Miscellaneous Fortran features

Character strings

```
 program test
  character :: a='A'
  character (len=3) :: b='xyz'
  character (len=6) :: c,d,e
  c = b//a//'qq'  ! catenation
  d = c
  d(1:2)='--'
  e='X'
  write(6,*) a,b,c,d,e
end program
```

Output of the program

```
A
xyz
xyzAqq
--zAqq
X
```

Additional control structures

case statement

```
select case (n)
case (0)
 x=0.0
case (1:5)
 x=1.0
end case
```

```
character code

select case (code)
case ('a')
 x=0.0
case ('b':'z')
 x=1.0
case default
 x=2.0
end case
```

Selection must be unambiguous.

If none of the cases corresponds to the selector, nothing is done.

The selector (n) can only be an `integer`, `character` or `logical`.

Jump statement

`goto` or `go to`:

```
      if (x > 0) goto 100
      ...

   100 continue
```

A label is a string of 1–5 digits.

The labelled statement should be an empty statement `continue` (not necessary but safer).

Usually goto statements are not needed.

Named statements

```
toobig: if (x > 1000) then
        ..
        end if toobig
```

The name is a kind of a comment. It is not a label where control can be transferred from elsewhere.

```
iter: do while ( .. )
      if (..) exit iter

      end do iter
```

```
outer: do i=1,100
inner: do j=1,100

      if (..) exit outer

      end do inner
      ...
      end do outer
```

Own data types

Cf. `record` in Pascal and `struct` in C.

Declaration of own types:

```
type star
 real :: ra, dec, magnitude
 character (len=20) :: name
end type
```

Declaration of a variable of the type `star`:

```
type (star) :: stella, s1, s2
type (star), dimension(10000) :: catalogue
```

Components can be accessed using the operator %:

```
stella%magnitude = 15.2
r= catalogue(i)%ra
```

Own types can appear in assignments:

```
s1 = s2
s2 = catalogue(i)
stella=(0.45199, -29.299, 8.80, 'HD2347')
```

Other operators can be defined, too (discussed later).

Dynamic allocation of variables

The required size of an array may depend on the input data.

If a procedure needs an auxiliary array, its size may not be known before the procedure is called.

Fortran 77:
- allocate a big table that is sufficient even in the worst case, or
- the workspace is given as a parameter by the calling program

Fortran90: arrays can be allocated dynamically during execution.

An array in a procedure is allocated when the procedure is invoked. The size can be a variable:

```
subroutine zz(x,n)
integer, intent(in) :: n
real, dimension (n) :: x
  ...
real, dimension(n,n) :: matrix
  ...
```

If the size of an array can be determined only during execution (e.g. in the main program), it can be allocated dynamically:

```
real, allocatable, dimension (:,:) :: matrix
  ...
read(*,*) n
allocate(matrix(n,n))
  ...
```

The function `allocated` can be used to check if memory has been allocated for an array:

```
if (.not. allocated(matrix)) &
  allocate(matrix(1:10, -10:10))
```

Local variables of a procedure vanish when the procecure ends. The allocated memory can also be released explicitly:

```
deallocate(matrix)
```