# Chapter Two
# Control Constructs In FORTRAN 90

## المقدمـــــة Introduction

لقد كان يعرف مثل هذا الفصل في نماذج Fortran القديمة مثل Fortran 77 بأسم عبارات السيطرة وكانت تشمل عبارات الأنتقال ( Goto Statements ) وعبارات الأنتقاء ( IF Statements ) ٠

أما النموذج الأخير Fortran 90 فقد دخل عالم اللغات المتقدمة والتي تعرف باللغات البنيوية المبنية من وحدات منفصلة Structured programming وبهذا التغيير الجذري للغة أصبح البرنامج عبارة عن وحدات بنائية تتكون كل وحدة من من تركيبة تسمى block ٠ وتقسم هذه التركيبات الى تركيبة الأنتقاء IF construct وتركيبة الأختيار CASE construct ويفضل أقران أي تركيبة بأسم وهذا الأسم يخضع لقواعد أسماء المتغيرات في Fortran 90 من حيث الشكل وفي العادة تقترن بداية التركيبة بهذا الأسم كما يظهر في نهاية التركيبة تماماً كما في أسم البرنامج ٠

## تركيبة الأنتقاء الشرطية ( IF Construct )

هناك ثلاثة أنواع من تركيبات الانتقاء في لغة Fortran 90 هي:

### 1٠ تركيبة ( IF…THEN Construct ):

وهي أبسط شكل لتركيبة الأنتقاء والصيغة العامة لها هي:

```
IF (Logical expression) THEN
  .....
  .....   } block of statements عبارات فورتران
  .....
END IF
```

T

وتعني هذه التركيبة إذا كان التعبير المنطقي صواب True أي كان الشرط متحققاً فأن مجموعة عبارات فورتران يتم تنفيذها وإذا لم يتحقق الشرط فأن العبارات لا تنفذ٠ وفي أي من الحالتين فإن العبارة التالية التي تنفذ هي العبارة التي تلي عبارة END IF ٠

مثال: أكتب برنامج لإيجاد القيمة الصغرى لقيمتين معلومتين ؟

الحل :

```
PROGRAM  SMALLER_VALUE
  IMPLICIT NONE
REAL :: MIN,A,B
PRINT *, "Enter The Values of A and B"
READ *, A , B
     MIN = A
     IF(MIN > B) THEN
    MIN = B
     END IF
PRINT "( 5X,A,F8.3)", "THE MINIMUM VALUE  = ",MIN
END PROGRAM  SMALLER_VALUE
```

## 2. تركيبة ( IF … THEN…ELSE ) :

وهي الشكل الأكثر تطوراً لهذه التركيبة هو أنتقاء أحد خيارين والصيغة العامة لهذه لها هي:

```
IF (Logical expression) THEN
  .......
  .......   } block of statements 1  عبارات فورتران
  .......
                      T
ELSE
  .......
  .......   } block of statements  عبارات فورتران
  .......
                      2
END IF
```

وتعني هذه التركيبة إذا كان التعبير المنطقي متحققاً فأن مجموعة عبارات فورتران T يتم تنفيذها وتهمل مجموعة عبارات فورتران F أما أذا لم يكن الشرط متحققاً فيتم تنفيذ مجموعة عبارات فورتران F وتهمل المجموعة T وفي أي من الحالتين فأن العبارة التالية في التنفيذ هي العبارة التي تلي عبارة END IF.

مثال: أكتب برنامج لإيجاد Y بموجب ما يلي :

$$Y = \begin{cases} x & x \geq 0 \\ -x & x < 0 \end{cases}$$

الحل :

```
PROGRAM VALUE_OF_Y
    IMPLICIT NONE
      REAL  :: X,Y
 PRINT *, "Enter The Value of X"
READ*, X
   IF(X > = 0.0) THEN
    Y=X
  ELSE
 Y=- X
  END IF
PRINT "(T10,A,F6.2)","The Value of Y =",Y
END PROGRAM VALUE_OF_Y
```

## 3. تركيبة IF العامة ( General form of IF Construct ) :

وهي تركيبة الأنتقاء العامة والصيغة العامة لهذه العبارة هي:

```
IF (Logical expression) THEN
  .....
  .....   } block of statements 1  عبارات فورتران
  .....                  T1
 ELSE IF (Logical expression) THEN
  .....
  .....   } block of statements 2  عبارات فورتران
  .....                  T2
 ELSE IF (Logical expression) THEN
  .....
  .....   } block of statements 3  عبارات فورتران
  .....                  T3
 ELSE
  .....
  .....   } block of statements  عبارات فورتران
  .....                  4
 END IF
```

وتعني هذه التركيبة بأن نختبر التعابير المنطقية من البداية وحسب ظهورها في البرنامج فإذا تحقق أي شرط تنفذ العبارات T التي تليه وإذا لم يتحقق أي من هذه الشروط فأن مجموعة العبارات F تنفذ وعلى أي حال فبعد تنفيذ أي مجموعة من العبارات ( سواء T أو F ) فأن العبارة التالية التي تنفذ هي تلك العبارة التنفيذية التي تلي END IF مباشرة .

مثال : أكتب برنامجاً لإيجاد Y بموجب ما يلي :

$$Y = \begin{cases} 1 & X < 0 \\ 4X^2 + 1 & 0 \le X \le 1 \\ 5X^2 + 2X + 6 & X > 1 \end{cases}$$

الحل :

```
PROGRAM VALUE_OF_Y
IMPLICIT NONE
REAL  :: X,Y
PRINT *, "Enter The Value of X"
READ*, X
IF(X < 0.0) THEN
Y=1.0
ELSE IF (X <= 1.0)THEN
Y=4*X**2+1
ELSE
Y=5*X**2+2*X+6
END IF
PRINT "(5x,A, F7.3,2x,A, F6.2)", "X =",X, "Y=",Y
END PROGRAM VALUE_OF_Y
```

## أساليب برمجة قديمة

بما أن لغة **Fortran** قديمة لذا كان من الضروري الاستفادة من الحجم الهائل للبرامج التي سبق كتابتها والتي تحتويها مكتبات الكمبيوتر ، ولذلك فأن مترجمات ( **Compilers** ) لغة **Fortran** الحديثة لا زالت تسمح باستخدام أساليب البرمجة القديمة ، ولكن ينصح المبرمجين القدماء بالتخلص من تلك الأساليب واستخدام أساليب **Fortran** الحديثة لما فيها من المرونة والسهولة وجمال الشكل ،

وفيما يلي بعض التفصيل لأساليب البرمجة القديمة وكيفية التعويض عنها بالأساليب الحديثة :

### عبارات الانتقال ( GO TO Statements )

هناك عدة أنواع من عبارات الانتقال من مكان إلى آخر في برنامج فورتران أهمها:

### 1. عبارة الانتقال غير المشروط ( Unconditional GO TO Statement ) :

أن أبسط طرق تغيير تسلسل تنفيذ عبارات البرامج هو باستعمال عبارة الانتقال غير المشروط والصيغة العامة لهذه العبارة هي:

**GO TO n**

حيث **n** : تمثل رقم العبارة المراد تنفيذها وعبارة **GO TO** هذه تعطي أمر للحاسب ليذهب مباشرة ودون أي شرط لينفذ العبارة رقم **n** ، وكمثال على عبارة الانتقال غير المشروط : **GO TO 30**

ملاحظة مهمة : لقد أصبحت جملة GO TO من أكثر العيوب ظهوراً في اللغات القديمة وعلى الرغم من أن معظم اللغات الحديثة لا زالت تحوي هذه العبارة فإن المبرمج ينصح بتجنبها قدر المستطاع ، وأن هذه الجملة يمكن الاستغناء عنها بأحد التركيبات التي تحويها اللغات الحديثة مثل EXIT , CYCLE , SELECT CASE .

## 2. عبارة الانتقال المشروطة ( Conditional GO TO Statement ):

وتعرف أيضاً بعبارة GO TO الحسابية (Computed Go To Statement) والصورة العامة لهذه العبارة هي:

**GO TO ($n_1$, $n_2$, $n_3$,..., $n_m$ ),I**

حيث :

$n_1$ , $n_2$ , $n_3$..., $n_m$ : هي أعداد صحيحة بدون أشارة تمثل أرقام عبارات تنفيذية.

I : يمثل متغير صحيح يشير إلى العبارة المراد تحويل التنفيذ اليها وتتراوح قيمته من 1 ⟵ m .

وتعمل هذه العبارة بعد أختيار قيمة المتغير الصحيح I بتحويل التنفيذ إلى رقم العبارة المناظرة له $n_I$ بمعنى أنه أذا كانت I = 1 فأن التحويل سينتقل للعبارة رقم $n_1$ وإذا كانت I = 2 فأن التحويل سينتقل للعبارة رقم $n_2$ وهكذا.

ومثال على هذه العبارة GO TO( 10 , 20 , 30 ) , N

هذا المثال يعني : إذا كان N = 1 سينتقل التنفيذ الى عبارة رقم 10 .

إذا كان N = 2 سينتقل التنفيذ الى عبارة رقم 20 .

إذا كان N = 3 سينتقل التنفيذ الى عبارة رقم 30 .

ملاحظة : هذه الجملة شأنها شأن جملة التفرع السابقة ، تشكل عيباً في لغة Fortran القديمة ويمكن الاستغناء عنها بالتركيبة الحديثة التالية :

```
SELECT CASE (I)
 CASE (1)
  block of statements 1
 CASE (2)
  block of statements 2
 CASE (3)
  block of statements 3
 END SELECT
```

## عبارات IF ( IF Statements ):

هناك نوعان من عبارة IF هما : عبارة IF الحسابية وعبارة IF المنطقية.

### 1. عبارة IF الحسابية ( Arithmetic IF Statement ) :

تأخذ عبارة IF الحسابية الشكل العام التالي:

**IF (A) $n_1$ , $n_2$ , $n_3$**

حيث:

A: أما متغير أو تعبير حسابي يراد أستعماله كشرط يتوقف على قيمته تنفيذ أحدى العبارات التي أرقامها $n_1$, $n_2$, $n_3$ .

$n_1$, $n_2$, $n_3$ : أرقام عبارات تنفيذية.

وتعمل عبارة IF الحسابية كالآتي : نجد قيمة التعبير الحسابي ( A ) وفي حالة :

١ - قيمة A سالبة ينتقل التنفيذ إلى جملة رقم $n_1$ ٠

٢ - قيمة A صفر ينتقل التنفيذ إلى جملة رقم $n_2$ ٠

٣ - قيمة A موجبة ينتقل التنفيذ إلى جملة رقم $n_3$ ٠

مثال: IF ( X – 4.0 ) 10 , 20 , 30

هذا المثال يعني:

١ - إذا كانت قيمة التعبير ( X – 4.0 ) سالبة يحول التنفيذ الى عبارة رقم 10 ٠

٢ - إذا كانت قيمة التعبير ( X – 4.0 ) صفر يحول التنفيذ الى عبارة رقم 20 ٠

٣ - إذا كانت قيمة التعبير ( X – 4.0 ) موجبة يحول التنفيذ الى عبارة رقم 30 ٠

مثال: IF ( K – 100 ) 10 , 20 , 10

هذا المثال يعني تحويل التنفيذ إلى العبارة رقم 20 إذا كانت K = 100 ويحول التنفيذ الى عبارة رقم 10 إذا كانت K ¹ 100 ٠

ملاحظة مهمة : هذه العبارة يجب الاستغناء عنها نهائياً لأنها تجعل البرنامج صعب التعقب ، فلو قرأت برنامجاً يحتوي مثل هذه الجملة وحاولت تفهمه ستجد نفسك في دوامة الانتقال من مكان إلى آخر داخل البرنامج ٠ وعليك الاستعاضة عنها عند كتابتك لبرنامج يحمل تركيبة IF ٠

## 2. عبارة IF المنطقية ( Logical IF Statement ) :

تأخذ عبارة IF المنطقية الشكل العام التالي :

```
IF ( B ) S
```

حيث :

B : تعبير منطقي قيمته TRUE أو FALSE ٠

S : جملة تنفيذية ( مثل GO TO n أو عبارة حسابية أو عبارة إدخال أو أخراج أو توقف ٠٠٠٠ )٠

ملاحظـــــــــــــات على عمل عبارة IF المنطقية :

١ - لا يجوز أن تكون ( S ) عبارة ( IF ) المنطقية أو عبارة (DO ) التكرارية أو عبارة الإنهاء (END )٠

٢ - في حالة ( B ) صواب يتم تطبيق ( S ) وفي حالة ( B ) خطأ لا يتم تطبيق ( S ) ويستمر البرنامج في تطبيق العبارة التالية ٠

مثال : IF ( A > B ) GO TO 20

```
IF (DEG = = 100) PRINT *, RAD
IF (EXP(X) < 15.5) READ *, X,Y,Z
```

مثال: أكتب برنامجاً لمئة طالب كل منهم أدى ثلاثة أمتحانات المطلوب إيجاد معدل كل طالب وأسمه وتسلسله .

الحل :

```
PROGRAM AVERAGE_OF_THREE_DEGREE
    IMPLICIT NONE
     REAL  :: D1,D2,D3,SUM, AVERAGE
   INTEGER :: No, I=1
    CHARACTER (10) :: NAME
 10  PRINT *, "Enter The Values of NO. and name and  three degree"
 READ*, No, Name,D1,D2,D3
   SUM=D1+D2+D3
  AVERAGE = SUM/3.0
   PRINT *, No, NAME, AVERAGE
    I=I+1
IF(I < = 100) Goto 10
END PROGRAM AVERAGE_OF_THREE_DEGREE
```

## تركيبة الحالات ( اختيار الحالة ) CASE Construct

تستخدم هذه التركيبة بشكل واسع في لغة **Fortran 90** وهذه التركيبة تشكل بديلاً لعبارة **GO TO** الحسابية في نسخ لغة **Fortran** القديمة وكذلك بديلاً لتركيبة الأنتقاء (**IF construct**) ولكنها تتميز عنها بوضوحها . غير أن تركيبة الحالات لا يمكن لها أن تحل محل تركيبة الأنتقاء في كل الحالات .

وتأخذ تركيبة الحالات الشكل العام التالي :

```
SELECT CASE (expression)
 CASE (case selector 1)
  block of statements 1
 CASE (case selector 2)
  block of statements 2
   .
   .
 CASE DEFAULIT
  block of statements
END SELECT
```

من الملاحظ أن هذه التركيبة محاطة بثنائية الجمل :

```
SELECT CASE (expression)
 END SELECT
```

وبإمكان التعبير المذكور في بداية التركيبة **expression** أن يعطي قيمة صحيحة ، نصية ، أو تعبير منطقي .

تنقسم التركيبة في داخله الى أفرع شكلها العام كما يلي :

```
CASE (case selector)
block of statements
```

في أحد الأفرع **block of statements** سيتم تنفيذ الجمل التابعة له أذا أنطبقت قيمة **case selector** على قيمة التعبير المذكور في بداية التركيب ، ولذلك فأن هاتين القيمتين يجب أن تكونا من نفس النوع .

وبإمكان حالة الأختيار **case selector** المذكورة في أفرع تركيبة الحالات أن تأخذ قيمة واحدة أو مجموعة من القيم وهذا يضفي مرونة وعملية لهذه التركيبة ، حيث يتم تطبيق أحدى مجموعات الجمل حسب شرط اختيار واحد أو ضمن مجال لاختيارات عديدة وهذا ما سوف توضحه الأمثلة التالية :

```
SELECT CASE(SPEED)
CASE (90:120)
PRINT *, "FAST"
CASE (50:89)
PRINT *, "LEGAL"
CASE (10:49)
PRINT *, "SLOW"
CASE DEFAULT
PRINT *, "DANGEROUS; TOO SLOW OR TOO FAST"
END SELECT
```

هذا الجزء من البرنامج يختار أحدى الجمل للطباعة بناءً على قيمة السرعة **SPEED** المذكور في بداية التركيبة ، هذا المتغير (**SPEED** ) يشكل أبسط أنواع التعابير ويمكنه أن يأخذ أي قيمة صحيحة .

أذا كانت قيمة المتغير **SPEED** من **90** الى **120** فإن الجملة **"FAST"** تطبع ،وأذا كانت **SPEED** من **50** الى **89** فتطبع كلمة **"FAST"** تطبع **"SLOW"** ، وعدا ذلك من القيم سواء أكبر من **120** أو أقل من **10** فتطبع الجملة **"DANGEROUS; TOO SLOW OR TOO FAST"**

مثال آخر :

```
SELECT CASE(TRAFFIC_LIGHT)
CASE ("RED")
PRINT *, "STOP"
CASE ("YELLOW")
PRINT *, "WAIT"
CASE ("GREEN")
PRINT *, "PASS"
CASE DEFAULT
PRINT *, "illegal value;", TRAFFIC_LIGHT
END SELECT
```

في تركيبة الحالات هذه تعبير الاختيار هو الآخر بسيط ومن نوع متغير قيمة هذا المتغير هي من نوع نصي ( **Character** ) ، حالات الاختيار تحوي كل منها على خيار واحد **"RED"** ، **"YELLOW"** أو **"GREEN"** وأما ما عدا ذلك فيتم تنفيذ الجملة الموجودة في الفرع **CASE DEFAULT**

ومثال آخر :

```
SELECT CASE (Y>0)
CASE (.TRUE.)
X = Y
CASE (.FALSE.)
X = -Y
END SELECT
```

لاحظ أن تعبير **Y > 0** يختلف عن حالة الاختيار في المثالين السابقين كونه تعبيراً وليس متغيراً ، قيمة هذا التعبير منطقية وتمثل أحد جوابين .TRUE. أو .FALSE. فإذا كانت قيمة **Y > 0** هي .TRUE. فإن الفرع الأول من التركيبة ( **X = Y** ) ينفذ وأما أذا كانت .FALSE. فإن الفرع الثاني (**X = -Y** ) هو الذي يجري تنفيذه ،

كما نلاحظ هنا أن **CASE DEFAULT** لم ترد بسبب عدم الحاجة اليها ،

وهذا يعني أن الفرع **CASE DEFAULT** هو فرع اختياري ويمكن الاستغناء عنه حسب الحاجة ،

وبأمكان تركيبة الحالات أن تأخذ شكلاً أكثر من الحالات السابقة ففي الحالات السابقة كانت فروع التركيبة ، تحوي جملة واحدة فقط ، ولكن كما أسلفنا في الشكل العام للتركيبة فأن الأفرع تحتوي على مجموعة جمل وقد يكون من بين الجمل في هذه المجموعة أي تركيبة أو دوران ،

وهذا ما يوضحه المثال التالي عن شهور السنة :

```
SELECT CASE(MONTH)
CASE (9,4,6,11)
NUMBER_OF_DAYS = 30
CASE (1,3,5,7:8,10,12)
NUMBER_OF_DAYS = 31
CASE (2)
IF LEAP_YEAR THEN
NUMBER_OF_DAYS = 29
ELSE
NUMBER_OF_DAYS = 28
END IF
CASE DEFAULT
PRINT *, MONTH ," is not a number of month"
END SELECT
```

والمقصود من هذا الجزء هو تحديد عدد أيام شهر معين ،
يعطى رقم هذا الشهر كقيمة للمتغير **MONTH** ،
وحسب هذا القيمة يتم تنفيذ فرع التركيبة
الأول أو الثاني أو الثالث وأذا كان رقم الشهر
أكبر من **31** أو أقل من **1** فتطبع الجملة
المذكورة في **CASE DEFAULT** ،

نلاحظ هنا أن جمل الاختيار تحتوي أكثر من حالة واحدة ، ففي الجملة **CASE (1,3,5,7:8,10,12)** يكفي أن تنطبق قيمة المتغير **MONTH** مع أي من القيم المدرجة في هذه الجملة كي ينفذ هذا الفرع ،
لاحظ أن المجال **7:8** أستخدم يهذا الشكل بسبب تعاقب **7** أو **8** وكان يصح أن يكتب المجال على شكل **7 ، 8** أي بفصلهما بواسطة فاصلة ،
نلاحظ في الفرع اللاحق لجملة **CASE (2)** أحتواء هذا الفرع على مجموعة جمل تحوي تركيبة **IF** أنطباقاً مع الشكل العام لتركيبة الحالات ،

أســـــــــــــئلة محلولة :

س1/ أكتب برنامج لقراءة قيمة X ثم أحسب Y من التعبيرين ٠

$Y = 5X^2 + 0.5X + 0.95$ IF $X \leq 2.1$

$Y = 7X^2 + 0.7X + 0.53$ IF $X > 2.1$

الحل:

```
PROGRAM VALUE_OF_Y
   IMPLICIT NONE
    REAL  :: X,Y
 PRINT *, "Enter The Value of X"
 READ*, X
   IF(X < = 2.1) THEN
    Y=5*X**2+0.5*X+0.95
  ELSE
  Y=7*X**2+0.7*X+0.53
   END IF
PRINT "(2(2X,A,F9.3))", "X =", X ,"and The Value of Y =", Y
END PROGRAM VALUE_OF_Y
```

س2/ أكتب برنامج لقراءة لإيجاد قيمة X من التعابير التالية :

في حالة J =1 $X = \sqrt{A^2 + B^2 + C^2}$

في حالة J =2 $X = |A^2 + 2B - C|$

في حالة J =3 $X = \ln(A + B + C)$

الحل:

```
PROGRAM VALUE_OF_X
IMPLICIT NONE
REAL :: A , B , C , X
INTEGER :: J
PRINT *,"ENTER THE VALUES OF A , B , C and J"
READ *, A , B , C
READ *, J
SELECT CASE (J )
Case (1)
X=SQRT(A**2+B**2+C**2)
Print *,"X =", X
Case (2)
X=ABS(A**2+2*B-C)
Print *,"X =", X
Case (3)
X=LOG(A+B+C)
Print *,"X =", X
CASE DEFAULT
Print *," No solution for X"
End Select
END PROGRAM VALUE_OF_X
```

ملاحظة :

لاحظ تم تعريف المتغير J كونه

متغير صحيح INTEGER

لأن المتغير العددي المستخدم في عبارة

SELECT CASE يجب أن يكون

متغير صحيح INTEGER

## HOME WORK :

س1 / أكتب برنامجاً لقراءة قيمة المتغيرين n , x وحساب وطبع قيمة Y المعرفة .

$Y = X + COS\ x + 5\ \tan^{-1}\sqrt{x}$ if n < 0

$Y = (X - Sin\ x)^2 - Sin\ 2x$ if n = 0

$Y = (X + e^{2x} + Sin\ (x-1.5))^2$ if n > 0

س2 / أكتب برنامجاً يطبع كلمة VOWEL أذا كانت قيمة المتغير C أحد الثوابت التالية ( A, E, I, O, U ) ويطبع كلمة CONSTANT أذا كانت قيمة المتغير C أي حرف آخر . وأن يطبع البرنامج أيضاً عبارة ERROR أذا كانت قيمة المتغير C عدا ذلك .

س3 / أكتب برنامجاً لحساب مجموع المتسلسلة التالية :

$$S = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \ldots + \frac{99}{100}$$

س4 / ما هي النتيجة المتوقعة للبرنامج التالي على شاشة الكومبيوتر ( تتبع خطوات الحل مطلوبة ) .

```
PROGRAM HOMEWORK
IMPLICIT NONE
REAL :: Z , R , P , Q , T, K
READ *, R , P , Q , T, K
Z = 5.0
 IF ((R < 3.0) .OR. (P+T <  10.5)) Z=Z+P
 IF ((P > Q) .AND. (K > 5)) Z=Z-P
 PRINT "(T10 , A, F6.2)",  "Z = ", Z
END PROGRAM HOMEWORK
```

علماً بأن القيم ( P = 3 , Q = 5 , R = 2 , T = 10.2 , K = 8 )

س5 / إذا فرضنا أ ن A=5.0 و B=2.0 فأوجد القيمة المنطقية للتعبير المنطقي المركب التالي :

B .LT. A .AND. A+B .EQ. 8.0 .OR. (.NOT. (B .EQ. 4.O))