How do you add one to something?
WG14 N3323

Title:	How do you add one to something?
Author, affiliation:	Aaron Ballman, Intel
Date:	2024-08-28
Proposal category:	Bug fixes
Target audience:	WG14 members, C implementers

Abstract: Clarifies what “appropriate type” means for the ++ and −− operators.

How do you add one to something?

Reply-to: Aaron Ballman (aaron@aaronballman.com)
Document No: N3323
Revises Document No: N3297
Date: 2024-08-28

Summary of Changes

N3323

- Updated the table of adjustments in the prose
- Slight tweak to clarify the proposed wording
- Changed pointer arithmetic to use `ptrdiff_t` instead of `int`

N3297

- Initial version

Introduction and Rationale

During discussion of WG14 N3259, which allowed ++ and -- to be used on complex types, the committee observed that "the value 1 of the appropriate type" is ambiguous. Consider an example like:

```
unsigned _BitInt(12) bi = 0;
bi++;
```

Is `1` of type `int`? `_BitInt(1)`? `unsigned _BitInt(12)`? Any of these answers is at least somewhat defensible and the standard is unclear on what we want the answer to be.

Generally, we want the type for `1` to be the same type as the type of the operand. However, special provisions should exist for:

Type	**Expression to yield the correct type for `1`**
`unsigned _BitInt(N)`	`1wbu`
`signed _BitInt(N)`	`1wb`
`_Complex` <type>	`(`<type>`)1`
`_DecimalN`	`(_DecimalN)1`
Other	`1`

Proposed Wording

The wording proposed is a diff from the WG14 N3301 working draft of ISO/IEC 9899. Green text is new text, while ~~red~~ text is deleted text.

Add a new paragraph before the existing 6.5.3.5p2:
The *adjustment* used to increment or decrement the operand is the value `1` with a type and value

representation as follows: if the operand has a pointer type, the adjustment has type `ptrdiff_t`; if the operand has complex type, the adjustment has the corresponding real type of the operand; if the operand has decimal floating type, the adjustment has the same type as the operand with a quantum exponent of `0`; otherwise, the adjustment has the same type as the operand.

Modify the existing 6.5.3.5p2:
The result of the postfix ++ operator is the value of the operand. As a side effect, the value of the operand object is incremented by the adjustment ~~(that is, the value 1 of the appropriate type is added to it)~~. …

Modify the existing 6.5.3.5p3:
The postfix −− operator is analogous to the postfix ++ operator, except that the value of the operand is decremented by the adjustment ~~(that is, the value 1 of the appropriate type is subtracted from it)~~.

Modify 6.5.4.2p2:
The value of the operand of the prefix ++ operator is incremented. The result is the new value of the operand after incrementation. The expression `++E` is similar ~~equivalent~~ to `(E+=1)`, except ~~where~~ the value 1 is the adjustment (6.5.3.5) ~~of the appropriate type~~. …

Acknowledgements

I would like to recognize the following people for their help in this work: Joseph Myers, Jens Gustedt, and Alejandro Colomar.