Tutorial Link https://codequotient.com/tutorials/Object-Oriented Programming in C++/5b38c12dc6a1d0259e728e56

TUTORIAL

Object-Oriented Programming in C++

Chapter

1. Object-Oriented Programming in C++

Topics

1.3 Defining Member Functions

Classes and Objects

A class can be visualized as a three- compartment box, as illustrated:

i) Class name (or identifier): identifies the class.

ii) Data Members or Variables (or attributes, states, fields): contains the static attributes of the class.

iii) Member Functions (or methods, behaviors, operations): contains the dynamic operations of the class.

Classes are created using the keyword class. The declaration of a class is enclosed with curly braces and terminated by a semi-colon. The member variables and functions are divided into two sections i.e. private and public. The private and public keywords are terminated by colon (:). The object cannot directly access the member variables and functions declared in private section. The private members of a class can only be accessed by a public member function of the same class. The syntax of class declaration is as under:

```
class < Name of Class>
{
        private:
                Declaration of Variables;
                Prototype declaration of Functions;
        public:
                Declaration of Variables;
                Prototype declaration of Functions;
} object-list; // the object –list is optional. If present, it
declares objects of the class.
```

The following example illustrates the class Circle declaration.

```
class Circle // class name
{
        private:
        double radius; // data members ( or variables)
        public:
        double getRadius(); // member functions
        double showArea();
};
```

Declaring Objects:

Defining objects of a class data type is known as class instantiation. When objects are created, only during that moment memory is allocated to them. For examples, suppose that we have a class Circle.

We can create instances of Circle in the function main () as follows:

```
Circle c1, c2, c3; // Construct 3 instances c1, c2, c3 of the class
Circle
```

Accessing class members:

The object can access the public member variables and functions of a class by using operator dot (.) and the syntax is as follows:

```
[Object name] [Operator][Member name];
```

To access the member functions getRadius & showArea of class Circle, the statement would be,

```
C1. getRadius();
C1. showArea();
```

Another method of accessing members of a class is using the arrow operator ' - >' and will be discussed later.

Defining Member Functions

The member function must be declared inside the class. They can be defined inside or outside the class.

The member function defined inside the class is treated as an inline function. If the function is defined outside the class, its prototype declaration must be done inside the class.

(i) **Member Function inside the class**

The following program illustrates the use of member function inside the class in the public section.

```
#include<iostream>
#include<cstdio>
#include<cmath>

```

```
using namespace std;

class Circle
{
  private:
  double radius;
  public:
  double getRadius( ) // Member function (Setter)
  {
    radius=1.5;
  }
  double showArea() // Member function
  {
    return radius*radius*3.1416;
  }
};

int main( )
{
  Circle C1;
  C1. getRadius( );
  cout<<"area = "<<C1.showArea();
  return 0;
}
```

(ii) **Member function outside the class**

To define a function outside the class the following steps must be taken:

- The prototype of function must be declared inside the class.
- The function name must be preceded by the class name and its return type separated by scope access operator.

The following example illustrates the function outside the class.

C++

```cpp
#include<iostream>
#include<cstdio>
#include<cmath>

using namespace std;

class Circle
{
  private:
  double radius;
  public:
  void getRadius( ) ; // Prototype declaration
  double showArea( ) ; // Prototype declaration
};

void Circle :: getRadius()
{
  radius=1.5;
}

double Circle :: showArea ()
{
  return radius*radius*3.1416;
}

int main( )
{
  Circle c1;
  c1. getRadius( );
  cout<<"area = "<< c1.showArea();
  return 0;
}
```