## constexpr Functions

## Chapter 2 Conditionally Safe Features

However, note that, because constructing an object of nonliteral type at compile time is not possible, there is no way to invoke h2 or h4 as part of a constant expression since the access of the member v in all of the above functions requires an already created object to exist. Pointers and references to nonliteral types can be **constexpr** provided they are not used to access their values at compile time:

```
Nlt arr[17];
constexpr Nlt& arr_0 = arr[0]; // OK, initializing a reference
constexpr Nlt* arr_0_ptr = &arr[0]; // OK, taking an address
constexpr Nlt& arr_0_ptr_deref = *arr_0_ptr; // OK, dereferencing but not using
static_assert(&arr[17] - &arr[4] == 13,""); // OK, pointer arithmetic
constexpr int arr_0_v = arr_0.v; // Error, arr[0] is not usable.
constexpr int arr_0_ptr_v = arr_0_ptr->v; // Error, " " " "
```

## Literal types defined

As discussed understanding which types are literal types is important for knowing what can and cannot be done during compile-time evaluation. We now elucidate how the language defines a literal type and, as such, how they are usable in two primary use cases:

- Literal types are eligible to be created and destroyed during the evaluation of a *constant expression*.
- Literal types are suitable to be used in the *interface* of a **constexpr** function, either as the return type or as a parameter type.

The criteria for determining whether a given type is a literal type can be divided into six parts:

1. Every scalar type is a literal type. Scalar types include all fundamental arithmetic (integral and floating point) types, all enumerated types, and all pointer types.

int	int is a <i>literal type</i> .
double	double is a <i>literal type</i> .
short*	<b>short</b> * is a <i>literal type</i> .
<b>enum</b> E { e_A };	E is a <i>literal type</i> .
Τ*	$T^*$ is a literal type (for any T).

Note that a pointer  $T^*$  is *always* a literal type, even when it points to a type T that itself is *not* a literal type.