Generic Programming

John Reid, JKR Associates and

Rutherford Appleton Laboratory

BCS Fortran Specialist Group

London, 28 September 2023

Abstract

WG5 has accepted a proposal from the Japanese National Body for extending the generic capability that has been present in Fortran since Fortran 90.

This will complement a much more ambitious US proposal that has been developed by a J3 subgroup since the 2019 WG5 meeting in Japan.

This talk will explain the Japanese proposal and give a very brief summary of the aims of the US proposal.

Generic procedures in Fortran 2023

Several versions of a procedure, differing in the type, kind, or rank of one or more arguments, may be wanted.

Can write each version separately with different names but merge them into one procedure with a single name.

Fortran 2023 example

```
    interface action
       procedure real_action, double_action
    end interface
contains
    subroutine real_action (a)
        real :: a
        :
    end subroutine
    subroutine double_action (a)
        double precision :: a
        :
     end subroutine
```

Generic procedures in Fortran 202y

- Declare the procedure as generic.
- It has no specific names (like many intrinsics).
- Declare alternative types, kinds, and ranks for some arguments.
- Use “meta select” blocks for code that varies between versions.
- The compiler needs to generate specific versions for all those actually invoked.

Fortran 202y example

Here is the equivalent of the code on slide 4

```
generic subroutine action (a)
    type(real, double precision)  :: a
    :
end subroutine
```

meta type construct

For code that depends on the type there is the meta type construct

```
meta select type (a)
   meta  type (real)
   :  ! code for real version
   meta  type (double precision)
   :  ! code for double-precision  version
end meta select
```

Multiple generic arguments

```
generic subroutine real_action (a, b)
type(real, double precision)  :: a , b
```

requires that in a call a and b have the same type so two versions are created.

```
generic subroutine real_action (a, b)
type(real, double precision)  ::  a
type(real, double precision)  ::  b
```

does not require that in a call a and b have the same type so four versions are created.

typeof

typeof statements can be used to copy the type and kind of a generic argument, for example

```
generic subroutine real_action (a, b)
   type(real, double precision)  :: a
   typeof (a) :: b
   typeof (a) :: c ! Local variable
```

Generic rank

Alternative ranks for a dummy argument that is assumed shape, allocatable, or a pointer may be declared with the rank attribute (new in Fortran 2023) having multiple values, for example

```
generic subroutine action (a, b)
type(real), rank(1:3)  :: a
type(real), rank(2,4), allocatable  :: b
```

rankof

The rankof clause is added to allow the rank of a generic argument to be copied, for example

```
generic subroutine real_action (a, b)
      type(real), rank(1:3)  :: a
      type(real), rankof (a) :: b
      type(real), allocatable, rankof (a) :: c ! Local variable
```

Meta select rank

For code that depends on the rank the meta select rank construct is available

```
meta select rank (a)
    meta rank (0)
    :  ! code for scalar version
   meta rank (1)
    :  ! code for rank-1  version
end meta select
```

Summary

- Declare the procedure as generic.
- No specific names.
- Declare alternative types, kinds, and ranks for arguments.
- Use meta select blocks for code that varies between versions.
- The compiler generates specific versions for those actually invoked.
- Compiler checks all versions and provides diagnostics against the user’s code.
- Not hugely difficult to implement.

J3 generics proposal

This uses the term template to refer to a generic entity that has dummy template parameters. Each wanted version must be instantiated from the template by specifying actual template parameters. A template parameter may be a type, value, or procedure. The association is like that of a procedure argument.

A template can define derived types, procedures, interfaces, variables, other templates, constants, or enumeration types.

Restrictions express relationships among template parameters. Templates may contain requires statements to express their requirements.

They aim to support containers such as a list, vector, dictionary, set, stack, or queue.

J3 generics: a personal view

I confess to not fully understanding the proposal but it appears to me to be too complicated.

It is also far from complete.

I had hoped to show you how to create a generic procedure comparable to that available from the Japanese proposal, but I have failed to see how to do this.

Addendum, 10 Oct 2023

The latest J3 generics paper, 23-222, shows that the intention is to allow the instantiate statement to rename entities accessed from a template. This allows me to construct code, see next slide, comparable to that in slide 6:

```
generic subroutine action (a)
    type(real, double precision)  :: a
    :
end subroutine
```

Fortran 202y example

equivalent of the code on slide 6

```
template actions(t)
   type, deferred :: t
contains
   subroutine action (a)
     type(t)  :: a
     :
   end subroutine
end template actions

   interface action
      procedure real_action, double_action
   end interface
contains
   instantiate actions (real), real_action => action
   instantiate actions (double precision), double_action => action
```