C SHARP

**INTRODUCTION**

C# (pronounced as C sharp) is a general-purpose, object-oriented programming language. It is one of the most popular languages used for developing desktop and web applications.

**VARIABLES:**

A variable is a name of memory location. It is used to store data. Its value can be changed and it can be reused many times. It is a way to represent memory location through symbol so that it can be easily identified.

|  |  |
| --- | --- |
| **VARIABLE TYPES** | **EXAMPLE** |
| Decimal | decimal |
| Boolean | True or False |
| Integer | Int , char, float, byte, long |
| Floating Point | Float and double |
| Nullable | Null value |

**Datatypes :**

There are three types of datatypes :

* Value Types: The value data types are integer-based and floating-point based. C# language supports both signed and unsigned literals. Ex. short, int, char, float, double etc.
* Reference Types: The reference data types do not contain the actual data stored in a variable, but they contain a reference to the variables. If the data is changed by one of the variables, the other variable automatically reflects this change in value. Ex. Object, String and Classes.
* Pointer Type: The pointer in C# language is a variable, it is also known as locator or indicator that points to an address of a value. The symbol used for pointer are & (address operator) and \* (Indication Operator).

**Operators:**

An operator is simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise etc.

* Arithmetic Operators (+, -, \* , /,%)
* Relational Operators (<, > ,<=, >=)
* Logical Operators ( | ,& , ^)
* Assignment Operators (==,+=,-=,\*=)

**Control Statements**

1. **If-else:** It is used for testing conditions. There are many types of if statements such as :

* if statement : if statement tests the condition. It is executed if condition is true.
* if-else statement : if-else statement tests the condition. It executes the if block if condition is true otherwise else block is execute.
* nested if statement :
* if-else-if ladder

1. **Switch- Case**: The switch statement executes one statement from multiple conditions. It is like if-else-if ladder statement.
2. **For Loop**: For loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop than while or do-while loops. In for loop there are also nested for loop and infinite for loop.
3. **While Loop**: while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop than for loop. There also nested while loop and infinite while loop available.
4. **Do-While Loop** : do-while loop is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once.
5. **Break Statement:** break is used to break loop or switch statement. It breaks the current flow of the program at the given condition. In case of inner loop, it breaks only inner loop.
6. **Continue Statement:** continue statement is used to continue loop. It continues the current flow of the program and skips the remaining code at specified condition. In case of inner loop, it continues only inner loop.
7. **Comments:**  comments are statements that are not executed by the compiler. By the help of comments, you can hide the program code also. There are two types of comments single line and multi lines comments . “//” is used for single line comment and “/\* \*/” is used for multiline comments.

**Functions:**

Function is a block of code that has a signature. Function is used to execute statements specified in the code block. A function consists of the following components

* Function name: It is a unique name that is used to make Function call.
* Return type: It is used to specify the data type of function return value.
* Body: It is a block that contains executable statements.
* Access specifier: It is used to specify function accessibility in the application.
* Parameters: It is a list of arguments that we can pass to the function during call.

There are two types of function call :

1. Call by Value: value-type parameters are that pass a copy of original value to the function rather than reference. It does not modify the original value. A change made in passed value does not alter the actual value
2. Call by Reference: a ref keyword to pass argument as reference-type. It passes reference of arguments to the function rather than copy of original value. The changes in passed values are permanent and modify the original variable value.

**Array**: Array is a collection of similar types of elements that have contiguous memory location. In C#, array is an object of base type System.Array. In C#, array index starts from 0. We can store only fixed set of elements in C# array.

Advantages:

* Code Optimization (less code)
* Random Access
* Easy to traverse data
* Easy to manipulate data
* Easy to sort data etc.

Dis-Advantages:

* Fixed-Size

There are 3 types of arrays

1. Single Dimensional Array : To create single dimensional array, you need to use square brackets [] after the type.
2. Multidimensional Array : The multidimensional array is also known as rectangular arrays in C#. It can be two dimensional or three dimensional. The data is stored in tabular form (row \* column) which is also known as matrix.
3. Jagged Array : Jagged array is also known as "array of arrays" because its elements are arrays. The element size of jagged array can be different.

**Params :**

params is a keyword which is used to specify a parameter that takes variable number of arguments. It is useful when we don't know the number of arguments prior. Only one params keyword is allowed and no additional parameter is permitted after params keyword in a function declaration.

**Collections:** collection represents group of objects. By the help of collections, we can perform various operations on objects such as

* store object
* update object
* delete object
* retrieve object
* search object
* sort object

1. **List<T> :** List<T> class is used to store and fetch elements. It can have duplicate elements. It is found in System.Collections.Generic namespace.
2. **HashSet** HashSet class can be used to store, remove or view elements. It does not store duplicate elements. It is suggested to use HashSet class if you have to store only unique elements. It is found in System.Collections.Generic. namespace.
3. **Stack<T>**: Stack<T> class is used to push and pop elements. It uses the concept of Stack that arranges elements in LIFO (Last In First Out) order. It can have duplicate elements. It is found in System.Collections.Generic namespace.
4. **Queue<T>:** Queue<T> class is used to Enqueue and Dequeue elements. It uses the concept of Queue that arranges elements in FIFO (First In First Out) order. It can have duplicate elements. It is found in System.Collections.Generic namespace.

**Object:** Object is a real world entity, for example, chair, car, pen, mobile, laptop etc. In other words, object is an entity that has state and behavior. Here, state means data and behavior means functionality. Object is a runtime entity, it is created at runtime. Object is an instance of a class. All the members of the class can be accessed through object.

**Class:** Class is a group of similar objects. It is a template from which objects are created. It can have fields, methods, constructors etc.

**Constructure**: constructor is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new object generally. The constructor in C# has the same name as class or struct. There can be two types of constructors in C#.

* Default constructor : A constructor which has no argument is known as default constructor. It is invoked at the time of creating object.
* Parameterized constructor : A constructor which has parameters is called parameterized constructor. It is used to provide different values to distinct objects.

**Destructor:** A destructor works opposite to constructor, It destructs the objects of classes. It can be defined only once in a class. Like constructors, it is invoked automatically. “~” symbol is used for destruct.

**This:** this is a keyword that refers to the current instance of the class. There can be 3 main usage of this keyword in C#.

* It can be used to refer current class instance variable. It is used if field names (instance variables) and parameter names are same, that is why both can be distinguish easily.
* It can be used to pass current object as a parameter to another method.
* It can be used to declare indexers.

**Static:** static is a keyword or modifier that belongs to the type not instance. So instance is not required to access the static members. In C#, static can be field, method, constructor, class, properties, operator.

**Enum:** Enum in C# is also known as enumeration. It is used to store a set of named constants such as season, days, month, size etc. Enum can be declared within or outside class and structs. Enum constants has default values which starts from 0 and incremented to one by one. But we can change the default value.

* Enum has fixed set of constants
* Enum improves type safety
* Enum can be traversed

**Inheritance:** Inheritance is a process in which one class acquires all the properties and behaviors of its parent class automatically. In such way, you can reuse, extend or modify the attributes and behaviors which is defined in other class. The class which inherits the members of another class is called derived class and the class whose members are inherited is called base class.

Advantages:

Code reusability: Now you can reuse the members of your parent class. So, there is no need to define the member again. So less code is required in the class.

There are different types of inheritances such as

1. Single Inheritance: When one class inherits another class, it is known as single level inheritance.
2. Multilevel Inheritance : When one class inherits another class which is further inherited by another class, it is known as multi level inheritance in C#. Inheritance is transitive so the last derived class acquires all the members of all its base classes.
3. Multilevel Inheritance: A derived class inherits from another derived class.
4. Hierarchical Inheritance: Multiple classes are derived from a single base class.

**Polymorphism :** Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways. There are two types of Polymorphism like Run time polymorphism and compile time polymorphism.

**Compile Time Polymorphism:**

* Method Overloading: Having two or more methods with same name but different in parameters, is known as method overloading in C#. The advantage of method overloading is that it increases the readability of the program because you don't need to use different names for same action.
* Parameter Overloading: Operator overloading in C# allows you to redefine how operators like +, -, \*, /, ==, !=, etc., behave for objects of your own custom classes. For example + operator is used for both addition and concatenation.

**Run Time Polymorphism:**

* Method Overriding: Method overriding in C# allows a subclass to provide a specific implementation of a method that is already defined in its superclass. In simple terms, it's like saying, "Even though my parent class has a method with this name, I want to give it a different behavior in my subclass."

**Access Modifier:**

|  |  |
| --- | --- |
| **Modifier** | **Description** |
| Public | The code is accessible for all classes. |
| Private | The code is only accessible within the same class |
| Protected | The code is accessible within the same class, or in a class that is inherited from that class. |
| Internal | The code is only accessible within its own assembly, but not from another assembly. |

**Abstraction:** Data abstraction is the process of hiding certain details and showing only essential information to the user. Abstraction can be achieved by abstract class.

* Abstract class: is a restricted class that cannot be used to create objects . To access it, it must be inherited from another class.
* Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the derived class.

**Encapsulation**: Encapsulation is the concept of wrapping data into a single unit. It collects data members and member functions into a single unit called class. The purpose of encapsulation is to prevent alteration of data from outside. This data can only be accessed by getter functions of the class. A fully encapsulated class has getter and setter functions that are used to read and write data. This class does not allow data access directly.