OVERVIEW OF C#

Taruna Rohdia

Abstract:- A type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure. In this way, the data structure becomes an *object* that includes both data and functions. In addition, programmers can create relationships between one object and another. Object-oriented technology is becoming increasingly popular in industrial software development environments. This technology helps in the development of a software product of higher quality and lower maintenance costs. One of the principal advantages of object-oriented programming techniques over procedural programming techniques is that they enable programmers to create modules that do not need to be changed when a new type of object is added. A programmer can simply create a new object that inherits many of its features from existing objects.

INTRODUCTION:- C# is a modern, general-purpose, object-oriented programming language. C# is designed for Common Language Infrastructure (CLI), which consists of the executable code and runtime environment that allows use of various high-level languages on different computer platforms and architectures. C# is an elegant and type-safe object-oriented language that enables developers to build a variety of secure and robust applications that run on the .NET Framework. You can use C# to create Windows client applications, XML Web services, distributed components, client-server applications, database applications, and much, much more. Visual C# provides an advanced code editor, convenient user interface designers, integrated debugger, and many other tools to make it easier to develop applications based on the C# language and the .NET Framework.

.NET FRAMEWORK: - The .Net framework is a revolutionary platform that helps in writing the following types of applications:

- Windows applications
- Web applications
- Web services

Integrated Development Environment (IDE) for C#

Microsoft provides the following development tools for C# programming:

- Visual Studio 2010 (VS)
- Visual C# 2010 Express (VCE)
- Visual Web Developer

Data Types:-

The variables in C#, are categorized into the following types:

- Value types
- Reference types
- Pointer types

Value Type:-

Value type variables can be assigned a value directly. They are derived from the class System.Value Type. The value types directly contain data. Some examples are int, char, and float, which stores numbers, alphabets, and floating point numbers.

Reference Type:-

The reference types do not contain the actual data stored in a variable, but they contain a reference to the variables. In other words, they refer to a memory location. Using multiple variables, the reference types can refer to a memory location. If the data in the memory location is changed by one of the variables, the other variable automatically reflects this change in value.

Pointer Type:-

Pointer type variables store the memory address of another type. Pointers in C# have the same capabilities as the pointers in C or C++.

Syntax for declaring a pointer type is:

type* identifier;

For example,

char* cptr;

int* iptr;

Type Conversion:-

Type conversion is converting one type of data to another type. It is also known as Type Casting. In C#, type casting has two forms:

- Implicit type conversion - These conversions are performed by C# in a type-safe manner. For example, conversions from smaller to larger integral types and conversions from derived classes to base classes.
- Explicit type conversion - These conversions are done explicitly by users using the pre-defined functions. Explicit conversions require a cast operator.

Example:-

```
using System;
namespace TypeConversionApplication
{
   class ExplicitConversion
   {
      static void Main(string[] args)
      {
         double d = 5673.74;
         int i;
         i = (int)d;
         Console.WriteLine(i);
         Console.ReadKey();

      }
   }
```

}

Encapsulation:- Encapsulation is defined 'as the process of enclosing one or more items within a physical or logical package'. Encapsulation, in object oriented programming methodology, prevents access to implementation details.

Encapsulation is implemented by using access specifiers. An access specifier defines the scope and visibility of a class member. C# supports the following access specifiers:

- Public
- Private
- Protected
- Internal
- Protected internal

Public Access Specifier:-

Public access specifier allows a class to expose its member variables and member functions to other functions and objects. Any public member can be accessed from outside the class.

Private Access Specifier:-

Private access specifier allows a class to hide its member variables and member functions from other functions and objects. Only functions of the same class can access its private members. Even an instance of a class cannot access its private members.

Protected Access Specifier:-

Protected access specifier allows a child class to access the member variables and member functions of its base class.

Internal Access Specifier:-

Internal access specifier allows a class to expose its member variables and member functions to other functions and objects in the current assembly. In other

words, any member with internal access specifier can be accessed from any class or method defined within the application in which the member is defined.

Protected Internal Access Specifier: - The protected internal access specifier allows a class to hide its member variables and member functions from other class objects and functions, except a child class within the same application.

Inheritance:-

Inheritance allows us to define a class in terms of another class, which makes it easier to create and maintain an application. This also provides an opportunity to reuse the code functionality and speeds up implementation time. When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the base class, and the new class is referred to as the derived class.

Multiple Inheritance in C#

C# does not support multiple inheritance. However, we can use interfaces to implement multiple inheritance.

Polymorphism:-

In object-oriented programming paradigm, polymorphism is often expressed as 'one interface, multiple functions'. Polymorphism can be static or dynamic.

In static polymorphism, the response to a function is determined at the compile time. In dynamic polymorphism, it is decided at run-time.

Static Polymorphism:-

The mechanism of linking a function with an object during compile time is called early binding. It is also called static binding.

Dynamic Polymorphism:-

C# allows creating abstract classes that are used to provide partial class implementation of an interface. Implementation is completed when a derived

class inherits from it. Abstract classes contain abstract methods, which are implemented by the derived class. The derived classes have more specialized functionality.

Conclusion:- C# is an object oriented language which is use to make web pages. By using other functionality we can make web page more attractive and dynamic. C# application types include Windows Console applications, Windows Forms applications, ASP.NET Web applications, ASP.NET Web Service applications, Smart Device applications, ActiveX applications, and setup and deployment applications.

References:-

1. http://searchsoa.techtarget.com/definition/object-oriented-programming
2. http://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html
3. http://ijcta.com/documents/volumes/vol2issue3/ijcta2011020343.pdf
4. http://stackoverflow.com/questions/6008116/exception-handling-in-c-sharp-research-paper
5. http://www.sciencedirect.com/science/article/pii/S1875389212002908
6. https://msdn.microsoft.com/en-us/library/ms228501(v=vs.90).aspx