BNF Syntax of the F™ Programming Language

F is maintained by The Fortran Company. If you are interested in supplying F tools, software, or related products, contact info@fortran.com.

This is a working draft and is subject to change

Version: 2003 April 22

NOTES

An F program is interpreted in the same way as a Fortran 95 program.

An F program is subject to all of the appropriate syntax constraints of Fortran 95, plus those listed here.

Additional source form rule:

- There is no ";" delimiter used to separate statements on a line.

The character data type has only default kind. There is no way to specify a kind parameter for character data.

In an OPEN statement the status= keyword is required and shall not be "UNKNOWN".

In an OPEN statement the action= keyword is required. If status is SCRATCH action must be READWRITE. If status is NEW or REPLACE action must not be READ.

In an OPEN statement for sequential access with status OLD the POSITION= keyword is required. The position must be REWIND or APPEND.

If a unit is connected to a file, it must be closed before a subsequent OPEN of the unit. There is no reopen of a connected file.

The F programming language does not directly support the concept of carriage control. This is a file/OS/output-device characteristic. Files intended to be "printed" on certain devices will have to be written in a certain way.

No characters, other than blanks, shall appear after the final ")" of a format.

The edit descriptors ES, EN, TL, TR, SS, and SP shall not have embedded blanks.

Statement keywords, etc., are usually printed in upper case to distinguish them from the surrounding text.

When a name is used in an F program, the case of the letters shall match that of the name when it is declared or defined.

Certain processors may produce an E (or D or d) on output of real numbers and may also accept that form for input. This is provided for compatibility with existing data files produced by nonF programs.

Character values returned from an INQUIRE statement will be in upper case.

```
R201      program
      is      program-unit
              [ program-unit ] ...

R202      program-unit
      is      main-program
      or      module

R1101      main-program
      is      program-stmt
                [ use-stmt ] ...
                [ specification-part ]
                [ execution-part ]
                [ subprogram-part ]
              end-program-stmt

R204      specification-part
      is      [ declaration-construct ] ...

R207      declaration-construct
      is      interface-block
      or      derived-type-def
      or      type-declaration-stmt
      or      intrinsic-stmt
      or      access-stmt
      or      optional-stmt

Constraint:      An access-stmt shall not appear
in a main-program or the subprogram-part of a module.

R208      execution-part
      is      [ executable-construct ] ...

R210      subprogram-part
      is     CONTAINS
               subprogram
               [ subprogram ] ...

R211      subprogram
      is     function-subprogram
      or     subroutine-subprogram

R215      executable-construct
      is      action-stmt
      or      case-construct
      or      do-construct
```

```
      or      if-construct
      or      where-construct
      or      forall-construct

R216      action-stmt
      is      allocate-stmt
      or      assignment-stmt
      or      backspace-stmt
      or      call-stmt
      or      close-stmt
      or      continue-stmt
      or      cycle-stmt
      or      deallocate-stmt
      or      endfile-stmt
      or      exit-stmt
      or      forall-stmt
      or      goto-stmt
      or      if-stmt
      or      inquire-stmt
      or      open-stmt
      or      pointer-assignment-stmt
      or      print-stmt
      or      read-stmt
      or      return-stmt
      or      rewind-stmt
      or      stop-stmt
      or      where-stmt
      or      write-stmt

Constraint:      The target of a go to statement shall
be a continue statement that occurs after the go to
statement in the same scoping unit.

R301      character
      is      alphanumeric-character
      or      special-character

R302      alphanumeric-character
      is      letter
      or      digit
      or      underscore

R303      underscore
      is      _

R304      name
      is      letter [ alphanumeric-character ] ...

Constraint:      The maximum length of a name is 31 characters.

Constraint:      All variables must be declared in type statements or
accessed by use or host association.

Constraint:      Entity names, type names, defined operator names, argument
keywords for non-intrinsic procedures, and non-intrinsic procedure names may
be in mixed upper and lower case; however, all occurrences of the names shall
use the same case convention.

Constraint:      All non-intrinsic procedures shall have an explicit interface.

Constraint:      Blank characters shall not appear within any name, keyword,
```

operator, delimiter, or literal-constant except that one or more blank characters may appear before or after the real-part or imag-part of a complex-literal-constant and one or more blanks may be used in keywords as follows:

keyword	alternate usage
elseif	else if
enddo	end do
endfile	end file
endfunction	end function
endif	end if
endinterface	end interface
endmodule	end module
endprogram	end program
endselect	end select
endsubroutine	end subroutine
endtype	end type
endwhere	end where
inout	in out
selectcase	select case

Constraint: No name, keyword, delimiter, or operator, shall be split onto more than one line via statement continuation. Keywords shall not be continued at the optional blank.

Constraint: No line shall begin with the & character.

R305 constant
is literal-constant
or named-constant

R306 literal-constant
is int-literal-constant
or real-literal-constant
or complex-literal-constant
or logical-literal-constant
or char-literal-constant

R307 named-constant
is name

R308 int-constant
is constant

Constraint: int-constant shall be of type integer.

R309 char-constant
is constant

Constraint: char-constant shall be of type character.

R310 intrinsic-operator
is power-op
or mult-op
or add-op
or concat-op
or rel-op
or not-op
or and-op
or or-op

```
      or      equiv-op

R311          defined-operator
      is      defined-unary-op
      or      defined-binary-op
      or      extended-intrinsic-op

R312          extended-intrinsic-op
      is      intrinsic-operator

Constraint:      A defined-unary-op and a defined-binary-op shall not
contain more than 31 letters and shall not be the same as any
intrinsic-operator or logical-literal-constant.

R313      label
     is   digit [ digit [ digit [ digit [ digit ] ] ] ]

Constraint:      At least one digit in a label shall be nonzero.

Constraint:      The only statement that may have a label is
the continue-stmt.

R401      signed-digit-string
      is      [ sign ] digit-string

R402      digit-string
      is      digit [ digit ] ...

R403      signed-int-literal-constant
      is      [ sign ] int-literal-constant

R404      int-literal-constant
      is      digit-string [ _ kind-param ]

R405      kind-param
      is      scalar-int-constant-name

R406      sign
      is      +
      or      -

Constraint:      The value of kind-param shall be nonnegative.

Constraint:      The value of kind-param shall specify a representation
method that exists on the processor.

R412      signed-real-literal-constant
      is      [ sign ] real-literal-constant

R413      real-literal-constant
      is      significand [ exponent-letter exponent ] [ _ kind-param ]

R414      significand
      is      digit-string .  digit-string

R415      exponent-letter
      is      E

R416      exponent
      is      signed-digit-string
```

Constraint: The value of kind-param shall specify an approximation method that exists on the processor.

```
R417      complex-literal-constant
      is      ( real-part , imag-part )

R418      real-part
      is      signed-real-literal-constant

R419      imag-part
      is      signed-real-literal-constant
```

Constraint: Both real-part and imag-part must either have no kind-param or have the same kind-param.

```
R420        char-literal-constant
      is      " [ rep-char ] ...  "
```

Note: Within a char-literal-constant the quote delimiter may be doubled to indicate a single instance of the delimiter.

```
R421      logical-literal-constant
      is      .TRUE.  [ _ kind-param ]
      or      .FALSE.  [ _ kind-param ]
```

Constraint: The value of kind-param shall specify a representation method that exists on the processor.

Constraint: No integer, real, logical, or character literal constant, or real-part or imag-part shall be split onto more than one line via statement continuation.

```
R422      derived-type-def
      is      derived-type-stmt
                [ private-stmt ]
                component-def-stmt
                [ component-def-stmt ] ...
              end-type-stmt

R423      derived-type-stmt
      is      TYPE [ , access-spec ] :: type-name
```

Constraint: The access-spec shall be present if the derived-type-stmt is in a module and shall not be present if it is in a main-program.

Constraint: A derived type type-name shall not be the same as the name of any intrinsic type defined in the Fortran standard nor the same as any other accessible derived type type-name.

```
R424      private-stmt
      is      PRIVATE

R425      component-def-stmt
      is      type-spec [ , component-attr-spec-list ] :: component-decl-list
```

Constraint: The character length specified by the char-length in a type-spec shall be a constant specification expression.

```
R426      component-attr-spec
      is      POINTER
      or      DIMENSION ( component-array-spec )
```

Note: F implementations allow ALLOCATABLE as an extension.

R427 component-array-spec
is explicit-shape-spec-list
or deferred-shape-spec-list

Constraint: If a component of a derived-type is of a type that is private, either the derived type definition shall contain the PRIVATE statement or the derived type shall be private.

Constraint: No component-attr-spec shall appear more than once in a given component-def-stmt.

Constraint: If the POINTER attribute is not specified for a component, a type-spec in the component-def-stmt shall specify an intrinsic type or a previously defined derived type.

Constraint: If the POINTER attribute is specified for a component, a type-spec in the component-def-stmt shall specify an intrinsic type or any accessible derived type including the type being defined.

Constraint: If the POINTER attribute is specified, each component-array-spec shall be a deferred-shape-spec-list.

Constraint: If the POINTER attribute is not specified, each component-array-spec shall be an explicit-shape-spec-list.

Constraint: Each bound in the explicit-shape-spec shall be a constant specification expression.

R428 component-decl
is component-name [initialization]

R430 end-type-stmt
is END TYPE type-name

Constraint: The type-name shall be the same as that in the corresponding derived-type-stmt.

R431 structure-constructor
is type-name (expr-list)

R432 array-constructor
is (/ ac-value-list /)

R433 ac-value
is expr
or ac-implied-do

R434 ac-implied-do
is (ac-value-list , implied-do-control)

R435 implied-do-control
is do-variable = scalar-int-expr , scalar-int-expr &
[, scalar-int-expr]

Constraint: Each ac-value expression in the array-constructor shall have the same type and kind type parameter.

R501 type-declaration-stmt

```
      is      type-spec [ , attr-spec ] ... :: entity-decl-list

R502      type-spec
      is      INTEGER [ kind-selector ]
      or      REAL [ kind-selector ]
      or      CHARACTER char-selector
      or      COMPLEX [ kind-selector ]
      or      LOGICAL [ kind-selector ]
      or      TYPE ( type-name )

R503      attr-spec
      is      PARAMETER
      or      access-spec
      or      ALLOCATABLE
      or      DIMENSION ( array-spec )
      or      INTENT ( intent-spec )
      or      OPTIONAL
      or      POINTER
      or      SAVE
      or      TARGET

R504      entity-decl
      is      object-name [ initialization ]

R505      initialization
      is      = initialization-expr
      or      => NULL()

R506      kind-selector
      is      ( KIND = scalar-int-constant-name )

Constraint:      The same attr-spec shall not appear more than once in a
given type-declaration-stmt.

Constraint:      The ALLOCATABLE attribute may be used only when declaring
an array that is not a dummy argument or a function result.
[ Note:  F extends this to allow dummy arguments and function results. ]

Constraint:      An array declared with a POINTER or an ALLOCATABLE
attribute shall be specified with an array-spec that is a
deferred-shape-spec-list.

Constraint:      An array-spec for an object-name that is a function result
that does not have the POINTER attribute shall be an explicit-shape-spec-list.

Constraint:      If the POINTER attribute is specified, neither the TARGET
nor INTENT attribute shall be specified.

Constraint:      If the TARGET attribute is specified, neither the POINTER
nor PARAMETER attribute shall be specified.

Constraint:      The PARAMETER attribute shall not be specified for dummy
arguments, pointers, allocatable arrays, or functions results.

Constraint:      The INTENT and OPTIONAL attributes may be specified only
for dummy arguments.  INTENT must be specified for every dummy argument
except a procedure or one with the POINTER attribute.

Constraint:      An entity shall not have the PUBLIC attribute if its type
has the PRIVATE attribute.
```

Constraint: The SAVE attribute shall not be specified for an object that is a dummy argument, a procedure, a function result, an automatic data object, or an object with the PARAMETER attribute.

Constraint: An array shall not have both the ALLOCATABLE attribute and the POINTER attribute.

Constraint: Initialization shall appear if the statement contains a PARAMETER attribute.

Constraint: Initialization shall not appear if object-name is a dummy argument, a function result, an allocatable array, or an automatic object.

Constraint: The value of scalar-int-constant-name in kind-selector shall be nonnegative and shall specify a representation method that exists on the processor.

R507 char-selector
is (LEN = char-len-param-value)

R510 char-len-param-value
is specification-expr
or *

Constraint: The char-len-param-value must be * for a dummy argument or a parameter.

Constraint: The char-len-param-value may be * only for a dummy argument or a parameter.

R511 access-spec
is PUBLIC
or PRIVATE

Constraint: An access-spec shall appear only in the specification-part of a module.

Constraint: An access-spec shall appear on every type-declaration-statement in a module.

R512 intent-spec
is IN
or OUT
or INOUT

Constraint: The INTENT attribute shall not be specified for a dummy argument that is a dummy procedure or a dummy pointer.

Constraint: A dummy argument with the INTENT(IN) attribute, or a subobject of such a dummy argument, shall not appear as

(1) The variable of an assignment-stmt,

(2) The pointer-object of a pointer-assignment-stmt,

(3) A DO variable (no dummy argument may be a DO variable),

(4) An input-item in a read-stmt,

(5) An internal-file-unit in a write-stmt,

(6) An IOSTAT= or SIZE= specifier in an input/output statement,

(7) A definable variable in an INQUIRE statement,

(9) A stat-variable or allocate-object in an allocate-stmt or a deallocate-stmt, or

(10) An actual argument in a reference to a procedure when the associated dummy argument has the INTENT(OUT) or INTENT(INOUT) attribute.

R513 array-spec
  is explicit-shape-spec-list
  or assumed-shape-spec-list
  or deferred-shape-spec-list

Constraint: The maximum rank is seven.

R514 explicit-shape-spec
  is [lower-bound :] upper-bound

R515 lower-bound
  is specification-expr

R516 upper-bound
  is specification-expr

Constraint: An explicit-shape array whose bounds depend on the values of nonconstant expressions shall be a function result, or an automatic array of a procedure.

R517 assumed-shape-spec
  is [lower-bound] :

Constraint: All dummy argument arrays shall be assumed-shape-arrays.

Constraint: Only dummy argument arrays shall be assumed-shape arrays.

R518 deferred-shape-spec
  is :

R521 optional-stmt
  is OPTIONAL :: dummy-arg-name-list

Constraint: Each optional-stmt shall occur only in the specification-part of a subprogram or an interface body (12.3.2.1).

Constraint: Each dummy-arg-name shall be the name of a procedure that is a dummy argument of the procedure in which the statement appears.

R522 access-stmt
  is access-spec :: access-id-list

R523 access-id
  is procedure-name
  or generic-spec

Constraint: Each procedure-name shall be the name of a procedure defined in the module.

Constraint: Each generic-spec or procedure defined in a module shall appear in an access-stmt in the module.

Constraint: A module procedure that has a dummy argument or function result of a type that has PRIVATE accessibility shall have PRIVATE accessibility and shall not have a generic identifier that has PUBLIC accessibility.

R601 variable
is scalar-variable-name
or array-variable-name
or subobject

Constraint: array-variable-name shall be the name of a data object that is an array.

Constraint: array-variable-name shall not have the PARAMETER attribute.

Constraint: scalar-variable-name shall not have the PARAMETER attribute.

Constraint: subobject shall not be a subobject designator (for example, a substring) whose parent is a constant.

R602 subobject
is array-element
or array-section
or structure-component
or substring

R603 logical-variable
is variable

Constraint: logical-variable shall be of type logical.

R604 default-logical-variable
is variable

Constraint: default-logical-variable shall be of type default logical.

R605 char-variable
is variable

Constraint: char-variable shall be of type character.

R607 int-variable
is variable

Constraint: int-variable shall be of type integer.

R608 default-int-variable
is variable

Constraint: default-int-variable shall be of type default integer.

R609 substring
is parent-string (substring-range)

R610 parent-string
is scalar-variable-name
or array-element
or scalar-structure-component

R611 substring-range

```
      is      [ scalar-int-expr ] : [ scalar-int-expr ]

Constraint:      parent-string shall be of type character.

R612      data-ref
      is      part-ref [ % part-ref ] ...

R613      part-ref
      is      part-name [ ( section-subscript-list ) ]

Constraint:      In a data-ref, each part-name except the rightmost shall be
of derived type.

Constraint:      In a data-ref, each part-name except the leftmost shall be
the name of a component of the derived type definition of the type of the
preceding part-name.

Constraint:      In a part-ref containing a section-subscript-list, the
number of section-subscripts shall equal the rank of part-name.

Constraint:      In a data-ref, there shall not be more than one part-ref
with nonzero rank.  A part-name to the right of a part-ref with nonzero rank
shall not have the POINTER attribute.

R614      structure-component
      is      data-ref

Constraint:      In a structure-component, there shall be more than one
part-ref and the rightmost part-ref shall be of the form part-name.

R615      array-element
      is      data-ref

Constraint:      In an array-element, every part-ref shall have rank zero
and the last part-ref shall contain a subscript-list.

R616      array-section
      is      data-ref [ ( substring-range ) ]

Constraint:      In an array-section, exactly one part-ref shall have
nonzero rank, and either the final part-ref shall have a
section-subscript-list with nonzero rank or another part-ref shall have
nonzero rank.

Constraint:      In an array-section with a substring-range, the rightmost
part-name shall be of type character.

R617      subscript
      is      scalar-int-expr

R618      section-subscript
      is      subscript
      or      subscript-triplet
      or      vector-subscript

R619      subscript-triplet
      is      [ subscript ] : [ subscript ] [ : stride ]

R620      stride
      is      scalar-int-expr
```

R621 vector-subscript
is int-expr

Constraint: A vector-subscript shall be an integer array expression of rank one.

R622 allocate-stmt
is ALLOCATE (allocation-list [, STAT = stat-variable])

R623 stat-variable
is scalar-int-variable

R624 allocation
is allocate-object [(allocate-shape-spec-list)]

R625 allocate-object
is variable-name
or structure-component

R626 allocate-shape-spec
is [allocate-lower-bound :] allocate-upper-bound

R627 allocate-lower-bound
is scalar-int-expr

R628 allocate-upper-bound
is scalar-int-expr

Constraint: Each allocate-object shall be a pointer or an allocatable array.

Constraint: The number of allocate-shape-specs in an allocate-shape-spec-list shall be the same as the rank of the pointer or allocatable array.

R630 pointer-object
is variable-name
or structure-component

Constraint: Each pointer-object shall have the POINTER attribute.

R631 deallocate-stmt
is DEALLOCATE (allocate-object-list [, STAT = stat-variable])

Constraint: Each allocate-object shall be a pointer or allocatable array.

R701 primary
is constant
or constant-subobject
or variable
or array-constructor
or structure-constructor
or function-reference
or (expr)

R702 constant-subobject
is subobject

Constraint: subobject shall be a subobject designator whose parent is a named-constant.

```
R703      level-1-expr
      is      [ defined-unary-op ] primary

R704      defined-unary-op
      is      . letter [ letter ] ...

Constraint:      A defined-unary-op shall not contain more than 31 letters.

R705      mult-operand
      is      level-1-expr [ power-op mult-operand ]

R706      add-operand
      is      [ add-operand mult-op ] mult-operand

R707      level-2-expr
      is      [ [ level-2-expr ] add-op ] add-operand

R708      power-op
      is      **

R709      mult-op
      is      *
      or      /

R710      add-op
      is      +
      or      -

R711      level-3-expr
      is      [ level-3-expr concat-op ] level-2-expr

R712      concat-op
      is      //

R713      level-4-expr
      is      [ level-3-expr rel-op ] level-3-expr

R714      rel-op
      is      ==
      or      /=
      or      <
      or      <=
      or      >
      or      >=

R715      and-operand
      is      [ not-op ] level-4-expr

R716      or-operand
      is      [ or-operand and-op ] and-operand

R717      equiv-operand
      is      [ equiv-operand or-op ] or-operand

R718      level-5-expr
      is      [ level-5-expr equiv-op ] equiv-operand

R719      not-op
      is      .NOT.

R720      and-op
```

```
      is      .AND.

R721      or-op
      is      .OR.

R722      equiv-op
      is      .EQV.
      or      .NEQV.

R723      expr
      is      [ expr defined-binary-op ] level-5-expr

R724      defined-binary-op
      is      . letter [ letter ] ...  .

Constraint:      A defined-binary-op shall not contain more than 31 letters.

R725      logical-expr
      is      expr

Constraint:      logical-expr shall be of type logical.

R726      char-expr
      is      expr

Constraint:      char-expr shall of be type character.

R728      int-expr
      is      expr

Constraint:      int-expr shall be of type integer.

R729      numeric-expr
      is      expr

Constraint:      numeric-expr shall be of type integer, real or complex.

R730      initialization-expr
      is      expr

Constraint:      initialization-expr shall be an initialization expression.

R731      char-initialization-expr
      is      char-expr

Constraint:      char-initialization-expr shall be an initialization expression.

R732      int-initialization-expr
      is      int-expr

Constraint:      int-initialization-expr shall be an initialization expression.

R733      logical-initialization-expr
      is      logical-expr

Constraint:      logical-initialization-expr shall be an initialization
expression.

R734      specification-expr
      is      scalar-int-expr
```

```
Constraint:      The scalar-int-expr shall be a restricted expression.

R735      assignment-stmt
      is      variable = expr

R736      pointer-assignment-stmt
      is      pointer-object => target

R737      target
      is      variable
      or      expr

Constraint:      The pointer-object shall have the POINTER attribute.

Constraint:      The variable shall have the TARGET attribute or be a
subobject of an object with the TARGET attribute, or it shall have the
POINTER attribute.

Constraint:      The target shall be of the same type, kind type parameters,
and rank as the pointer.

Constraint:      The target shall not be an array with vector section subscripts

Constraint:      The expr shall deliver a pointer result.

R738      where-stmt
      is     WHERE ( mask-expr ) where-assignment-stmt

R739      where-construct
      is      where-construct-stmt
                [ where-body-construct ] ...
              [ masked-elsewhere-stmt ]
                [ where-body-construct ] ...
              [ elsewhere-stmt ]
                [ where-body-construct ] ...
              end-where-stmt

R740      where-construct-stmt
      is      WHERE ( mask-expr )

R741      where-body-construct
      is      where-assignment-stmt
      or      where-stmt
      or      where-construct

R742      where-assignment-stmt
      is      assignment-stmt

R743      mask-expr
      is      logical-expr

R744      masked-elsewhere-stmt
      is      ELSEWHERE ( mask-expr )

R745      elsewhere-stmt
      is      ELSEWHERE

R746      end-where-stmt
      is      ENDWHERE

Constraint:      In each where-assignment-stmt, the mask-expr and the
```

variable being defined must be arrays of the same shape.

Constraint: A where-assignment-stmt that is a defined assignment shall be elemental.

```
R747    forall-construct
    is    forall-construct-stmt
            [ forall-body-construct ] ...
          end-forall-stmt

R748    forall-construct-stmt
    is    FORALL (forall-triplet-spec-list [ , scalar-mask-expr ] )

R750    forall-triplet-spec
    is    index-name = subscript : subscript [ : stride ]

R751    forall-body-construct
    is    forall-assignment-stmt
    or    where-construct
    or    where-stmt
    or    forall-construct
    or    forall-stmt

R752    forall-assignment-stmt
    is    assignment-stmt
    or    pointer-assignment-stmt

R753    end-forall-stmt
    is    END FORALL
```

Constraint: The scalar-mask-expr shall be scalar and of type logical.

Constraint: Any procedure referenced in the scalar-mask-expr, including one referenced by a defined operation, shall be pure.

Constraint: An index-name shall be a named variable of type integer, shall not be a dummy argument, shall not have the POINTER attribute, shall not be initialized, shall not have the save attribute, shall not be accessed by use or host association, and shall be used in the scoping unit only as an index-name.

Constraint: A subscript or stride in a forall-triplet-spec shall not contain a reference to any index-name of the forall-triplet-spec in which it appears.

Constraint: A statement in a forall-body-construct shall not define an index-name of the forall-construct.

Constraint: Any procedure referenced in a forall-body-construct, including one referenced by a defined operation or assignment, shall be pure.

```
R754    forall-stmt
    is    FORALL (forall-triplet-spec-list [ , scalar-mask-expr ] ) &
            forall-assignment-stmt

R801    block
    is     [ executable-construct ] ...

R802    if-construct
    is     if-then-stmt
```

```
                block
            [ else-if-stmt
              block ] ...
            [ else-stmt
              block ]
            end-if-stmt

R803      if-then-stmt
      is      IF ( scalar-logical-expr ) THEN

R804      else-if-stmt
      is      ELSEIF ( scalar-logical-expr ) THEN

R805      else-stmt
      is      ELSE

R806      end-if-stmt
      is      ENDIF

R807      if-stmt
      is      IF ( scalar-logical-expr ) action-stmt

Constraint:      The action-stmt shall not be an if-stmt,
end-program-stmt, end-function-stmt, or end-subroutine-stmt.

R808      case-construct
      is      select-case-stmt
              [ case-stmt
                block ] ...
              [ CASE DEFAULT
                block ]
              end-select-stmt

R809      select-case-stmt
      is      SELECT CASE ( case-expr )

R810      case-stmt
      is      CASE case-selector

R811      end-select-stmt
      is      END SELECT

R812      case-expr
      is      scalar-int-expr
      or      scalar-char-expr

R813      case-selector
      is      ( case-value-range-list )

R814      case-value-range
      is      case-value
      or      case-value :
      or      : case-value
      or      case-value : case-value

R815      case-value
      is      scalar-int-initialization-expr
      or      scalar-char-initialization-expr

Constraint:      For a given case-construct, each case-value shall be of the
same type as case-expr.  For character type, length differences are allowed.
```

Constraint: For a given case-construct, the case-value-ranges shall not overlap; that is, there shall be no possible value of the case-expr that matches more than one case-value-range.

R816 do-construct
is block-do-construct

R817 block-do-construct
is do-stmt
do-block
end-do

R818 do-stmt
is [do-construct-name :] DO [loop-control]

Constraint: The do-construct-name shall not be the same as the name of any accessible entity.

Constraint: The same do-construct-name shall not be used on more than one do-stmt in a scoping unit.

R821 loop-control
is do-variable = scalar-int-expr, scalar-int-expr &
[, scalar-int-expr]

R822 do-variable
is scalar-int-variable

Constraint: A do-variable shall be a named variable of type integer, shall not be a dummy argument, shall not have the POINTER attribute, shall not be initialized, shall not have the SAVE attribute, shall not be accessed by use or host association, and shall be used in the scoping unit only as a do-variable.

R823 do-block
is block

R824 end-do
is ENDDO [do-construct-name]

Constraint: If the do-stmt is identified by a do-construct-name, the corresponding end-do shall specify the same do-construct-name. If the do-stmt is not identified by a do-construct-name, the corresponding end-do shall not specify a do-construct-name.

R834 cycle-stmt
is CYCLE [do-construct-name]

Constraint: If a cycle-stmt refers to a do-construct-name, it shall be within the range of that do-construct; otherwise, it shall be within the range of at least one do-construct.

R835 exit-stmt
is EXIT [do-construct-name]

Constraint: If an exit-stmt refers to a do-construct-name, it shall be within the range of that do-construct; otherwise, it shall be within the range of at least one do-construct.

R836 goto-stmt

```
      is      GO TO label

Constraint:      The label shall be the statement label of a continue-stmt
that appears after the goto-stmt in the same scoping unit as the goto-stmt.

R839      continue-stmt
      is      label CONTINUE

R840      stop-stmt
      is      STOP

R901      io-unit
      is      external-file-unit
      or      *
      or      internal-file-unit

R902      external-file-unit
      is      scalar-int-expr

R903      internal-file-unit
      is      char-variable

Constraint:      The char-variable shall not be an array section with a
vector subscript.

R904      open-stmt
      is      OPEN ( connect-spec-list )

R905      connect-spec
      is      UNIT = external-file-unit
      or      IOSTAT = scalar-default-int-variable
      or      FILE = file-name-expr
      or      STATUS = scalar-char-expr
      or      ACCESS = scalar-char-expr
      or      FORM = scalar-char-expr
      or      RECL = scalar-int-expr
      or      POSITION = scalar-char-expr
      or      ACTION = scalar-char-expr

R906      file-name-expr
      is      scalar-char-expr

Constraint:      A connect-spec-list shall contain exactly one UNIT =
io-unit, exactly one STATUS= scalar-char-expr, and exactly one ACTION =
scalar-char-expr and may contain at most one of each of the other specifiers.

R907      close-stmt
      is      CLOSE ( close-spec-list )

R908      close-spec
      is      UNIT = external-file-unit
      or      IOSTAT = scalar-default-int-variable
      or      STATUS = scalar-char-expr

Constraint:      A close-spec-list shall contain exactly one UNIT = io-unit
and may contain at most one of each of the other specifiers.

R909      read-stmt
      is      READ ( io-control-spec-list ) [ input-item-list ]
      or      READ format [ , input-item-list ]
```

```
R910      write-stmt
      is      WRITE ( io-control-spec-list ) [ output-item-list ]

R911      print-stmt
      is      PRINT format [ , output-item-list ]

R912      io-control-spec
      is      UNIT = io-unit
      or      FMT = format
      or      REC = scalar-int-expr
      or      IOSTAT = scalar-default-int-variable
      or      ADVANCE = scalar-char-expr
      or      SIZE = scalar-default-int-variable

Constraint:      An io-control-spec-list shall contain exactly one UNIT =
io-unit and may contain at most one of each of the other specifiers.

Constraint:      A SIZE= specifier shall not appear in a write-stmt.

Constraint:      If the unit specifier specifies an internal file, the
io-control-spec-list shall not contain a REC= specifier.

Constraint:      If the REC= specifier is present, the format, if any, shall
not be an asterisk specifying list-directed input/output.

Constraint:      An ADVANCE= specifier may be present only in a formatted
sequential input/output statement with explicit format specification whose
control information list does not contain an internal file unit specifier.

Constraint:      If a SIZE= specifier is present, an ADVANCE= specifier also
shall appear.

R913      format
      is      char-expr
      or      *

R914      input-item
      is      variable
      or      ( variable-list , implied-do-control )

R915      output-item
      is      expr
      or      ( expr-list , implied-do-control )

R919      backspace-stmt
      is      BACKSPACE ( position-spec-list )

R920      endfile-stmt
      is      ENDFILE ( position-spec-list )

R921      rewind-stmt
      is      REWIND ( position-spec-list )

R922      position-spec
      is      UNIT = external-file-unit
      or      IOSTAT = scalar-default-int-variable

Constraint:      A position-spec-list shall contain exactly one UNIT =
external-file-unit, and may contain at most one IOSTAT specifier.

R923      inquire-stmt
```

```
      is      INQUIRE ( inquire-spec-list )
      or      INQUIRE ( IOLENGTH = scalar-default-int-variable )
output-item-list

R924      inquire-spec
      is      UNIT = external-file-unit
      or      FILE = file-name-expr
      or      IOSTAT = scalar-default-int-variable
      or      EXIST = scalar-default-logical-variable
      or      OPENED = scalar-default-logical-variable
      or      NUMBER = scalar-default-int-variable
      or      NAMED = scalar-default-logical-variable
      or      NAME = scalar-char-variable
      or      ACCESS = scalar-char-variable
      or      SEQUENTIAL = scalar-char-variable
      or      DIRECT = scalar-char-variable
      or      FORM = scalar-char-variable
      or      FORMATTED = scalar-char-variable
      or      UNFORMATTED = scalar-char-variable
      or      RECL = scalar-default-int-variable
      or      NEXTREC = scalar-default-int-variable
      or      POSITION = scalar-char-variable
      or      ACTION = scalar-char-variable
      or      READ = scalar-char-variable
      or      WRITE = scalar-char-variable
      or      READWRITE = scalar-char-variable

Constraint:      An inquire-spec-list shall contain one FILE= specifier or
one UNIT= specifier, but not both, and at most one of each of the other
specifiers.

R1002     format-specification
      is      ( [ format-item-list ] )

R1003     format-item
      is      [ r ] data-edit-desc
      or      control-edit-desc
      or      [ r ] ( format-item-list )

R1004     r
      is      int-literal-constant

Constraint:      r shall be positive.

Constraint:      r shall not have a kind parameter specified for it.

R1005     data-edit-desc
      is      I w [ .  m ]
      or      F w .  d
      or      ES w .  d [ E e ]
      or      L w
      or      A [ w ]

R1006     w
      is      int-literal-constant

R1007     m
      is      int-literal-constant

R1008     d
      is      int-literal-constant
```

```
R1009     e
      is      int-literal-constant

Constraint:      w and e shall be nonnegative for I and F.  w and e shall
be positive for the other edit descriptors.

Constraint:      w, m, d, and e shall not have kind parameters specified for
them.

R1010     control-edit-desc
      is      position-edit-desc
      or      [ r ] /
      or      :
      or      sign-edit-desc

R1012     position-edit-desc
      is      T n
      or      TL n
      or      TR n

R1013     n
      is      int-literal-constant

Constraint:      n shall be positive.

Constraint:      n shall not have a kind parameter specified for it.

R1014     sign-edit-desc
      is      S
      or      SP
      or      SS

R1101     main-program
      is      program-stmt
                [ use-stmt ] ...
                [ derived-type-def ] ...
                [ specification-part ] ...
                [ execution-part ]
              end-program-stmt

R1102     program-stmt
      is      PROGRAM program-name

R1103     end-program-stmt
      is      END PROGRAM program-name

Constraint:      In a main-program, the execution-part shall not contain a
RETURN statement.

Constraint:      The program-name in the end-program-stmt shall be identical
to the program-name specified in the program-stmt.

Constraint:      An automatic object shall not appear in the
specification-part of a main program.

R1104x    module
      is      public-module
      or      private-module

R1104y    public-module
```

```
      is      module-stmt
                use-stmt
                [ use-stmt ] ...
                PUBLIC
              end-module-stmt

R1104     private-module
      is      module-stmt
                [ use-stmt ] ...
                [ PRIVATE ]
                [ specification-part ]
                [ subprogram-part ]
              end-module-stmt

Constraint:      A PRIVATE statement shall appear if any use-stmts appear.
A PRIVATE statement shall not appear if no use-stmts appear.

Constraint:      Every function-subprogram or subroutine-subprogram in a
private-module shall be listed in an access-stmt.

R1105     module-stmt
      is      MODULE module-name

R1106     end-module-stmt
      is      END MODULE module-name

Constraint:      The module-name is specified in the end-module-stmt shall
be identical to the module-name specified in the module-stmt.

Constraint:      An automatic object shall not appear in the
specification-part of a module.

R1107     use-stmt
      is      USE module-name [ , rename-list ]
      or      USE module-name , ONLY : [ only-list ]

R1108     rename
      is      local-name => use-name

R1109     only
      is      generic-spec
      or      only-use-name
      or      only-rename

R1110     only-use-name
      is      use-name

R1111     only-rename
      is      local-name => use-name

Constraint:      Each generic-spec shall be a public entity in the module.

Constraint:      Each use-name shall be the name of a public entity in the
module.

Constraint:      use-name shall not be the name of an intrinsic procedure.

Constraint:      In a use-stmt a use-name shall appear only once.

Constraint:      No two accessible entities may have the same local name.
```

```
R1201     interface-block
      is      INTERFACE [ generic-spec ]
                 [ interface-specification ] ...
              END INTERFACE [ generic-spec ]

R1202     interface-specification
      is      interface-body
      or      module-procedure-stmt

R1205     interface-body
      is      function-stmt
                 [ specification-part ]
              end-function-stmt
      or      subroutine-stmt
                 [ specification-part ]
              end-subroutine-stmt

Constraint:      An interface-body shall specify the intents of all dummy
arguments except pointer and procedure arguments.

Constraint:      Each procedure dummy argument shall appear in exactly one
interface body.

R1206     module-procedure-stmt
      is      MODULE PROCEDURE procedure-name-list

R1207     generic-spec
      is      generic-name
      or      OPERATOR ( defined-operator )
      or      ASSIGNMENT ( = )

Constraint:      Every generic-spec in a private-module shall be listed in
an access-stmt.

Constraint:      If generic-spec is also the name of an intrinsic procedure
the generic name shall appear in a previous intrinsic statement in the module.

Constraint:      An external procedure shall not be used as an actual argument.

R1209     intrinsic-stmt
      is      INTRINSIC :: intrinsic-procedure-name-list

Constraint:      Each intrinsic-procedure-name shall refer to an
intrinsic procedure in the following list:

      abs
      acos
      adjustl
      adjustr
      aimag
      aint
      all
      allocated
      anint
      any
      asin
      associated
      atan
      atan2
      bit_size
      btest
```

```
ceiling
char
cmplx
conjg
cos
cosh
count
cpu_time
cshift
date_and_time
digits
dot_product
eoshift
epsilon
exp
exponent
floor
fraction
huge
iand
ibclr
ibits
ibset
ichar
ieor
index
int
ior
ishft
ishftc
kind
lbound
len
len_trim
log
log10
logical
matmul
max
maxexponent
maxloc
maxval
merge
min
minexponent
minloc
minval
modulo
mvbits
nearest
nint
not
null
pack
precision
present
product
radix
random_number
random_seed
range
```

```
      real
      repeat
      reshape
      rrspacing
      scale
      scan
      selected_int_kind
      selected_real_kind
      set_exponent
      shape
      sign
      sin
      sinh
      size
      spacing
      spread
      sqrt
      sum
      system_clock
      tan
      tanh
      tiny
      transpose
      trim
      ubound
      unpack
      verify
```

Constraint: In a reference to any intrinsic function that has a kind argument the corresponding actual argument must be a named constant.

```
R1210     function-reference
      is      function-name ( [ actual-arg-spec-list ] )

R1211     call-stmt
      is      CALL subroutine-name ( [ actual-arg-spec-list ] )

R1212     actual-arg-spec
      is      [ keyword = ] actual-arg

R1213     keyword
      is      dummy-arg-name

R1214     actual-arg
      is      expr
      or      variable
      or      procedure-name
```

Constraint: The keyword = may be omitted from an actual-arg-spec only if the keyword = has been omitted from each preceding actual-arg-spec in the argument list.

Constraint: Each keyword shall be the name of a dummy argument of the procedure.

Constraint: In a reference to a function, a procedure-name actual-arg shall be the name of a function.

Constraint: A procedure-name actual-arg shall not be the name of an intrinsic function or a generic-name.

```
R1216     function-subprogram
      is      function-stmt
                [ specification-part ]
                [ execution-part ]
              end-function-stmt

R1217     function-stmt
      is      [ prefix ] FUNCTION function-name
                     ( [ dummy-arg-name-list ] ) RESULT ( result-name )

Constraint:      The function-name shall not appear in any specification
statement in the scoping unit of the function subprogram.

R1218     prefix
      is      prefix-spec [ prefix-spec ]...

R1219     prefix-spec
      is      RECURSIVE
      or      PURE
      or      ELEMENTAL

Constraint:      A prefix shall contain at most one of each prefix-spec.

Constraint:      If ELEMENTAL is present, RECURSIVE shall not be present.

R1220     end-function-stmt
      is      END FUNCTION function-name

Constraint:      result-name shall not be the same as function-name.

Constraint:      The function-name in the end-function-stmt shall be
identical to the function-name specified in the function-stmt.

R1221     subroutine-subprogram
      is      subroutine-stmt
                [ specification-part ]
                [ execution-part ]
              end-subroutine-stmt

R1222     subroutine-stmt
      is      [ prefix ] SUBROUTINE subroutine-name ( [ dummy-arg-list ] )

R1223     dummy-arg
      is      dummy-arg-name

R1224     end-subroutine-stmt
      is      END SUBROUTINE subroutine-name

Constraint:      The subroutine-name in the end-subroutine-stmt shall be
identical to the subroutine-name specified in the subroutine-stmt.

R1226     return-stmt
      is      RETURN

Constraint:      The return-stmt shall be in the scoping unit of a function
or subroutine subprogram.

R1227     contains-stmt
      is      CONTAINS

Constraint:      Every function-subprogram shall satisfy the constraints of a
```

pure function, whether or not the keyword PURE appears, except that a function in which PURE does not appear my contain a PRINT statement.

The following constraints for Section 12 apply to the syntax rules defining function subprograms (R1216-R1220) and pure subroutine subprograms (R1222-R1224).

Constraint: The specification-part of a function shall specify that all dummy arguments have INTENT(IN) except procedure arguments and arguments with the POINTER attribute.

Constraint: The specification-part of a subroutine shall specify the intents of all dummy arguments except procedure arguments and arguments with the POINTER attribute.

Constraint: A local variable declared in the specification-part of a function or pure subroutine shall not have the SAVE attribute.

Constraint: The specification-part of a function or pure subroutine shall specify that all dummy arguments that are procedures are pure.

Constraint: If a procedure that is not an intrinsic procedure is used in a context that requires it to be pure, its interface shall be explicit in the scope of that use. The interface shall specify that the procedure is pure.

Constraint: In a function or pure subroutine any variable that is accessed by host or use association, is a dummy argument to a function, or is a dummy argument with INTENT(IN) to a pure subroutine shall not be used in the following contexts:

(1) As the variable of an assignment-stmt;

(2) As a DO variable or implied DO variable;

(3) As an input-item in a read-stmt from an internal file;

(4) As an internal-file unit in a write-stmt;

(5) As an IOSTAT=specifier in an input or output statement with an internal file;

(6) As the pointer-object of a pointer-assignment-stmt;

(7) As the target of a pointer-assignment-stmt;

(8) As the expr of an assignment-stmt in which the variable is of a derived type if the derived type has a pointer component at any level of component selection;

(9) As an allocate-object or stat-variable in an allocate-stmt or deallocate-stmt; or

(10) As an actual argument associated with a dummy argument with INTENT(OUT) or INTENT(IN OUT) or with the POINTER attribute.

Constraint: Any procedure referenced in a pure procedure, including one referenced via a defined operation or assignment, shall be pure.

Constraint: Any procedure referenced in a function,

including one referenced via a defined operation or assignment, shall be a function or a pure procedure.

Constraint: A function or pure subroutine shall not contain a print-stmt, open-stmt, close-stmt, backspace-stmt, end-file-stmt, rewind-stmt, or inquire-stmt, except that a function not explicitly declared pure may contain a print-stmt.

Constraint: A function or pure subroutine shall not contain a read-stmt or write-stmt whose io-unit is an external-file-unit or *.

Constraint: A function or pure subroutine shall not contains a stop-stmt.

The following constraints for Section 12 apply to the syntax rules defining elemental procedures:

Constraint: All dummy arguments shall be scalar and shall not have the POINTER attribute.

Constraint: For a function, the result shall be scalar and shall not have the POINTER attribute.

Constraint: A dummy argument, or a subobject thereof, shall not appear in a specification-expr except as the argument to one of the intrinsic functions BIT_SIZE, KIND, LEN, or the numeric inquiry functions.

Constraint: A dummy argument shall not be a procedure.

The Fortran Company
11155 E. Mountain Gate Place
Tucson, Arizona 85749 USA
+1-520-256-1455 ** 760-1397 (fax)
info@fortran.com