

Variable Types

This lesson gives an overview of all the types of variables in C# like int, bool, double, char and float and much more!

WE'LL COVER THE FOLLOWING



- Integer
- Float
- Char
- Boolean
- Sizes
 - Stay away from any possible error!

Integer

An *integer* is a number that does not have any decimal places. It is a whole number, for example, **1,2,3,4** are all integers. **4.3** is not. If you were to try and place the number **4.3** into an integer, the number would be truncated to **4**.





There are further different types in an integer as well. All of them differ in size which we'll discuss further down the lesson.





Let's take a look at them one by one.

```
using System;

namespace VariablesExampleOne
{
    class Program
    {
        static void Main(string[] args)
        {
            sbyte a = -128;
            Console.WriteLine("The variable a contains " + a);
            byte b = 255;
            Console.WriteLine("The variable b contains " + b);
            short c = 32767;
            Console.WriteLine("The variable c contains " + c);
        }
    }
}
```



A `char` is an **8-bit** integer. This means that an `unsigned char` can store between **0 and 255**, and a `signed char` can store between **-128 and 127**.

Unsigned chars are commonly used to store text in *ASCII* format. A `char` can be initialized to hold either a number or a character, but it will store only the *ASCII* value.

```
using System;

namespace VariablesExampleOne
{
    class Program
    {
        static void Main(string[] args)
        {
            char a = 'a';
            Console.WriteLine("The variable a contains " + a);
        }
    }
}
```



Note: *ASCII* is a system where a numerical value is assigned to every character you can think of. For a complete conversion chart visit <http://ascii-code.com/>

Boolean

The `bool` (boolean) type is a 1-byte data type that is either true or false. A true being any number other than zero and false being zero. The true keyword uses the value 1 to assign true.

```
using System;

class BooleanExample {
    static void Main() {
        bool canJump = false;
        bool canDo = true;
        Console.WriteLine("Value of canJump is: {0}", canJump);
        Console.WriteLine("Value of canDo is: {0}", canDo);
    }
}
```



Sizes

This table portrays how the variable sizes vary with the variable types.

C# Alias	Size (bits)	Range
sbyte	8	-128 to 127
byte	8	0 to 255
short	16	-32,768 to 32,767
ushort	16	0 to 65,535
char	16	A unicode character of code 0 to 65,535
int	32	-2,147,483,648 to 2,147,483,647
uint	32	0 to 4,294,967,295
long	64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
ulong	64	0 to 18,446,744,073,709,551,615
float	32	1.5×10^{-45} to 3.4×10^{38}
double	64	5.0×10^{-324} to 1.7×10^{-308}

decimal	128	1.0×10^{-28} to 7.9×10^{28}
bool	32	true or false
string	16*length	A unicode string with no special upper bound.
object	32/64	Platform dependent (a pointer to an object).

Stay away from any possible error!

If you'll try to store some value that exceeds the specific range of a variable type, you'll get a runtime error:

```
using System;

namespace VariablesExample
{
    class Program
    {
        static void Main(string[] args)
        {
            sbyte a = -129;
            Console.WriteLine("The variable a contains " + a);
        }
    }
}
```



Here, the type **sbyte** can store integers from **-128 to 127**. **-129** is out of its limit, and therefore, it will throw an error.

Cool, right? Let's move onto variable casting!

Cool, right? Let's move onto variable casting.