoked using class name with scope resolution operator.

4>It is also possible to invoked the static member function using objects.

5>When one of the object changes the value of data member variables,the effect is visible to all the objects of the class.

write a program to declare static member and display the value of static data member.

#include<iostream.h>

#include<conio.h>

class number

{

 private:

 static int c;

 public:

 void count()

 {

 ++c;

 cout<<"\n the value of c="<<c;

 }

};

int number::c=0;

void main()

{

 clrscr();

 number n1,n2,n3;

 n1.count();

 n2.count();

 n3.count();

 getch();

}

 write a program to show the difference between static and non static data member.

#include<iostream.h>

#include<conio.h>

class number

{

 private:

 int k;

 static int c;

 public:

 void input()

 {

 k=0;

 }

 void count()

 {

 ++k;

 ++c;

 cout<<"\n the vlaue of c="<<c <<"and the value of k="<<k;

 }

};

int number::c=0;

void main()

{

 clrscr();

 number n1,n2,n3;

 n1.input();

 n2.input();

 n3.input();

 n1.count();

 n2.count();

 n3.count();

 getch();

}

Static private member function:-

Static member function can also be declared in private section. The private static function must be invoked using static public function.

Static public member variable:-

The static public member variable can also be accessed in main function like other variables. The static member variable can be accessed using class name and scope resolution operator. The scope resolution operator is also used when variable of same name are declared in global or local scope.

**Static object:-**

In C it is common to declare variable static that gets initialized to zero. The object is a composition of one or more member variables. There is a mechanism called constructor to initialize member variables of the object with desired values. The keyword static can be used to initialized all class data members with zero. Declaring object itself as static can do it thus all its associated members get initialized to zero.

Syntax:-

Static classname objectname;

Example:-

Static number n;

Here n is the object of number class which is declared as static.

write a program to declare static member function and call them from main function.

#include<iostream.h>

#include<conio.h>

class number

{

 private:

 static int c;

 public:

 static void count()

 {

 ++c;

 }

 static void display()

 {

 cout<<"\n the value of c="<<c;

 }

};

int number::c=0;

void main()

{

 clrscr();

 number::count();

 number::display();

 number::count();

 number::display();

 getch();

}

write a program to implement private static member function.

#include<iostream.h>

#include<conio.h>

class number

{

 private:

 static int c;

 static void count()

 {

 ++c;

 }

 public:

 static void display()

 {

 count();

 cout<<"\n the value of c ="<<c;

 }

};

int number::c=0;

void main()

{

 clrscr();

 number:: display();

 number ::display();

 getch();

}

write a program to implement static object.

#include<iostream.h>

#include<conio.h>

class number

{

 private:

 int a,b;

 public:

 void display()

 {

 cout<<"\n the value of a ="<<a;

 cout<<"\n the value of b="<<b;

 }

};

void main()

{

 clrscr();

 static number n;

 n.display();

 getch();

}