C++ namespace

By – Mohammad Imran

Namespace

Introduction

There are two students with the same name in a classroom. To distinguish the two students, we need to call them by their names and surnames. A situation like this can occur in C++ as well. It might happen that a function we created with the name **sqrt(),** but this is already present in the **cmath** library of C++. In this case, the compiler cannot differentiate between the two function names. Hence, it will yield an error. To avoid this confusion, namespaces are used.

Namespaces provide a method to avoid name conflicts in a project. Namespaces in C++ are used as additional information to differentiate two or more variables, functions, or classes having the same names.

By – Mohammad Imran

Namespace

Defining Namespaces

Creating a namespace in C++ is similar to creating a class. We use the keyword namespace followed by the name of the namespace to define a namespace in C++.

A namespace definition begins with the keyword **namespace** followed by the **namespace name.**

```
namespace MyNamespace
{
   // Function, class, and variable declarations.
}
```

Namespace

Defining Namespaces

In the scope of the namespace, we can declare variables, functions, user-defined data types (like classes), and even nested namespaces.

It is optional to declare all the namespace members all at once. We can define a namespace in multiple parts, and then we can use the scope of each part to define the different identifiers. A namespace is the sum of each separately defined part. A namespace defined in multiple parts is called a discontiguous namespace. We can do so because a namespace in C++ can be characterized in different sections.

By – Mohammad Imran

Using a Namespace in C++

There are three ways to use a namespace in C++:

- ✓ Using Scope Resolution Operator (::)
- ✓ Using Directive
- ✓ Using Declaration

Defining Namespace

Using Scope Resolution Operator

The scope resolution operator (**::**) can be used with the name of the namespace to call/access any member (variable, function, or class) declared inside a namespace.

Defining Namespace

Using Scope Resolution Operator | **Example - 1**

```
#include <iostream>
using namespace std;
namespace ns1
{
   void greet()
   {
      cout << "Namespace - 1" << endl;
   }
}
namespace ns2
{
   void greet()
   {
      cout << "Namespace - 2" << endl;
   }
   int var = 10;
   int func()
   {
      return var * 3;
   }
}
int main()
{
   ns1::greet();
   ns2::greet();
   cout << ns2::func();
   return 0;
}
```

OUTPUT

```
Namespace - 1
Namespace - 2
30
```

By – Mohammad Imran

Defining Namespace

Using Directive

With the help of using namespace directive, we can import an entire namespace from another program file into our program file. The imported namespace will have a global scope. We can also use this directive to import a namespace into another namespace or even another program.

Here is the **syntax** for the using directive.

Syntax:

```
using namespace namespace_name;
```

Defining Namespace

Using Directive

Create your own namespace in a separate file with one or more function and save it with **mymath.**

Because I want to create **mymath** namespace with different math functions to use in a program.

By – Mohammad Imran

Defining Namespace

Using Directive

```
namespace mymath
{
   int sqrt(int n)
   {
      if (n == 0 || n == 1)
         return n;
      int i = 1, check = 1;
      while (check <= n)
      {
         i++;
         check = i * i;
      }
      return i - 1;
   }
```

```
   void checkChar( char c )
   {
      if(  (c  >=  'a'  &&  c  <=  'z')
         || (c >= 'A' && c <= 'Z') )
      {
        cout << "Character" << endl;
      }
      else
      {
          cout << "Not a Character"
            << endl;
      }
   }
}
```

By – Mohammad Imran

Defining Namespace

Using Directive | **Example - 2**

```
#include <iostream>
using namespace std;
#include "mymath.cpp"
using namespace mymath;

int main()
{
   int num;
   cout << "Enter number : ";
   cin >> num;
   int srt = sqrt(num);
   cout << "Square root of " << num;
   cout << " = " << srt;
   return 0;
}
```

OUTPUT

```
Enter number : 9
Square root of 9 = 3
```

By – Mohammad Imran

Defining Namespace

Using Declaration

The **using declaration** differs slightly from the **using directive**. In the **using declaration**, we only import one namespace member at a time with the help of the scope resolution operator. The member imported is only available in the current scope.

Syntax:

```
using namespace namespace_name::member_name;
```

Defining Namespace

Using Declaration

Example - 3

```
#include <iostream>
using namespace std;
namespace First
{
   void sayHello()
   {
      cout << "Hello First Namespace" << endl;
   }
}
namespace Second
{
   void sayHello()
   {
      cout << "Hello Second Namespace" << endl;
   }
}
```

By – Mohammad Imran

Defining Namespace

Using Declaration

Example - 3

```
using namespace First;

int main()

{

   sayHello();

   return 0;

}
```

OUTPUT

```
First Namaespace
```

By – Mohammad Imran

C++
Class and Object

By – Mohammad Imran

Class and Object in C++

C++ is an object-oriented programming language that is used to model real-world entities into programs. Object-oriented programming languages achieve this using classes and objects.

Everything in **C++** is associated with classes and objects, classes are the building blocks of C++ language. It is a user-defined data type, which acts as a blueprint from which objects are created. Classes hold their own data members and member functions, which are accessed using an instance of that class.

By – Mohammad Imran

Class in C++

A class in C++ is a user-defined type or data structure declared with a keyword class that has data and functions as its members. A class can be used by declaring an instance of that class which is nothing but the object in C++.

In real life, we see dogs with different breeds or classes but there are some common characteristics such as they have tails, they bark, they sniff, etc. So, instead of creating different classes for every dog, we can define a single class with common properties of dogs. Hence, Classes act as a user-defined data type to create objects with similar properties.

By – Mohammad Imran

Defining Class in C++

A Class is defined by the keyword class followed by a class name (user's choice) and a block of curly brackets with a semicolon after the block. The block starts with access specifiers followed by data members and member functions.

NOTE: class uses access specifier within class body. There are three types of access specifier in C++ such as public, private and protected. We will discuss later about access specifier.

By – Mohammad Imran

Defining Class in C++

Syntax

```
class ClassName
{
    Access specifier:
    Data members;
    Member Functions()
    {
        // member function defintion
    }
};
```

By – Mohammad Imran

Defining Class in C++

Example

```
class Dog                                   // class ClassName
{
  public:                                   // Access specifiers
   string breed, color;                     // Data members
   void displayColor()                      // Member functions
   {
      cout << color << " ";
   }
   void displayBreed()
   {
      cout << breed << " ";
   }
};                                          // end with semicolon
```

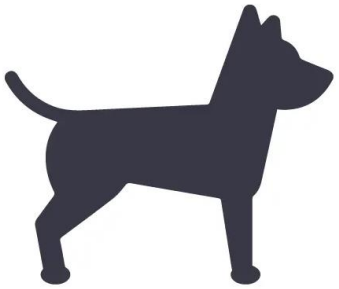
By – Mohammad Imran

Object in C++

A class is merely a blueprint of data. An Object is a data structure that is an instance of a class. So when a class is created no memory is allocated but when an instance is created by declaring an object, memory is then allocated to store data and perform required functions on them. We can visualize trees, birds, and dogs as different classes and their instances like neem tree, peacock, and German shepherd dog respectively. Hence, classes and objects in C++ are the main ingredients of object-oriented programming.

By – Mohammad Imran

Class and Object in C++

Class	Object
	

By – Mohammad Imran

Object in C++

Declaring Object

When a class is defined only the blueprint of data structure is defined no memory is allocated. To use data and its member function we can declare objects. We can declare objects by mentioning the class followed by user-defined object name.

Syntax –

```
ClassName ObjectName;
```

Example –

```
Dog d;
```

By – Mohammad Imran

Significance of Class and Object

The concept of class and object in C++ makes it possible to incorporate real-life analogy to programming. It gives the data the highest priority using classes. The following features prove the significance of class and object in C++:

Data Hiding: A class prevents the access of the data from the outside world using access specifiers. It can set permissions to restrict the access of the data.

By – Mohammad Imran

Significance of Class and Object

Code Reusability: You can reduce code redundancy by using reusable code with the help of inheritance. Other classes can inherit similar functionalities and properties, which makes the code clean.

Data Binding: The data elements and their associated functionalities are bound under one hood, providing more security to the data.

Flexibility: You can use a class in many forms using the concept of polymorphism. This makes a program flexible and increases its extensibility.

Object Oriented Paradigm

Object oriented paradigm is a significant methodology for the development of any software. Most of the architecture styles or patterns such as pipe and filter, data repository, and component-based can be implemented by using this paradigm.

The major aspects of Object Oriented Programming (OOP) paradigm are as follows:

By – Mohammad Imran

Object Oriented Paradigm

Reduced Maintenance :

The primary goal of object-oriented development is the assurance that the system will enjoy a longer life while having far smaller maintenance costs. Because most of the processes within the system are encapsulated, the behaviors may be reused and incorporated into new behaviors.

Object Oriented Paradigm

Real-World Modeling :

Object-oriented system tend to model the real world in a more complete fashion than do traditional methods. Objects are organized into classes of objects, and objects are associated with behaviors. The model is based on objects, rather than on data and processing.

Object Oriented Paradigm

Improved Reliability and Flexibility :

Object-oriented system promise to be far more reliable than traditional systems, primarily because new behaviors can be "built" from existing objects. Because objects can be dynamically called and accessed, new objects may be created at any time. The new objects may inherit data attributes from one, or many other objects. Behaviors may be inherited from super-classes, and novel behaviors may be added without effecting existing systems functions.

Object Oriented Paradigm

High Code Re-usability :

When a new object is created, it will automatically inherit the data attributes and characteristics of the class from which it was spawned. The new object will also inherit the data and behaviors from all super classes in which it participates.

Object Oriented Analysis

In the system analysis or object-oriented analysis phase of software development, the system requirements are determined, the classes are identified and the relationships among classes are identified.

The three analysis techniques that are used in conjunction with each other for object-oriented analysis are object modelling, dynamic modelling, and functional modelling.

By – Mohammad Imran

Object Modeling

Object modelling develops the static structure of the software system in terms of objects. It identifies the objects, the classes into which the objects can be grouped into and the relationships between the objects. It also identifies the main attributes and operations that characterize each class.

Object Modeling

The process of object modelling can be visualized in the following steps –

- ✓ Identify objects and group into classes
- ✓ Identify the relationships among classes
- ✓ Create user object model diagram
- ✓ Define user object attributes
- ✓ Define the operations that should be performed on the classes
- ✓ Review glossary

Dynamic Modeling

Dynamic Modelling can be defined as “a way of describing how an individual object responds to events, either internal events triggered by other objects, or external events triggered by the outside world”.

Dynamic Modeling

The process of dynamic modelling can be visualized in the following steps

- ✓ Identify states of each object
- ✓ Identify events and analyze the applicability of actions
- ✓ Construct dynamic model diagram, comprising of state transition diagrams
- ✓ Express each state in terms of object attributes
- ✓ Validate the state–transition diagrams drawn

Class and Object in C++

Example - 4

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

class Test
{
  public:
   int x;
};
int main()
{
   Test t;
   t.x = 100;
   cout << t.x;
   return 0;
}
```

OUTPUT

100

By – Mohammad Imran

Class and Object in C++

Example - 5

```
#include <iostream>
using namespace std;
class Addition
{
   int x, y, res;
  public:
   void initialize()
   {
      x = 20;
      y = 5;
   }
   void add()
   {
      res = x + y;
      cout << "Sum = " << res;
   }
};
```

```
int main()
{
      Addition ad;
      ad.initialize();
      ad.add();
      return 0;
}
```

OUTPUT

```
Sum = 25
```

By – Mohammad Imran

Class and Object in C++

Example - 6

```
#include <iostream>
using namespace std;
class Employee
{
   int salary;
  public:
   void setSalary(int n)
   {
      salary = n;
   }
   int getSalary()
   {
      return salary;
   }
};
```

```
int main()
{
   Employee emp;
   emp.setSalary(50000);
   cout << emp.getSalary();
   return 0;
}
```

OUTPUT

50000

By – Mohammad Imran

Coding Questions

By – Mohammad Imran

Question - 1

Write a program in C++ to find the sum and difference of two number but the condition is both task should be define in separate file in the form of class with function.

For Example –

File1.cpp contains the function to find difference of two numbers

File2.cpp contains the function to find the sum of two numbers

Click here to see code

By – Mohammad Imran

Question - 2

Write a program in C++ to display strings using class and function in different way.

Using

- Simple Function
- Argument Function
- Return with Pointer Argument

Click here to see code

By – Mohammad Imran

Question - 3

Problem - Consider an unsorted array with N number of elements. A number **k** less than the size of the array is given; we have to find the **kth** largest element in the best possible ways.

Input

9
14, 5, 6, 12, 14, 8, 10, 6, 25
3 // K = 3

Output

Sorted Array = 5 6 6 8 10 12 14 14 25

Kth largest element = 12

Click here to see code

By – Mohammad Imran