Tipos de enteros nativos en C y C++

Tipo de dato	Número de bits garantizados	Número de bits usual	Modificadores	Rango usual de valores
char	≥ 8	8	**signed**	[-128, 127]
			unsigned	[0, 255]
short	≥ 16	16	**signed**	[-32768, 32767]
			unsigned	[0, 65535]
int	≥ 16	32	**signed**	[-2147483648, 2147483647]
			unsigned	[0, 4294967295]
long	≥ 32	32[1]	**signed**	[-2147483648, 2147483647]
			unsigned	[0, 4294967295]
long long	≥ 64	64	**signed**	[-9223372036854775808, 9223372036854775807]
			unsigned	[0, 18446744073709551615]

En todas las plataformas modernas, el tipo **char** tiene 8 bits. Nosotros definiremos un *byte* como una unidad de información que ocupa 8 bits, por lo que un **char** ocupa un byte. Tanto en C como en C++ se define **sizeof(char)** = 1. Es decir, **sizeof(T)** es la cantidad de bytes que ocupa una variable de tipo **T**.

Todos los tipos enteros son **signed** por omisión, excepto **char** que depende de la plataforma (en algunas plataformas, **char** actúa como **signed** y en otras como **unsigned**). Si no investigan para su plataforma y tampoco especifican el modificador, sólo pueden usar el rango [0, 127] de manera segura.

Tipos enteros de ancho fijo en C y C++ (**stdint.h** / **cstdint**)

Tipo de dato	Número de bits garantizados	Rango garantizado de valores	Rango usual de valores
int8_t[2]	8	[-127, 127]	[-128, 127]
uint8_t[3]	8	[0, 255]	[0, 255]
int16_t	16	[-32767, 32767]	[-32768, 32767]
uint16_t	16	[0, 65535]	[0, 65535]
int32_t	32	[-2147483647, 2147483647]	[-2147483648, 2147483647]
uint32_t	32	[0, 4294967295]	[0, 4294967295]
int64_t	64	[-9223372036854775807, 9223372036854775807]	[-9223372036854775808, 9223372036854775807]
uint64_t	64	[0, 18446744073709551615]	[0, 18446744073709551615]

1 En Linux de 64 bits, **long** es de 64 bits.

2 Tanto **cin** o **cout** confunden **int8_t** con **signed char** y lo procesan como caracter, no como entero**.** Pueden leerlo como entero usando **scanf** con **"%hhd"** y pueden imprimirlo usando **printf** con **"%d"**.

3 Tanto **cin** o **cout** confunden **uint8_t** con **unsigned char** y lo procesan como caracter, no como entero**.** Pueden leerlo como entero usando **scanf** con **"%hhu"** y pueden imprimirlo usando **printf** con **"%u"**.

Otros tipos enteros

Tipo de dato	Encabezado de inclusión	Descripción o rango de valores
bool	**stdbool.h** (C) Disponible sin encabezado (C++)	[0, 1][4]
size_t	**stddef.h** **cstddef**	Entero sin signo capaz de almacenar cualquier índice de un arreglo
intptr_t	**stdint.h** **cstdint**	Entero con signo con al menos la misma cantidad de bits que un apuntador
uintptr_t	**stdint.h** **cstdint**	Entero sin signo con al menos la misma cantidad de bits que un apuntador

Cómo examinar las propiedades de los tipos de datos

```
// Examinar las propiedades de los tipos de datos al estilo C con limits.h o climits
// http://www.cplusplus.com/reference/climits/
#include <climits>
#include <iostream>
using namespace std;

int main( )
{
  cout << CHAR_MIN << " " << CHAR_MAX << "\n";
  cout << SHRT_MIN << " " << SHRT_MAX << "\n";
  cout << INT_MIN << " " << INT_MAX << "\n";
  cout << LONG_MIN << " " << LONG_MAX << "\n";
  cout << LLONG_MIN << " " << LLONG_MAX << "\n";
}
```

```
// Examinar las propiedades de los tipos de datos al estilo C++ con limits
// http://www.cplusplus.com/reference/limits/numeric_limits/
#include <limits>
#include <iostream>
using namespace std;

int main( )
{
  cout << numeric_limits<char>::min( ) << " " << numeric_limits<char>::max( ) << "\n";
  cout << numeric_limits<short>::min( ) << " " << numeric_limits<short>::max( ) << "\n";
  cout << numeric_limits<int>::min( ) << " " << numeric_limits<int>::max( ) << "\n";
  cout << numeric_limits<long>::min( ) << " " << numeric_limits<long>::max( ) << "\n";
  cout << numeric_limits<long long>::min( ) << " " << numeric_limits<long long>::max( ) << "\n";
}
```

4 A pesar de su rango limitado, **sizeof(bool)** ≥ 1. Esto quiere decir que un variable de tipo **bool** consume al menos un byte a pesar de que teóricamente basta un bit para representar el rango entero [0, 1].