The complete form of a variable declaration is

```
type (parameters), attributes :: name
```

Parameters define the representation of the variable. Attributes define array sizes and other information related to memory allocation.

Since the compiler must accept also declarations written in the old f77 style, the following forms are equivalent:

```
real x
real :: x
```

Also the following ones:

```
real x(10)
real :: x(10)
real, dimension(10) :: x
```

If a variable is given an initial value in the declaration, only the form with the colons is possible:

```
real :: a=1.0
real, dimension(10) :: x = 0.0
real, dimension(0:2) :: d=(/ 1.0, 2.0, 5.0 /)
```

Parameters of variables

The `kind` parameter is an integer that is used to specify which one of the representations supported by the hardware will be used for storing the variable. Since the representations are machine dependent, the actual values are not defined in the standard and may be different in different environments. In principle it would be possible to say e.g. `kind=1`, but this is not a good idea since it would affect the portability of the program.

Intrinsic functions should be used to select a suitable value for the kind parameter:

```
integer (kind=selected_int_kind(5)) :: lkm
real (kind=selected_int_kind(5)) :: x
real (kind=selected_int_kind(5, 30)) :: y
```

In the first case, at most 5 digits are needed to express the integer. In the two others there must be space at least for 5 decimals, and in the last one the range must be at least 10^{-30}–10^{30}.

Attributes of variables

The following list contains all available attributes.

`dimension`: array size

`parameter`: constant that cannot be altered

`save`: memory allocated statically; value remains between procedure invocation.s

`allocatable`: array to be allocated dynaamically

`pointer`: pointer to a variable

`target`: variable that may be pointed by a pointer

`intent`: usage of a procedure argument

`optional`: optional argument; may be missing in the procedure call

`external`: the variable is a name of a procedure

`intrinsic`: the variable is a name of an intrinsic function

`private`: private variable of a module

`public`: public variable of a module

Warning concerning local variables

If a local variable of a procedure has the `save` attribute, its value will not disappear when the procedure end but remains till the next invocation.

If a local variable is initialized in the declaration it has automatically the save property. It is NOT reinitialized every time the procedure is invoked.

```
program savetest
  write(*,*) f(1.0)
  write(*,*) f(1.0)
contains
real function f(x)
  real, intent(in) :: x
  real :: y=0.0
  y=y+1
  f=x+y
end function
end program

2.000000
3.000000
```

If such a side effect is not wanted, the variable must be initialixed with an assignment statement:

```
real function f(x)
  real, intent(in) :: x
  real :: y
  y=0.0
  ...
```