3.5 DEFINING DATA MEMBERS AND MEMBER FUNCTION

3.5.1 Defining Data Member

All the variables inside the class are called data members. These variables are of any basic data type, derive data type or user-defined data type or objects of other class.

Any function declare inside a class is called a member function of that class. Only the member functions can have access to the private data members and private functions.

};

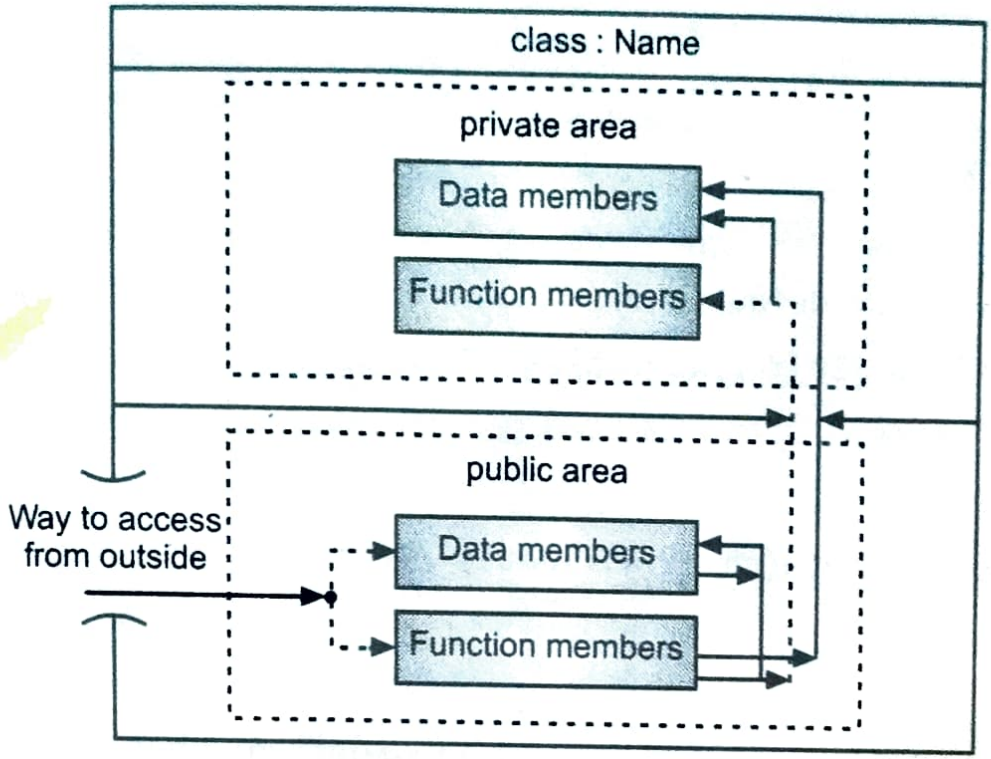

Fig. 3.2: Data Member and Member Function

3.5.2 Defining Member Functions inside and Outside Class Definition

- Member functions of a class are defined in following two ways:
  1. Inside the class definition, and
  2. Outside the class.
- In both ways, only syntax of definition of member function changes but body of function (code) is remain same.

1. Inside the Class Definition:

- Functions which are defined inside the class are similar to normal functions and they are handling automatically and inline functions. Normally, functions which are small are defined inside the class.

Example of student class:

```
class student
{
    int roll;
    char name[20];
    public:
        void getdata()          //definition of member function
        {
            cin>>roll>>name;
        }
        void putdata()          //definition of member function
        {
            cout<<roll<<name;
        }
};
```

2. Outside the Class:

- The functions are define after the class declaration, however the prototype of function should be appear inside the class i.e. declaration only (not definition). Since, the function has defined outside the class, there should be some mechanism to know the identity of function to which class they belong we have to use scope resolution operator (::) to identify the function belongs to which class.
- We can also have a function with the same name and same argument list in different classes, since the scope resolution operator will resolve the scope of the function.
- **Syntax:**

```
return type class_name :: function_name (argument list if any)
{
    ..............  //body of function
}
```

- **Example:**

```
class student
{
    ..............
    ..............
    public:
        void getdata(); //declaration of member function
        ..............
        ..............
};
void student::getdata()
{
    ..............
}
```

Program 3.1: Program for class declaration and creation of objects.

```
#include<iostream>
using namespace std;
class date
{
   private:
      int d;
      int m;
      int y;
   public:
      void get (int Day, int Month, int Year);
      void display(); //declaration
};
void date::get (int Day,int Month,int Year)        //definition
{
   d=Day;
   m=Month;
   y=Year;
}
void date::display() //definition
{
   cout<<d<<"-"<<m<<"-"<<y<<endl;
}
int main()
{
   date d1, d2, d3;//date objects d1, d2 & d3
   d1.get (26, 3, 1968);      //call member function
   d2.get(15, 9, 1978);
   d3.get(12, 3, 1992);
   cout<<"Birth Date of the first child:";
   d1.display();
   cout<<"Birth Date of the second child:";
   d2.display();
   cout<<"Birth Date of the third child:";
   d3.display();
}
```

Output:

```
Birth Date of the first child   :  26 - 3 - 1968
Birth Date of the second child  :  15 - 9 - 1978
Birth Date of the third child   :  12 - 3 - 1992
```

Const member functions:

- If a member function does not alter or modify any data in the class, in this condition we may declare it as a const member function as:

```
void multi (int, int) const;
double get_balance() const;
```

- The const qualifier is appended to the function prototype.
- A const member function guarantees that it will never modify any of its class's member data.

3.6 SIMPLE C++ PROGRAM USING CLASS

- We usually give class some meaningful name like student.
- This student name now becomes a new type identifier that can be used to declare instances of that class.
- Consider the following simple class example a class called student having roll_no and name as data members.

Program 3.2: Simple C++ program using class.

```
#include <iostream>
#include<cstring>
using namespace std;
class student
{
   int roll_no;
   char name[30];
   public:
         void setdata()
         {
               roll_no = 10;
               strcpy (name, "SAIKRISHNA");
         }
         void getdata()
         {
               cout<<"\n"<<" RollNo:"<<roll_no;
               cout<<"\n"<<"Name:"<<name;
         }
};
int main()
{
   student s;
   s.setdata();
   s.getdata();
}
```

Output:

```
RollNo:10
Name:SAIKRISHNA
```

functions of a class student.

- It is a general practice to declare data members as private and member functions as public. The general notation of representation of a class student is shown in Fig. 3.3.

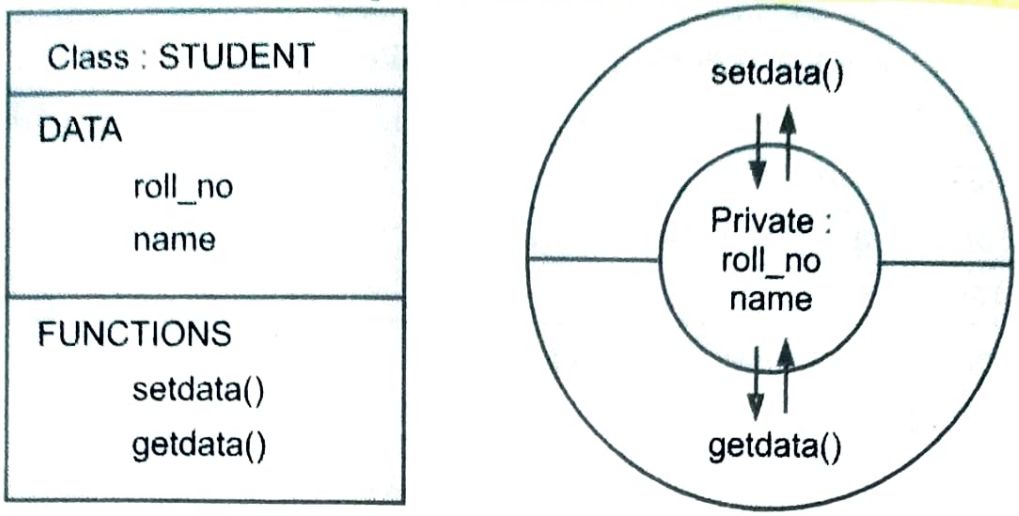

Fig. 3.3: Representation of Class

3.7 MEMORY ALLOCATION FOR OBJECTS

- A class is a template from which instances i.e. objects can be created. All the objects created from a class look like and exhibit similar behaviour.
- When a class is declared no storage is allocated for data members. Memory is allocated for objects when they are declared.
- The member functions are created and allocated space in memory only once when they are defined in the class.
- All the objects of that class share the same member functions. But when an object is created, then separate memory space is allocated for the data members of that object, because data members of each object have different values.
- For example, consider the following class:

```
class sample
{
    int x, y;
    public:
        void initialize ( )
        {
            x = 20;
            y = 30;
        }
};
sample s1, s2;
```

- For the above class two objects s1 and s2 are declared. There are two data members in the class x and y which are of integer data type. So total 4 bytes will be allocated for object s1 and 4 bytes for s2.
- Fig. 3.4 illustrates the memory allocation for objects s1 and s2.

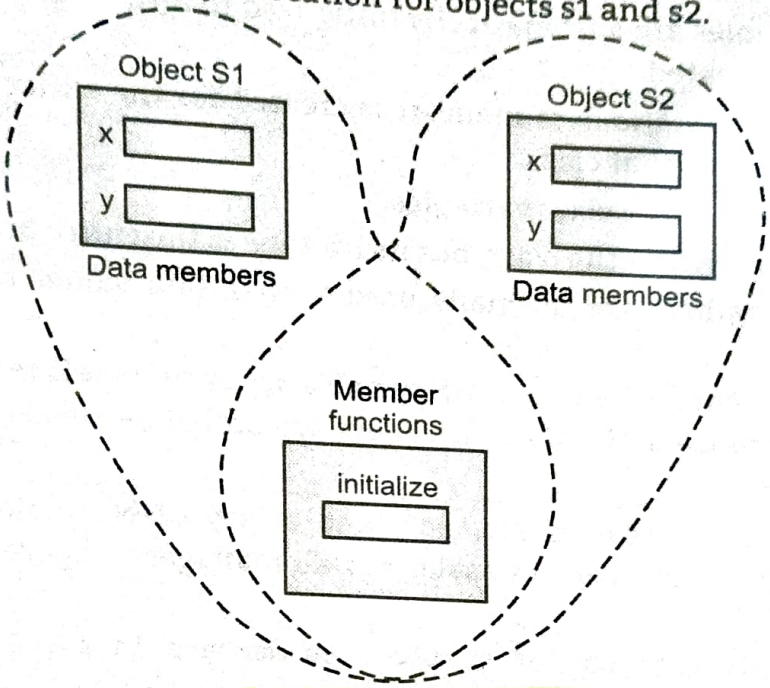

Fig. 3.4: Memory Allocation of Object

- In short the data members are allocated separate space for each object and all objects shares the same member functions.