

The New C Standard (Excerpted material)

An Economic and Cultural Commentary

Derek M. Jones

derek@knosof.co.uk

6.8.2 Compound statement

compound state-
ment
syntax

```

compound-statement:
    { block-item-listopt }
block-item-list:
    block-item
    block-item-list block-item
block-item:
    declaration
    statement

```

Commentary

Compound statements are commonly used to group together a sequence of statements that are required to be executed under the same set of conditions. While compound statements also provide a mechanism to limit the scope of declared identifiers and the lifetime of objects, developers do not often make use of this functionality (see Figure ??).

C90

The C90 syntax required that declarations occur before statements and the two not be intermixed.

```

compound-statement:
    { declaration-listopt statement-listopt }
declaration-list:
    declaration
    declaration-list declaration
statement-list:
    statement
    statement-list statement

```

C++

The C++ Standard uses the C90 syntax. However, the interpretation is the same as the C99 syntax because C++ classifies a declaration as a statement.

Other Languages

Many languages use the keywords **begin/end** instead of a pair of braces. While Fortran, prior to the 1995 standard, did not provide any syntactic mechanism for grouping statements together in blocks. Subroutines were the only explicit statement grouping construct. Developers used conditional and unconditional jumps to create groups of statements.

Some languages (e.g., Pascal) do not allow declarations to occur within nested blocks.

In some languages (e.g., Pascal) the semicolon is a statement separator, not a terminator as in other languages (e.g., Ada). So it need not occur after the last statement in a compound statement. But it may need to occur after the compound statement.

```

1  A := B;
2  begin C := D end;
3  E := F

```

In C the semicolon is part of the syntax of certain statements. The syntax for compound statement does not include a semicolon.

```

1  a = b;
2  { c = d; }
3  e = f;

```

Ada supports the optional occurrence of a token between the **end** and **;**, echoing the corresponding opening header. For instance, `end if;` terminating an **if** statement.

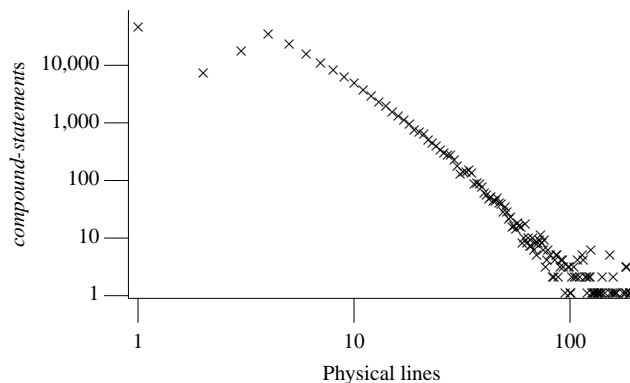


Figure 1729.1: Number of *compound-statements* containing the given number of physical lines (including the opening and closing braces and any nested *compound-statements*, but excluding the lines between the braces denoting the start/end of the function definition). Based on the translated form of this book's benchmark programs.

Common Implementations

A compound statement could be translated into a sequence of machine code instructions making up a basic block, provided it contains no statements or operators that cause conditional execution to occur. For instance, an **if** statement, or the logical-AND operator.

Coding Guidelines

The issues of mixing declarations and statements, and whether to locate declarations at the start of a block or near the point of first use are discussed elsewhere.

Many of the visual layout schemes for source code can be differentiated by where they place opening brace, relative to other tokens. A more realistic analysis, compared to the glowing claims of readability made by the proponents authors of these schemes, is given elsewhere. Some layout schemes locate the open brace after non-white-space characters (e.g., `if (x < y) {`), rather than on a line by itself (and possibly other white-space characters). There are several reasons for believing that the former usage results in a larger number of mistakes being made than the latter. These include: (1) the `{` token not being visible during a visual scan of the left edge of the visible source, and (2) non-white-space characters appear along the visual line connecting the opening and closing brace characters (which require cognitive effort to search and are a possible cause of interference).

Given the lack of experimental measurements of reading performance for different layout schemes, there is no empirical evidence to support any guideline recommendation relating to the visual layout of blocks.

Usage

Usage information on the number of declarations occurring in nested blocks is given elsewhere (see Figure ??).

Semantics

1730 A *compound statement* is a block.

Commentary

This defines the term *compound statement*.

C90

The terms *compound statement* and *block* were interchangeable in the C90 Standard.

Coding Guidelines

The issue associated with enclosing the bodies associated with selection and iteration statements in a pair of matching braces is discussed in the C sentences for those statements. The term most commonly used by

mixing
declarations
and statements

reading
practice

reading
kinds of

compound
statement
is a block

block

developers is *block*, rather than *compound statement*. In C99, but not C90, the term *block* includes constructs that are not compound statements. Authors of coding guideline documents need to ensure that their usage of terminology is consistent.

References