score in a ten-pin bowling game, or number of stories in a building. For efficient storage in a **class** or **struct**, however, we may well decide to represent such quantities more compactly using a **short** or **char**; see also the aliases found in C++11's `<cstdint>`.

Sometimes the size of the virtual address space for the underlying architecture itself dictates how large an integer we will need. For example, on a 64-bit platform, specifying the *distance* between two pointers into a contiguous array or the size of the array itself could well exceed the size of an **int** or **unsigned int**, respectively. Using either **long long** or **unsigned long long** here would, however, not be indicated as the respective platform-dependent integer types (**typedef**s) `std::ptrdiff_t` and `std::size_t` are provided expressly for such use and avoid wasting space where it cannot be used by the underlying hardware.

Occasionally, however, the decision of whether to use an **int** is neither platform dependent nor clear cut, in which case using an **int** is almost certainly a bad idea. Suppose we were asked to provide a function, as part of a financial library, that, given a date, returns the number of shares of some particular stock, identified by its security id, `SecId`, traded on the New York Stock Exchange (NYSE).[2] Since the average daily volume of even the most heavily traded stocks — roughly 70 million shares — appears to be well under the maximum value a signed **int** supports (more than 2 billion on our production platforms), we might at first think to write the function to return **int**:

```
int volYMD(SecId equity, int year, int month, int day);  // (1) BAD IDEA
```

One obvious problem with this interface is that the daily fluctuations in turbulent times might exceed the maximum value representable by a 32-bit **int**, which, unless detected internally, would result in **signed integer overflow**, which is both **undefined behavior** and potentially a pervasive defect enabling avenues of deliberate attack from outside sources.[3] What's more, the growth rate of some companies, especially technology startups, has been at times seemingly exponential. To gain an extra insurance factor of two, we might opt to replace the return type **int** with an **unsigned int**:

```
unsigned volYMD(SecId stock, int year, int month, int day);  // (2) BAD IDEA!
```

Use of an **unsigned int**, however, simply delays the inevitable as the number of shares being traded is almost certainly going to grow over time.

Furthermore, the algebra for unsigned quantities is entirely different from what one would normally expect from an **int**. For example, if we were to try to express the day-over-day change in volume by subtracting two calls to this function and if the number of shares traded were to have decreased, then the **unsigned int** difference would wrap, and the result would be a typically large, erroneous value. Because integer literals are themselves of type **int** and not **unsigned**, comparing an unsigned value with a negative signed one does not

[2]There are more than 3,200 listed symbols on the NYSE. Composite daily volume of NYSE-listed securities across all exchanges ranges from 3.5 to 6 billion shares, with a high reached in March 2020 of more than 9 billion shares.

[3]For an overview of integer overflow in C++, see **ballman**. For a more focused discussion of secure coding in CPP using CERT standards, see **seacord13**, Chapter 5, "Integer Security," pp. 225–307.