C# - 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.ValueType**.

The value types directly contain data. Some examples are **int, char, and float**, which stores numbers, alphabets, and floating point numbers, respectively. When you declare an **int** type, the system allocates memory to store the value.

The following table lists the available value types in C# 2010 −

Type	Represents	Range	Default Value
bool	Boolean value	True or False	False
byte	8-bit unsigned integer	0 to 255	0
char	16-bit Unicode character	U +0000 to U +ffff	'\0'
decimal	128-bit precise decimal values with 28-29 significant digits	$(-7.9 \times 10^{28}$ to $7.9 \times 10^{28}) / 10^{0}$ to 28	0.0M
double	64-bit double-precision floating point type	$(+/-)5.0 \times 10^{-324}$ to $(+/-)1.7 \times 10^{308}$	0.0D
float	32-bit single-precision floating point type	-3.4×10^{38} to $+ 3.4 \times 10^{38}$	0.0F
int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
long	64-bit signed integer type	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
sbyte	8-bit signed integer type	-128 to 127	0
short	16-bit signed integer type	-32,768 to 32,767	0
uint	32-bit unsigned integer type	0 to 4,294,967,295	0
ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0
ushort	16-bit unsigned integer type	0 to 65,535	0

To get the exact size of a type or a variable on a particular platform, you can use the **sizeof** method. The expression *sizeof(type)* yields the storage size of the object or type in bytes. Following is an example to get the size of *int* type on any machine −

Live Demo

```
using System;

namespace DataTypeApplication {
   class Program {
      static void Main(string[] args) {
         Console.WriteLine("Size of int: {0}", sizeof(int));
         Console.ReadLine();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result −

```
Size of int: 4
```

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. Example of **built-in** reference types are: **object**, **dynamic,** and **string**.

Object Type

The **Object Type** is the ultimate base class for all data types in C# Common Type System (CTS). Object is an alias for System.Object class. The object types can be assigned values of any other types, value types, reference types, predefined or user-defined types. However, before assigning values, it needs type conversion.

When a value type is converted to object type, it is called **boxing** and on the other hand, when an object type is converted to a value type, it is called **unboxing**.

```
object obj;
obj = 100; // this is boxing
```

Dynamic Type

You can store any type of value in the dynamic data type variable. Type checking for these types of variables takes place at run-time.

Syntax for declaring a dynamic type is −

```
dynamic <variable_name> = value;
```

For example,

```
dynamic d = 20;
```

Dynamic types are similar to object types except that type checking for object type variables takes place at compile time, whereas that for the dynamic type variables takes place at run time.

String Type

The **String Type** allows you to assign any string values to a variable. The string type is an alias for the System.String class. It is derived from object type. The value for a string type can be assigned using string literals in two forms: quoted and @quoted.

For example,

```
String str = "Tutorials Point";
```

A @quoted string literal looks as follows −

```
@"Tutorials Point";
```

The user-defined reference types are: class, interface, or delegate. We will discuss these types in later chapter.

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;
```

We will discuss pointer types in the chapter 'Unsafe Codes'.