Annex A

(informative)

Glossary of technical terms

The following is a list of the principal technical terms used in the standard and their definitions. A reference in parentheses immediately after a term is to the section where the term is defined or explained. The wording of a definition here is not necessarily the same as in the standard.

action statement (2.1) : A single statement specifying or controlling a computational action (R216).

actual argument (12.4.1) : An expression, a variable, a procedure, or an alternate return specifier that is specified in a procedure reference.

allocatable variable (5.1.2.2) : A variable having the ALLOCATABLE attribute. It may be referenced or defined only when it has space allocated. If it is an array, it has a shape only when it has space allocated. It may be a named variable or a structure component.

argument (12) : An actual argument or a dummy argument.

argument association (16.7.1.1) : The relationship between an actual argument and a dummy argument during the execution of a procedure reference.

array (2.4.5) : A set of scalar data, all of the same type and type parameters, whose individual elements are arranged in a rectangular pattern. It may be a named array, an array section, a structure component, a function value, or an expression. Its rank is at least one. Note that in FORTRAN 77, arrays were always named and never constants.

array element (2.4.5, 6.2.2) : One of the scalar data that make up an array that is either named or is a structure component.

array pointer (5.1.2.5.3) : A pointer to an array.

array section (2.4.5, 6.2.2.3) : A subobject that is an array and is not a structure component.

array-valued : Having the property of being an array.

assignment statement (7.5.1.1) : A statement of the form "variable = expression".

associate name (8.1.4.1) : The name by which a selector of a SELECT TYPE or ASSOCIATE construct is known within the construct.

association (16.7) : Name association, pointer association, storage association, or inheritance association.

assumed-shape array (5.1.2.5.2) : A nonpointer dummy array that takes its shape from the associated actual argument.

assumed-size array (5.1.2.5.4) : A dummy array whose size is assumed from the associated actual argument. Its last upper bound is specified by an asterisk.

attribute (5) : A property of a data object that may be specified in a type declaration statement (R501).

automatic data object (5.1) : A data object that is a local entity of a subprogram, that is not a dummy argument, and that has a character length or array bound that is specified by an expression that is not an initialization expression.

base type (4.5.3) : An extensible type that is not an extension of another type. A type that is declared with the EXTENSIBLE attribute.

belong (8.1.5.4.3, 8.1.5.4.4) : If an EXIT or a CYCLE statement contains a construct name, the statement **belongs** to the DO construct using that name. Otherwise, it **belongs** to the innermost DO construct in which it appears.

binding label (12.5.2.7, 15.2.7.1) : A value of type default character that uniquely identifies how a variable, common block, subroutine, or function is known to a companion processor.

block (8.1) : A sequence of executable constructs embedded in another executable construct, bounded by statements that are particular to the construct, and treated as an integral unit.

block data program unit (11.4) : A program unit that provides initial values for data objects in named common blocks.

bounds (5.1.2.5.1) : For a named array, the limits within which the values of the subscripts of its array elements shall lie.

character (3.1) : A letter, digit, or other symbol.

class (5.1.1.8) : A class named N is the set of types extended from the type named N.

characteristics (12.2) :

(1) Of a procedure, its classification as a function or subroutine, whether it is pure, whether it is elemental, whether it has the BIND attribute, the value of its binding label, the characteristics of its dummy arguments, and the characteristics of its function result if it is a function.

(2) Of a dummy argument, whether it is a data object, is a procedure, is a procedure pointer, is an asterisk (alternate return indicator), or has the OPTIONAL attribute.

(3) Of a dummy data object, its type, type parameters, shape, the exact dependence of an array bound or type parameter on other entities, intent, whether it is optional, whether it is a pointer or a target, whether it is allocatable, whether it has the VALUE, ASYNCHRONOUS, or VOLATILE attributes, whether it is polymorphic, and whether the shape, size, or a type parameter is assumed.

(4) Of a dummy procedure or procedure pointer, whether the interface is explicit, the characteristics of the procedure if the interface is explicit, and whether it is optional.

(5) Of a function result, its type, type parameters, which type parameters are deferred, whether it is polymorphic, whether it is a pointer or allocatable, whether it is a procedure pointer, rank if it is a pointer or allocatable, shape if it is not a pointer or allocatable, the exact dependence of an array bound or type parameter on other entities, and whether the character length is assumed.

character length parameter (2.4.1.1) : The type parameter that specifies the number of characters for an entity of type character.

character string (4.4.4) : A sequence of characters numbered from left to right 1, 2, 3, ...

character storage unit (16.7.3.1) : The unit of storage for holding a scalar that is not a pointer and is of type default character and character length one.

collating sequence (4.4.4.1) : An ordering of all the different characters of a particular kind type parameter.

common block (5.5.2) : A block of physical storage that may be accessed by any of the scoping units in a program.

companion processor (2.5.10): A mechanism by which global data and procedures may be referenced or defined. It may be a mechanism that references and defines such entities by means other than Fortran. The procedures can be described by a C function prototype.

component (4.5) : A constituent of a derived type.

component order (4.5.4) : The ordering of the components of a derived type that is used for intrinsic formatted input/output and for structure constructors.

conformable (2.4.5) : Two arrays are said to be **conformable** if they have the same shape. A scalar is conformable with any array.

conformance (1.5) : A program conforms to the standard if it uses only those forms and relationships described therein and if the program has an interpretation according to the standard. A program unit conforms to the standard if it can be included in a program in a manner that allows the program to be standard conforming. A processor conforms to the standard if it executes standard-conforming programs in a manner that fulfills the interpretations prescribed in the standard and contains the capability of detection and reporting as listed in 1.5.

connected (9.4.3) :

(1) For an external unit, the property of referring to an external file.

(2) For an external file, the property of having an external unit that refers to it.

constant (2.4.3.1.2) : A data object whose value shall not change during execution of a program. It may be a named constant or a literal constant.

construct (7.5.3, 7.5.4, 8.1) : A sequence of statements starting with an ASSOCIATE, DO, FORALL, IF, SELECT CASE, SELECT TYPE, or WHERE statement and ending with the corresponding terminal statement.

construct entity (16) : An entity defined by a lexical token whose scope is a construct.

control mask (7.5.3) : In a WHERE statement or construct, an array of type logical whose value determines which elements of an array, in a *where-assignment-stmt*, will be defined.

data : Plural of datum.

data entity (2.4.3) : A data object, the result of the evaluation of an expression, or the result of the execution of a function reference (called the function result). A data entity has a data type (either intrinsic or derived) and has, or may have, a data value (the exception is an undefined variable). Every data entity has a rank and is thus either a scalar or an array.

data object (2.4.3.1) : A data entity that is a constant, a variable, or a subobject of a constant.

data type (2.4.1) : A named category of data that is characterized by a set of values, together with a way to denote these values and a collection of operations that interpret and manipulate the values. For an intrinsic type, the set of data values depends on the values of the type parameters.

datum : A single quantity that may have any of the set of values specified for its data type.

decimal symbol (9.8.1.6, 10.5, 10.7.8) : The character that separates the whole and fractional parts in the decimal representation of a real number in a file. By default the decimal symbol is a decimal point (also known as a period). The current decimal symbol is determined by the current decimal edit mode.

declared type (5.1.1.8, 7.1.4) : The data type that a data entity is declared to have. May differ from the data type during execution (the dynamic type) for polymorphic data entities.

default initialization (4.5) : If initialization is specified in a type definition, an object of the type will be automatically initialized. Nonpointer components may be initialized with values by default; pointer components may be initially disassociated by default. Default initialization is not provided for objects of intrinsic type.

default-initialized (4.5.1.2) : A subcomponent is said to be default-initialized if it will be initialized by default initialization.

deferred binding (4.5.1.5) : A type-bound procedure binding that specifies the NULL() intrinsic. A deferred binding shall not be invoked.

deferred type parameter (4.3) : A nonkind type parameter whose value is not specified in the declaration of an object, but instead is specified when the object is allocated or pointer-assigned.

definable (2.5.4) : A variable is **definable** if its value may be changed by the appearance of its designator on the left of an assignment statement. An allocatable variable that has not been allocated is an example of a data object that is not definable. An example of a subobject that is not definable is C (I) when C is an array that is a constant and I is an integer variable.

defined (2.5.4) : For a data object, the property of having or being given a valid value.

defined assignment statement (7.5.1.3, 12.3.2.1.2) : An assignment statement that is not an intrinsic assignment statement and is defined by a subroutine and a generic interface that specifies ASSIGNMENT (=).

defined operation (7.1.3, 12.3.2.1.1) : An operation that is not an intrinsic operation and is defined by a function that is associated with a generic identifier.

deleted feature (1.7) : A feature in a previous Fortran standard that is considered to have been redundant and largely unused. See section B.1 for a list of features which that in a previous Fortran standard, but are not in this standard. A feature designated as an obsolescent feature in the standard may become a deleted feature in the next revision.

derived type (2.4.1.2, 4.5) : A type whose data have components, each of which is either of intrinsic type or of another derived type.

designator : See object designator.

disassociated (2.4.6) : A pointer is **disassociated** following execution of a DEALLOCATE or NULLIFY statement, or following pointer association with a disassociated pointer.

dummy argument (12.5.2.1, 12.5.2.2, 12.5.2.4, 12.5.4) : An entity whose name appears in the parenthesized list following the procedure name in a FUNCTION statement, a SUBROUTINE statement, an ENTRY statement, or a statement function statement.

dummy array : A dummy argument that is an array.

dummy pointer : A dummy argument that is a pointer.

dummy procedure (12.1.2.3) : A dummy argument that is specified or referenced as a procedure.

dynamic type (5.1.1.8, 7.1.4) : The type of a data entity during execution of a program. The dynamic type of a data entity that is not polymorphic is the same as its declared type.

effective item (9.5.2) : A scalar object resulting from expanding an input/output list according to the rules in 9.5.2.

elemental (2.4.5, 7.5.1.3, 12.7) : An adjective applied to an operation, procedure, or assignment statement that is applied independently to elements of an array or corresponding elements of a set of conformable arrays and scalars.

entity : The term used for any of the following: a program unit, procedure, abstract interface, operator, generic interface, common block, external unit, statement function, type, data entity, statement label, construct, type alias, or namelist group.

executable construct (2.1) : An action statement (R216) or an ASSOCIATE, CASE, DO, FORALL, IF, SELECT TYPE, or WHERE construct.

executable statement (2.3.1) : An instruction to perform or control one or more computational actions.

explicit initialization (5.1) : Explicit initialization may be specified for objects of intrinsic or derived type in type declaration statements or DATA statements. An object of a derived type that specifies default initialization may not appear in a DATA statement.

explicit interface (12.3.1) : For a procedure referenced in a scoping unit, the property of being an internal procedure, a module procedure, an intrinsic procedure, an external procedure that has an interface body, a recursive procedure reference in its own scoping unit, or a dummy procedure that has an interface body.

explicit-shape array (5.1.2.5.1) : A named array that is declared with explicit bounds.

expression (2.4.3.2, 7.1) : A sequence of operands, operators, and parentheses (R722). It may be a variable, a constant, a function reference, or may represent a computation.

extended type (4.5.3) : An extensible type that is an extension of another type. A type that is declared with the EXTENDS attribute.

extensible type (4.5.3) : A type from which new types may be derived using the EXTENDS attribute. A type that is declared with either the EXTENSIBLE attribute or the EXTENDS attribute.

extension type (4.5.3) : A base type is an extension type of itself only. An extended type is an extension type of itself and of all types for which its parent type is an extension.

extent (2.4.5) : The size of one dimension of an array.

external file (9.2) : A sequence of records that exists in a medium external to the program.

external linkage : The characteristic describing that a C entity is global to the program; defined in clause 6.2.2 of the C standard.

external procedure (2.2.3.1) : A procedure that is defined by an external subprogram or by a means other than Fortran.

external subprogram (2.2) : A subprogram that is not in a main program, module, or another subprogram. Note that a module is not called a subprogram. Note that in FORTRAN 77, a block data program unit is called a subprogram.

external unit (9.4) : A mechanism that is used to refer to an external file. It is identified by a nonnegative integer.

file (9) : An internal file or an external file.

file storage unit (9.2.4) : The unit of storage for an unformatted or stream file.

final subroutine (4.5.1.9) : A subroutines that is called automatically by the processor during finalization.

finalizable (4.5.1.9) : A type that has final subroutines, or that has a finalizable component. An object of finalizable type.

finalization (4.5.10) : The process of calling user-defined final subroutines immediately before destroying an object.

function (2.2.3) : A procedure that is invoked in an expression and computes a value which is then used in evaluating the expression.

function result (12.5.2.1) : The data object that returns the value of a function.

function subprogram (12.5.2.1) : A sequence of statements beginning with a FUNCTION statement that is not in an interface block and ending with the corresponding END statement.

generic identifier (12.3.2.1) : A lexical token that appears in an INTERFACE statement and is associated with all the procedures in the interface block.

generic interface (4.5.1.5, 12.3.2.1) : An interface specified by a generic procedure binding or a generic interface block.

generic interface block (12.3.2.1) : An interface block with a generic specification.

global entity (16.1.1) : An entity identified by a lexical token whose scope is a program. It may be a program unit, a common block, or an external procedure.

host (2.2) : Host scoping unit.

host association (16.7.1.3) : The process by which a contained scoping unit accesses entities of its host.

host scoping unit (2.2) : A scoping unit that immediately surrounds another scoping unit.

implicit interface (12.3.1) : A procedure referenced in a scoping unit other than its own is said to have an implicit interface if the procedure is an external procedure that does not have an interface body, a dummy procedure that does not have an interface body, or a statement function.

inherit (4.5.3) : To acquire from a parent. Components or procedure bindings of an extended type that are automatically acquired from its parent type without explicit declaration in the extended type are said to be inherited.

inheritance association (4.5.3.1, 16.7.4) : The relationship between the inherited components and the parent component in an extended type.

inquiry function (13.1) : A function that is either intrinsic or is defined in an intrinsic module and whose result depends on properties of the principal argument other than the value of the argument.

intent (5.1.2.7) : An attribute of a dummy argument data object that indicates whether it is used to transfer data into the procedure, out of the procedure, or both.

instance of a subprogram (12.5.2.3) : The copy of a subprogram that is created when a procedure defined by the subprogram is invoked.

interface block (12.3.2.1) : A sequence of statements from an INTERFACE statement to the corresponding END INTERFACE statement.

interface body (12.3.2.1) : A sequence of statements in an interface block from a FUNCTION or SUBROUTINE statement to the corresponding END statement.

interface of a procedure (12.3) : See procedure interface.

internal file (9.3) : A character variable that is used to transfer and convert data from internal storage to internal storage.

internal procedure (2.2.3.3) : A procedure that is defined by an internal subprogram.

internal subprogram (2.2) : A subprogram in a main program or another subprogram.

intrinsic (2.5.7) : An adjective that may be applied to data types, operators, assignment statements, procedures, and modules. Intrinsic data types, operators, and assignment statements are defined in this standard and may be used in any scoping unit without further definition or specification. Intrinsic procedures are defined in this standard or provided by a processor, and may be used in a scoping unit without further definition or specification. Intrinsic modules are defined in this standard or provided by a processor, and may be accessed by use association; procedures and types defined in an intrinsic module are not themselves intrinsic.

Intrinsic procedures and modules that are not defined in this standard are called nonstandard intrinsic procedures and modules.

invoke (2.2.3) :

(1) To call a subroutine by a CALL statement or by a defined assignment statement.

(2) To call a function by a reference to it by name or operator during the evaluation of an expression.

keyword (2.5.2) : A word that is part of the syntax of a statement or a name that is used to identify an item in a list.

kind type parameter (2.4.1.1, 4.4.1, 4.4.2, 4.4.3, 4.4.4, 4.4.5) : A parameter whose values label the available kinds of an intrinsic type.

label : See statement label.

length of a character string (4.4.4) : The number of characters in the character string.

lexical token (3.2) : A sequence of one or more characters with a specified interpretation.

line (3.3) : A sequence of 0 to 132 characters, which may contain Fortran statements, a comment, or an INCLUDE line.

linked (12.5.3) : When a C function with external linkage has the same binding label as a Fortran procedure, they are said to be linked. It is also possible for two Fortran entities to be linked.

literal constant (2.4.3.1.2, 4.4) : A constant without a name. Note that in FORTRAN 77, this was called simply a constant.

local entity (16.1.2) : An entity identified by a lexical token whose scope is a scoping unit.

local variable (2.4.3.1.1) : A variable local to a particular scoping unit; not imported through use or host association, not a dummy argument, and not a variable in common.

main program (11.1) : A program unit that is not a module, external subprogram, or block data program unit.

many-one array section (6.2.2.3.2) : An array section with a vector subscript having two or more elements with the same value.

module (2.2.4, 11.3) : A program unit that contains or accesses definitions to be accessed by other program units.

module procedure (2.2.3.2) : A procedure that is defined by a module subprogram.

module subprogram (2.2) : A subprogram that is in a module but is not an internal subprogram.

name (3.2.1) : A lexical token consisting of a letter followed by up to 30 alphanumeric characters (letters, digits, and underscores). Note that in FORTRAN 77, this was called a symbolic name.

name association (16.7.1) : Argument association, use association, or host association.

named : Having a name. That is, in a phrase such as ''named variable,'' the word ''named'' signifies that the variable name is not qualified by a subscript list, substring specification, and so on. For example, if X is an array variable, the reference ''X'' is a named variable while the reference ''X(1)'' is an object designator.

named constant (2.4.3.1.2) : A constant that has a name. Note that in FORTRAN 77, this was called a symbolic constant.

NaN (14.6) : A Not-a-Number value of IEEE arithmetic. It represents an undefined value or a value created by an invalid operation.

nonexecutable statement (2.3.1) : A statement used to configure the program environment in which computational actions take place.

numeric storage unit (16.7.3.1) : The unit of storage for holding a scalar that is not a pointer and is of type default real, default integer, or default logical.

numeric type (4.4) : Integer, real or complex type.

object (2.4.3.1) : Data object.

object designator (2.5.1) : A name, followed by zero or more of the following: component selectors, array section selectors, array element selectors, and substring selectors.

obsolescent feature (1.7) : A feature that is considered to have been redundant but that is still in frequent use.

operand (2.5.8) : An expression that precedes or succeeds an operator.

operation (7.1.2) : A computation involving one or two operands.

operator (2.5.8) : A lexical token that specifies an operation.

override (4.5.1, 4.5.3) : When explicit initialization or default initialization overrides default initialization, it is as if only the overriding initialization were specified. If a procedure is bound to an extensible type, it overrides the one that would have been inherited from the parent type.

parent type (4.5.3) : The extensible type from which an extended type is derived.

parent component (4.5.3.1) : The component of an entity of extended type that corresponds to its inherited portion.

passed-object dummy argument (4.5.1) : The dummy argument of a type-bound procedure or procedure pointer component that becomes associated with the object through which the procedure was invoked.

pointer (2.4.6) : An entity that has the POINTER attribute.

pointer assignment (7.5.2) : The pointer association of a pointer with a target by the execution of a pointer assignment statement or the execution of an assignment statement for a data object of derived type having the pointer as a subobject.

pointer assignment statement (7.5.2) : A statement of the form "pointer-object => target".

pointer associated (6.3, 7.5.2) : The relationship between a pointer and a target following a pointer assignment or a valid execution of an ALLOCATE statement.

pointer association (16.7.2) : The process by which a pointer becomes pointer associated with a target.

polymorphic (5.1.1.8) : Able to be of differing types during program execution. An object declared with the CLASS keyword is polymorphic.

preconnected (9.4.4) : A property describing a unit that is connected to an external file at the beginning of execution of a program. Such a unit may be specified in input/output statements without an OPEN statement being executed for that unit.

present (12.4.1.6) : A dummy argument is **present** in an instance of a subprogram if it is associated with an actual argument and the actual argument is a dummy argument that is present in the invoking subprogram or is not a dummy argument of the invoking subprogram.

procedure (2.2.3, 12.1) : A computation that may be invoked during program execution. It may be a function or a subroutine. It may be an intrinsic procedure, an external procedure, a module procedure, an internal procedure, a dummy procedure, or a statement function. A subprogram may define more than one procedure if it contains ENTRY statements.

procedure interface (12.3) : The characteristics of a procedure, the name of the procedure, the name of each dummy argument, and the generic identifiers (if any) by which it may be referenced.

processor (1.2) : The combination of a computing system and the mechanism by which programs are transformed for use on that computing system.

processor dependent (1.5) : The designation given to a facility that is not completely specified by this standard. Such a facility shall be provided by a processor, with methods or semantics determined by the processor.

program (2.2.1) : A set of program units that includes exactly one main program.

program unit (2.2) : The fundamental component of a program. A sequence of statements, comments, and INCLUDE lines. It may be a main program, a module, an external subprogram, or a block data program unit.

prototype : The C analog of a function interface body; defined in 6.7.5.3 of the C standard.

pure procedure (12.6) : A procedure that is a pure intrinsic procedure (13.1, 13.7), is defined by a pure subprogram, or is a statement function that references only pure functions.

rank (2.4.4, 2.4.5) : The number of dimensions of an array. Zero for a scalar.

record (9.1) : A sequence of values or characters that is treated as a whole within a file.

reference (2.5.5) : The appearance of an object designator in a context requiring the value at that point during execution, the appearance of a procedure name, its operator symbol, or a defined assignment statement in a context requiring execution of the procedure at that point, or the appearance of a module name in a USE statement. Neither the act of defining a variable nor the appearance of the name of a procedure as an actual argument is regarded as a reference.

result variable (2.2.3, 12.5.2.1) : The variable that returns the value of a function.

rounding mode (14.3) : The method used in IEEE arithmetic to choose the result of a floating-point operation that cannot be represented exactly. There are four modes; nearest, towards zero, up (towards ∞), and down (towards $-\infty$).

scalar (2.4.4) :

(1) A single datum that is not an array.

(2) Not having the property of being an array.

scope (16) : That part of a program within which a lexical token has a single interpretation. It may be a program, a scoping unit, a construct, a single statement, or a part of a statement.

scoping unit (2.2) : One of the following:

(1) A program unit or subprogram, excluding any scoping units in it,

(2) A derived type definition, or

(3) An interface body, excluding any scoping units in it.

section subscript (6.2.2) : A subscript, vector subscript, or subscript triplet in an array section selector.

selector : A syntactic mechanism for designating

(1) Part of a data object. It may designate a substring, an array element, an array section, or a structure component.

(2) The set of values for which a CASE block is executed.

shape (2.4.5) : For an array, the rank and extents. The shape may be represented by the rank-one array whose elements are the extents in each dimension.

size (2.4.5) : For an array, the total number of elements.

specification expression (7.1.6) : An expression with limitations that make it suitable for use in specifications such as nonkind type parameters or array bounds.

specification function (7.1.6) : A nonintrinsic function that may be used in a specification expression.

standard-conforming program (1.5) : A program that uses only those forms and relationships described in this standard, and that has an interpretation according to this standard.

statement (3.3) : A sequence of lexical tokens. It usually consists of a single line, but a statement may be continued from one line to another and the semicolon symbol may be used to separate statements within a line.

statement entity (16) : An entity identified by a lexical token whose scope is a single statement or part of a statement.

statement function (12.5.4) : A procedure specified by a single statement that is similar in form to an assignment statement.

statement label (3.2.4) : A lexical token consisting of up to five digits that precedes a statement and may be used to refer to the statement.

storage association (16.7.3) : The relationship between two storage sequences if a storage unit of one is the same as a storage unit of the other.

storage sequence (16.7.3.1) : A sequence of contiguous storage units.

storage unit (16.7.3.1) : A character storage unit, a numeric storage unit, a file storage unit, or an unspecified storage unit.

stride (6.2.2.3.1) : The increment specified in a subscript triplet.

struct : The C analog of a sequence derived type; defined in 6.2.5 of the C standard.

structure (2.4.1.2) : A scalar data object of derived type.

structure component (6.1.2) : A part of an object of derived type. It may be referenced by an object designator.

structure constructor (4.5.8) : A syntactic mechanism for constructing a value of derived type.

subcomponent (6.1.2) : A subobject that is a structure component.

subobject (2.4.3.1) : A portion of a data object that may be referenced or defined independently of other portions. It may be an array element, an array section, a structure component, a substring, or the real or imaginary part of a complex object.

subprogram (2.2) : A function subprogram or a subroutine subprogram. Note that in FORTRAN 77, a block data program unit was called a subprogram.

subroutine (2.2.3) : A procedure that is invoked by a CALL statement or by a defined assignment statement.

subroutine subprogram (12.5.2.2) : A sequence of statements beginning with a SUBROUTINE statement that is not in an interface block and ending with the corresponding END statement.

subscript (6.2.2) : One of the list of scalar integer expressions in an array element selector. Note that in FORTRAN 77, the whole list was called the subscript.

subscript triplet (6.2.2) : An item in the list of an array section selector that contains a colon and specifies a regular sequence of integer values.

substring (6.1.1) : A contiguous portion of a scalar character string. Note that an array section can include a substring selector; the result is called an array section and not a substring.

target (2.4.6, 5.1.2.13, 6.3.1.3) : A data entity that has the TARGET attribute, or an entity that is associated with a pointer.

transformational function (13.1) : A function that is either intrinsic or is defined in an intrinsic module and that is neither an elemental function nor an inquiry function. It usually has array

arguments and an array result whose elements have values that depend on the values of many of the elements of the arguments.

type (4) : Data type.

type-bound procedure (4.5.1.5) : A procedure binding in a type definition. The procedure may be referenced by the *binding-name* via any object of that dynamic type, as a defined operator, or by defined assignment.

type-compatible (5.1.1.8) : All entities are type-compatible with other entities of the same data type. Unlimited polymorphic entities are type-compatible with all entities of extensible type; other polymorphic entities are type-compatible with entities whose dynamic type is an extension type of the polymorphic entity's declared type.

type declaration statement (5) : An INTEGER, REAL, DOUBLE PRECISION, COMPLEX, CHARACTER, LOGICAL, or TYPE (*type-name*) statement.

type parameter (2.4.1.1) : A parameter of a data type. KIND and LEN are the type parameters of intrinsic types. The type parameters of a derived type are defined in the derived type definition.

type parameter order (4.5.5) : The ordering of the type parameters of a derived type that is used for derived-type specifiers.

ultimate component (4.5) : For a derived type or a structure, a component that is of intrinsic type, has the ALLOCATABLE attribute, or has the POINTER attribute, or an ultimate component of a derived-type component that does not have the POINTER attribute or the ALLOCATABLE attribute.

undefined (2.5.4) : For a data object, the property of not having a determinate value.

unsigned : A qualifier of a C numeric type indicating that it is only comprised of nonnegative values; defined in clause 6.2.5 of the C standard. There is nothing analogous in Fortran.

unspecified storage unit (16.7.3.1) : A unit of storage for holding a pointer or a scalar that is not a pointer and is of type other than default integer, default character, default real, double precision real, default logical, or default complex.

use association (16.7.1.2) : The association of names in different scoping units specified by a USE statement.

variable (2.4.3.1.1) : A data object whose value can be defined and redefined during the execution of a program. It may be a named data object, an array element, an array section, a structure component, or a substring. Note that in FORTRAN 77, a variable was always scalar and named.

vector subscript (6.2.2.3.2) : A section subscript that is an integer expression of rank one.

void : A C type comprising an empty set of values; defined in clause 6.2.5 of the C standard. There is nothing analogous in Fortran.

whole array (6.2.1) : A named array.