Chapter 3

Program Unit Parts

A *program* is built from one or more program units.

A *program-unit* is one of: R3-1
- *main-program*
- *external-subroutine-subprogram*
- *external-function-subprogram*
- *block-data-program-unit*

A *program unit* (R3-1) may be a main program, an external subprogram, or a block data program unit. An external subprogram may be a function subprogram or a subroutine subprogram. A function subprogram defines a *function* and a subroutine subprogram defines a *subroutine*. Functions and subroutines are collectively called *procedures*. A block data program unit specifies initial values for named common block objects. Block data program units are discussed in Section 9.2, procedures are discussed in Chapter 12, and main programs are discussed in Section 3.5.

A program unit may be divided into smaller parts: the *specification part* specifies information about entities in the program unit, and the *execution part* specifies the computational behavior of the program unit as a series of *executable statement*s. The following sections describe the form of these parts and the ordering relationships among them.

3.1 Executable and Nonexecutable Statements

Program execution is a sequence of computational actions. An *executable statement* (R3-8, page 18) is an instruction to perform or control one or more of these actions.

The major classes of executable statements are

- assignment statements

Copyright © 1989–1994 Thinking Machines Corporation

- input/output statements
- control statements

The *nonexecutable* statements prescribe the environment in which computational actions take place. A nonexecutable statement provides the compiler with information about the arrangement and storage of data, the form and structure of data to be transmitted on input or output, or the kinds of program and subprogram units. The classes of nonexecutable statements are

- `IMPLICIT` statements
- `PARAMETER` statements
- declarations, including type declaration statements, specification statements, and interface blocks
- statement function statements
- `FORMAT` statements
- `ENTRY` statements
- subprogram statements and the `PROGRAM` statement

3.2 Specification Part

The *specification part* (R3-2) of a program unit, if present, specifies information about entities to be used in the execution part.

A *specification-part* is: R3-2
specification-part-stmt
[specification-part-stmt]...

A *specification-part-stmt* is: R3-3
- *implicit-stmt*
- *parameter-stmt*
- *declaration*
- *stmt-function-stmt*
- *data-stmt*
- *format-stmt*
- *entry-stmt*

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

A *declaration* is one of: R3-4

- *type-decl-stmt*
- *specification-stmt*
- *interface-block*

- An *implicit-stmt* must precede any *declaration*, *stmt-function-stmt*, or *data-stmt* of a *specification-part*.
- A *parameter-stmt* must precede any *stmt-function-stmt* of a *specification-part*.
- A *declaration* must follow any *implicit-stmt* and must precede any *stmt-function-stmt* of a *specification-part*.
- A *data-stmt* must follow any *implicit-stmt* of a *specification-part*.
- A *stmt-function-stmt* must follow any *declaration* or *parameter-stmt* of a *specification-part*.

A *declaration* (R3-4) is a statement or a block that specifies the attributes of one or more named program entities. This may involve specifying the data type of a named object, or specifying the shape of a named array object.

3.2.1 Specification Statement

A *specification statement* (R3-5) is a form of declaration that introduces new entities with specified attributes, or specifies additional attributes of entities already declared.

A *specification-stmt* is: R3-5

- *common-stmt*
- *data-stmt*
- *dimension-stmt*
- *equivalence-stmt*
- *external-stmt*
- *intent-stmt*
- *intrinsic-stmt*
- *save-stmt*

3.3 Execution Part

The *execution part* (R3-6) of a program unit, if present, begins with the first executable statement of a program unit and extends to its `END` statement.

Copyright © 1989–1994 Thinking Machines Corporation

An *execution-part* is: R3-6

executable-stmt
[execution-part-construct]...

An *execution-part-construct* is one of: R3-7

- *executable-stmt*
- *data-stmt*
- *format-stmt*
- *entry-stmt*

Executable statements may be interspersed with `DATA` statements, `FORMAT` statements, and `ENTRY` statements.

3.3.1 Executable Statements, Constructs, and Action Statements

An *executable statement* (R3-8) is a statement or construct that performs or controls one or more computational actions.

An *executable-stmt* is one of: R3-8

- *action-stmt*
- *executable-construct*

An *executable-construct* is one of: R3-9

- *block-if-construct*
- *case-construct*
- *loop-construct*
- *where-construct*

An executable statement that may contain other executable statements is called an *executable construct* (R3-9).

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

An *action-stmt* is one of: R3-10

- *assign-stmt*
- *assigned-goto-stmt*
- *arithmetic-if-stmt*
- *assignment-stmt*
- *call-stmt*
- *computed-goto-stmt*
- *continue-stmt*
- *cycle-stmt*
- *exit-stmt*
- *end-action-stmt*
- *forall-stmt*
- *goto-stmt*
- *logical-if-stmt*
- *input-output-stmt*
- *pause-stmt*
- *return-stmt*
- *stop-stmt*
- *where-stmt*

An *action-stmt* (R3-10) does not contain any other executable statements.

3.3.2 END Statements

Each program unit and subprogram must have an `END` statement, which may be labeled, and which must be the last statement of the program unit or subprogram. The `END` statements include the *end-action-stmt*s.

An *end-action-stmt* is one of: R3-11

- `END` *[* `PROGRAM` *[* program-*name* *]]*
- `END` *[* `SUBROUTINE` *[* subroutine-*name* *]]*
- `END` *[* `FUNCTION` *[* function-*name* *]]*

An *end-action-stmt* is executable, and may be the target of a branch statement. Executing an `END PROGRAM` statement terminates program execution. Executing an `END FUNCTION` statement or `END SUBROUTINE` statement is equivalent to executing a `RETURN` statement.

Nonexecutable statements beginning with the keyword `END` also include the `END INTERFACE`, `END BLOCK DATA`, `END IF`, `END DO`, `END WHERE`, and `END SELECT` statements. All except `END INTERFACE` and `END BLOCK DATA` may be branch target statements.

Copyright © 1989–1994 Thinking Machines Corporation

3.4 Executable Programs and Compilation Files

An *executable program* is a collection of one or more *compilation file*s (R3-12), each of which contains one or more program units.

A *compilation-file* is: R3-12
program-unit
[program-unit]...

- One and only one of the program units comprising an *executable-program* must be a *main-program* program unit.

The compilation files of an executable program are text files that may be compiled independently, in any order. The object code resulting from the compilations is linked to form an executable CM Fortran program, as described in the *CM Fortran User's Guide*.

3.5 Main Programs: PROGRAM Statement

A *main program* (R3-13) is a program unit that does not begin with a `FUNCTION`, `SUBROUTINE`, or `BLOCK DATA` statement. It may begin with a `PROGRAM` statement.

A *main-program* is: R3-13
[`PROGRAM` program-*name]*
[specification-part]
[execution-part]
`END` *[* `PROGRAM` *[* program-*name]]*

- The *execution-part* of a *main-program* must not contain a `RETURN` statement or an `ENTRY` statement.

- The program-*name* may be included in the `END` statement only if the optional `PROGRAM` keyword is used and, if included, the name must be identical to the program-*name* specified in the `PROGRAM` statement.

A `PROGRAM` statement may appear only as the first statement of a main program. The program-*name* is global to the program, and must not be the same as the name of any other program unit or common block in the executable program, nor the same as any local name in the main program. The name of a program is not used in CM Fortran.

The specification part of a main program may declare only explicit-shape arrays with constant bounds (no automatic arrays may be declared).

Version 2.2, October 1994
Copyright © 1989–1994 Thinking Machines Corporation

The sequence of statements in the *execution-part* specifies the actions of the main program during program execution, which begins with the first executable statement of the program and proceeds as described in Chapter 11. Execution of a main program ends with execution of the `END PROGRAM` statement of the main program, or with the execution of a `STOP` statement in any program unit.

An example of a main program:

```
PROGRAM CUBIT
PARAMETER ( CUBESIZE = 200 )
REAL C(CUBESIZE,CUBESIZE,CUBESIZE)  ! specification part
CALL READ( C )                      ! execution part
CALL PROCESS( C )
CALL DISPLAY( C )
END PROGRAM CUBIT
```

3.5.1 Order of Statements in a Program or Subprogram

The syntax rules given in this chapter specifying the order of statements within a main program or subprogram are illustrated in Figure 3. In the figure, vertical lines delineate kinds of statements that may be interspersed and horizontal lines delineate kinds of statements that must not be interspersed.

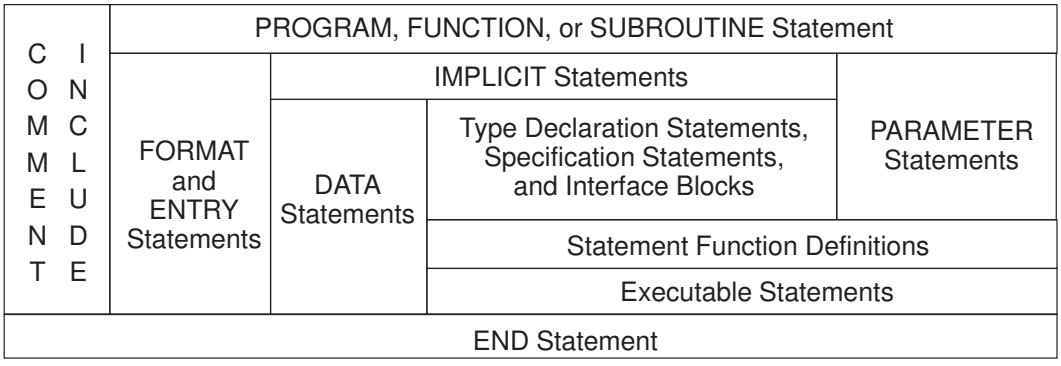

Figure 3. Required order of statements in a main program unit or external subprogram.

- `ENTRY` statements may not appear in a main program unit.

A similar figure is given in Section 9.2.1 for block data program units, and in Section 12.6.1.1 for interface blocks.

Copyright © 1989–1994 Thinking Machines Corporation