CLASSES AND OBJECTS IN C++

C STRUCTURES

A structure is a convenient tool for handling a group of logically related data items.

It is a user defined data type with a template that serves to define its data properties.

Once the structure type is created we can create variables of that type using declaration similar to built in type declaration.

C structure does not support data - hiding

SPECIFYING A CLASS

Class is a group of objects that share common properties and relationships .

In C++, a class is a new data type that contains member variables and member functions that operates on the variables.

A class is defined with the keyword class.

It allows the data to be hidden, if necessary from external use.

When we defining a class, we are creating a new abstract data type that can be treated like any other built in data type.

Generally a class specification has two parts:-

a) Class declaration

b) Class function definition

The class declaration describes the type and scope of its members. The class function definition describes how the class functions are implemented.

SYNTAX

```
class class-name
{
        private:
                variable declarations;
                function declaration ;
        public:
                variable declarations;
                function declaration;
};
```

CREATING OBJECTS

- Once a class has been declared we can create variables of that type by using the class name.
- Example: item x;
- creates a variables x of type item.
- In C++, the class variables are known as objects. Therefore x is called an object of type item.

```
class item
        { ----------
          ----------
        ---------- }x ,y ,z;
```

would create the objects x ,y ,z of type item.

DEFINING MEMBER FUNCTION

- Member can be defined in two places
  - Outside the class definition
  - Inside the class definition
- Outside the class definition
  - Member function that are declared inside a class have to be defined separately outside the class.
  - Their definition are very much like the normal functions.
  - An important difference between a member function and a normal function is that a member function incorporates a membership.
  - The member ship label class-name :: tells the compiler that the function function - name belongs to the class class-name . That is the scope of the function is restricted to the classname specified in the header line. The :: symbol is called scope resolution operator

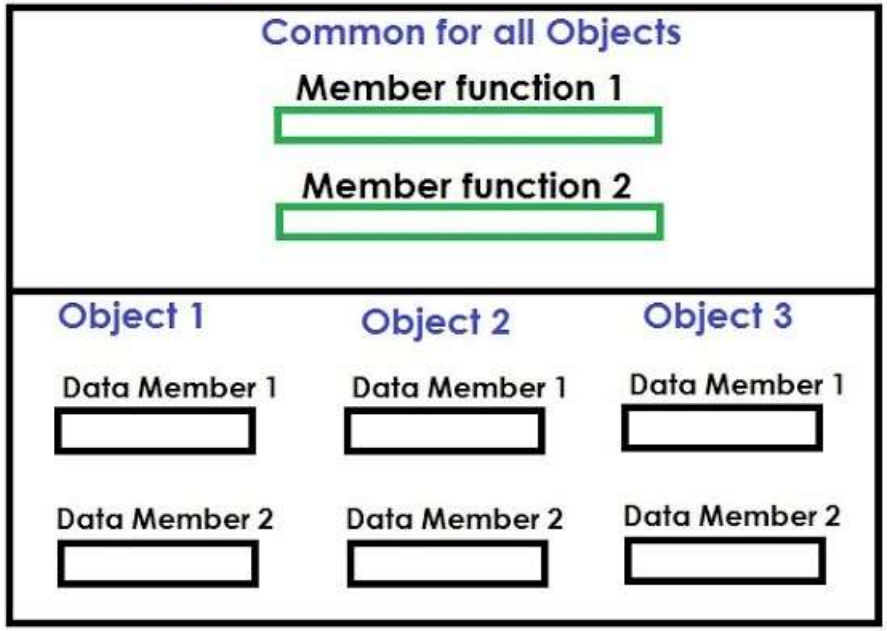
- INSIDE THE CLASS DEFINATION
  - Another method of defining a member function is to replace the function declaration by the actual function definition inside the class .
  - Example:

```
class item {
        lnt number;
        float cost;
        public:
                void getdata (int a ,float b);
                void putdata(void)
                        {
                                cout<<number<<<cost<<endl;
                        }
        };
```

NESTING OF MEMBER FUNCTION

- A member function can be called by using its name inside another member function of the same class. This is known as nesting of member functions
- Although it is a normal practice to place all the data items in a private section and all the functions in public, some situations may require contain functions to be hidden from the outside calls.
- A private member function can only be called by another function that is a member of its class. Even an object can not invoke a private function using the dot operator

Memory Allocation For Objects



Example:

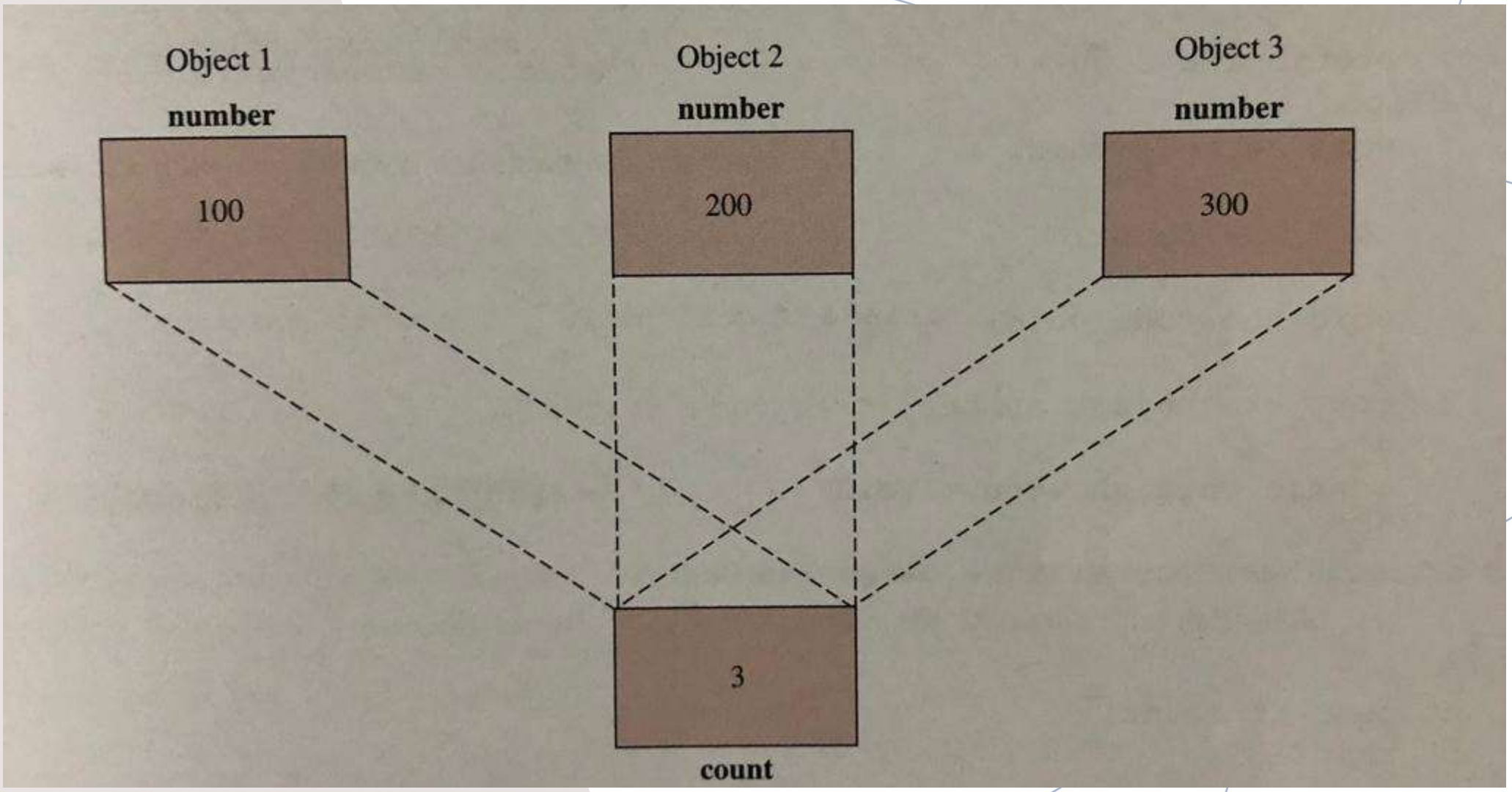
```
class product
{
        int pno;
        float cost;
        public:
        void getdata(int a, float b)
        {
                pno=a;
                cost=b;
        }
        void putdata()
        {
                cout<<pno<<"   "<<cost;
        }
};
void main()
{
        product p1,p2,p3;
        ....
}
```

STATIC

- **Static Variables :** Variables in a function, Variables in a class
  Static Members of Class : Class objects and Functions in a class
- **Static variables in a Function**: When a variable is declared as static, space for **it gets allocated for the lifetime of the program**. Even if the function is called multiple times, space for the static variable is allocated only once and the value of variable in the previous call gets carried through the next function call.

STATIC DATA MEMBER

- A data member of a class can be qualified as static
- Static data members have special characteristics
  - It is initialized to zero when the first object of its class is created.No other initialization is permitted
  - Only one copy of that member is created for the entire class and is shared by all the objects of that class, no matter how many objects are created
  - It is visible only with in the class but its life time is the entire program. Static variables are normally used to maintain values common to the entire class. For example a static data member can be used as a counter that records the occurrence of all the objects


Object 1
number
100
Object 2
number
200
Object 3
number
300
3
count

STATIC MEMBER FUNCTION

- **Static functions in a class**: Just like the static data members or static variables inside the class, static member functions also does not depend on object of class.
- We are allowed to invoke a static member function using the object and the '.' operator but it is recommended to invoke the static members using the class name and the scope resolution operator.
  Static member functions are allowed to access only the static data members or other static member functions, they can not access the non-static data members or member functions of the class.

OBJECTS AS FUNCTION ARGUMENTS

- Like any other data type, an object may be used as A function argument.
- This can be done in two ways
  - 1. A copy of the entire object is passed to the function.
  - 2. Only the address of the object is transferred to the function
- The first method is called pass-by-value. Since a copy of the object is passed to the function, any change made to the object inside the function do not effect the object used to call the function.
- The second method is called pass-by-reference . When an address of the object is passed, the called function works directly on the actual object used in the call. This means that any changes made to the object inside the functions will reflect in the actual object .The pass by reference method is more efficient since it requires to pass only the address of the object and not the entire object.

FRIENDLY FUNCTIONS

- The function declaration should be preceded by the keyword friend.
- The function is defined else where in the program like a normal C ++ function .
- The function definition does not use their the keyword friend or the scope operator :: .
- The functions that are declared with the keyword friend are known as friend functions.
- A function can be declared as a friend in any no of classes.
- A friend function, as though not a member function , has full access rights to the private members of the class