C++ CLASSES AND OBJECTS

Abstract data type (ADT) :

* Data abstraction refers to providing only essential information to the outside world and hiding the background details.
* In C++, classes provides great level of data abstraction, that provide only sufficient public methods to the outside world.
* Thus C++ classes are known as abstract data types (ADT) and are used to define our own abstract data types.

Class :

* Class is a template or blueprint that defines the characteristics of object. It is an organization of data and functions which operate on them.
* It acts as the user-defined data type to define its variable called objects.
* It is also defined as collection of similar objects.

Object :

* Object is an instance of a class that
  - combines both data and member functions.
* Basic run time entity of OOP.
* May represent a person, a place or any item the program can handle.
* Takes space in memory and have an associated address

Class components :

* The components of a class are data and the functions which operate on the data .

* The data of a class are called data members and the functions are called member functions .

* Class specification has two parts :

(i) Class declaration

↳ describes the type and scope of its members (data & function)

(ii) Class function definitions

↳ defines the implementation of class fns

(i)

* Syntax for class declaration :

```
class   <name of class>
{
 private :
     data members ;
     member functions ( ) ;
  public :
      data members ;
      member functions ( ) ;
   protected :
       data members ;
       member functions ( ) ;
} ;
```

- Here private , public and protected are called access specifiers (or) visibility labels . (or) access modifiers .

Access modifiers (or) access specifiers

* Access specifiers are used to identify access rights for the data members and member functions of the class.
* Used to define various levels of data abstraction in a class.
* Depending upon the access level, the class member is allowed to access or denied.
* There are 3 main types of access specifiers in C++.

(a) Private:

- A private member within the class can be accessed only by the members of the same class.
- The private member is not accessible from outside the class.
- By default, the class members are private.

(b) Public:

- Public members are accessible from outside the class

(c) Protected:

- Protected members are not accessible from outside the class.
- But can be accessed from the derived class (inheritance).

(ii) Defining member functions

* The member functions can be defined in two places
    - Inside class declaration → can be defined directly.
    - Outside class declaration
        ↳ should use the scope resolution :: operator and class name along with function name.

(Eg)

```
#include <iostream.h>
class date
{
private:
    int day;
    int month;
public:
    int year;
    void set (int dd, int mm, int yy);
    void show ()          // Defining member fn inside class
     {
     cout << day <<"-"<< month <<"-"<< year << endl;
     }
};
void date :: set (int dd, int mm, int yy)  // Defining member
                                              fn. outside class.
  {
   day = dd;
   month = mm;
   year = yy;
  }
```

↳ The scope resolution (::) operator specifies the class to which the member being defined belongs.

Instantiation of objects (or) object creation

* Instantiation is used to create an object from a class (ie) class instance.
* Memory is allocated when the objects are created.

Syntax:

```
classname objectname ;
```

(Eg) date d1;

Accessing class members

- using dot(.)operator, the data members and member functions can be accessed.

(Eg)
```
void main()
{
  date d1, d2 ;
  d1.set (8, 3, 1983);
  d2.set (5, 8, 2007);
  cout << " D.O.B. of first person\n";
  d1.show();
  cout << "D.O.B. of second person\n";
  d2.show ();
}
```

Note:

- Private data can be accessed only through member fns.
  (Eg) d1.set (8, 3, 1983);
  d1.day = 8 ; // invalid.
- Public variable can be accessed directly.
  (Eg) d1. year = 1983 ; // valid.

Example C++ program with class

```
# include <iostream.h>
class person
 {
   char name [20];
   int age ;
   public:
   void getdata ();        }  member functions.
   void display ();        }
 };
void person :: getdata ( void)
  {
   cout << " Enter name " ;
   cin >> name ;
   cout << "Enter age" ;
   cout >> age ;
  }
void person :: display ()
  {
   cout << "\nName" << name ;
   cout << "\n Age" << age ;
  }
  main( )
  {
   person p ;
   p . get data ( ) ;
   p . display ( ) ;
  }
```

Here,
p → class variables (or) objects

Arrays within a class

- Arrays can be used as member variables in a class.

(Eg)
```
#include <iostream.h>
const int size =5;
class student
{
int rollno;
int marks [size];
public:
 void getmarks();
 void total();
};
void student :: getmarks ()
{
cout << "Enter roll no : \n";
cin>> rollno;
for (int i=0; i<size; i++)
 {
 cout << "Enter marks in subject "<< (i+1) <<endl',
 cin >> marks [i];
 }
}
void student :: total()
{
int total =0;
for(int i=0; i<size; i++)
 {
 total += marks[i];
 }
cout << " Total marks \n"<< total ;
}
```

```
void main( )
{
 student s1 , s2 ;        // Creates obj for 2 students
 s1. getmarks( );         // Get marks in 5 subjects for 1st student
 s1 . total ( );          // Find total marks for 1st student
 s2 . getmarks( );        // Get marks for 2nd student
 s2 . total ( );          // find total marks for 2nd student.
}
```

- Here the memory space for array is allocated when object of class is declared.

Array of objects

* Array of variables of type class.

- For the above program, an array of objects can be created as follows, each element representing individual student.

```
void main()
{
 student s[5] ;
 for(int i=0; i<5; i++)
  {
  cout<<"\n Enter details of student "<<(i+1) <<endl;
   s[i]. getmarks();
  }
 for(int i=0; i<5; i++)
  {
  cout<<" Total marks for student "<<(i+1)<<endl;
   s[i]. total ( );
  }
}
```