Topic: Class and Objects Constructors and Destructors

CSET-334 : Programming in C++

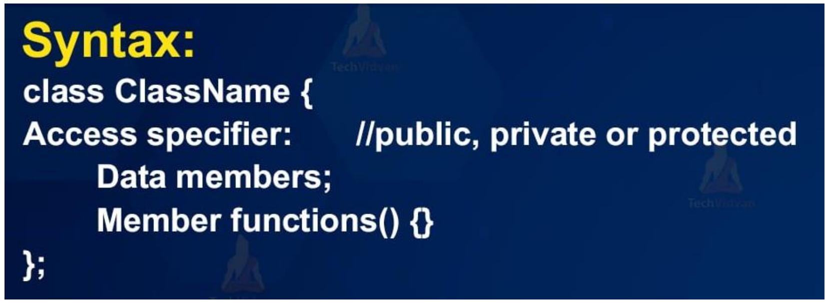
Classes and Objects in C++

- The main purpose of C++ programming is to add object orientation to the C programming language and **classes are the central feature of C++** that supports object-oriented programming and are often called **user-defined types**.
- When you define a class, you define a **blueprint for a data type**.
- A class is used to specify the form of an **object** and it combines data representation and methods for manipulating that data into one neat package.
- The data and functions within a class are called members of the class.

Classes and Objects: Syntax

- A class definition starts with the keyword class followed by the class name; and the class body, enclosed by a pair of curly braces.
- A class definition must be followed either by a semicolon or a list of declarations.

```
 class Box
  {
       public:
       double length; // Length of a box
       double breadth; // Breadth of a
box
        double height; // Height of a box
   } ;
```

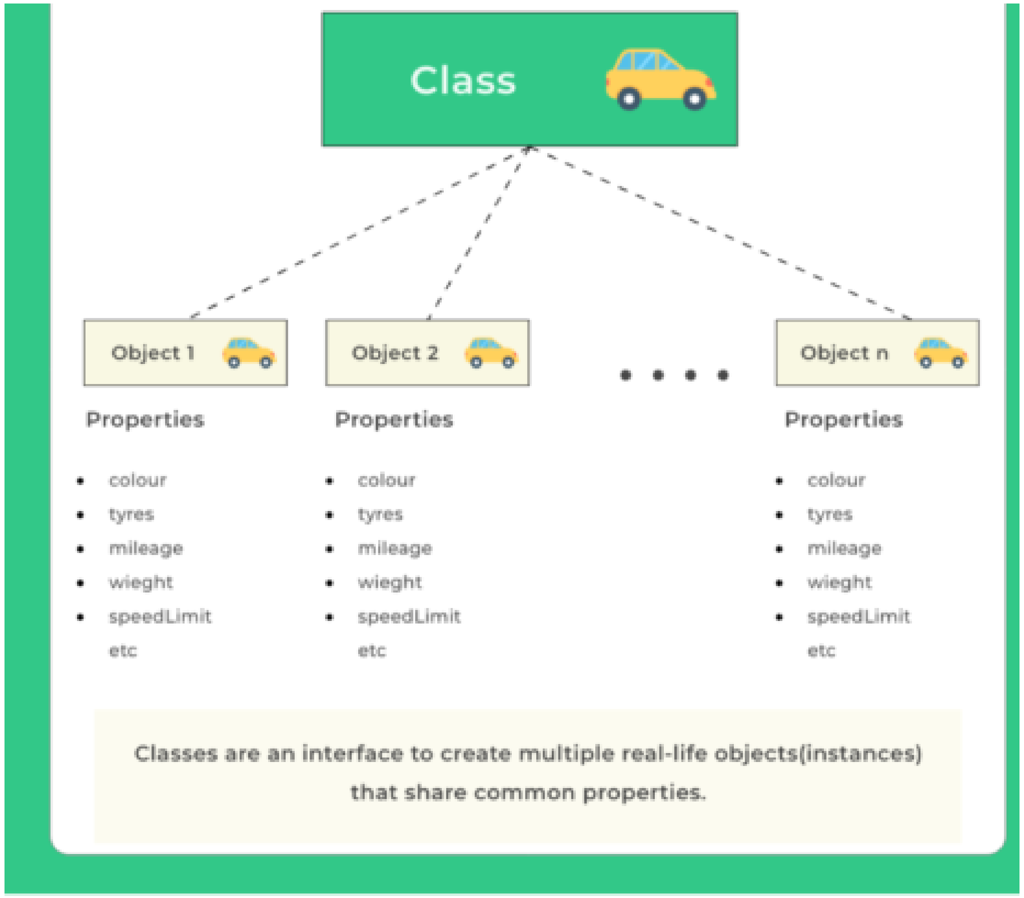


```
Syntax:
class ClassName {
Access specifier:      //public, private or protected
      Data members;
      Member functions() {}
};
```

Objects

- An instance of a class is called an object. When a class is defined, no memory is allocated, we only define the specifications for its object.
- Memory is allocated when we create an object of a class.
- Data members and member functions of a class can be used and accessed by creating objects.
- We can create multiple objects of a class.
- **Creating Objects in C++**

```
ClassName  ObjectName;
```


Class
Object 1
Object 2
Object n
Properties
colour
tyres
mileage
wieght
speedLimit
etc
Classes are an interface to create multiple real-life objects(instances) that share common properties.

ACCESS SPECIFIERS

- Access Specifiers (modifiers) are keywords in C++ .
- Access Specifiers define how the members of a class can be accessed.
- It sets some restrictions on the class members not to get directly accessed by the outside functions.
- In C++, there are three access specifiers –

- PUBLIC
- PRIVATE
- PROTECTED
-

PUBLIC ACCESS SPECIFIERS -:

- The Public keyword is used to create public member.(data and function)
- The Public members are accessible from any part of program.
- It means all the class members declared under Public will be available to everyone.
- The data members and member functions declared public can be accessed by other classes too.

```
  class ABC
{
      public :                    // Public access modifiers
       int x;                     // data member
       void display();      //member function

   };
```

PRIVATE ACCESS SPECIFIERS:-

- Private keyword means that no one can access the class members declared private, outside the class.
- If someone tries to access the private number of a class, they will get a compile time error.

- The private keyword is used to create private members (data and function).
- The private members can only be accessed from within the class.
- However, friend classes and friend functions can access private members.

```
class ABC
{
     private :                  // Private access modifiers
       int x;                   // data member
        void display();        //member function

   };
```

PROTECTED ACCESS SPECIFIERS -:

- Protected is the last access specifier and it is similar to private , it make class member inaccessible outside the class. But they can be accessed by any sub class of that class.
- The protected keyword is used to create protected members (data and function).
- The protected members can be accessed within the class and from the derived class.

```
class ABC
          {
              protected :          // Protected access modifiers
               int x;                    // data member
                void display();      //member function

            };
```

Constructors in C++

- **What is constructor?**
  A constructor is a member function of a class which initializes objects of a class. In C++, Constructor is automatically called when objects(instance of class) are created. It is special member function of the class.
- **How constructors are different from a normal member function?**
- A constructor is different from normal functions in following ways:
- Constructor has same name as the class itself
- Constructors don’t have return type
- A constructor is automatically called when an object is created.
- If we do not specify a constructor, C++ compiler generates a default constructor for us (expects no parameters and has an empty body).

Types of Constructors

- **Default Constructors:** Default constructor is the constructor which doesn’t take any argument. It has no parameters.

- Cpp program to illustrate the concept of Constructors

```
#include <iostream>
using namespace std;
class construct
{
     public:
     int a, b;
construct()                              // Default Constructor
{
       a = 10;
        b = 20;
 }
};

int main()
{
    construct c; // Default constructor called automatically
     cout << "a: " << c.a << endl
    << "b: " << c.b;
   return 1;
    }
```

Parameterized Constructors

- It is possible to pass arguments to constructors.
- Typically, these arguments help initialize an object when it is created. To create parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor's body, use the parameters to initialize the object.

```
CPP program to illustrate parameterized constructors
#include <iostream>
using namespace std;
class Point
 {
         private:
         Int x, y;
         public:

         Point(int x1, int y1)  // Parameterized Constructor
       {
            x = x1;
            y = y1;
        }

        int getX()
         {
             return x;
         }
       int getY()
        {
              return y;
          }
     };
```

```
int main()
  {
      Point p1(10, 15);   // Constructor called
      cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();
    return 0;
   }
 Output:
 p1.x = 10, p1.y = 15
```

- **Uses of Parameterized constructor:**
  - It is used to initialize the various data elements of different objects with different values when they are created.
  - It is used to overload constructors.
- **Can we have more than one constructors in a class?**
  Yes, It is called Constructor Overloading.

Copy Constructor:

A copy constructor is a member function which initializes an object using another object of the same class. The copy constructor is used to:

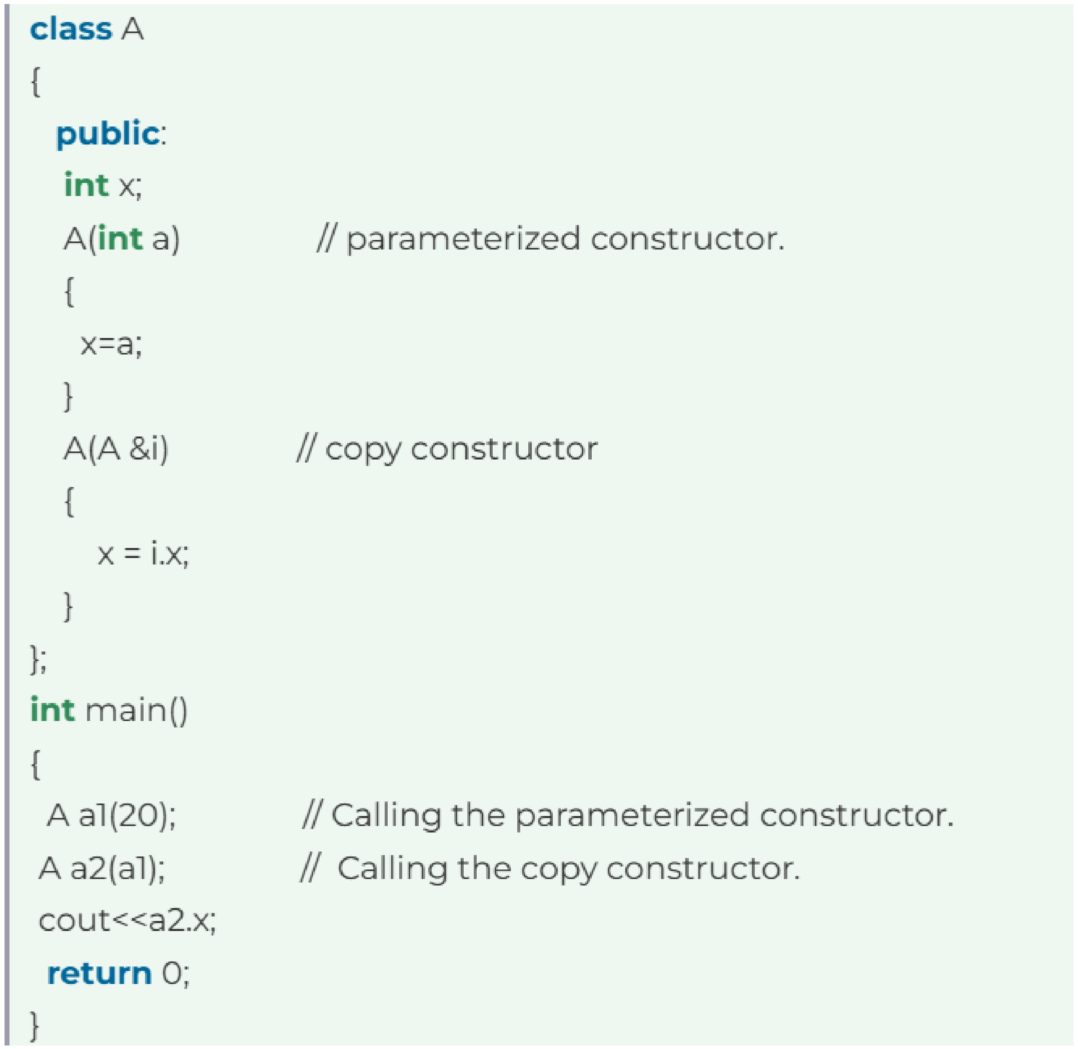
- Initialize one object from another of the same type.
- Copy an object to pass it as an argument to a function.
- Copy an object to return it from a function.

- The most common form of copy constructor is shown here:

```
classname (const classname &obj)
    {
            // body of constructor
     }
```

Here, obj is a reference to an object that is being used to initialize another object.

Example of Copy Constructor



```
class A
{
  public:
   int x;
   A(int a)              // parameterized constructor.
   {
    x=a;
   }
   A(A &i)             // copy constructor
   {
      x = i.x;
   }
};
int main()
{
 A a1(20);             // Calling the parameterized constructor.
 A a2(a1);             //  Calling the copy constructor.
 cout<<a2.x;
  return 0;
}
```

Output: 20

- **What is destructor?**
  Destructor is a member function which destructs or deletes an object.
- **When is destructor called?**
  A destructor function is called automatically when the object goes out of scope:
  (1) the function ends
  (2) the program ends
  (3) a block containing local variables ends
  (4) a delete operator is called
- **How destructors are different from a normal member function?**
  Destructors have same name as the class preceded by a tilde (~)
  Destructors don't take any argument and don't return anything

```
class String
   {
        private:
        char *s;
        int size;
         public:
        String(char *); // constructor
       ~String();      // destructor
       };
     String::String(char *c)
     {
         size = strlen(c);
         s = new char[size+1];
         strcpy(s,c);
      }
String::~String()
    {
       delete []s;
     }
```

- **Can there be more than one destructor in a class?**
  No, there can only one destructor in a class with class name preceded by ~, no parameters and no return type.
- **When do we need to write a user-defined destructor?**
  If we do not write our own destructor in class, compiler creates a default destructor for us. The default destructor works fine unless we have dynamically allocated memory or pointer in class. When a class contains a pointer to memory allocated in class, we should write a destructor to release memory before the class instance is destroyed. This must be done to avoid memory leak.

Nesting of Member function

- Only the public members of a class can be accessed by the object of that class, using dot operator.
- However, a member function can call another member function of the same class directly without using the dot operator.
- This is called as *nesting of member functions*.

A member function can call another member function of the same class for that you do not need an object

Illustration : The use of Nesting of Member Function

```
#include<iostream>
using namespace std
class nest
{
    int a;
    int square_num( )
    {
        return a* a;
    }
    public:
    void input_num( )
    {
    cout<<"\nEnter a number ";
    cin>>a;
    }
    int cube_num( )
    {
        return a* a*a;
    }
void disp_num()
{
    int sq=square_num();      //nesting of member function
    int cu=cube_num();  //nesting of member function
    cout<<"\nThe square of "<<a<<" is  " <<sq;
    cout<<"\nThe cube of "<<a<<" is  " <<cu;
} };
int main()
{
    nest n1;
    n1.input_num();
    n1.disp_num();
    return 0;
}
```

Output:

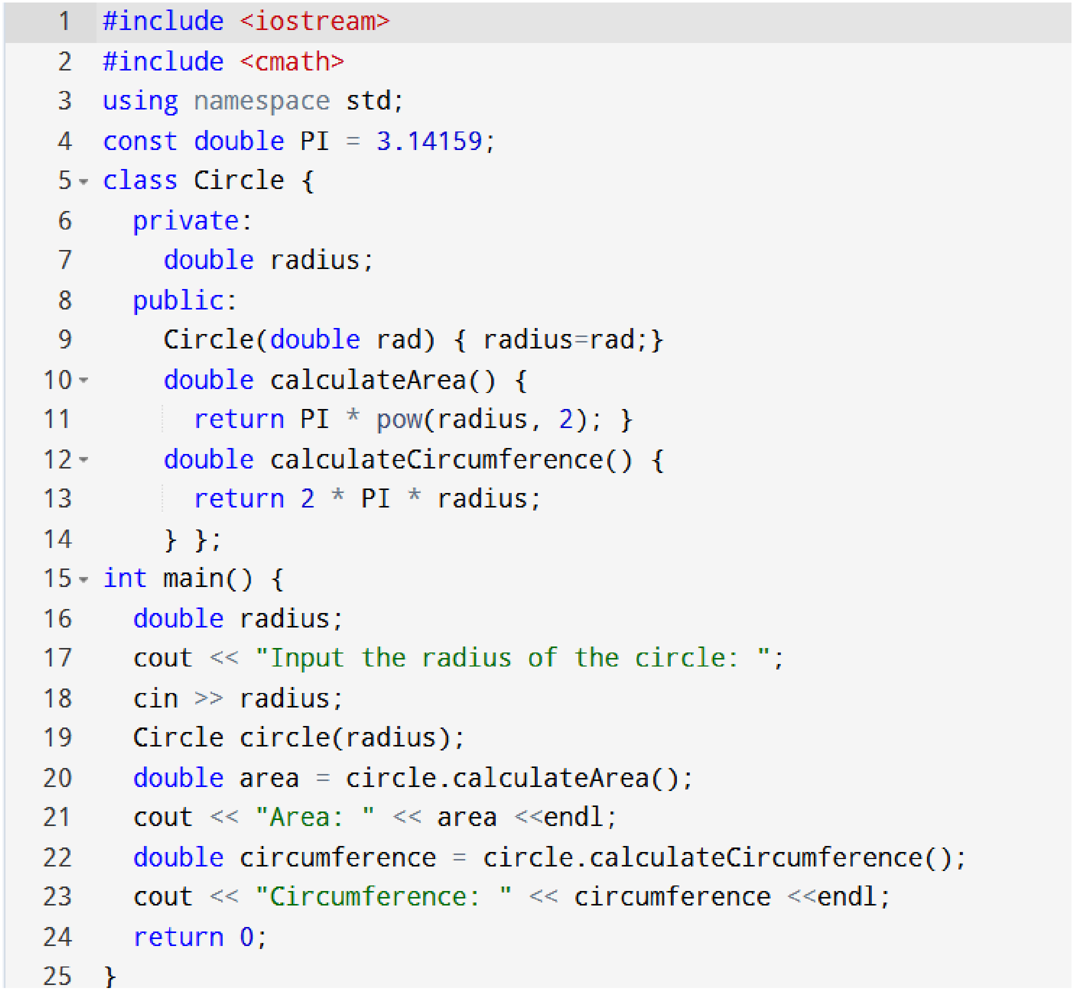
```
Enter a number 5
The square of 5 is 25
The cube of 5 is 125
```

Exercise Questions

Write a C++ program to implement a class called Circle that has private member variables for radius. Include member functions to calculate the circle's area and circumference.

Write a C++ program to create a class called Car that has private member variables for company, model, and year. Implement member functions to get and set these variables.

Solution



```cpp
#include <iostream>
#include <cmath>
using namespace std;
const double PI = 3.14159;
class Circle {
  private:
    double radius;
  public:
    Circle(double rad) { radius=rad;}
    double calculateArea() {
      return PI * pow(radius, 2); }
    double calculateCircumference() {
      return 2 * PI * radius;
    } };
int main() {
  double radius;
  cout << "Input the radius of the circle: ";
  cin >> radius;
  Circle circle(radius);
  double area = circle.calculateArea();
  cout << "Area: " << area <<endl;
  double circumference = circle.calculateCircumference();
  cout << "Circumference: " << circumference <<endl;
  return 0;
}
```

Thank You