C.5 Constant Expressions

In places such as array bounds (§5.2), case labels (§6.3.2), and initializers for enumerators (§4.8), C++ requires a *constant expression*. A constant expression evaluates to an integral or enumeration constant. Such an expression is composed of literals (§4.3.1, §4.4.1, §4.5.1), enumerators (§4.8), and ***const***s initialized by constant expressions. In a template, an integer template parameter can also be used (§C.13.3). Floating literals (§4.5.1) can be used only if explicitly converted to an integral type. Functions, class objects, pointers, and references can be used as operands to the ***sizeof*** operator (§6.2) only.

Intuitively, constant expressions are simple expressions that can be evaluated by the compiler before the program is linked (§9.1) and starts to run.

C.6 Implicit Type Conversion

Integral and floating-point types (§4.1.1) can be mixed freely in assignments and expressions. Wherever possible, values are converted so as not to lose information. Unfortunately, value-destroying conversions are also performed implicitly. This section provides a description of conversion rules, conversion problems, and their resolution.

C.6.1 Promotions

The implicit conversions that preserve values are commonly referred to as *promotions*. Before an arithmetic operation is performed, *integral promotion* is used to create ***int***s out of shorter integer types. Note that these promotions will *not* promote to ***long*** (unless the operand is a ***wchar_t*** or an enumeration that is already larger than an ***int***). This reflects the original purpose of these promotions in C: to bring operands to the ''natural'' size for arithmetic operations.

The integral promotions are:

- A ***char***, ***signed char***, ***unsigned char***, ***short int***, or ***unsigned short int*** is converted to an ***int*** if ***int*** can represent all the values of the source type; otherwise, it is converted to an ***unsigned int***.
- A ***wchar_t*** (§4.3) or an enumeration type (§4.8) is converted to the first of the following types that can represent all the values of its underlying type: ***int***, ***unsigned int***, ***long***, or ***unsigned long***.
- A bit-field (§C.8.1) is converted to an ***int*** if ***int*** can represent all the values of the bit-field; otherwise, it is converted to ***unsigned int*** if ***unsigned int*** can represent all the values of the bit-field. Otherwise, no integral promotion applies to it.
- A ***bool*** is converted to an ***int***; ***false*** becomes ***0*** and ***true*** becomes ***1***.

Promotions are used as part of the usual arithmetic conversions (§C.6.3).

C.6.2 Conversions

The fundamental types can be converted into each other in a bewildering number of ways. In my opinion, too many conversions are allowed. For example: