Representation of Pointers and Nullptr_t

Document: n3563

Author: Martin Uecker

Date: 2025-05-25

This paper proposes a non-normative change to move all rules about representation of pointer types and of nullptr_t that are now scattered across the standard into a new section "6.2.6.4 Pointer types and nullptr_t" under 6.2.6. "Representations of types".

It is further proposed to move the remaining semantic properties of nullptr_t from the library section to "6-2.5 Types and"6.7.11 Initialization".

Overall, I believe this makes the standard easier to follow and improves separation of the library from core language semantics.

Wording (n3550)

6.2.5 Types

25 Any number of derived types …

26 The type of the nullptr constant, i.e. nullptr_t, is an unqualified complete scalar type that is different from all pointer or arithmetic types and is neither an atomic or array type and has exactly one value, nullptr.

~~33 A pointer to void shall have the same representation and alignment requirements as a pointer to a character type.41) Similarly, pointers to qualified or unqualified versions of compatible types shall have the same representation and alignment requirements. All pointers to structure types shall have the same representation and alignment requirements as each other. All pointers to union types shall have the same representation and alignment requirements as each other. Pointers to other types may not have the same representation or alignment requirements~~

6.2.6.2 Integer types

6.2.6.3 Pointer types and nullptr_t

1 A pointer to void shall have the same representation and alignment requirements as a pointer to a character type.41) Similarly, pointers to qualified or unqualified versions of compatible types shall have the same representation and alignment requirements. All pointers to structure types shall have the same representation and alignment requirements as each other. All pointers to union types shall have the same representation and alignment requirements as each other. Pointers to other types may not have the same representation or alignment requirements.

2 The size and alignment of nullptr_t is the same as for a pointer to character type. An object representation of the value nullptr is

the same as the object representation of a null pointer value of type void*. All other object representations are non-value representations for this type.

6.7.11 Initialization

14 If an object that has automatic storage duration is not initialized explicitly, its representation is indeterminate. If an object that has static or thread storage duration is not initialized explicitly, or any object is initialized with an empty initializer, then it is subject to default initialization, which initializes an object as follows:

— if it has pointer type, it is initialized to a null pointer

– **if it has type nullptr_t, it is initialized to nullptr.**

– ...

7.22.3 The nullptr_t type

2 The nullptr_t type is the type of the nullptr predefined constant. It has only a very limited use in contexts where this type is needed to distinguish nullptr from other expression types. ~~It is an unqualified complete scalar type that is different from all pointer or arithmetic types and is neither an atomic or array type and has exactly one value, nullptr. Default or empty initialization of an object of this type is equivalent to an initialization by nullptr.~~

~~3 The size and alignment of nullptr_t is the same as for a pointer to character type. An object representation of the value nullptr is the same as the object representation of a null pointer value of type void*. An lvalue conversion of an object of type nullptr_t with such an object representation has the value nullptr; if the object representation is different, the behavior is undefined.~~