First steps in Fortran - Introduction

Every program in Fortran has to be enclosed be enclosed by starting line with keyword `program` followed by a unique name. This name cannot be a keyword from the Fortran language. Other than that there is no restriction.

```
program MyProgramName
```

Last line has to end the program with keywords `end program` followed by the name of the program

```
end program MyProgramName
```

The simplest 'hello world' type program in Fortran may look like this

```
program hello
  print *,'Hello world from Fortran'
end program hello
```

In this example we have used function `print` which displays strings and variables on the standard output (in our case on the screen).

To get this first example working we need to create a file `hello.f90` and put this simple program in that file. After that we need to compile this code using a Fortran language compiler. We will use `gfortran` from GCC package http://gcc.gnu.org

```
$ vim hello.f90
$ gfortran hello.f90 -o hello.x
$ ./hello.x
Hello world from Fortran
```

Variables

All variables need to be defined at the top of the proram/function/subroutine and not as when needed (like in C/C++)

```
program variables

  integer :: a
  real :: b
  logical :: c

  a = 1
  b = 3.1415
  c = .false.

  print *,' Integer a ', a
  print *,' Real b ', b
  pritn *,' Logical c ', c

end program variables
```

the output of this short program will be

```
  Integer a            1
  Real b    3.14150000
  Logical c  F
```

Conditional

An `if` conditional statement for flow control and variable or expression checks looks like this

```
program conditional

  integer :: val

  val = .true.

  if ( val ) then
    print *,' True value '
  else
    print *,' False value '
  endif

end program conditional
```

Loops

Fortran has several loop - form constructs, however, `do` loop is the most commonly used and can emulate any other loop together with conditional `if`

```
program loop

  integer :: max_value = 10
  integer :: i

  do i=1,max_value
    print *,'Iteration i=', i
  enddo

end program loop
```

Functions

A subprogram which returns a value (only one)

```
program simple_function

  implicit none

  integer :: val = -4
  logical :: res = .true.

  res = sign_test(val)

  print *,'Is the value positive ? : ', res

  contains

  logical function sign_test(input) result(output)

    integer :: input

    if( input >= 0 ) then
      output = .true.
    else
      output = .false.
    endif

  end function sign_test

end program simple_function
```

Subroutines

A subprogram which does not return a value but may modify more than one variables passed to it as argument

```
program simple_subroutine

  implicit none
  integer :: i,j

  i = 1
  j = 2

  call print_pair(i,j)

end program simple_subroutine

subroutine print_pair(a,b)

  integer, intent(in) :: a,b

  print *,'Pair of numbers : (',a,',',b,')'

end subroutine print_pair
```

Few more informations

- Fortran language is case insensitive. This means that variables and function/subroutines names can be a mixture of upper and lower case characters and compiler will convert them to uniform set. This means that names `Variable`, `VARIABLE` and `variable` are all the same in Fortran.
- Fortran variables and function/subroutines names can contain white spaces and compiler will remove them. This means that `integer :: myvariable` is the same as `integer :: my variable`.
- For large programs where source code is split between many files the order of compilation may be important. This is crucial when user define a `module` which is used by other functions and subroutines in other source code file. The `module` has to be present (compiled before) during the compilation of function/subroutine that wants to use it.