Fortran 2008 Overview

July 31, 2012

1 Introduction

This document describes those parts of the Fortran 2008 language which are not in Fortran 2003, and which are now supported by Release 5.3 of the NAG Fortran Compiler.

2 Overview of Fortran 2008

The new features of Fortran 2008 that are supported by the NAG Fortran Compiler can be grouped as follows:

- the `BLOCK` construct,
- additional intrinsic functions for bit manipulation,
- `ISO_FORTRAN_ENV` additions,
- miscellaneous and convenience features.

3 The `BLOCK` construct

This construct allows declarations of entities within executable code. For example,

```
Do i=1,n
  Block
    Real tmp
    tmp = a(i)**3
    If (tmp>b(i)) b(i) = tmp
  End Block
End Do
```

Here the variable `tmp` has its scope limited to the `BLOCK` construct, so will not affect anything outside it. This is particularly useful when including code by `INCLUDE` or by macro preprocessing.

All declarations are allowed within a `BLOCK` construct except for `COMMON`, `EQUIVALENCE`, `IMPLICIT`, `INTENT`, `NAMELIST`, `OPTIONAL` and `VALUE`; also, statement function definitions are not permitted.

`BLOCK` constructs may be nested; like other constructs, branches into a `BLOCK` construct from outside are not permitted. A branch out of a `BLOCK` construct "completes" execution of the construct.

Entities within a `BLOCK` construct that do not have the `SAVE` attribute (including implicitly via initialisation), will cease to exist when execution of the construct is completed. For example, an allocated `ALLOCATABLE` variable will be automatically deallocated, and a variable with a `FINAL` procedure will be finalised.

4 Additional intrinsic functions for bit manipulation

- The elemental intrinsic functions `BGE`, `BGT`, `BLE` and `BLT` perform bitwise (i.e. unsigned) comparisons. They each have two arguments, `I` and `J`, which must be of type Integer but may be of different kind. The result is default Logical.

  For example, `BGE(INT(Z'FF',INT8),128)` is true, while `INT(Z'FF',INT8)>=128` is false.

- The array reduction intrinsic functions `IALL`, `IANY` and `IPARITY` reduce arrays using bitwise operations. These are exactly the same as `SUM` and `PRODUCT`, except that instead of reducing the array by the `+` or `*` operation, they reduce it by the `IAND`, `IOR` and `IEOR` intrinsic functions respectively. That it, each element of the result is the bitwise-and, bitwise-or, or bitwise-exclusive-or of the reduced elements. If the number of reduced elements is zero, the result is zero for `IANY` and `IPARITY`, and `NOT(`*zero*`)` for `IALL`.
- The elemental intrinsic functions `LEADZ` and `TRAILZ` return the number of leading (most significant) and trailing (least significant) zero bits in the argument `I`, which must be of type Integer (of any kind). The result is default Integer.
- The elemental intrinsic functions `MASKL` and `MASKR` generate simple left-justified and right-justified bitmasks. The value of `MASKL(I,KIND)` is an integer with the specified kind that has its leftmost `I` bits set to one and the rest set to zero; `I` must be non-negative and less than or equal to the bitsize of the result. If `KIND` is omitted, the result is default integer. The value of `MASKR` is similar, but has its rightmost `I` bits set to one instead.
- The array reduction intrinsic function `PARITY` reduces Logical arrays. It is exactly the same as `ALL` and `ANY`, except that instead of reducing the array by the `.AND.` or `.OR.` operation, it reduces it by the `.NEQV.` operation. That is, each element of the result is `.TRUE.` if an odd number of reduced elements is `.TRUE.`.
- The elemental intrinsic function `POPCNT(I)` returns the number of bits in the Integer argument `I` that are set to 1. The elemental intrinsic function `POPPAR(I)` returns zero if the number of bits in `I` that are set to 1 are even, and one if it is odd. The result is default Integer.

5 `ISO_FORTRAN_ENV` additions

The intrinsic module `ISO_FORTRAN_ENV` contains additional named constants as follows.

- The additional scalar integer constants `INT8`, `INT16`, `INT32`, `INT64`, `REAL32`, `REAL64` and `REAL128` supply the kind type parameter values for integer and real kinds with the indicated bit sizes.
- The additional named array constants `CHARACTER_KINDS`, `INTEGER_KINDS`, `LOGICAL_KINDS` and `REAL_KINDS` list the available kind type parameter values for each type (in no particular order).

6 Miscellaneous and convenience features

- In a structure constructor, the value for an allocatable component may be omitted: this has the same effect as specifying `NULL()`.
- In a `STOP` statement, the *stop-code* may be any scalar constant expression of type integer or default character. (In the NAG Fortran Compiler this also applies to the `PAUSE` statement, but that statement is no longer standard Fortran.)
- `ENTRY` statements are regarded as obsolescent.
- An empty internal subprogram part, module subprogram part or type-bound procedure part is now permitted following a `CONTAINS` statement. In the case of the type-bound procedure part, an ineffectual `PRIVATE` statement may appear following the unnecessary `CONTAINS` statement.
- The `FUNCTION` and `SUBROUTINE` keywords on the END statement for an internal or module procedure is now optional (when the procedure name does not appear).
- A type-bound procedure declaration statement may now declare multiple type-bound procedures. For example, instead of

```
PROCEDURE,NOPASS :: a
PROCEDURE,NOPASS :: b=>x
PROCEDURE,NOPASS :: c
```

  the single statement

```
PROCEDURE,NOPASS :: a, b=>x, c
```

will suffice.

- The `NEWUNIT=` specifier has been added to the `OPEN` statement; this allocates a new unit number that cannot clash with any other logical unit (the value will be a special negative value). For example,

```
INTEGER unit
OPEN(FILE='output.log',FORM='FORMATTED',NEWUNIT=unit)
WRITE(unit,*) 'Logfile opened.'
```

  The `NEWUNIT=` specifier can only be used if either the `FILE=` specifier is also used, or if the `STATUS=` specifier is used with the value `'SCRATCH'`.

- Fortran 2008 extends the rules that are used for generic resolution and for checking that procedures in a generic are unambiguous. The new rules are that
  - a dummy procedure is distinguishable from a dummy variable;
  - an `ALLOCATABLE` dummy variable is distinguishable from a `POINTER` dummy variable that does not have `INTENT(IN)`.

7 References

The Fortran 2008 standard, IS 1539-1:2010(E), is available from ISO as well as from many national standards bodies. A number of books describing the new standard are available; the recommended reference book is "Modern Fortran Explained" by Metcalf, Reid & Cohen, Oxford University Press, 2011 (ISBN 978-0-19-960141-7).