Table 23.6 Character Macros in `<limits.h>`

Name	*Value*	*Description*
`CHAR_BIT`	≥8	Number of bits per byte
`SCHAR_MIN`	≤–127	Minimum `signed char` value
`SCHAR_MAX`	≥+127	Maximum `signed char` value
`UCHAR_MAX`	≥255	Maximum `unsigned char` value
`CHAR_MIN`	†	Minimum `char` value
`CHAR_MAX`	††	Maximum `char` value
`MB_LEN_MAX`	≥1	Maximum number of bytes per multibyte character in any supported locale (see Section 25.2)

†`CHAR_MIN` is equal to `SCHAR_MIN` if `char` is treated as a signed type; otherwise, `CHAR_MIN` is 0.
††`CHAR_MAX` has the same value as either `SCHAR_MAX` or `UCHAR_MAX`, depending on whether `char` is treated as a signed type or an unsigned type.

`unsigned long int`. Table 23.7 lists these macros and shows the maximum or minimum value of each; the formula used to compute each value is also given.

C99 Note that C99 provides three macros that describe the characteristics of the `long long int` types.

Table 23.7 Integer Macros in `<limits.h>`

Name	*Value*	*Formula*	*Description*
`SHRT_MIN`	≤–32767	$-(2^{15}-1)$	Minimum `short int` value
`SHRT_MAX`	≥+32767	$2^{15}-1$	Maximum `short int` value
`USHRT_MAX`	≥65535	$2^{16}-1$	Maximum `unsigned short int` value
`INT_MIN`	≤–32767	$-(2^{15}-1)$	Minimum `int` value
`INT_MAX`	≥+32767	$2^{15}-1$	Maximum `int` value
`UINT_MAX`	≥65535	$2^{16}-1$	Maximum `unsigned int` value
`LONG_MIN`	≤–2147483647	$-(2^{31}-1)$	Minimum `long int` value
`LONG_MAX`	≥+2147483647	$2^{31}-1$	Maximum `long int` value
`ULONG_MAX`	≥4294967295	$2^{32}-1$	Maximum `unsigned long int` value
`LLONG_MIN`†	≤–9223372036854775807	$-(2^{63}-1)$	Minimum `long long int` value
`LLONG_MAX`†	≥+9223372036854775807	$2^{63}-1$	Maximum `long long int` value
`ULLONG_MAX`†	≥18446744073709551615	$2^{64}-1$	Maximum `unsigned long long int` value

†C99 only

The macros in `<limits.h>` are handy for checking whether a compiler supports integers of a particular size. For example, to determine whether the `int` type can store numbers as large as 100,000, we might use the following preprocessing directives:

```
#if INT_MAX < 100000
#error int type is too small
#endif
```

`#error` directive ➤ *14.5*

If the `int` type isn't adequate, the `#error` directive will cause the preprocessor to display an error message.