As in N1434

```
module FOO_M

  interface FOO
    module procedure REAL_FOO, DOUBLE_FOO
  end interface

  submodule :: REAL_SUB_FOO
    subroutine REAL_FOO ( A, B )
      real :: A, B
    end subroutine REAL_FOO
  submodule :: DOUBLE_SUB_FOO
    subroutine DOUBLE_FOO ( A, B )
      double precision :: A, B
    end subroutine DOUBLE_FOO

end module FOO_M
```

Notes on N1434 case

- The generic interface and the specific interfaces are created separately. This increases development and maintenance costs.

- One is *required* to specify the submodule in which the body of a procedure is to be found. This may be desirable in some cases, to help the human reader or for the compiler to double-check your layout. It may be an undesirable restriction on flexibility in other cases.

As in 01-371

```
module FOO_M

  interface FOO
    submodule subroutine REAL_FOO ( A, B )
      real :: A, B
    end subroutine REAL_FOO
    submodule(double_sub_foo) subroutine &
      & DOUBLE_FOO ( A, B )
      double precision :: A, B
    end subroutine DOUBLE_FOO
  end interface

end module FOO_M
```

- The "submodule" prefix on a procedure header in an interface block indicates its body is in a submodule.

- It is optional whether one specifies the submodule in which the body is to be found.

In either case

```
submodule(foo_m) REAL_SUB_FOO

contains

  ! The "submodule" prefix here indicates
  ! it's a continuation, with interface in
  ! FOO_M or one of its ancestors (FOO_M
  ! doesn't have ancestors in this case,
  ! because it's a module).

  submodule subroutine REAL_FOO ! ( A, B )
  ! real :: A, B
  ...
  end subroutine REAL_FOO
end submodule REAL_SUB_FOO
```

Notice that the interface is *not* respecified. Can you think of a graceful way to allow it?