```
char i = 4;

struct S6
{
    int j = sizeof(i);  // refers to S6::i, not ::i
    int i = 5;
};

S6 s6;  // OK, s6.j initialized to 4.
```

The **this** pointer can also be safely used as part of a default member initializer. As with any other uses of **this** inside a constructor, care must be exercised because the object referred to by **this** will be in a partially constructed state:

```
int getSomeRuntimeValue();

struct S7
{
    S7* d_selfPtr = this;                   // OK
    int d_bad = this->d_later;              // Bug, d_later not yet initialized
    int d_later = getInitialDLaterValue();  // OK
    static int getInitialDLaterValue();
};
```

Unlike variables at function or global scope and unlike static data members, a default member initializer for a member that is an array of unknown bound will not determine the array bound:

```
struct S8
{
    static int d_s[];        // OK, d_s has unknown bounds.
    int d_a[]  = {1, 2, 3};  // Error, d_a is an array of unknown bound.
    int d_b[3] = {1, 2, 3};  // OK, bound explicitly specified
};

int a[]       = {1, 2, 3};  // OK, the length of a is deduced to 3.
int S8::d_s[] = {4, 5, 6};  // OK, the length of S8::d_s is deduced to 3.
```

Interactions with unions

Default member initializers can also be used with union members. However, only one variant member of a union can have a default member initializer, since that will determine the default initialization of the entire union:

```
union U0
{
    char d_c = 'a';
```