[**https://msdn.microsoft.com/en-us/library/s3f49ktz.aspx**](https://msdn.microsoft.com/en-us/library/s3f49ktz.aspx)

**Data Type Ranges**

**Visual Studio 2015**

[Other Versions](javascript:void(0))

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=636107049706926704

Visual C++ 32-bit and 64-bit compilers recognize the types in the table later in this article.

* **int** (**unsigned** **int**)
* **\_\_int8** (**unsigned** **\_\_int8**)
* **\_\_int16** (**unsigned** **\_\_int16**)
* **\_\_int32** (**unsigned** **\_\_int32**)
* **\_\_int64** (**unsigned** **\_\_int64**)
* **short** (**unsigned** **short**)
* **long** (**unsigned** **long**)
* **long** **long** (**unsigned** **long** **long**)

If its name begins with two underscores (**\_\_**), a data type is non-standard.

The ranges that are specified in the following table are inclusive-inclusive.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type Name** | **Bytes** | **Other Names** | **Range of Values** |
| int | 4 | signed | –2,147,483,648 to 2,147,483,647 |
| unsigned int | 4 | unsigned | 0 to 4,294,967,295 |
| \_\_int8 | 1 | char | –128 to 127 |
| unsigned \_\_int8 | 1 | unsigned char | 0 to 255 |
| \_\_int16 | 2 | short, short int, signed short int | –32,768 to 32,767 |
| unsigned \_\_int16 | 2 | unsigned short, unsigned short int | 0 to 65,535 |
| \_\_int32 | 4 | signed, signed int, int | –2,147,483,648 to 2,147,483,647 |
| unsigned \_\_int32 | 4 | unsigned, unsigned int | 0 to 4,294,967,295 |
| \_\_int64 | 8 | long long, signed long long | –9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| unsigned \_\_int64 | 8 | unsigned long long | 0 to 18,446,744,073,709,551,615 |
| bool | 1 | none | false or true |
| char | 1 | none | –128 to 127 by default  0 to 255 when compiled by using [/J](https://msdn.microsoft.com/en-us/library/0d294k5z.aspx) |
| signed char | 1 | none | –128 to 127 |
| unsigned char | 1 | none | 0 to 255 |
| short | 2 | short int, signed short int | –32,768 to 32,767 |
| unsigned short | 2 | unsigned short int | 0 to 65,535 |
| long | 4 | long int, signed long int | –2,147,483,648 to 2,147,483,647 |
| unsigned long | 4 | unsigned long int | 0 to 4,294,967,295 |
| long long | 8 | none (but equivalent to \_\_int64) | –9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| unsigned long long | 8 | none (but equivalent to unsigned \_\_int64) | 0 to 18,446,744,073,709,551,615 |
| enum | varies | none | See [3691ceca-05fb-4b82-b1ae-5c4618cda91a#bkmkRemarks](https://msdn.microsoft.com/en-us/library/s3f49ktz.aspx#NotExistJustToMakeTheAElementVisible) later in this article |
| float | 4 | none | 3.4E +/- 38 (7 digits) |
| double | 8 | none | 1.7E +/- 308 (15 digits) |
| long double | same as double | none | Same as double |
| wchar\_t | 2 | \_\_wchar\_t | 0 to 65,535 |

Depending on how it's used, a variable of **\_\_wchar\_t** designates either a wide-character type or multibyte-character type. Use the **L** prefix before a character or string constant to designate the wide-character-type constant.

**signed** and **unsigned** are modifiers that you can use with any integral type except **bool**. Note that **char**, **signed char**, and **unsigned char** are three distinct types for the purposes of mechanisms like overloading and templates.

The **int** and **unsigned** **int** types have a size of four bytes. However, portable code should not depend on the size of **int** because the language standard allows this to be implementation-specific.

C/C++ in Visual Studio also supports sized integer types. For more information, see [\_\_int8, \_\_int16, \_\_int32, \_\_int64](https://msdn.microsoft.com/en-us/library/29dh1w7z.aspx) and [Integer Limits](https://msdn.microsoft.com/en-us/library/296az74e.aspx).

For more information about the restrictions of the sizes of each type, see [Fundamental Types (C++)](https://msdn.microsoft.com/en-us/library/cc953fe1.aspx).

The range of enumerated types varies depending on the language context and specified compiler flags. For more information, see [C Enumeration Declarations](https://msdn.microsoft.com/en-us/library/whbyts4t.aspx) and [Enumerations (C++)](https://msdn.microsoft.com/en-us/library/2dzy4k6e.aspx).

**See Also**

[Keywords (C++)](https://msdn.microsoft.com/en-us/library/2e6a4at9.aspx)  
[Fundamental Types (C++)](https://msdn.microsoft.com/en-us/library/cc953fe1.aspx)