Subject: Correct C1107 and definition of ultimate components
From: Van Snyder

1 Introduction

C1107 [250:23-25] refers to "...of a type for which *component-initialization* is specified...." This doesn't cover the case of a component of a component of ..., but almost certainly should. The term "subcomponent" is wrong because it extends to components of allocatable components, and "ultimate component" is wrong because it excludes components of derived type. What's needed is to resurrect the definition **direct component** from Fortran 95 – but done correctly. By "direct" the Fortran 95 standard clearly didn't mean "within a single type definition." Rather, it meant "the processor doesn't have to chase a pointer, even one disguised as allocatable."

The definition of "ultimate components" is defective by not carrying through its recursion correctly; indeed, since ultimate components are of intrinsic type, there is no hope for it to do so. That is an additional reason to resurrect "direct components." Having a correct definition for direct components, constructing a correct definition for ultimate components is trivial.

Should this be an interp?

2 Edits

Edits refer to 04-007. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by associated text, while a page and line number followed by + (-) indicates that associated text is to be inserted after (before) the indicated line. Remarks are noted in the margin, or appear between [and] in the text.

The **direct components** of an object of derived type are 44:24-26

(1) components declared in its type definition, and
(2) components declared in the type definitions of its nonpointer nonallocatable direct components of derived type.

The **ultimate components** of an object are its direct components that are of intrinsic type or that have the POINTER or ALLOCATABLE attribute.

C1107 (R1104) If a direct component (4.5) of a variable declared in the specification part of a module has component initialization, the variable shall have the SAVE attribute. 250:23-25

3 As an interp request

NUMBER: TBD
TITLE: Default initialization of module variables and SAVE, ultimate components
KEYWORDS: DEFAULT INITIALIZATION, MODULE VARIABLES, SAVE, ultimate components
DEFECT TYPE: Erratum
STATUS: TBD

QUESTION 1a:

Does the following program conform to the Fortran 2003 standard?

```
module M
  type :: T1
    integer :: A = 0
  end type T1
  type :: T2
    type(t1) :: B
  end type T2
  type(t2) :: X
```

```
end module M
```

QUESTION 1b:

Alternative question: Add the ALLOCATABLE attribute to the B component of type T2.

ASWER 1:

The answer depends upon which way one interprets C1107: either "type for which *component initialization* is specified" applies only to components declared in the type of the object (1a), or to components "at all levels of component selection" (1b).

ANSWER 1a:

Unfortunately, this program conforms, although it was intended that it not conform. The problem is that C1107 [250:23-25] is defective: it applies only to components declared within the object's type definition; therefore component initialization is not specified for type T2. Edits are provided to correct this defect.

ANSWER 1b:

Unfortunately, this program does not conform, although it was intended that it conform. The problem is that C1107 [250:23-25] is defective: it applies to components at all levels of component selection; therefore, component initialization is specified for type T2. Edits are provided to correct this defect.

QUESTION 2:

Does the following program conform to the Fortran 2003 standard?

```
program P
  type :: T1
    integer, pointer :: A1
  end type T1
  type :: T2
    type(t1) :: A2
    integer :: B = 2
  end type T2
  type :: T3
    type(t2) :: B2
    integer :: C = 3
  end type T3
  type(t3) :: X
  integer, target :: Y
  x%b2%a2%a1 => y
  print *, x
end program P
```

ANSWER 2:

Unfortunately, this program conforms, although it was intended that it not conform. The problem is the use of "ultimate components" in the second paragraph after NOTE 9.35 in subclause 9.5.2, together with a defective definition for "ultimate components". Edits are provided to correct this defect.

QUESTION 3:

Does the following program conform to the 2003 standard:

```
program E
  type :: T1
    sequence
    integer, allocatable ::  A1
  end type T1
  type :: T2
```

```
    sequence
    type(t1) :: A2
    integer :: B = 2
  end type T2
  type :: T3
    sequence
    type(t2) :: B2
    integer :: C = 3
  end type T3
  type(t3) :: X, Y
  equivalence ( X, Y )
end program E
```

ANSWER 3:

Unfortunately, this program conforms, although it was intended that it not conform. The problem is the use of "ultimate components" in C576 [96:7-8]. The subobjects X%B2%A2%A1 and Y%B2%A2%A1 are not ultimate components. If the A1 component of type T1 were declared within T2, the allocatable subobjects X%B2%A1 and Y%B2%A1 would be ultimate components of X and Y, and the program would not conform. Edits are provided to correct this defect.

QUESTION 4:

In the following program

```
program P
  type :: T1
    integer, allocatable :: A1
  end type T1
  type :: T2
    type(t1) :: A2
    integer :: B = 2
  end type T2
  type :: T3
    type(t2) :: B2
    integer :: C = 3
  end type T3
  type(t3), allocatable :: X
  allocate ( x )
end program P
```

what is the allocation status of X%B2%A2%A1 after execution of the allocate statement?

ANSWER 4:

Unfortunately, the standard does not provide an interpretation for this program. The last paragraph of 6.3.1.1 [113:20-21] probably intended that X%B2%A2%A1 is unallocated after execution of the allocate statement, but because the definition of "ultimate components" extends through only two levels of component selection, it provides no interpretation. Edits are provided to correct this defect.

QUESTION 5:

Does the following program conform to the 2003 standard?

```
program P
  type :: T1
    integer, allocatable :: A1
  end type T1
  type :: T2
```

```
    type(t1) :: A2
    integer :: B = 2
  end type T2
  type :: T3
    type(t2) :: B2
    integer :: C = 3
  end type T3
  type(t3) :: X
  namelist /out/ x
  write ( *, out )
end program P
```

ANSWER 5:

One might think that the program does not conform because because it violates the second paragraph of 9.5.3.6 **Namelist formatting** (allocatable components shall be processed by user-defined derived-type input/output subroutines). Unfortunately, it does not conform because the standard provides no interpretation for it (how does one output an unallocated variable?), because the definition of "ultimate components" extends through only two levels of component selection. Edits are provided to correct this defect.

QUESTION 6:

In the following program

```
program P
  type :: T1
    integer, pointer :: A1 => NULL()
  end type T1
  type :: T2
    type(t1) :: A2
    integer :: B = 2
  end type T2
  type :: T3
    type(t2) :: B2
    integer :: C = 3
  end type T3
  type(t3) :: X
  call sub ( x )
contains
  subroutine SUB ( Y )
    type(t3), intent(out) :: Y
  end subroutine SUB
end program P
```

what is the pointer association status of X%B2%A2%A1 after execution of the CALL SUB statement?

ANSWER 6:

The standard does not provide an interpretation because the definition of "ultimate components" extends through only two levels of component selection. Edits are provided to correct this defect.

QUESTION 7:

In the following program

```
program P
  type :: T1
    integer, pointer :: A1
```

```
  end type T1
  type :: T2
    type(t1) :: A2
    integer :: B = 2
  end type T2
  type :: T3
    type(t2) :: B2
    integer :: C = 3
  end type T3
  type(t3) :: X
  nullify ( x%b2%a2%a1 )
  call sub ( x )
contains
  subroutine SUB ( Y )
    type(t3), intent(out) :: Y
  end subroutine SUB
end program P
```

what is the pointer association status of X%B2%A2%A1 after execution of the CALL SUB statement?

ANSWER 7:

X%B2%A2%A1 remains disassociated because the definition of "ultimate components" extends through only two levels of component selection. It is clear, however, that the intent is that the pointer association status of X%B2%A2%A1 becomes undefined. Edits are provided to correct this defect.

EDITS:

The definition of "ultimate components" cannot be corrected in its current form because recursion via derived types is required, while the fundamental property of "ultimate components" is that they are of intrinsic type. As a result, the definition of "ultimate components" encompasses only two levels of derived-type components. Instead, the definition of "direct components" is resurrected from Fortran 95, after having been removed in Fortran 2003, (and corrected because it was defective in Fortran 95) and that definition used to correct both problems.

The **direct components** of an object of derived type are 44:24-26

(1) components declared in its type definition, and

(2) components declared in the type definitions of its nonpointer nonallocatable direct components of derived type.

The **ultimate components** of an object are its direct components that are of intrinsic type or that have the POINTER or ALLOCATABLE attribute.

C1107 (R1104) If a direct component (4.5) of a variable declared in the specification part of a module has component initialization, the variable shall have the SAVE attribute. 250:23-25

SUBMITTED BY: Van Snyder

HISTORY: