Table 27.2 <stdint.h> Limit Macros for Other Integer Types

Name	*Value*	*Description*
PTRDIFF_MIN	≤–65535	Minimum ptrdiff_t value
PTRDIFF_MAX	≥+65535	Maximum ptrdiff_t value
SIG_ATOMIC_MIN	≤–127 (if signed) 0 (if unsigned)	Minimum sig_atomic_t value
SIG_ATOMIC_MAX	≥+127 (if signed) ≥255 (if unsigned)	Maximum sig_atomic_t value
SIZE_MAX	≥65535	Maximum size_t value
WCHAR_MIN	≤–127 (if signed) 0 (if unsigned)	Minimum wchar_t value
WCHAR_MAX	≥+127 (if signed) ≥255 (if unsigned)	Maximum wchar_t value
WINT_MIN	≤–32767 (if signed) 0 (if unsigned)	Minimum wint_t value
WINT_MAX	≥+32767 (if signed) ≥65535 (if unsigned)	Maximum wint_t value

```
i = 100000;
```

is problematic, because the constant 100000 might be too large to represent using type int (if int is a 16-bit type). However, the statement

```
i = INT32_C(100000);
```

is safe. If int_least32_t represents the int type, then INT32_C(100000) has type int. But if int_least32_t corresponds to long int, then INT32_C(100000) has type long int.

<stdint.h> has two other parameterized macros. INTMAX_C converts an integer constant to type intmax_t, and UINTMAX_C converts an integer constant to type uintmax_t.

27.2 The <inttypes.h> Header (C99) Format Conversion of Integer Types

Q&A

The <inttypes.h> header is closely related to the <stdint.h> header, the topic of Section 27.1. In fact, <inttypes.h> includes <stdint.h>, so programs that include <inttypes.h> don't need to include <stdint.h> as well. The <inttypes.h> header extends <stdint.h> in two ways. First, it defines macros that can be used in ...printf and ...scanf format strings for input/output of the integer types declared in <stdint.h>. Second, it provides functions for working with greatest-width integers.