START PROGRAMMING IN FORTRAN

J-C. Soetens

University of Bordeaux / 2021-2022

If you want to know more :

http://www.idris.fr/formations/fortran/

First simple program in Fortran

Name of the file : ex1.f

```
1234567890123456789012345…                                                    72
       program example_1
c      my first program
       write(*,*) ''Hello world''
       end
```

One instruction per line.
Upper and lower case are not significant, blank lines and spaces are not significant.

To create an executable, you need to compile the code :
your prompt > ifort –o ex1 ex1 .f

To execute the program, submit the new command to the system :
your prompt > ex1
or
your prompt > ./ex1
if your *home directory* is not in your *path* (variable of the unix system).

Operators

**	raise to the power of
*	multiplication
/	division
+	addition
-	subtraction

Types of variables

It is recommended to impose the declaration of all variables : *implicit none*.
Types : integer, double precision, logical, character.

Name of the file : ex2.f

```
      program example_2
      implicit none

      double precision TC, TK, Kfactor
      parameter(Kfactor = 273.15d0)

c     input
      write(*,*) ''Enter the temperature in C :''
      read(*,*) TC
      if (TC .lt. (-Kfactor)) then
        write(*,*) ''this temperature do not exist''
        STOP
      end if
c     algorithm
      TK = TC + Kfactor
c     output
      write(*,*) ''The temperature in Kelvin is : '', TK
c     output using a format
      write(*,''(The Temperature in Kelvin is :'', F8.2)'') TK

      end
```

Intrinsic functions

Some functions are so important that they are provided as part of the language.
As we will systematically use real variables in double precision, the needed intrinsic functions name will start with the letter 'D'.

Examples :

`DABS(X)` absolute value of any X

`DCOS(X)` cosine of argument in radians
`DSIN(X)` sine of argument in radians
`DTAN(X)` tangent of argument in radians

`DACOS(X)` inverse cosine in the range (0,ð) in radians
`DASIN(X)` inverse sine in the range (-ð/2,ð/2) in radians
`DATAN(X)` inverse tangent in the range (-ð/2,ð/2) in radians

`DEXP(X)` exponential function
`DLOG(X)` natural logarithm: if W is real it must be positive,
`DLOG10(X)` logarithm to base 10
`DSQRT(X)` square root function

Logical controls

The `if` statement is the way of changing what happens in a program according to a condition.

Syntax :

```
      if (logical expression)then
c       instructions(s) in case the logical expression is true
        …
      else
c       instruction(s) in case the logical expression is false
        …
      end if
```

Main operators :
`.lt.` less than
`.le.` less than or equal
`.eq.` equal
`.ge.` greater than or equal
`.gt.` greater than
`.ne.` not equal
`.not.` not
`.and.` and
`.or.` inclusive or

Loops

In case you need to repeat a set of instructions, different ways exist to do this repetition.

Case 1, you know the number of repetitions : `do loop`.

```
     do variable  = start, stop [,step]
c      instructions to do
       …
     end do
```

with :

variable	is an integer variable
start	is the initial value var is given
stop	is the final value
step	optional, is the increment by which var is changed

Case 2, you do not know the number of repetitions : `do while`

```
      do while (logical expression)
c       instructions
        …
      end do
```

WARNING ! Possibly an infinite loop if the logical expression is always true (never false).

Arrays

Important in scientific programming : manipulation of vectors and matrices.

Examples of declarations :

```
      integer V(10)
      integer M(10,10)
      double precision N(2,50)
```

V is a vector of 10 (integer) elements.
V(1) is a pointer to the first element, V(10) is a pointer to the tenth element.

M is a table of 10 x 10=100 (integer) elements.
In the memory of the computer, it is a stack of elements with the first argument running in first, so the order is M(1,1), M(2,1)…M(10,1), M(1,2), M(2,2)…M(10,2) …M(8,10), M(9,10), M(10,10).

N is a table of 2 x 50=100 (double precision) elements.

Name of the file : ex3.f

```
      program example_3
      implicit none

      double precision M(10,10)
      integer i, j

c     initialize all the elements of the table M to zero
      do j=1, 10
        do i=1, 10

          M(i,j) = 0.0d0

        end do
      end do

c     use of table M to to something…

      end
```

Functions

As for intrinsic functions, you can define our own functions for use in a program. This is a very powerful feature because it allows to writte code once while it can be used many times. Such own functions can be programmed and tested separately, even build a library.

Name of the file : ex4.f

```
      program example_4
      implicit none

      double precision V1(3), V2(3)
      double precision scalar_product
      integer i

c     initialize the 3-D vectors V1 and V2
      do i=1, 3
        read(*,*) V1(i), V2(i)
      end do

      write(*,*) scalar_product(3, V1, V2)

      end
```

Name of the file : sp.f

```
      function scalar_product(n, A, B)
      implicit none

       integer N
       double precision A(N), B(N)
      double precision scalar_product
      integer i

      scalar_product = 0.0d0
      do i=1, N
        scalar_product = scalar_product + A(i)*B(i)
      end do

      end
```

To test the function scalar_product solely (so only the Fortran syntax) :

your prompt > ifort –c sp.f

To build an executable with the two files :

your prompt > ifort –o ex4 ex4.f sp.f

Subroutines

Subroutines are very similar to functions but they do not return a value.

Example of code of a subroutine :

```
c       subroutine MY_FIRST_SUBROUTINE(argI, argJ,…,argZ)
        implicit none

c       declarations of arguments
        …

c       work to do
        …

        end
```

Somewhere in a program

```
c       programm example_5
        implicit none

c       declarations
        …

c       core of the program
        …
        call MY_FIRST_SUBROUTINE(arg1, arg2,…,argN)
        …

        end
```

Many things to explain !
List of arguments, declarations, local and global variables, etc.