__int8, __int16, __int32, __int64

Article • 08/03/2021

Microsoft-specific

Microsoft C/C++ features support for sized integer types. You can declare 8-, 16-, 32-, or 64-bit integer variables by using the `__intN` type specifier, where `N` is 8, 16, 32, or 64.

The following example declares one variable for each of these types of sized integers:

C++

```cpp
__int8 nSmall;      // Declares 8-bit integer
__int16 nMedium;    // Declares 16-bit integer
__int32 nLarge;     // Declares 32-bit integer
__int64 nHuge;      // Declares 64-bit integer
```

The types `__int8`, `__int16`, and `__int32` are synonyms for the ANSI types that have the same size, and are useful for writing portable code that behaves identically across multiple platforms. The `__int8` data type is synonymous with type `char`, `__int16` is synonymous with type `short`, and `__int32` is synonymous with type `int`. The `__int64` type is synonymous with type `long long`.

For compatibility with previous versions, `_int8`, `_int16`, `_int32`, and `_int64` are synonyms for `__int8`, `__int16`, `__int32`, and `__int64` unless compiler option /Za (Disable language extensions) is specified.

Example

The following sample shows that an `__intN` parameter will be promoted to `int`:

C++

```cpp
// sized_int_types.cpp

#include <stdio.h>

void func(int i) {
    printf_s("%s\n", __FUNCTION__);
}

int main()
{
    __int8 i8 = 100;
    func(i8);   // no void func(__int8 i8) function
                // __int8 will be promoted to int
}
```

Output

```
func
```

See also

Keywords
Built-in types
Data Type Ranges

Feedback