Class

- A class is a way to bind data and associated function together.
- A class is an expanded concept of a data structure, instead of holding only data , it can hold both data and function.
- The data is to be hidden from external use.
- Classes are generally declared using the keyword **class**, with the following format:

```
class class_name
{
 private:
   variable declaration;
 public:
    function declaration;
 ...
};
```

- The body of the declaration can contain members that can be either data or function declaration, and optionally access specifier.

- The variable declared inside the class is known as **data member** and function are known as **member functions**.

- Access specifier are keyword in object oriented language that set the accessibility of classes, method and other member.

- Access specifier is one of the following keyword: public, private, protected.

- These specifier modify the access rights that the member following them acquire:
  - **private** members of class are accessible only from within other member of same class or from their friends.
  - **protected** members are accessible form members of their same class and from their friends but also from members of their derived classes.
  - **public** members are accessible from anywhere the object is visible.

- By default, all members of class declared with the class keyword have private access for all its member. Therefore, any member that is declared before one other class specifier automatically has private access.

Object

- Once a class has been created, we can create variable of that type(class type) by using following syntax which is called **object.**
- Syntax:

  class_name variable_name;

Ex:

  student s;

- we can create any number of objects belonging to that class by declaring more than one object in one statement. This statement are written in main().

- The objects can also be defined by placing their name immediately after the closing brace of the class.

- Syntax:

```
class class_name
{
....
}object1,object2,...;
```

- Ex:

```
class student
{
   ...
}s1,s2;
```

- <u>Accessing class member:</u>

  A object can be declared in the main(),and member functions are declared in class in public section so always a member function can be called by using object.

- Syntax:

  object_name.member_function(arguments);

Ex:

  s.getdata();

- A data member can also be access by using object only , if data member is declared as public.

- If data member is declared private then you can not access it by using object directly in object.

Defining member function

- A member function can be defined in two places in the class:
  1. inside the class definition
  2. outside the class definition

1) **Inside the class definition:**
   To write a member function inside the class instead of only declaration(prototype).

Ex:

```
class item
{
    int num;
   float cost;
```

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

2) Outside the class definition:

- To write function we need to declare function inside the class and definition(function body) is written outside the class.
- The general form of a member function definition:

```
return_type  class_name::function_name(arument)
{
    function body
}
```

- The membership label class_name :: tells the compiler that the function function_Name belongs to the class class_name.
- **::** is *scope resolution operator*.

Ex:

```
....
void item::getdata(int a, float b)
{
    number=a;
    coat=b;
}
void item::putdata(void)
{
  cout<<"number  "<<number;
  cout<< "cost "<<cost;
}
...
```

Nesting Member Function

- A member function can be called by using its name inside another member function the same class is called **nesting member function.**
- Ex:

```
#include<iostream>
using namespace std;

class number
{
 private:
    int a,b,s1,s2;
 public:
 int getdata(int m,int n);

 int sum();
 int sub();
```

```
int show()
{
   cout<<"\n Enter number1: ";
   cin>>a;
   cout<<"\n Enter number2: ";
   cin>>b;
   cout<<"\n Answer of Addition:"<<sum()<<endl;
   cout<<"\n Answer of Addition:"<<sub()<<endl;

}

};
int number::getdata(int m,int n)
{
   a=m;
   b=n;
```

```
int number::sum()
{
 s1=a+b;
 return(s1);
}

int number :: sub()
{
   s2=a-b;
   return(s2);
}

int main()
{
  number x;
  x.getdata(10,20);
  x.show();
  return 0;
```

Private Member Function

- Generally we declare , data members are in private section and member function in public section, that's why we call a member function from main() through object.

- But if we declare a member function in private section then we can not call directly from the main(), because it's private function.

- To call private function , we have to create public function of that class and we call this private function inside that public function , then the public function called by object from main().

Ex:

```
#include<iostream>
using namespace std;

class value
{
  private:
    int a,b;
    void getdata();
  public:
     void show();
};
void value::getdata()
{
  cout<<"Enter number1: ";
  cin>>a;
   cout<<"Enter number2: ";
  cin>>b;
}
```

```
void value::show()
{
   getdata();
   cout<<"Two numbers are "<<a <<"\n"<<b;
}

int main()
{
  value v;
  v.show();
  return 0;
}
```

Array within class:

- The arrays can be used as member variable in a class.

- An array is collection of same data type or group of data item that store in a common name.

- Syntax:

  data_type name[size]={list of value};

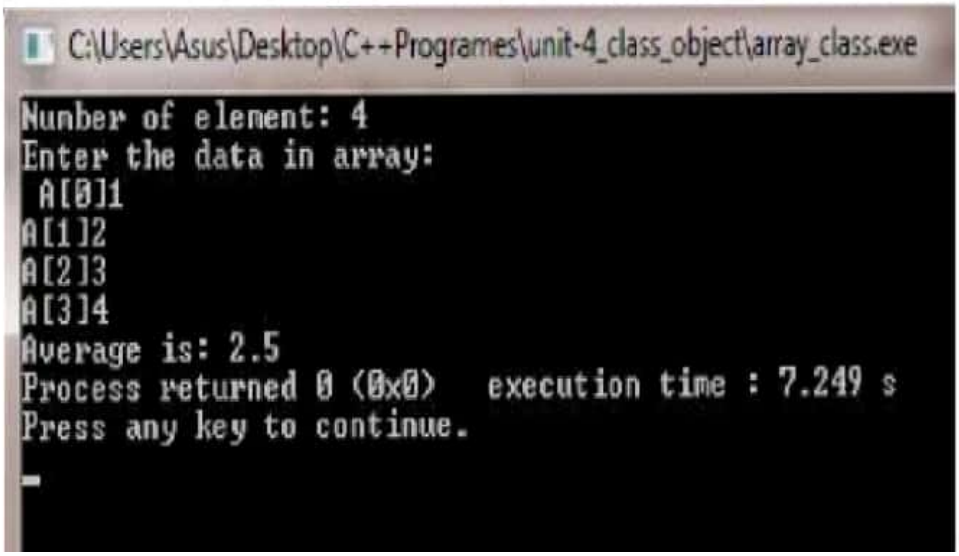
Like

int number[4]={1,2,3,4};

Ex:

```
#include<iostream>
using namespace std;

class average
{
   private:
    int n,A[20];
   public:
     void getdata()
     {
      cout<<"Number of element: ";
      cin>>n;
      cout<<"Enter the data in array:\n ";
      for(int i=0;i<n;i++)
      {
         cout<<"A["<<i<<"]";
         cin>>A[i];
      }
```

```
 float avg()
    {
        float sum=0,ans;
        for(int i=0;i<n;i++)
        sum=sum+A[i];
        ans=sum/n;
        cout<<"Average is: "<<ans;
    }
};
int main()
{
  average a;
  a.getdata();
  a.avg();
  return 0;
}
```



Memory allocation for object

- The memory space for objects are allocated when they are declared , not when the class is specified.
- For member function , when member function are created , it will occupy the memory space only once when they are defining in a class.
- So all objects created for that class can use same member functions , so no separate space is allocated for member functions when the object are created.
- For data member , only space for data members is allocated separately for each object when is created.

- The separate space allocation for data member is essential because the data member will hold different data values for different objects.

- For example, a class student have three data members such as **reg_no**, **age, per** and two member functions **getdata()** and **show().**

- If we create three object S1 ,S2, S3 then,
  object S1 takes up space for: reg_no , age , per
  object S2 takes up space for: reg_no , age , per
  object S3 takes up space for: reg_no , age , per
  But it will access common member function getdata() and show(), so it will take up space only **one time** when class is created.

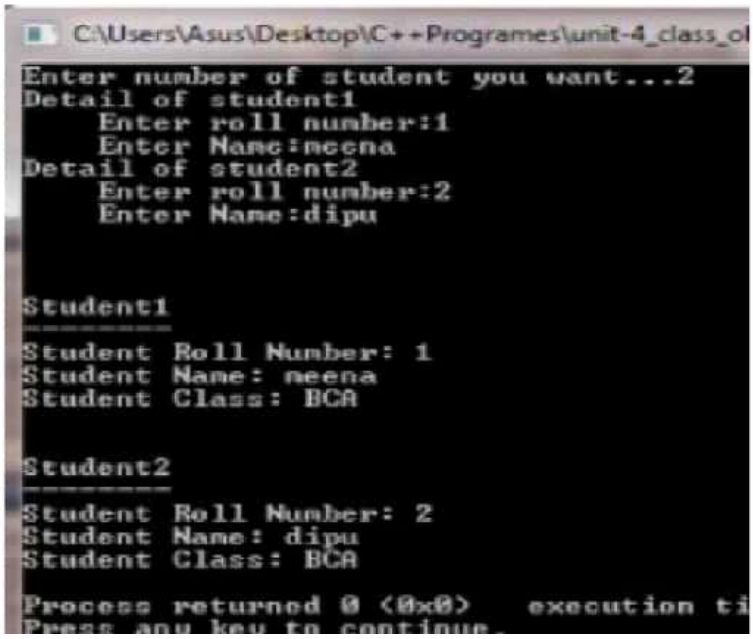
Static data member

- Static variable are normally used to maintain values common to the entire class.
- For example, a static data member can be used as a counter that record occurrences of all the objects.
- A static member variable has certain characteristic:
  1. It automatically initialized **zero** when the first object is created , no other initialization is permitted. Where a simple variable have initially garbage value.
  2. Only one copy of that member is created for entire class and shared by all objects of that class, no matter how many objects are created.
  3. It is visible only within a class, but its life time is the entire program.

Ex:

```
#include<iostream>
using namespace std;

class student{
      int roll_no;
      char name[15];
      static char course[15];
   public:
      void getdata() {
        cout<<"    Enter roll number:";
         cin>>roll_no;
         cout<<"    Enter Name:";
         cin>>name;
      }
      void putdata()  {
         cout<<"Student Roll Number: "<<roll_no<<"\n";
         cout<<"Student Name: " <<name<<"\n";
         cout<<"Student Class: " <<course<<"\n";

      }
```

```
char student::course[15]="BCA";
int main()
{
   int n;
  cout<<"Enter number of student you want...";
  cin>>n;
   student s[n]; // array of object
   for(int i=0;i<n;i++)
   {
     cout<<"Detail of student"<<i+1<<"\n";
     s[i].getdata();
   }
   cout<<"\n";
   for(int i=0;i<n;i++)
   {
    cout<<"\n\nStudent"<<i+1<<"\n";
     cout<<"--------\n";
     s[i].putdata();
   }
   return 0;
}
```



Static member function:

- A member function that is declared **static** has the following properties:
  - A **static** function can have access to only other static members(function or variable) declared in the same class.
  - A **static** member function can be called using the class name.

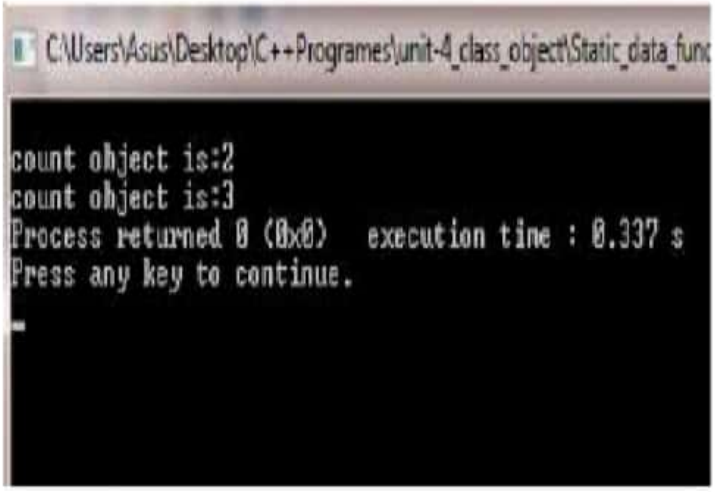
like, class_name :: Function_name();

test :: getdata();

Ex:

```
#include<iostream>
using namespace std;

class stat_fun
{
  int obj;
 static int count;
 public:
     void stat()
     {
       obj=++count;
     }
     void showObject()
     {
        cout<<"\n object number is: "<<obj;
     }
     static void showcount()
     {
       cout<<"\ncount object is:"<<count;
     }
```

```
int stat_fun::count;
int main()
{
  stat_fun o1,o2;
  o1.stat();
  o1.stat();
  stat_fun::showcount();

  stat_fun o3;
  o3.stat();

  stat_fun::showcount();


  return 0;
}
```



Arrays of object:

- As an array can be of any data type including struct. Similarly, we can also have arrays of variable that are of the type class. Such variables are called **array of objects.**
- For example:

```
class student {
        private: float per;
        public: int regno,age;
                void getdata();
                void show();
        };
```

For this class if we required 100 student , then we are not declare different s1,s2,...,s100 object because it's very critical task. For this problem we use array of object.

Ex:

```
#include<iostream>
using namespace std;

class student{
      char name[15];
      float age;
   public:
      void getdata();
      void putdata();
};
void student :: getdata()
{
   cout<<"Enter Name: ";
   cin>>name;
   cout<<"Enter Age: ";
   cin>>age;
}
void student :: putdata()
{
   cout<<"Name: "<<name<<"\n";
   cout<<"Age: "<<age <<"\n";
```

```
const int size=2;
int main()
{
  student s[size]; // array of object
  for(int i=0;i<size;i++)
  {
    cout<<"Detail of student"<<i+1<<"\n";
    s[i].getdata();
  }
  cout<<"\n";
  for(int i=0;i<size;i++)
  {
    cout<<"\n\nStudent"<<i+1<<"\n";
    cout<<"-------\n";
    s[i].putdata();
  }
  return 0;
}
```