Name of Teacher: Miss Radhika M. Patil

Class: B.Sc. Computer Science (Entire)- II **Semester : 3**

Course Title: Object Oriented Programming using C++

Static Data Member:

- Data members are the variables declared inside the class .
- Static is a keyword in C++ used to give a special characteristic to an element.
- Static elements are allocated the memory only once in a program and they have a scope till the program lifetime.
- Static data members of a class are those members which are shared by all the objects.
- Static data member has a single piece of storage and is not available as separate copy with each object like other non static data members.
- Static data members must be initialized explicitly always outside the class. If not initialized, then linker will give an error.
- Static data member is a data member which is shared by all the objects of a class , so only one copy of static data member is kept in memory.

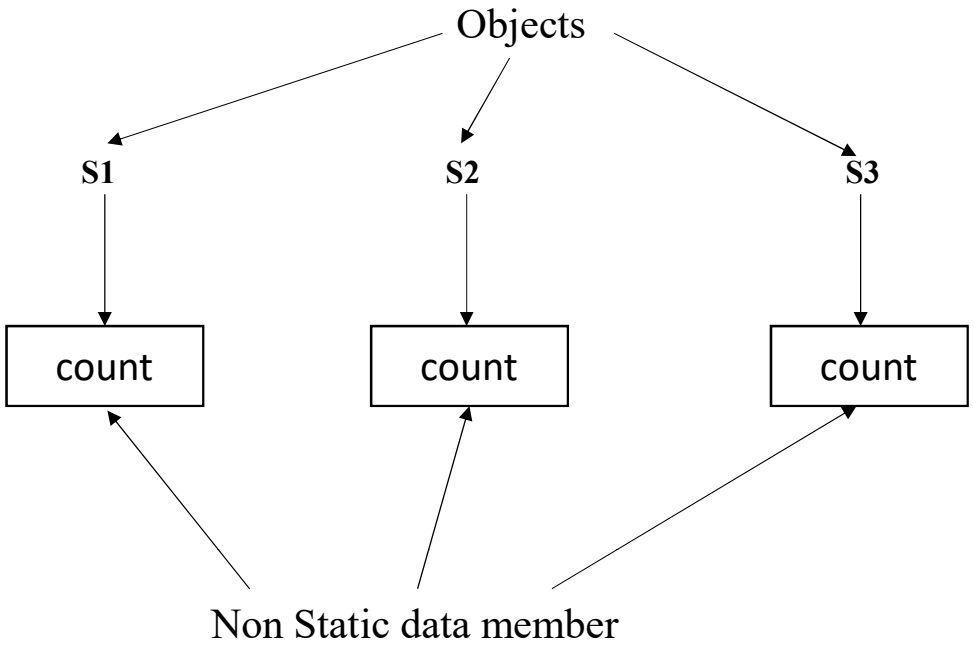

Fig. Memory Allocation of non static data members for objects

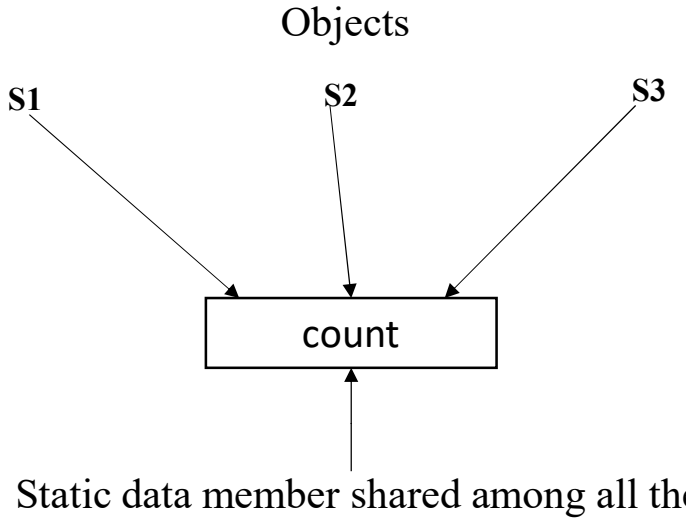

Fig a. Memory Allocation of static data members for objects

Properties of static data member:

1. Only one copy of static data member is created and is shared by all objects of a class , no matter how many objects are created.
2. It is initialized when first object of a class is created, no other initialization is allowed here.
3. Static data members are normally used to maintain the values which are **common to the entire class.**

Note: Static data members are not initialized using constructor because they are not dependant on object initialization.

- **Syntax for static data member declaration is as follows:**

Syntax:

```
static data_type var_name;
```

Example:

```
static int count;
```

- **Initialization of static data member:**

Syntax:

```
data_type class_name::var_name=value;
```

Example:

```
int Sample::count=10;
```

Once we define static data member, user **can not redefine it.**

Example:

```
class Sample
{
            static int count;
            int num;
   public:
            void getdata(int);
            void getcount();
};
void Sample::getdata(int x)
{
     num=x;
     cout<<"\n num="<<num;
     count++;
}

void Smaple::getcount()
{
     cout<<"\n count="<<count;

}

int Sample::count=0;

void main()
{
     Sample s1,s2,s3;
     s1.getcount();
     s2. getcount();
     s3.getcount();
     s1.getdata(100);
     s2. getdata(200);
     s3.getddata(300);
     cout<<"\n After reading data:";
     s1.getcount();
     s2. getcount();
     s3.getcount();
getch();
}
```

Output:

count=0

count=0

count=0

num=100

num=200

num=300

count=3

count=3

count=3

Static Member Function:

- Like member variable or data member, member function can also be declared as static.
- Static member function can **access only static data members and function of same class**.
- The non-static members are not available to these function.
- The static member function can be **invoked/called** using its **class name, without using its object**.
- Static keyword makes function free from individual object of class and its scope is global in the class.

Features of static member function:

1. Static member function can access only static data member of function.
2. Static member function can be invoked using class name and not using object of a class.
3. When one of the object changes the value of static data member then effect is visible to all objects of class.

- **Defining static member function:**

Static member function is defined by using keyword static before member function to be declared as static.

Syntax:

```
static return_type function_name()
  {
      //statements;
  }
```

- **Accessing or Invoking static member function:**

A normal member function is accessed using object and dot operator (.). It can be accessed using class name and scope resolution operator (::).

Syntax:

```
class_name::fun_name();
```

Example:

```
Sample::display();
```

Program:

```
class Test
{
        static int count;
        int code;
    public:
        void setcode()
        {
            code=++count;
        }
        void showcode()
        {
            cout<<"\n Object No.:"<<code;
        }
        static void showcount()
        {
            cout<<"\n Count="<<count;
        }
};

int Test::count=0;

void main()
{
    Test t1,t2;
    t1.setcode();
    t2.setcode();
    Test::showcount();
    Test t3;
    t3.setcode();
    Test::showcount();
    t1.showcode();
    t2.showcode();
    t3.showcode();
getch();
}
```

Output:

Count=2

Count=3

Object No.:1

Object No.:2

Object No.:3

Array of Objects:

- Like array of other user defined data types, array of type class can also be created.
- Array of type class contains object of a class and its individual elements. Thus an array of class type is known as array of objects.
- An array of object is declared in same way as an array of any built in data type.

Syntax for declaring an array of object is:

```
class_name  obj_name[size];
```

Example:

```
Student s[5];
```

Program: Program to demonstrate concept of array of objects.

```
class Books
{
            char title[20];
             float price;
   public:
              void getdata();
              void putdata();
};

void Books::getdata()
{
     cout<<"\n Enter title:";
     cin>>title;
     cout<<"\n Enter price:";
     cin>>price;
}

void Books::putdata()
{
      cout<<"\n Title:"<<title;
      cout<<"\n Price:"<<price;
}
```

```
void main()
{
      Books b[5];
      for(int i=0;i<2;i++)
      {
            cout<<"\n Enter details of book:"<<i+1;
             b[i].getdata();
      }

       for(int j=0;j<2;j++)
       {
                cout<<"\n Books are:";
                 b[j].putdata();
       }
  getch();
}
```

Output:

```
Enter details of book 1:
Enter Title: RDBMS
Enter price: 250
Enter details of book 2:
Enter Title: C++
Enter price: 200
Enter details of book 3:
Enter Title: Operating System
Enter price: 280
Books are:
Title: RDBMS
Price: 250
Title: C++
Price: 200
Title: Operating System
Price: 280
```

In above program array being type class Books with size 5 is declared. When array of object is declared, memory is allocated in same way as to multidimensional array. E.g. For array b separate copy of title and price is created for each member i.e. b[0], b[1], b[2], b[3] and b[4]. However, member functions are stored at different places in memory and share among all array elements.

Assignment: Write an OOP to input and display student details using array of object.

Solve this assignment and attach the screenshot of program in pdf file along with the notes.

Friend Function:

- When data is declared as private inside the class then its not accessible from outside the class.
- A function that is not a member or an external class will not be able to access private data.
- A programmer may have a situation where programmer needs to access private data from non-member function and external classes.
- For handling such situation concept of friend function is useful.
- Friend function is ordinary function or member of another class.
- A class can allow non-member function and other classes to access its own private data by making them friends.

How to define and use friend function in C++:

- A friend function is written as any other normal function except function declaration of this function is preceded with friend keyword.
- Friend function must have a class to which it is declared as a function passed to it in argument.

Syntax:

```
friend return_type fun_name (argument);
```

- **Important points regarding friend function**

1. Keyword friend is placed only in function declaration of the friend function and not in function definition
2. It is possible to declare friend function in any number of classes
3. It is possible to declare print function as either private or public
4. Friend function can be invoked without using an object of a class
5. Friend function use an object of a class as its argument.
6. Friend function is not a member of a class in which it is declared.

Program:

```
class FriendDemo
{
            int a,b;
    public:
            void test()
             {
                        a=100;
                        b=200;
             }
        friend void compute(FriendDemo);
};

void compute(FriendDemo f)
{
      int c;
      c=f.a+f.b;
      cout<<"\n Addition is:"<<c;
}

void main()
{
        FriendDemo d;
        d.test();
        compute(d);
        getch();
}
```

Output:
Addition is: 300

Here, function compute() is a non member function of class FriendDemo. In order to make this function have an access to private data i.e. a and b of class FriendDemo, declare it as friend so that it can access the private data of class FriendDemo.

- **Friend Class:**

1. Like friend function, we can also declare the class as friend of another class.
2. This will allow that the second one can access private and protected members of first class.

Example:

```
class Square;
class Rectangle {
            int w, h;
    public:
            int area()
            {
                return(w*h);
             }
            void convert(Square);
};
```

```
class Square
{
            int side;
  public:
            void setSide(int x)
            {
                    side=x;
            }
      friend class Rectangle;
};

void Rectangle::convert(Square s)
{
      w=s.side;
       h=s.side;
}
```

```
void main
{
      Square sq;
      Rectangle r;
      sq.setSide(10);
      r.convert(sq);
      int a=r.area();
       cout<<"\n Area of Rectangle is:"<<a;
       getch();
}
```

Output:

Area of Rectangle is: 100

In this example , we have declared class Rectangle as friend of Square class so that Rectangle can access private and protected members of a Square class . But Square class can not access protected and private members of Rectangle class.

THANK YOU…