```
const int b;         // Error, const scalar variable must be initialized.
extern const int c;  // OK, declaration
const int d = c;     // OK, not constant initialized (c initializer not seen)

int ca1[c];          // Error, initializer of c is not visible.
int da1[d];          // Error, initializer of d is not a compile-time constant.

const int c = 7;
int ca2[c];          // OK, initializer is visible
int da2[d];          // Error, initializer of d is not a compile-time constant.

const int e = 17;    // OK
int ea[e];           // OK
```

For an **integral constant** to be usable at compile time (i.e., as part of constant expression), three requirements must be satisfied.

1. The variable must be marked **const**.
2. The initializer for a variable must have been seen by the time it is used, and it must be a constant expression; this information is needed for a compiler to be able to make use of the variable in other constant expressions.
3. The variable must be of *integral* type, e.g., **bool**, **char**, **short**, **int**, **long**, **long long**, as well as the **unsigned** variations on these and any additional **char** types; see also Section 1.1."**long long**" on page 89.

This restriction to integral types provides support for those values where compile-time constants are most frequently needed while limiting the complexity of what compilers were required to support at compile time.

Use of **constexpr** when declaring a variable or variable template (see Section 1.2."Variable Templates" on page 157) enables a much richer category of types to participate in constant expressions. This generalization, however, was not made for mere **const** variables because they are not *required* to be initialized by compile-time constants:

```
int f() { return 0; }  // f() is not a compile-time constant expression.

               int x0 = f();  // OK
         const int x1 = f();  // OK, but x1 is not a compile-time constant.
constexpr      int x2 = f();  // Error, f() is not a constant expression.
constexpr const int x3 = f();  // Error, f()  "  "  "     "         "
```

As the example code above demonstrates, variables marked **constexpr** *must* satisfy the same requirements needed for integral constants to be usable in constant expressions. Unlike other integral constants, their initializers *must* be constant expressions, or else the program is **ill formed**.