C# Variable

* 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.

The basic variable type available in C# can be categorized as:

|  |  |
| --- | --- |
| **Variable Type** | **Example** |
| Decimal types | decimal |
| Boolean types | True or false value, as assigned |
| Integral types | int, char, byte, short, long |
| Floating point types | float and double |
| Nullable types | Nullable data types |

Syntax to declare a variable: **type variable\_list;**

### **Rules for defining variables**

1. A variable can have alphabets, digits and underscore.
2. A variable name can start with alphabet and underscore only. It can't start with digit.
3. No white space is allowed within variable name.
4. A variable name must not be any reserved word or keyword e.g. char, float etc.

# Data Types

**Data types in**[C#](https://www.geeksforgeeks.org/introduction-to-c-sharp/)**is mainly divided into three categories**

* **Value Data Types**
* **Reference Data Types**
* **Pointer Data Type**

1. **Value Data Types :** In [C#](https://www.geeksforgeeks.org/introduction-to-c-sharp/), the Value Data Types will directly store the variable value in memory and it will also accept both signed and literals.
2. **Signed & Unsigned Integral Types :**

| **Alias** | **Type Name** | **Type** | **Size(bits)** | **Range** | **Default Value** |
| --- | --- | --- | --- | --- | --- |
| sbyte | System.Sbyte | signed integer | 8 | -128 to 127 | 0 |
| short | System.Int16 | signed integer | 16 | -32768 to 32767 | 0 |
| Int | System.Int32 | signed integer | 32 | -231 to 231-1 | 0 |
| long | System.Int64 | signed integer | 64 | -263 to 263-1 | 0L |
| byte | System.byte | unsigned integer | 8 | 0 to 255 | 0 |
| ushort | System.UInt16 | unsigned integer | 16 | 0 to 65535 | 0 |
| uint | System.UInt32 | unsigned integer | 32 | 0 to 232 | 0 |
| ulong | System.UInt64 | unsigned integer | 64 | 0 to 263 | 0 |

1. **Floating Point Types :**

| **Alias** | **Type name** | **Size(bits)** | **Range (aprox)** | **Default Value** |
| --- | --- | --- | --- | --- |
| float | System.Single | 32 | ±1.5 × 10-45 to ±3.4 × 1038 | 0.0F |
| double | System.Double | 64 | ±5.0 × 10-324 to ±1.7 × 10308 | 0.0D |

1. **Decimal Types :**

| **Alias** | **Type name** | **Size(bits)** | **Range (aprox)** | **Default value** |
| --- | --- | --- | --- | --- |
| decimal | System.Decimal | 128 | ±1.0 × 10-28 to ±7.9228 × 1028 | 0.0M |

1. **Character Types :**

| **Alias** | **Type name** | **Size In(Bits)** | **Range** | **Default value** |
| --- | --- | --- | --- | --- |
| char | System.Char | 16 | U +0000 to U +ffff | ‘\0’ |

**Boolean Types :**

| **Alias** | **Type name** | **Values** |
| --- | --- | --- |
| bool | System.Boolean | True / False |

# C# Access Modifiers / Specifiers

These are the keywords that are used to specify accessibility or scope of variables and functions in the C# application.

## Difference between the Call by Value and Call by Reference

The following table lists the differences between the call-by-value and call-by-reference methods of parameter passing.

| **Call By Value** | **Call By Reference** |
| --- | --- |
| While calling a function, we pass the values of variables to it. Such functions are known as “Call By Values”. | While calling a function, instead of passing the values of variables, we pass the address of variables(location of variables) to the function known as “Call By References. |
| In this method, the value of each variable in the calling function is copied into corresponding dummy variables of the called function. | In this method, the address of actual variables in the calling function is copied into the dummy variables of the called function. |
| With this method, the changes made to the dummy variables in the called function have no effect on the values of actual variables in the calling function. | With this method, using addresses we would have access to the actual variables and hence we would be able to manipulate them. |
| In call-by-values, we cannot alter the values of actual variables through function calls. | In call by reference, we can alter the values of variables through function calls. |
| Values of variables are passed by the Simple technique. | Pointer variables are necessary to define to store the address values of variables. |
| This method is preferred when we have to pass some small values that should not change. | This method is preferred when we have to pass a large amount of data to the function. |