ISO/IEC JTC 1/SC 22/WG14
WG 21, SG 22

N2818 v2
P2378R1

2021-9-27

Properly define blocks as part of the grammar
proposal for C23

Jens Gustedt[1] and Martin Uecker[2]

[1] INRIA and ICube, Université de Strasbourg, France
[2] University Medical Center Göttingen, Germany

History: Version 1 of this paper was voted 12-1-8 in favor by WG14. It also asked for the clarification about the status of some rarely occuring compound literals; that material has been moved to N2819.
Changes in v2:

— Remove references to lambdas.
— Expand changed syntax sections for readability.
— Clarify the lexical range of the block to consist of three parts, the parameter type list, a possible following attribute specifier sequence and then the function body.

1. INTRODUCTION

Blocks are a fundamental concept in C for the definition of visibility scopes of identifiers and for the lifetime of objects. Currently, there is no closed definition what a block is and the different definitions that compose the term have to be collected in different places that spread over several clauses. In particular, the fact that dependent statements of iteration or selection statements form blocks of their own is easily overlooked and leads to misunderstandings for example concerning the lifetime of compound literals.
We propose to change that situation by introducing terms *primary block* and *secondary block* in the syntax and by referring to the other definitions of blocks, namely functions definitions and lambda expressions (if added to C23), in a summary definition.

2. AMBIGUITIES

The current text is not completely clear about the scope of function parameters in function definitions. Whereas it seems clear from examples that previous parameters can already be used as size expression in a VLA parameter, it is less clear whether it is visible in the attribute specifier list that follows the parameter type list.
Current practice and expectation seems to be to make the scope of visibility a lexical scope, that is have it stretch from the parameter declaration to the end of the function body. We propose to make this explicit. This is also consistent with the reach of *function prototype scope* where the scope also stretches to the end of the whole declarator, thus including possible attributes that follow the parameter type list.

3. IMPACT

The proposed changes are meant for clarification and should have no impact on the validity of user code and no implementation would have to change.

4. PROPOSED CHANGES

The following two are the principal changes proposed to the grammar.

CHANGE 1. *Replace the content of clause* `6.8 p1` *by*

statement:
  labeled-statement
  expression-statement
  attribute-specifier-sequence$_{opt}$ primary-block
  attribute-specifier-sequence$_{opt}$ jump-statement

primary-block:
  compound-statement
  selection-statement
  iteration-statement

secondary-block:
  statement

CHANGE 2. *Change the start of* `6.8 p3` *as follows*

A block is either a primary block, a secondary block, or the block associated with a function definition; it *allows a set of declarations and statements to be grouped into one syntactic unit.* Whenever a *block B* appears in the syntax production as part of the definition of an enclosing block *A*, scopes of identifiers and lifetimes of objects that are associated with *B* do not extend to the parts of *A* that are outside of *B*. *The initializers of objects that have automatic storage duration, ...*

Then, we also need to change the grammar of the three different forms of primary blocks.

CHANGE 3. *In* `6.8.2` *(Compound statement) change* `p2` *as follows:*

A compound statement that is a function body together with the parameter type list and the optional attribute specifier sequence between them forms the block associated with the function definition in which it appears. Otherwise, it *is a block* that is different from any other block.

CHANGE 4. *In* `6.8.4 p1` *and* `p2` *(Selection statements) replace the syntax term* ~~statement~~ *by* secondary-block.

Syntax
1 selection-statement:
  if *(expression)* ~~statement~~secondary-block
  if *(expression)* ~~statement~~secondary-block *else* ~~statement~~secondary-block
  switch *(expression)* ~~statement~~secondary-block

CHANGE 5. *In* `6.8.4 p2` *replace:*

2 A selection statement selects among a set of ~~statements~~secondary blocks *depending on the value of a controlling expression.*

CHANGE 6. *Remove* `6.8.4 p3`.

~~A selection statement is a block whose scope is a strict subset of the scope of its enclosing block. Each associated substatement is also a block whose scope is a strict subset of the scope of the selection statement.~~

CHANGE 7. *In* `6.8.5 p1` *(Iteration statements) replace the syntax term* ~~statement~~ *by* secondary-block*:*

> ***Syntax***
> 1 *iteration-statement:*
> **while** (*expression*) ~~statement~~secondary-block
> **do** ~~statement~~secondary-block **while** (*expression*) ;
> **for** (*expression*$_{opt}$; *expression*$_{opt}$; *expression*$_{opt}$) ~~statement~~secondary-block
> **for** (*declaration expression*$_{opt}$; *expression*$_{opt}$) ~~statement~~secondary-block

CHANGE 8. *Modify* `6.8.4 p4`.

> *An iteration statement causes a* ~~statement~~secondary block *called the loop body to be executed repeatedly until the controlling expression compares equal to 0. The repetition occurs regardless of whether the loop body is entered from the iteration statement or by a jump.*[170)]

CHANGE 9. *Remove* `6.8.4 p5`.

> ~~An iteration statement is a block whose scope is a strict subset of the scope of its enclosing block. The loop body is also a block whose scope is a strict subset of the scope of the iteration statement.~~

And finally, we make the association of the parameter type list and the function body into a single block explicit.

CHANGE 10. *Change* `6.9.1 p9` *(Function definitions) as follows*

> The parameter type list, the attribute specifier sequence of the declarator that follows the parameter type list, and the compound statement of the function body form a single block.[FNT1)] *Each parameter has automatic storage duration; its identifier, if any*[177)]*, is an lvalue.*[178)] *The layout of the storage for parameters is unspecified.*

and add the footnote

> [FNT1)] The visibility scope of a parameter in a function definition starts when its declaration is completed, extends to following parameter declarations, to possible attributes that follow the parameter type list, and then to the entire function body. The lifetime of each instance of a parameter starts when the declaration is evaluated starting a call and ends when that call terminates.

CHANGE 11. *Apply the analogous changes to the language syntax summary in Annex* `A.2.3` *and* `A.2.4`.

5. QUESTION FOR WG14

QUESTION 1. *Shall the indicated changes of N2818 be integrated into C23?*