For a variable of other than **const** integral type to be usable in a constant expression, certain criteria must hold:

1. The variable must be annotated with **constexpr**, which implicitly also declares the variable to be **const**[1]:

```
struct S  // simple (aggregate) literal type
{
    int i;  // built-in integer data member
};

void test1()
{
    constexpr S s{1};  // OK, literal type constant expression initialized
    s = S();           // Error, constexpr implies const.
    static_assert(s.i == 1, "");  // OK, subobjects of constexpr objects are
    constexpr int j = s.i;        //     usable in constant expressions.
    constexpr const int k = 1;    // OK, redundant keyword const
    const constexpr int l = 2;    // OK, keywords in either order
}
```

   In the example above, we have, for expedience of exposition, used braced initialization to initialize the aggregate; see Section 2.1."Braced Init" on page 215. Note that non**mutable** subobjects of **constexpr** objects are also effectively **constexpr** and can be used freely in constant expressions even though they themselves are not explicitly declared **constexpr**.

2. All **constexpr** variables must be initialized with a constant expression when they are defined. Hence, every **constexpr** variable must have an initializer, and that initializer must be a valid constant expression; see Section 2.1."**constexpr** Functions" on page 257:

```
          int g() { return 17; }  // a nonconstexpr function
constexpr int h() { return 34; }  // a constexpr function

constexpr int v1;        // Error, v1 is not initialized.
constexpr int v2 = 17;   // OK
constexpr int v3 = g();  // Error, g() is not constexpr.
constexpr int v4 = h();  // OK

void test2(int c)
{
    constexpr int v5 = c;          // Error, c is not a compile-time constant.
    constexpr int v6 = sizeof(c);  // OK, c is not evaluated.
}
```

[1] C++20 added the **constinit** keyword to identify a variable that is initialized at compile time (with a constant expression) but may subsequently be modified at run time.