

constexpr Functions

Chapter 2 Conditionally Safe Features

defined, defaulted, or deleted, (b) all **nonstatic data members** have trivial default constructors and no default member initializers, and (c) all base classes are **nonvirtual** and have trivial default constructors. Hence, it is possible to **value-initialize** a type that has an explicitly defaulted but not explicitly **constexpr** default constructor:

```
struct S1 // example of a nonconstexpr trivial default constructor
{
    int d_i; // not initialized by S1()
    S1() = default; // trivial, nonconstexpr
};
static_assert(S1().d_i == 0, ""); // OK, value initialization
static_assert(S1{}.d_i == 0, ""); // OK, value initialization
```

Aggregate initialization might produce the even more surprising effect of successful initialization even when matching constructors are **deleted** (see Section 1.1. “Deleted Functions” on page 53), including a **deleted** default constructor⁷:

```
struct S2 // a type having a non-trivial default constructor
{
    constexpr S2() { } // non-trivial, constexpr
};

struct S3 // example of an aggregate having deleted constructors
{
    int d_i; // not initialized
    S2 d_s2; // has non-trivial constructor

    S3() = delete; // non-trivial, nonconstexpr
    S3(int a) = delete; // nonconstexpr
};
static_assert(S3().d_i == 0, ""); // Error, invokes deleted constructor
static_assert(S3{}.d_i == 0, ""); // OK, aggregate initialization
static_assert(S3{7}.d_i == 7, ""); // OK, aggregate initialization
```

Notice that failing to use **braced initialization** results in **value initialization**, rather than **aggregate initialization**, and therefore attempts to invoke the **deleted** default constructor of S3.

7. For a **union**, exactly one of its **data members** must be initialized with a **constant expression** via (1) a **default member initializer** (see Section 2.1. “Default Member Init” on page 318), (2) a **constexpr** constructor, or (3) **aggregate initialization**:

```
// unions having no explicit constructors
union U0 { bool b; char c; }; // OK, neither member initialized
```

⁷Since C++20, a type having any **user-declared** constructors, which includes defaulted and deleted constructors, is no longer considered an **aggregate** and thus **aggregate initialization** does not apply to such types.