Sub-programs (function and subroutine)

1. introduction

If a program is too long, it will become more complex to write, read (understand) and maintain (change or correct errors).

Therefore, it is useful to divide the program into a group of programming units called partial programs, which allows the solution to be simplified from a large, complex program to a group of simple partial programs that help reach the solution. This makes it much easier to write it, read it (understand it) and maintain it (change or correct errors). In addition, developing the program in software modules allows the work to be distributed among several programmers, each of whom will be responsible for developing one or more software modules. Also, a software module can be used multiple times within the same program or in other programs; Which has a significant impact on the reusability of the code.

اذا كان البرنامج طويلا جدا ، فسوف يزيد تعقيدا في كتابته، قراءته (فهمه) وكذلك صيانته (تغيير أو تصحيح الأخطاء)

لذلك فمن المفيد تقسيم البرنامج إلى مجموعة من الوحدات البرمجية الصغيرة تسمى برامج جزئية مما يسمح بتبسيط الحل من برتامج كبير معقد الى مجموعة من البرامج الجزئية البسيطة تهدف للوصول الى الحل. وهذا يجعل الأمر أسهل إلى حد كبير في كتابته، قراءته (فهمه) وكذلك صيانته(تغيير أو تصحيح الأخطاء) . بالإضافة إلى ذلك، تطوير البرنامج في وحدات برمجية يسمح بتوزيع العمل بين عدة مبرمجين، سيكون كل منهم مسؤولاً عن تطوير وحدة برمجية أو أكثر . و يمكن أيضا استخدام الوحدة البرمجية عدة مرات داخل البرنامج الأساسي أو في برامج أخرى.

2. Sub-programs

A specific piece of processing consisting of a block of instructions (more than two instructions It can be played multiple times in different locations in the main program. With a purpose to avoid repeating the same commands several times in the program, we define them only once using a partial program. so, every time The programmer needs to execute the defined set of instructions he calls the Sub program by typing its name. Sub programs are divided into 2 types: **functions** and procedures (subroutine) .

2. البرامج الجزئية

جزء محدد من المعالجة يتكون من كتلة من التعليمات (أكثر من تعليمتين) ويمكن تشغيلها عدة مرات في مواقع مختلفة في البرنامج الرئيسي. لغرض تجنب تكرار نفس الأوامر عدة مرات في البرنامج، نقوم بتعريفها مرة واحدة فقط باستخدام برنامج جزئي، و في كل مرة يحتاج المبرمج إلى تنفيذ مجموعة محددة من التعليمات يقوم باستدعاء البرنامج الجزئي عن طريق كتابة اسمه. تنقسم البرامج الجزئية إلى نوعين: الدوال والإجراءات .

2.1. function: subprogram resolve a problem witch it one value at result like the result of addition ,product, subtraction … of three integer numbers (input : 3

numbers, **output : 1 number**(result)) or subprogram calculate factorial, power ,… (output : **1 number**(result)) so we can used **function**

The general syntax for declared a **functions** is:

```
type function name-fct (<arg 1>,…,<arg n>) result(arg r)
implicit none
// déclaration of arguments <arg1 >…<argn>
// déclaration of local variables
// Set of instructions

end function name-fct
```

In the declaration of function we specified by **type** function the type of result argument *<arg r > returned.*

Example
This example resolve by Fortran function f : **R3→ R** defined : f(a, b, c) = a + b * c

```
real function f (a, b, c) result(r)
implicit none
  real :: a, b, c
  r = a + b*c
end function f
```

The calling of function by main program by its name with his arguments value.

Example : p= f(4,6,2) , p=f(a,b,c) , p= f(4,6,2) + f(a,b,c)….// a,b,c contains real values

2.2 **Subroutine:** subprogram resolve a problem witch haven't a value at result returned like the result of filling array ,display arrays (**output : array filling or displaying**)

The general syntax for declared a **Subroutine** is:

```
Subroutine name-Sub (<arg 1>,…,<arg n>)
   implicit none
 // déclaration of arguments <arg1 >…<argn>
 // déclaration of local variables
 // Set of instructions
end Subroutine name-Sub
```

Example : subroutine for filling array of 10 integers elements and displaying

```
Subroutine filling (t,n)
implicit none
integer  :: t(20), n  // argument declaration
integer ::   i          // Local variable declaration

Do i=1,n
  read*, t(i)
end Do

end Subroutine filling

Subroutine Display (t1,n)
implicit none
integer  :: t1(20), n  // argument declaration
integer ::   i          // Local variable declaration

Do i=1,n
  print*, t1(i)
end Do

end Subroutine Display
```

For calling a *subroutine* by main program we use the keyword **call**.

Example : **call** filling(A ,10) , **call** filling(B ,n) , **call** Display(B,5)

witch : A, B are Arrays of 20 elements and **n** integer value indicate length of array

2. Fortran program structure contains sub-programs:

```
program name-main-program
implicit none
// déclaration global variables
// set of instructions  ( we  can calling subroutines and functions declared after
contains keyword  )
contains
Subroutines  name-Sub1 (<arg 1>,…,<arg n>)
 implicit none
// déclaration of arguments <arg1 >…<argn>
// déclaration of local variables
// Set of instructions
end  Subroutine name-Sub
```

Subroutines *name-Sub2(<arg 1>,…,<arg n>)*

….

type ***function*** *name-fct 1 (<arg 1>,…,<arg n>)* ***result****(arg r)*

implicit none

// déclaration of arguments <arg1 >…<argn>

// déclaration of local variables

// Set of instructions

end function *name-fct1*

type ***function*** *name-fct 2 (<arg 1>,…,<arg n>)* ***result****(arg r)*

…..

end program *name-main-program*

For using function or calling subroutine by main program it must declared in **contains** part of the program so if we have more then program witch need using or calling same subprogram program it must declared in **contains** part for any one (redundancy code)

لاستخدام دالة أو استدعاء إجراء معين بواسطة برنامج رئيسي ، يجب أن يتم تعريفه ضمن البرنامج الذي يقوم باستدعائه ، لذلك إذا كان لدينا مجموعة من البرنامج الرئيسية التي تحتاج إلى استخدام أو استدعاء نفس البرامج الجزئية ، فيجب تعريفها في كل هذه البرنامج الرئيسية (تكرار البرامج الجزئية)

For ameliorate the visibility of code, we can declared subprograms at separed files called ***module***.

لتحسين استغلال البرامج الجزئية و تفادي تكرارها ، نستطيع الإعلان عنها في ملفات برمجية منفصلة تسمى ***module*** ويمكن لمختلف البرامج الرئيسية استغلالها بكتابة اسم ***module*** الذي تنتمي إليه

The module is detached file contains :
A subroutines, and functions declaration, witch can used for one or more different main programs.
Fortran general structure module :

module *name-module*
implicit none
// declaration of local variables
// déclaration subroutines
// déclaration fonctions
end module *name-module*

Example

```
program Array
use calc
implicit none
integer :: i,  j, t(20), s,n
```

```
print*,"Enter Array dimension")
read*,n
call filling( t,n )
s=sum( t,n )
call Display (t,n )
print*, "sum array=" , s
end program matrice
```

The only change in this code is the addition of the **use** statement indicating that the Array program uses subprograms declared in the module named **calc**.

Here is the code for the calc.f95 file:

```
   module calc
   implicit none
   contains
   Subroutine filling (t,n)
   implicit none
   integer  :: t(20), n
   integer ::   i
   Do i=1,n
     read*, t(i)
   end Do
   end Subroutine filling

   Subroutine Display (t1,n)
   implicit none
   integer  :: t1(20), n
   integer ::   i
   Do i=1,n
     print*, t1(i)
   end Do
   end Subroutine Display

integer function sum(t,m) result(s)
implicit none
   integer  :: t1(20), n
   integer ::   i
   s=0
   Do i=1,n
     s=s+ t1(i)
   end Do
end function sum
end module calc
```