depends on either `sizeof(int) < sizeof(long)` or `sizeof(long) < sizeof(long long)`, portability is needlessly restricted. Relying instead on only the guaranteed[4] property that `sizeof(int) < sizeof(long long)` avoids such portability issues since the relative sizes of the `long` and `long long` integral types continue to evolve.

When precise control of size *in the implementation* (as opposed to in the interface) matters, consider using one of the standard signed, `intn_t`, or unsigned, `uintn_t`, integer aliases provided, since C++11, in `<cstdint>` and summarized here in Table 1.

Table 1: Useful `typedefs` found in `<cstdint>` (since C++11)

Exact Size (optional)[a]	Fastest integral type having at least N bits	Smallest integer type having at least N bits
int8_t	int_fast8_t	int_least8_t
int16_t	int_fast16_t	int_least16_t
int32_t	int_fast32_t	int_least32_t
int64_t	int_fast64_t	int_least64_t
uint8_t	uint_fast8_t	uint_least8_t
uint16_t[a]	uint_fast16_t	uint_least16_t
uint32_t	uint_fast32_t	uint_least32_t
uint64_t	uint_fast64_t	uint_least64_t

[a] The compiler doesn't need to fabricate the exact-width type if the target platform doesn't support it.

Note: Also see `intmax_t`, the maximum width integer type, which might be different from all of the above.

See Also

- "Binary Literals" (§1.2, p. 142) explains how programmers can specify binary constants directly in the source code; large binary values might fit only in a `long long` or even `unsigned long long`.
- "Digit Separators" (§1.2, p. 152) describes visually separating digits of large `long long` literals.

Further Reading

- For rationale behind adding the 64-bit integral type to the language, see the original proposal, "Adding the `long long` type to C++," **adamczyk05**.

[4] Due to the unfathomable amount of software that would stop working if an `int` were ever anything but exactly *four* bytes, we — along with the late Richard Stevens of Unix fame (see **stevens93**, section 2.5.1., "ANSI C Limits," pp. 31–32, specifically row 6, column 4 of Figure 2.2, p. 32) — are prepared to *guarantee* that it will never become as large as a `long long` for any general-purpose computer.