

## Part I: The C# Language

| Name              | CTS Type                    | Values                                  |
|-------------------|-----------------------------|---|
| <code>bool</code> | <code>System.Boolean</code> | <code>true</code> or <code>false</code> |

You cannot implicitly convert `bool` values to and from integer values. If a variable (or a function return type) is declared as a `bool`, you can only use values of `true` and `false`. You will get an error if you try to use zero for `false` and a non-zero value for `true`.

### The Character Type

For storing the value of a single character, C# supports the `char` data type.

| Name              | CTS Type                 | Values   |
|-------------------|--------------------------|--|
| <code>char</code> | <code>System.Char</code> | Represents a single 16-bit (Unicode) character |

Although this data type has a superficial resemblance to the `char` type provided by C and C++, there is a significant difference. C++ `char` represents an 8-bit character, whereas a C# `char` contains 16 bits. This is part of the reason that implicit conversions between the `char` type and the 8-bit `byte` type are not permitted.

Although 8 bits may be enough to encode every character in the English language and the digits 0–9, they aren't enough to encode every character in more expansive symbol systems (such as Chinese). In a gesture toward universality, the computer industry is moving away from the 8-bit character set and toward the 16-bit Unicode scheme, of which the ASCII encoding is a subset.

Literals of type `char` are signified by being enclosed in single quotation marks, for example `'A'`. If you try to enclose a character in double quotation marks, the compiler will treat this as a string and throw an error.

As well as representing `chars` as character literals, you can represent them with four-digit hex Unicode values (for example `'\u0041'`), as integer values with a cast (for example, `(char)65`), or as hexadecimal values (`'\x0041'`). They can also be represented by an escape sequence, as shown in the following table.

| Escape Sequence | Character             |
|-----------------|-----------------------|
| <code>\'</code> | Single quotation mark |
| <code>\"</code> | Double quotation mark |
| <code>\\</code> | Backslash             |
| <code>\0</code> | Null                  |
| <code>\a</code> | Alert                 |