Shatt Al-Arab University

Department Civil Engineering

Programming by FORTRAN 90

Second Stage

2024 – 2025

Dr. Wisam Abdulla Najim ALHalfi

Chapter One

Introduction to Programming in FORTRAN90

مقدمة الى البرمجة بلغة FORTRAN

وهي اختصار لعبارة **FOR**mula **TRAN**slation وتعني ترجمة الصيغ أو المعادلات وحيث تعد لغة **FORTRAN** من أقدم لغات البرمجة ذات المستوى العالي وشيوعاً واستعمالاً لمختلف الأغراض العلمية والهندسية. وقد ظهرت هذه اللغة في منتصف الخمسينات، حيث بدأ عام 1954 من قبل فريق من الباحثين برئاسة المهندس جون باكس (JOHN BACKUS) بالعمل من قبل شركة IBM الأمريكية على تطوير لغة برمجة تقبل برنامجاً مكتوباً بلغة قريبة من لغة الانسان, ويتم تحويله الى شفرة قابلة للتنفيذ على الحاسب الالي وبعد 3 سنوات أي في عام 1957 ظهر اول مفسر فورتران. وفي تلك الفترة كان استخدام الحاسب الالي يكاد يكون حكراً على العلماء والمهندسين والرياضيين ومن الطبيعي ان تكون هذه اللغة المطورة حديثاً قد جاءت لتواكب احتياجاتهم اذ تتميز لغة **FORTRAN** بقدراتها على اجراء العمليات الحسابية المعقدة وحل المعادلات الرياضية.

وقد شهدت هذه اللغة العديد من التطورات والاضافات التي وسعت امكانياتها وجعلتها من اهم اللغات التي تستخدم في مجال الرياضيات. الإصدار الأول **FORTRAN I** صدر في شهر ابريل من عام 1957 وفي العام التالي 1958 صدر الإصدار الثاني **FORTRAN II** الذي وفر إمكانيات ترجمة الروتينات الفرعية بشكل مستقل. وفي عام 1966 صدرت **FORTRAN 66** من قبل **ANSI** (America National Standards Institute) التي كانت الأكثر استخداماً في الأوساط العلمية لفترة طويلة. وفي عام 1978 اصبح الإصدار **FORTRAN 77** من قبل **ANSI** هو اللغة القياسية و تضمن معاملة السلاسل الحرفية و جملة If – Else.

والاصدار **FORTRAN 90** ظهر عام 1991 وتضمن المؤشرات والمصفوفات الديناميكية. وهنالك إصدارات أخرى مستمرة للغة فورتران هي (**FORTRAN 95, FORTRAN 2003, …**) ولغة فورتران كأي لغة أخرى تتألف من مجموعة من الكلمات والعبارات والتركيبات المؤلفة طبقاً لقواعد معينة ينبغي التقيد بها دوماً. مما يحد من انتشار هذه اللغة انها تحتاج الى مترجم خاص بها **FORTRAN Compiler** ومن أشهر تلك المترجمات Lahey FORTRAN, Microsoft FORTRAN Compiler, Power Station,…

البرنامج:

هو مجموعة من التعليمات التي تخبر الحاسوب للقيام بمهمة معينة. لإنشاء برنامج يعمل، ويعمل بكفاءة سوف نحتاج إلى القيام بما يلي قبل البدء في كتابة البرنامج:

1. التأكد من فهم أهداف البرنامج.
2. تحديد المعلومات التي يحتاجها البرنامج كمعطيات (Input) وكذلك التي سوف تنتج عنه كمخرجات (نتائج) (Output).
3. فهم كيفية إجراء العمليات الحسابية (أي فهم الخوارزميات التي يجب استخدامها) ، وبالترتيب الذي ينبغي القيام به.

من السهل جدا تخطي واحدة أو أكثر من الخطوات (1 – 3) والبدء في كتابة البرنامج دون التفكير في الكيفية التي سيعمل بها، ولكن ما لم يكن البرنامج صغير جدا ، فإن هذا ربما يؤدي إلى برنامج سيئ التنظيم ومن الصعب متابعته. معظم البرامج لا تعمل بشكل متكامل من المرة الأولى، لكنها وفي الأرجح سوف تعمل وسيكون من السهل التصحيح عليها إذا كانت منظمة بشكل جيد من البداية. خصوصاً اذا كان المبرمج قد وضع بعض التعليقات التي تبين عمل كل مجموعة من الإيعازات Block.

4. إنشاء البرنامج الذي يحل مشكلة محددة وحسب المدخلات والمخرجات (Input/Output) المطلوبة التي كنا حددناها مسبقاً.
5. اختبار البرنامج بدقة. (كأن نتأكد أن هذا البرنامج عام ولا يعمل فقط للمدخلات التي بين أيدينا فقط).

تشغيل برنامج FORTRAN90

بعد الانتهاء من عملية تثبيت البرنامج، الان ابدأ بتشغيله للتعرف على اهم سمات بيئة تطويره ويمكن تشغيل البرنامج بأحدى الطرق التالية:

- قم بالضغط المزدوج بزر الماوس الايسر double click على ايقونة الاختصار short cut الخاصة بالبرنامج والموجودة على شريط المهام وتعد هذه الطريقة من أسهل وأسرع الطرق لتشغيل برنامج **FORTRAN**.
- او من خلال الاتجاه الى قائمة ابدأ Start ثم نختار All programs لتظهر لك قائمة فرعية بكافة البرامج المثبتة على نظام التشغيل الخاص بجهازك، فاختر من هذه القائمة الفرعية **FORTRAN.**

Start Menu ⟶ All Programs ⟶ FTN95 ⟶ Plato

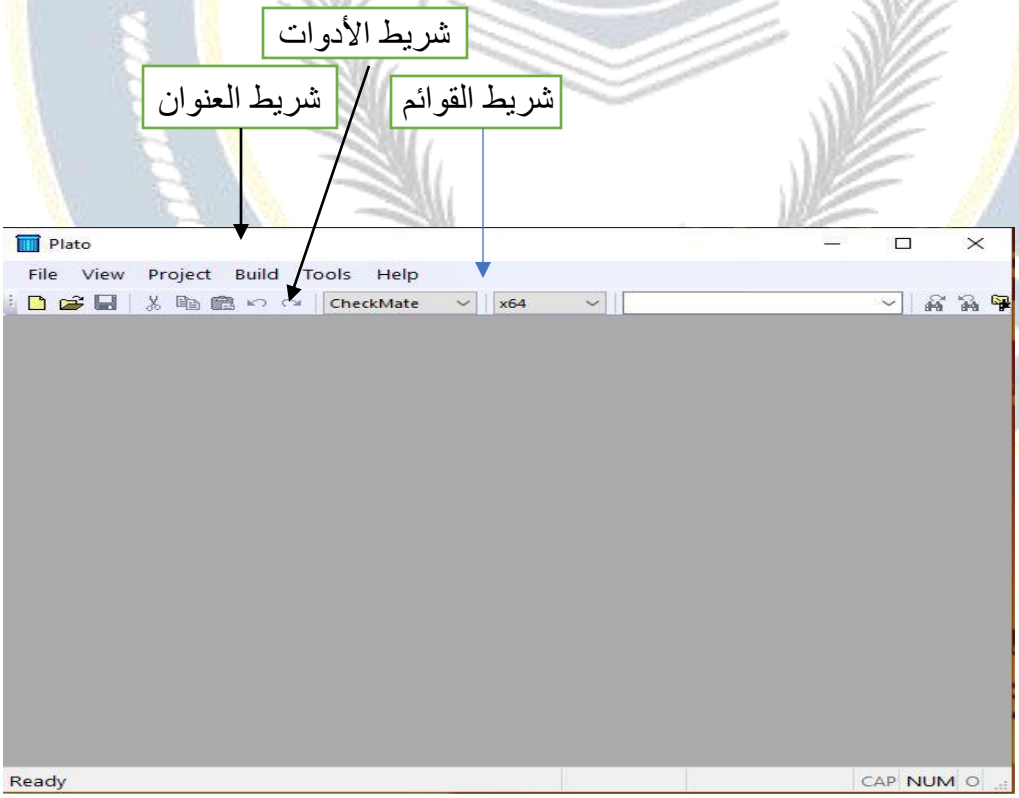
وبأستخدام احدى الطريقتين السابقة لتشغيل برنامج **FORTRAN** سوف تظهر لك الواجهة الرئيسية للبرنامج وهي واجهة التخاطب الأساسية مع المستخدم والتي تسمى سطح مكتب برنامج **FORTRAN** وتتضمن هذه النافذة كافة النوافذ المرتبطة بسطح مكتب البرنامج التي سنتعرف عليها بالتفصيل.

مكونات واجهة برنامج FORTRAN

لا تختلف سمات واجهة برنامج **FORTRAN** كثيرا عن سمات البرامج التي تعمل تحت نظام التشغيل WINDOW مثل برنامج Office وغيرها فكلاهما يستخدم نفس العناصر.

يتكون سطح مكتب برنامج **FORTRAN** من العناصر التالية:

1. شريط العنوان Title Bar.
2. شريط القوائم Menu Bar.
3. شريط الأدوات Tool Bar.
4. شريط الحالة Status Bar.



شريط العنوان Title Bar

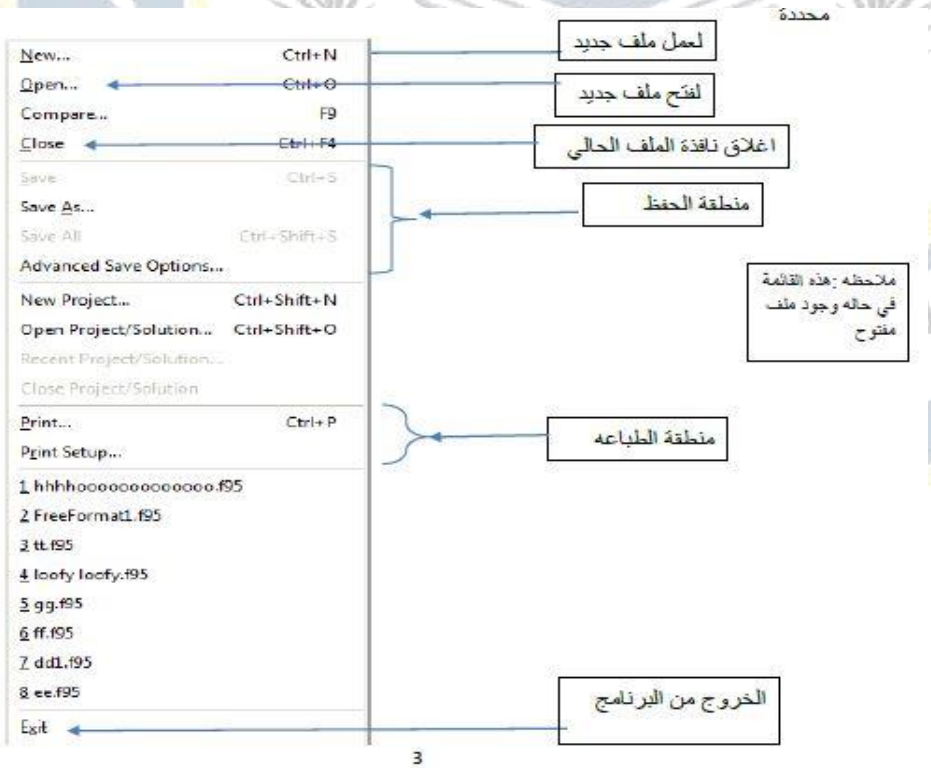
يحتوي هذا الشريط على اسم ورمز البرنامج واسم الملف او النافذة المفتوحة حاليا، كما يحتوي في اقصى يمينه على مفاتيح التحكم الثلاثة:

- Close عند الضغط على هذا المفتاح يتم أغلاق النافذة المفتوحة حاليا.
- Maximize/Restore عند الضغط على هذا المفتاح يتم تكبير النافذة الى الحد الأقصى ملئ الشاشة او استعادة النافذة الى حد اصغر.
- Minimize عند الضغط على هذا المفتاح يتم تصغير النافذة على شريط المهام Taskbarولاستعادتها مرة أخرى يتم الضغط على اسم النافذة على شريط المهام.

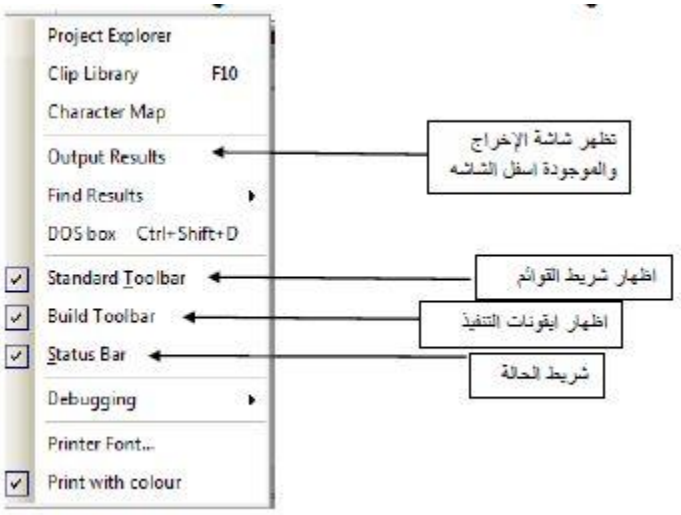
شريط القوائم Menu Bar:

يوجد شريط القوائم أسفل شريط العنوان مباشرة ويحتوي على قوائم برنامج Fortran الأساسية والتي تحتوي على الأوامر والخيارات اللازمة للتعامل مع البرنامج. ويضم هذا الشريط القوائم التالية:

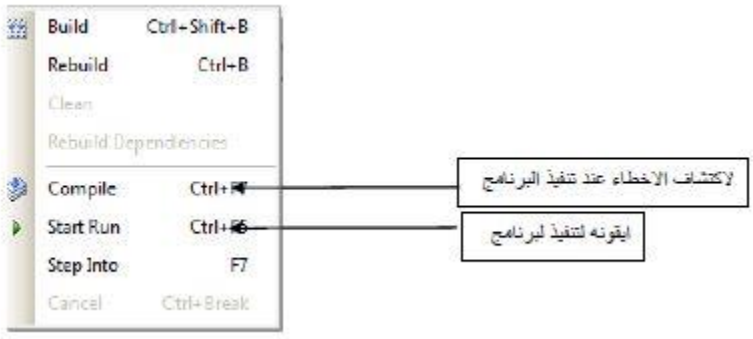
قائمة ملف File: تتكون هذه القائمة من الخيرات التي تنفذ كل منها وظيفة محددة.



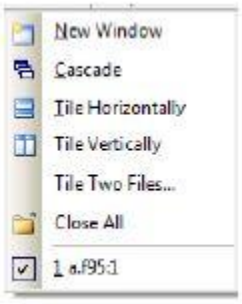
قائمة View: توجد في هذه القائمة العديد من النوافذ كما مبين في الشكل أدناه.



قائمة Build: يحتوي هذا الشريط على أيقونات لبعض الأدوات.



قائمة Window: يمكن ترتيب النوافذ المختلفة وغلقها وكذلك فتح نافذة كما مبين في الشكل.



شريط القوائم Menu Bar:

يحتوي هذا الشريط على أيقونات لبعض الأدوات الموجودة في شريط القوائم ويعتبر وسيلة سريعة يمكننا من خلالها تنفيذ الأوامر بطريقة سريعة بدلا من فتح القوائم والبحث بداخلها عن الأوامر المطلوبة.

شريط الحالة Status Bar:

يمثل حالة البرنامج فأما أن يكون جاهزا Ready لتنفيذ الأوامر التي يقوم المستخدم بإدخالها أو يكون مشغول Busy بأجراء حسابات معينة.

لأنشاء ملف والعمل عليه:

الملفات: هي نوع من أنواع الملفات التي يعمل عليها برنامج كوسيلة لإدخال الأوامر والرموز البرمجية، حيث يتم تحرير أوامر البرنامج في ملف.

هنالك طريقتان لأنشاء الملف:

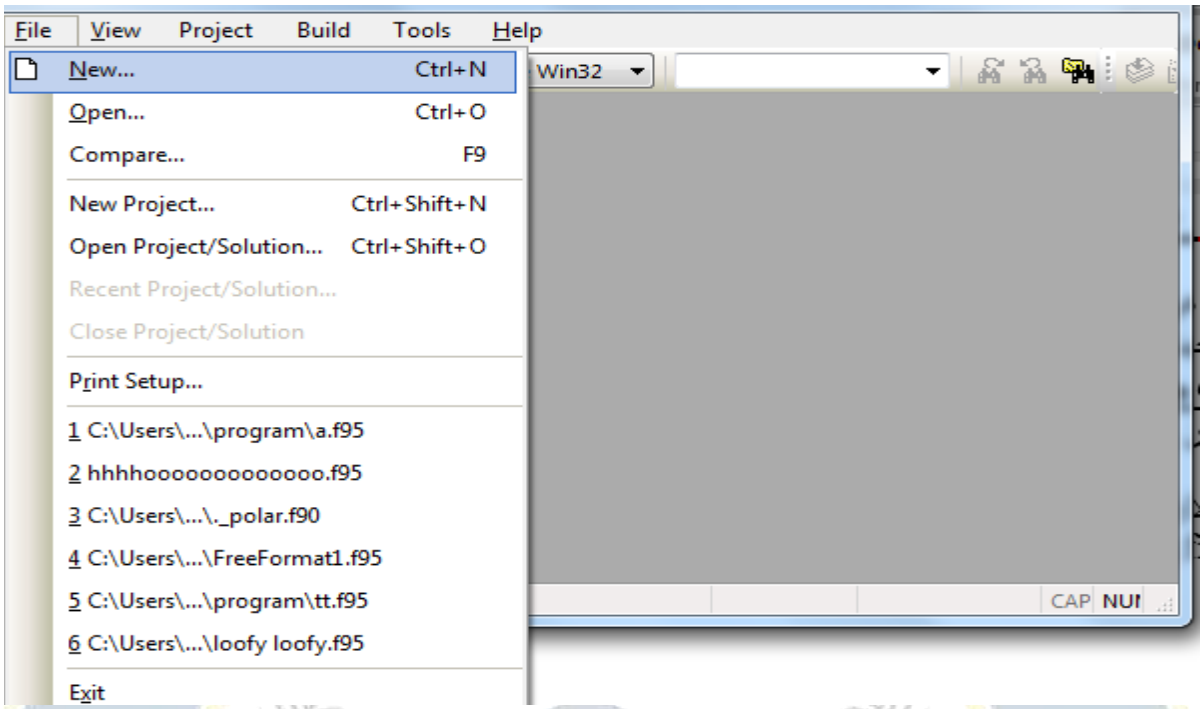
1. File ⟶ New ⟶ Free format file

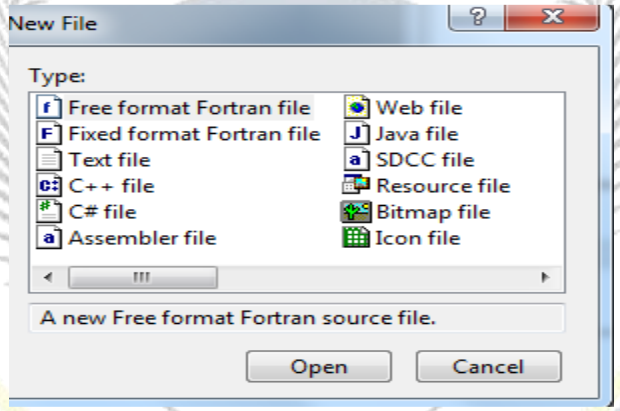
2. File ⟶ New ⟶ Fix format file

نختار احدى الطريقتين: ففي الطريقة الأولى تبدء من أي عمود و غير مقيد بعمود معين ويكون امتداد الملف (Extension) (.f95, .f90, inc).

وفي الطريقة الثانية تبدء من العمود السابع و مقيد بعمود معين ويكون امتداد الملف (Extension) (,for. (.f, ins.

كما موضح في الأشكال التالية:





الأجزاء الرئيسية لبرنامج فورتران 90 (The main parts of FORTRAN90):

يتكون البرنامج من اربع أجزاء رئيسية:

1. اسم البرنامج.
2. الإعلان عن المتغيرات (قسم التهيئة).
3. هيكل البرنامج الرئيسي.
4. البرامج الفرعية.

اسم البرنامج (Program Name)

السطر الأول من البرنامج يختص باسم البرنامج والشكل العام لهذا الأمر هو: Program name

اسم البرنامج يفضل ان يكون مختصراً ومعبراً عن الغرض من البرنامج على سبيل المثال Program average. ويمكن ان تضع رقم يشير الى الإصدار أو أخر نسخة عدلتها من البرنامج على سبيل المثال Program average3. وهذا الاسم اختياري لا يوضع فراغ داخل الاسم بل تستخدم الـ Underscore (_) عندما يتكون من اكثر من كلمة ولا يجوز استخدام إشارة (-).

مثال: Program Summation and Average غير صحيح لوجود فراغات بين الكلمات.

مثال: ProgramSummation-and-Average غير صحيح لوجود إشارة الطرح بين الكلمات.

مثال: ProgramSummation_and_Average تعبير صحيح.

قسم التهيئة: الإعلان عن المتغيرات (Initialization section: declaration of variables)

قسم التهيئة يحدد بعض القواعد للبرنامج مثلاً يستخدم للإعلان عن جميع المتغيرات التي يمكن استخدامها لتخزين أرقام، رموز، مصفوفات.... ، الخ ، ضمن البرنامج. في فورتران 90 هناك عدة أنواع من المتغيرات يتم تعريفها مثل الأعداد الصحيحة أو الحقيقية أو الرموز الأبجدية (integer, real and character) وحتى يمكن تحديد أنواع جديدة. إعلان البيانات يمثل تعيين أنواع المتغيرات الخاصة التي سوف تستخدمها في البرنامج ويمكن أن يستخدم أيضا لتعيين القيم الأولية. قواعد الفورتران لا تصر على أن تقوم بتعريف المتغيرات ، لكنه أمر جيد للبرمجة ويساعد على تجنب الكثير من الأخطاء.

أن السبب الرئيس للإعلان عن أنواع المتغيرات هو سبب فني يخص ذاكرة الحاسبة فالحاسبة تحتاج أماكن صغيرة من الذاكرة لخزن الرقم الصحيح، بينما تحتاج الى أماكن اكبر من الذاكرة لخزن الأرقام الحقيقية وكذلك الرموز، لذلك عندما يعلم البرنامج نوع هذه المتغيرات منذ البداية فسوف يخصص لها الأماكن المناسبة بدون إسراف أو تبذير (أن هذا الإسراف لن يكون واضحاً أو حتى ملحوظاً في البرامج الصغيرة ولكن له الأثر الكبير في البرامج الضخمة).

هيكل البرنامج الرئيسي (Main Program Body)

الهيكل الرئيسي للبرنامج يجب أن يحتوي على جميع العبارات (Statements) القابلة للتنفيذ بلغة فورتران والتي تحقق الغرض من البرنامج للوصول إلى النتيجة لحل مسألة معينة أو إخراج نتائج مطلوبة. العبارات (Statements) في العادة تتضمن أوامر تنفذ هذه العبارات بالتسلسل ابتداء من الأعلى الى الأسفل وأخر عبارة في البرنامج يجب أن تكون End.

البرامج الفرعية (Subprograms)

إذا احتوى البرنامج على مجموعة من العبارات (blocks) تؤدي وظيفة معينة وتتكرر باستمرار بالبرنامج فيمكن جعل البرنامج أكثر وضوحاً وذلك بإزالة هذه المجموعة من العبارات والأوامر(blocks) ووضعها في برنامج فرعي يقوم البرنامج الرئيسي باستدعائها كلما تطلبت الحاجة الى ذلك ويمكن تكوين أكثر من برنامج فرعي داخل للبرنامج الرئيسي.

أنواع البيانات (Data types)

أولا: الأحرف والرموز: أن الأحرف والرموز المسموح استخدامها تشمل ما يأتي:

1. **الحروف الأبجدية الإنكليزية:** تتكون من الحروف الكبيرة والصغيرة وهي ستة وعشرون حرفاً هي: A,B,C,……Z وتسمح فورتران90 بأستعمال الحروف الأبجدية الصغيرة a,b,c,…z .
2. **الأرقام الحسابية:** وهي 0,1,2,3,….9 .
3. **الرموز الخاصة** وتشمل:

Name	Character	Name	Character	Name	Character
Colon	:	Dollar sign	$	Underscore	_
Exclamation mark	!	Quotation mark	"	Question mark	?
Percent sign	%	Less than	<	Blank (Space)	
Ampersand	&	Greater than	>	Left parenthesis	(
Semicolon	;	Minus sign	-	Right parenthesis	)
Equal sign	=	Asterisk	*	Comma	,
Plus sign	+	Slash	/	Apostrophe	‘

ثانياً: الثوابت (Constants)

الثوابت هي الكميات التي تبقى ثابتة أثناء تنفيذ البرنامج وتنقسم إلى قسمين: الثوابت العددية (Numerical Constants) والثوابت غير عددية (Non- Numerical Constants).

الثوابت العددية (Numerical Constants):

هي عبارة عن أعداد رقمية وكل عدد يمكن أن يحوي على فاصلة عشرية وكما يمكن أن يحمل الإشارة السالبة أو الموجبة وهناك ثلاث أنواع من الثوابت العددية :

1. **الثوابت الصحيحة (Integer Constants):** تعرف بانها أعداد صحيحة أو صفرية مسبقة أو غير مسبقة بإشارة وتظهر هذه الأعداد بدون الفاصلة العشرية (.). على سبيل المثال (0, 5, +300, 313 -,....).

قواعد الثوابت العددية الصحيحة:

1. لا يجوز استخدام الفاصلة العشرية (.) في كتابتها.
2. لا يجوز استخدام الفارزة (,) في كتابتها ٠
3. يمكن وضع إشارة (+)) أمام العدد للدلالة على أنه موجب وكذلك الحال بالنسبة للإشارة (-) عندما يكون العدد سالباً وعند عدم وجود الإشارة يعتبر العدد موجبا.
2. **الثوابت الحقيقة (Real Constants):** وتشمل هذه المجوعة من الثوابت الأعداد الحقيقية الكسرية وتكتب بأستخدام الفاصلة العشرية والصورة العامة لها في لغة فورتران تكون بالشكل التالي:

$$n.m \quad n. \quad n.mE \pm s \quad n.E \pm s \quad .mE \pm s \quad nE \pm s$$

حيث: n تمثل العدد الصحيح. و m تمثل الجزء الكسري أي الأرقام الموجودة على يمين الفاصلة. و E تمثل علامة الضرب (x) والأساس 10.

وعلى سبيل المثال (13.25, -14.1, +5.00,...)

و (0.0123 = 1.23E-02)

و (987652.0 = 9.87652E+05).

قواعد الثوابت العددية الحقيقية:

1. العدد الحقيقي الكسري يحتوي على فاصلة عشرية.
2. الفارزة غير مسموح بها في أي عدد.
3. يمكن إضافة أي عدد من الأصفار في مقدمة العدد أو مؤخرته دون التأثير في قيمته.
4. يمكن للعدد أن تكون له إشارة تسبقه.
5. يمكن كتابة العدد الحقيقي بالدقة المضاعفة (Double precision) نفس طريقة كتابة العدد الحقيقي لكن نستبدل الرمز E بدل منه D للتعبير عن الأس مثلاً(1.23D-2 = 0.0123D0 = 0.0123)

3. **الثوابت المركبة (Complex Constants):** هو ثابت عددي ذو فاصلة عشرية وهو مكون من جزئين يسمى أحدهما الجزء الحقيقي (Real Part) ويسمى الآخر بالجزء الخيالي (Imaginary Part) ويكتب عادة بين قوسين ويفصل بين جزئيه هذين بفارزة.

وعلى سبيل المثال:

صورة الأعداد المركبة الرياضية	صورة الأعداد بلغة فورتران
3.2-1.86i	(3.2,-1.86)
4.0+5.0i	(4.0,5.0)
-21.0+3.24i	(-21.0,3.24)

الثوابت غير العددية (Non numerical Constants): وتشمل على الأنواع التالية:

1. **الثوابت المنطقية (Logical Constants):** هذا النوع من الثوابت يختلف عن الأنواع سابقة الذكر في أنه ليست له قيمة عددية وإنما يكون على أحد حالتين إما أن تكون صواب (TRUE) أو خطأ (FALSE). وحسابياً TRUE تأخذ قيمة الرقم ١ أو أي رقم آخر عدا صفر و FALSE تأخذ الرقم صفر.
2. **الثوابت الرمزية (Character Constants):** وهذه الثوابت تتكون من سلسلة أو عدد من الرموز وتستخدم عادة في كتابة العناوين أو تمييز بعض نتائج البرامج وتكتب بين حاصرتين علويتين (من النوع المفرد أو المزدوج) وهذا النوع من الثوابت غير عددي أي لا يخضع للعمليات الحسابية.

وعلى سبيل المثال: , "Amount" , "10+20" , 'Quality'

"program's logic" , '2008'

ملاحظـــــة: هنالك تعبير واحد يتعامل مع هذا النوع من الثوابت وهو تعبير الوصل (Concatenation) ورمزه (//) وتكون عملية الوصل نتيجة لوصل نصين ينتج عنهما نص ثالث مثل عملية الوصل :

"NO SMOKING" تكون نتيجتها "NO"//"SMOKING"

ثالثاً: المتغيرات (Variables)

المتغير هو كمية تعطي أسماً معيناً مثل x أوy ويسمح لها بالتغير أي بأخذ قيم مختلفة خلال تنفيذ البرنامج. هي أيضاً أسماء داخل البرنامج وقد تعبر عن أرقام صحيحة أو حقيقية أو رموز ...الخ، ولكن فرقها عن الثوابت أن القيم الموكلة الى هذه الأسماء قد تتغير داخل البرنامج، وهي تستخدم للمدخلات, المخرجات, والنتائج الوسطية (input, output, intermediate results). في لغة فورتران 90 فهي تحوي على اسم المتغير الذي يتكون من الأحرف الأبجدية والرموز الرقمية وخط الشارحة السفلى (underscore)

شروط كتابة اسم المتغير في لغة FORTRAN90:

1. يتكون من رمز أو مجموعة رموز وهذه الرموز هي أما حروف أو أرقام والرمز (_) (underscore) فقط.
2. يسمى المتغير بحيث لا يتجاوز عدد رموزه عن 31 رمزاً (حروف وأرقام والرمز _ فقط).
3. يجب أن يبدأ أسم المتغير بحرف ولا يجوز أن يبدأ برقم أو (_).
4. يستحسن استعمال أسم للمتغير يدل على نوع استخدامه ويفضل تجنب الحرف الواحد.

وهنالك نوعان من المتغيرات في لغة فورتران هما: المتغيرات العددية والمتغيرات غير عددية.

المتغيرات العددية (Numerical Variables): وهي المتغيرات التي تستخدم لتخزين الثوابت العددية في وحدة الذاكرة وهي على أنواع أهمها :

1. **المتغيرات الصحيحة (Integer Variables):** هي المتغيرات التي يسمح لها بأخذ قيمة ثابت صحيح فقط وتخضع أسماء المتغيرات الصحيحة لقاعدة الحرف الأول أذا لم تعرف في بداية البرنامج في عبارة نوع وهي أن كل متغير يبدأ بأحد الأحرف الستة الآتية فهو متغير صحيح:

I , J , K , L , M , N

فمثلاً الأسماء التالية : MASS_1 , K5 , I , JM , KK , NUM_2

2. **المتغيرات الحقيقة (Real Variables):** وهي المتغيرات التي تستخدم لخزن الأعداد الحقيقية المحتوية على كسور عشرية في الذاكرة، ويخضع أسم المتغير الحقيقي لقاعدة الحرف الأول اذا لم يعرف في البداية للبرنامج في عبارة نوع وهي أن كل متغير يبدأ بأي حرف عدا الحروف الستة الأتية (**I , J , K , L , M , N**).

فمثلاً الأسماء الأتية: Value_1 , X5 , Y , ABC , Calculate_of_X هي متغيرات حقيقية.

3. **المتغيرات المركبة (Complex Variables):** وهي المتغيرات التي تستخدم لخزن الثوابت المركبة (تم تعريفها سابقاً) في الذاكرة، ويجب تعريف هذه المتغيرات في بداية البرنامج بعبارة Complex.

المتغيرات غير العددية (Non numerical Variables): وتشمل على الأنواع التالية:

1. **المتغيرات المنطقية (Logical Variables):** وهي المتغيرات التي تستخدم لخزن الثوابت المنطقية (تم تعريفها سابقاً) في الذاكرة، ويجب تعريف هذه المتغيرات في بداية البرنامج بعبارة Logical.
2. **المتغيرات الرمزية (Character Variables):** هذا النوع من المتغيرات ليس له قيم عددية، كالمتغيرات العددية، وإنما له قيم رمزية ويتكون المتغير الرمزي من مجموعة من الرموز Character or Strings والتي يمكن أن تكون من الحروف والأرقام والرموز الخاصة التي سبقت الإشارة اليها. ويستخدم هذا النوع من المتغيرات كعناوين أو رموز توضيحية يحتاج أليها المبرمج لتبيان أسماء المتغيرات أو معانيها أو للتعبير عن قيم ثوابت رمزية.

أمثلة على هذا النوع: CHARACTER (Len=25) :: NAME

حيث: NAME: اسم المتغير الرمزي.

(Len=25): طول المتغير الرمزي أي عدد رموزه فيحجز له في الذاكرة 25 موقع.

CHARACTER (25) :: NAME

لان كلمة Len هي اختيارية Optional.

CHARACTER (*) :: ANSWER

وهذا يعني استخدام الصيغة الحرة (*) لمقدار الحجز لقيمة المتغير الرمزي ANSWER. (أي مهما كانت قيمة الثابت الرمزي المراد الحجز له).

ملاحظـــــــة: إذا لم يذكر أي رقم بعد عبارة CHARACTER فأنه يفهم من ذلك أن طول المتغير الرمزي المعلن عنه لا يتعدى رمزاً واحداً فقط. على سبيل المثال

```
CHARACTER :: RLGION
RLGION = 'I'
```

حيث I: هو رمز واحد فقط يمثل القيمة الرمزية للمتغير الرمزي RLGION والحرف I هنا اختصار للثابت الرمزي ISLAM وقد استعمل الحرف Iوحدة لان عبارة CHARACTER لم تحجز غير مكان واحد فقط.

مثال: CHARACTER *10,X,Y

وتعني ان المتغيرين X,Y متغيرات رمزية وتم الحجز لقيم كل منهما بمقدار 10.

مثال: CHARACTER *6,L*4,M,N,S*8

ومعناه: المتغيرات M,N يحجز لهما 6 رموز والمتغير الرمزي L يحجز له 4 رموز والمتغير الرمزي S يحجز له 8 رموز.

قواعد استعمال المتغيرات الرمزية (Rules for character variables)

1. يجب الإعلان عن أي متغير رمزي في البرنامج وذلك باستخدام عبارة character في بداية البرنامج وقبل اسم المتغير الرمزي.
2. لا تستخدم الحاصرات العلوية مع المتغيرات الرمزية وإنما مع الثوابت الرمزية.
3. ينبغي أن يكون الرقم المستعمل بعد عبارة character صحيح العدد.
4. يجوز استعمال المتغيرات المؤشرة والمصفوفات في جملة الحجز character.
   مثل character(2)::month(12)
5. لا يجوز أن يستعمل أسم متغير رمزي في أكثر من جملة character واحدة.

أمثلة:

CHARACTER (LEN=8):: DateINFO

CHARACTER (LEN=4):: Year, Month*2, Day*2

CHARACTER (LEN=2):: Hour, Minute, Second*6

عبارات النوع (Type Statements)

هي عبارات التصريح declaration بنوع المتغيرات التي تستخدم في البرنامج وتوضع هذه العبارات عادة في بداية البرنامج وهي عبارة اختيارية Optional ولكنها مهمة جداً كما تم ذكره في لغة فورتران 90 لأنه في كثير من الأحوال يحتاج المبرمج إلى استخدام بعض أسماء المتغيرات التي لا تخضع لقاعدة الحرف الأول السابقة وفي هذه الحالة نستطيع استخدام عبارات تحديد النوع للتغلب على المشكلة، وكذلك لتجنب الوقوع في خطأ استخدام متغيرات من نفس النوع في العمليات الحسابية وأيضاً سهولة قراءة البرنامج وهناك **خمسة** أنواع من عبارات النوع المستخدمة في لغة فورتران 90هي :

INTEGER, REAL, LOGICAL, COMPLEX and CHARACTER

والصيغة العامة لكتابة عبارات النوع هي: **Type,specifier::list**

حيث: Type: عبارة النوع المراد تعريف المتغيرات بها (INTEGER, REAL, LOGICAL, …).

Specifier: وهي بعض العبارات التي تحدد المتغير إضافة إلى عبارة النوع وسنأتي على ذكرها.

List: هي أسماء المتغيرات المراد تعريفها مفصولة عن بعضها بفارزة (,).

مثال: INTEGER :: A2, B_1, BETA, QUEEN

أن كل من A2, B_1, BETA, QUEEN هي متغيرات صحيحة (دون التقيد بقاعدة الحرف الأول).

مثال: REAL :: I_10,JAK2,MAX,MIN

أن كل من I_10,JAK2,MAX,MIN هي متغيرات حقيقية (دون التقيد بقاعدة الحرف الأول).

مثال: LOGICAL :: FLAG,MAR

أن كل منFLAG,MAR هي متغيرات منطقية.

مثال: COMPLEX :: Total, A_ 1

أن كل من1 _A ,Total هي متغيرات مركبة.

هناك نوع أخر من عبارات النوع هي عبارات النوع الضمنية مثل (**Implicit**).

IMPLICIT INTEGER :: (P-T)

IMPLICIT REAL :: (J-N)

حيث تعني العبارة الأولى أن أي متغير يبدأ بالحروف من P الى T هو متغير صحيح. والعبارة الثانية تعني أن أي متغير يبدأ بالحروف من J الى N هو متغير حقيقي.

عبارة Implicit none: هي عبارة مهمة جداً في لغة فورتران 90 ويكون موقعها في بداية البرنامج بعد أسم البرنامج وقبل عبارات التصريح بالنوع وتعني إلغاء قاعدة الحرف الأول في تعريف نوع المتغيرات وأن كل المتغيرات يجب أن يعرف نوعها بأحد عبارات النوع التي تم ذكرها سابقاً ورغم أنها عبارة اختيارية ألا أنه يجب استخدامها في كل برامج لغة فورتران90 لأنها تساعد أن يكون البرنامج أمين من الأخطاء.

هيكلية برنامج FORTRAN90 (FORTRAN 90 Program Structure)

يتكون برنامج لغة فورتران 90 من البرنامج الرئيسي ووحدات أخرى من البرامج ومنها البرامج الفرعية: ويكون الشكل العام للبرنامج الرئيسي هو:

```
PROGRAM program-name
IMPLICIT NONE
Specification statements
...
Executable statements
...
END PROGRAM program-name
```

عبارة Program

وهي عبارة اختيارية، يصح البرنامج بدونها ولكن من الجيد التعود على كتابتها في كل برنامج مع أنها لا تأثير لها على البرنامج سوى أعطاء أسم يستفيد منه مستخدم البرنامج في المستقبل.

و أسم البرنامج (program-name) هو أسم يعطى إلى البرنامج ويفضل أن يكون أسم البرنامج يدل على عمل البرنامج وهذا الأسم تنطبق عليه شروط كتابة أسم المتغير أي هو كلمة أو مجموعة كلمات متكونة من (حروف وأرقام والرمز_) بشرط أن لا يتعدى طوله (31) رمزاً وأن لا يبدأ برقم.

مثل: My_Program , Test , Calculate_Value_Of_Z , ...

عبارة End

وهي عبارة غير تنفيذية ينتهي بيها كل برنامج والتي تشير الى نهاية نص البرنامج وفي لغة FORTRAN90 يفضل أن تكون عبارة End ويكتب بعدها نفس العبارة الأولى أي عبارة (program) كما موضح بالمثال التالي:

```
PROGRAM Calculate
Print *, 10 + 20
END PROGRAM Calculate
```

عبارة الإيضاح والتعليق Comments

وهي عبارة غير تنفيذية تستخدم لتوضيح وشرح بعض خطوات البرنامج أو وظيفة البرنامج وبعض المعلومات عنه، ولذلك فمن الممكن أن يوجد أكثر من عبارة تعليق واحدة في بداية البرنامج وفي ثناياه. وصيغة هذه العبارة هي ظهور رمز علامة التعجب (!) Exclamation Mark في بداية السطر وبعد نهاية عبارة معينة وكل شيء يكتب بعد الرمز (!) يعتبر غير تنفيذي وهو توضيح أو شرح أو تعليق ما.

مثال:

```
PROGRAM Who
!This program reads a name
!Then prints it
READ *, NAME ! Input Statement
PRINT *, NAME ! Output Statement
END PROGRAM Who
```

ملاحظـــة: لا يجوز أن يحتوي البرنامج على اكثر من عبارة End.

عبارة تحديد الثوابت Parameters

أذا كان لدينا أسماء لثوابت معينة وأردنا تحديد هذه الثوابت خلال البرنامج فيتم ذلك من خلال المحدد في أحد عبارات النوع كما ذكر سابقاً ويكتب بالصيغة العامة التالية:

TYPE, PARAMETER:: CONS_1 = 100 , CONS_2 = 200, …

أمثلة:

```
REAL, PARAMETER :: pi = 3.1415926 , e =2.17828
INTEGER, PARAMETER :: MAXIMUM = 100
LOGICAL, PARAMETER :: TRUE = .true. , FALSE = .false.
CHARACTER(LEN=3), PARAMETER :: YES = "yes" , NO= "no"
```

الغرض من استخدام أسماء للثوابت هو زيادة الوضوح، وكذلك أعطاء فرصة للكمبيوتر من أي خطأ غير مقصود عند استعمال المتغيرات. وعلى سبيل المثال لاحظ البرنامج الآتي:

```
PROGRAM METER_TO_INCHES
! This program Convert Meter To Inches
IMPLICIT NONE
REAL :: METERS
REAL, PARAMETER :: INCHES_PER_METER=39.37
READ *, METERS
PRINT *, METERS * INCHES_PER_METER
END PROGRAM METER_TO_INCHES
```

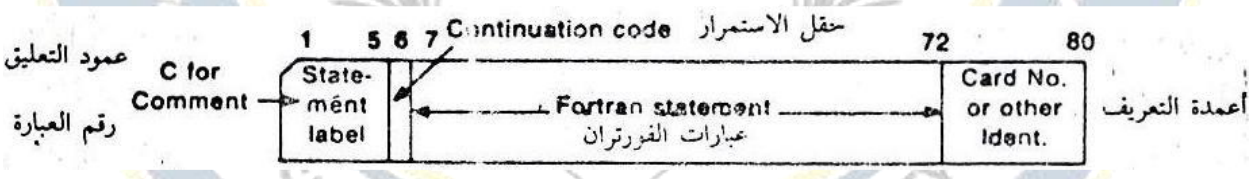
ملاحظـــة: يجب الانتباه إلى أن أسم الثابت لا يمكن تغيير قيمته خلال البرنامج فلا يمكن أن تظهر العبارة التالية في البرنامج مثلاً: (INCHES_PER_METER=39.3701) ولو ظهرت هذه الجملة فإن البرنامج سيظهر خطأ ينبه المبرمج لذلك.

القواعد الشكلية للبرامج المكتوبة بلغة FORTRAN

لقد تغيرت القواعد الشكلية التي كانت تستخدم في الإصدارات السابقة للغة **FORTRAN** ولأسباب تاريخية كانت لغة **FORTRAN** تكتب في بطاقات، فعليه أن صيغة عبارات **FORTRAN** يجب أن تكون متوافقة مع شكل هذه البطاقات وكانت كل بطاقة مقسمة الى أربعة أقسام متميزة هي:

1. الأعمدة من (1) إلى (5) تعرف بأعمدة ترقيم العبارة (Statement Label).
2. العمود السادس يعرف بعمود الاستمرارية (Continuation Code) وهو الذي يستعمل للدلالة على استمرارية العبارة أي أن ما موجود في البطاقة هو تكملة لما هو موجود في البطاقة السابقة لها.
3. الأعمدة من (7) إلى (72) تستعمل لكتابة عبارة فورتران وتعرف بمجال العبارة (Statement Field).
4. الأعمدة من (73) إلى (80) تستعمل لترقيم البطاقات الاختياري وتعرف بمجال ترقيم البطاقة أو حقل التعريف (Identification Field) وللمبرمج وضع أي حروف أو أرقام أو كلاهما معاً لتشير الى البرنامج والسطر حيث يفيد هذا الحقل في الرجوع لأي سطر في البرنامج دون الرجوع الى محتوياته، وكذلك يمكن ترتيب البرنامج إذا حدث أي خلل أو تغيير في هذا الترتيب.

أن كافة عبارات فورتران تخضع لهذا التصنيف عدا عبارات التعليق والشرح (Comment Statement) والتي تبدأ عادة بطبع حرف (C) في العمود الأول. كما في الشكل التالي:



أما في الإصدار FORTRAN90 فقد تحررت هذه القيود و أصبح شكل البرنامج كما موضح في الشكل في صفحة 16 ويمكن أن نذكر ان القواعد الشكلية السابقة لاتزال مقبولة في FORTRAN90 غير أن لا يجوز الخلط بين القواعد الشكلية السابقة والحالية في نفس الملف.

قواعد السطر بلغة FORTRAN90 (F90 Line Rules):

البرنامج الواحد يتكون من مجموعة من العبارات وكل عبارة تبدأ بسطر منفرد والسطر الواحد في لغة FORTRAN90 يحتوي على (132) رمزاً. لذا لا يمكن كتابة سطر أكثر من (132) رمزاً ولذلك يجب

أن تكتب العبارة بأكثر من سطر بإضافة الرمز& (Ampersand) الى نهاية السطر للأشارة الى استمرارية السطر ويمكن كتابة أسطر استمرار إلى حد 39 سطر وكما في المثال التالي:

```
PRINT *, "LINE ONE", &
          "LINE ONE" &
         , "LINE ONE"
```

كما أنه يمكن كتابة أكثر من جملة في لغة FORTRAN في سطر واحد وذلك بفصل الجملتين عن بعضهما بواسطة الفاصلة المنقوطة (;) كما في المثال: SUM = O ; PRODUCT = 1

ملاحظة: بإمكان المبرمج أن يحدد قيماً ابتدائية للمتغيرات عند الإعلان عنها بعبارات النوع مثل :

```
INTEGER :: SUM =O , PRODUCT = 1
REAL :: Offset = 0.1, Length = 10.0 , Tolerance = 1.E-7
```

العمليات الحسابية (Arithmetic Operation):

توجد خمس عمليات حسابية أساسية, وهذه العمليات مع الرموز المقابلة لها في لغة FORTRAN هي:

العملية	الرمز	مثال
الجمع	+	A+B
الطرح	–	A – B
الضرب	*	A*B تعني A.B
القسمة	/	A/B تعني A÷B
الرفع الى الأس	**	A**B تعني A^B

التعبير الحسابي (Arithmetic Expression):

كان الغرض الأساسي للبرمجة بلغة FORTRAN هو أجراء العمليات الحسابية، ثم تطورت فيما بعد لتشمل أجراء عمليات على رموز متنوعة. فالتعبير الحسابي هو أجراء العمليات الحسابية على بعض الثوابت والمتغيرات والدوال وذلك لإيجاد قيمة عددية، والتعبير الحسابي في صورته البسيطة يتكون من ثابت واحد أو متغير واحد أو دالة واحدة كما في المثال: A/B*(X+12.5) , (X+Y)**3 , T-R3

وفي لغة FORTRAN90 يمكن ربط المتغيرات والثوابت من النوع الى الرمزي (CHARACTER) بواسطة وسيلة الوصل (//). كما في المثال: X // Y // "abcde"

قواعد كتابة وحساب قيمة التعبير الحسابي

1. يجب أن لا يظهر رمزان من رموز العمليات الحسابية بجانب بعضها البعض مباشرة فمثلاً من الخطأ أن يترجم التعبير الجبري الى فورتران بالتعبير الحسابي A/-B حيث ظهر الرمزان - ،/ بجانب بعضها دون أي فاصل بينهما ولكن يمكن أن يترجم الى A/(-B) حيث استخدمنا القوس للفصل أو أن يترجم الى A/B- وهذا يعطي نفس القيمة المطلوبة.
2. قاعدة الأسبقية (RULE OF PRECEDENCE) ويتم تنفيذ التعبير الحسابي على أساس قاعدة الأسبقية الآتية:
   i. تنفيذ ما بداخل الأقواس ( ).
   ii. الدوال أن وجدت.
   iii. تنفيذ عمليات الرفع للاس **.
   iv. تنفيذ عمليات الضرب * أو القسمة / وحسب أسبقيتها من اليسار الى اليمين.
   v. تنفيذ عمليات الجمع + أو الطرح ــ وحسب أسبقيتها من اليسار الى اليمين.

ملاحظـــــــات مهمة:

1. في حالة وجود عدة أقواس داخلية وخارجية ((( ))) فتعطي الأسبقية لما هو داخل الأقواس الداخلية أولاً ثم الخارجية التي تليها ثم التي تليها وهكذا....
2. يجوز أن يكون الأس (القوة) تعبيراً وليس مجرد ثابت أو متغير وكذلك يجوز أن تكون الكمية التي ترفع إلى هذا الأس تعبيراً.
3. لا يسمح برفع قيمة سالبة إلى قوة حقيقية ولا برفع الصفر إلى قوة صفرية.
4. عند أجراء عملية حسابية من نفس النوع ينتج ذلك النوع.

   مثال على ذلك: $3./2. = 1.5\ (Real),\ \ 3/2 = 1\ (Integer)$
5. عند أجراء عملية حسابية مختلفة يحول الصحيح إلى حقيقي (كسر) وينتج حقيقي.
6. مثال على ذلك: $3./2 = 1.5\ (Real),\ \ 3/2. = 1.5\ (Real)$

مثال: حول كل علاقة رياضية مما يلي إلى عبارة واحدة مقابلة لها بلغة فورتران؟

1. $A = \frac{4}{3}\pi R^3$
2. $k = \left(\frac{\tau}{\mu}\right)^n$
3. $x = \frac{-B+\sqrt{B^2-4AC}}{2A}$
4. $y = (ax^4 + 10^{-5} + \alpha \cos^2 x)^{3/4}$

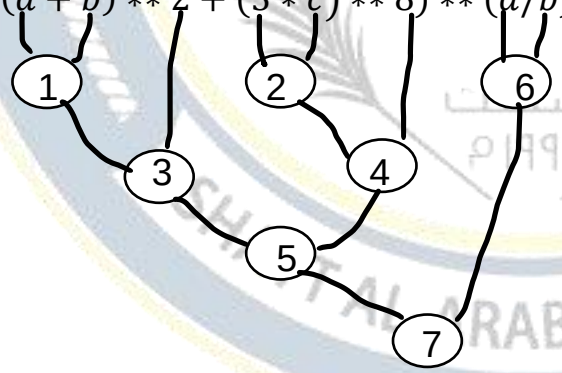
الحل:

1. $A = (4./3.) * 3.141597 * R ** 3$
2. $K = (Tau/Mu) ** n$
3. $X = (-B + Sqrt(B ** 2 - 4.* A * C))/2.* A$
4. $Y = (a * x ** 4 + 1.0E - 5 + alfa * \cos(x) ** 2) ** (3./4.)$

مثال: اكتب التعبير التالي بلغة فورتران وبين تسلسل العمليات $(x = [(a+b)^2 + (3c)^8]^{a/b})$.

الحل:

$$x = ((a+b) ** 2 + (3 * c) ** 8) ** (a/b)$$



عمليات المقارنة (Relational Operators):

يمكن أجراء عمليات مقارنة بين القيم العددية وتكون نتيجة هذه التعابير منطقية (True أو False)

الجدول التالي يمثل تعابير المقارنة المستخدمة في لغة فورتران:

الرمز الرياضي	المعنى	الرمز بلغة Fortran77	الرمز بلغة Fortran90	مثال
$=$	يساوي	.EQ.	==	$A == B$
$\neq$	لا يساوي	.NQ.	/=	$A/= C$
$<$	أصغر من	.LT.	<	$Y < X$
$\leq$	أصغر من أو يساوي	.LE.	<=	$Y <= Z$
$>$	أكبر من	.GT.	>	$M > N$
$\geq$	أكبر من أو يساوي	.GE.	>=	$M >= W$

العوامل المنطقية:

1. $.AND.$ ومعناها (و) وتكون صواب إذا المقارنة على الجانبين صواب.
2. $.OR.$: ومعناها (أو) وتكون صواب إذا المقارنة على الجانبين صواب أو أحدهما صواب
3. $.NOT.$ ومعناها (لا) وتكون صواب في حالة المقارنة على اليمين خطأ.
4. $.EQV.$ ومعناها (مكافئ) وتكون صواب إذا المقارنة على الجانبين صواب أو المقارنة على الجانبين خطأ.
5. $.NEQV.$ ومعناها (لا يكافئ) وتكون صواب إذا المقارنة على الجانبين أحدهما صواب والأخرى خطأ.

ملاحظـــــــات:

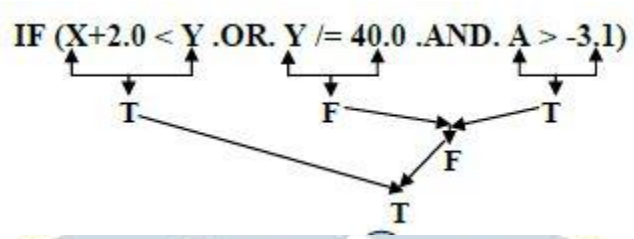
1. العامل $.NOT.$ يجب أن لا يسبق بأي عبارة أو متغير أما العاملين $.AND.$ و $.OR.$ فيجب أن يكونا مسبوقين ومتبوعين بتعبير أو متغيرات.
2. قاعدة الأسبقية للعوامل المنطقية تعطي أولوية التنفيذ لما هو داخل الأقواس الداخلية أولاً ثم الخارجية التي تليها ثم التي تليها وهكذا... ويعطي العامل $.NOT.$ الأولوية بين العوامل المنطقية ويليه العامل $.AND.$ ويليه العامل $.OR.$ ثم $.EQV.$ أو $.NEQV.$

مثال: افرض $A = 10, Y = 40.0 \ \& \ X = 30.0$ جد قيمة التعبير المنطقي التالي:

$$IF(X + 2.0 < Y \ .OR. Y/= 40.0 \ .AND. A > -3.1)$$

الحل:

نبدأ بالعامل $.AND.$ حسب قاعدة الأسبقية ثم العامل $.OR.$



الدوال المكتبية (Library Functions):

لكل حاسب الكرتوني مكتبة للدوال الرياضية وهي على شكل برامج معدة لبعض الصيغ الرياضية ومخزونة بصورة دائمة داخل الحاسب لأستخدامها عند الطلب من قبل المستفيدين ومخططي البرامج على أن تذكر فقط ضمن برامجهم ولها أولوية التنفيذ. وتحتوي لغة FORTRAN العديد من الدوال المكتبية المعرفة لدى هذه اللغة ونذكر فيما يلي بعضاً من هذه الدوال:

1. دوال تحويل نوع المتغير (Numeric Type Conversion Function):

قد يستخدم بعض المبرمجين الأنواع المختلفة (الصحيح والحقيقي) من المتغيرات في نفس العبارة الحسابية وأن عملية المزج بالأنواع هذه قد تؤدي في بعض الأحيان الى نتيجة غير متوقعة.

وعلى سبيل المثال عند أجراء عملية القسمة التالية : $PI/2$ حيث $PI = 3.1415$ كما هو معروف لذا تكون نتيجة القسمة تساوي (1) والسبب في ظهور مثل هذه النتيجة أن المقسوم عليه من نوع صحيح Integer ولذلك فإن الكمبيوتر يقوم بإسقاط الكسر في عملية القسمة أي أن الكمبيوتر يقوم بإتباع قاعدة القسمة وكأنها أعداد صحيحة ،ولإلزام الكمبيوتر بإجراء القسمة على أنها قسمة أعداد حقيقية وهنالك طريقتين:

أحداهما أن يكتب التعبير على الشكل الآتي: $PI/2.0$ بإضافة كسر عشري للمقسوم عليه. أو باستعمال الدالة المكتبية *REAL* لتغيير النوع الى الحقيقي كما يلي: $PI/REAL(2)$.

وهنالك دوال مكتبة أخرى لتغيير النوع مثل INT التي تقوم بأسقاط الجزء الكسري من قيمة المتغير.

فلو كان عندنا متغير X من نوع *REAL* وكانت قيمة X هي 1.5 فان نتيجة الدالة INT(X) ستكون (1) أي بأسقاط الكسر عنه.

هناك عمليات تغيير من نوع آخر تتم تلقائياً عند أجراء المساواة فبعد أجراء العملية التالية: $I = X$ على فرض ان قيمة X هي 1.5 وان المتغير I من نوع صحيح فان قيمة I تكون (1) أي ان قيمة X حولت الى نوع صحيح اولاً ثم أجريت عملية المساواة.

وعلى الرغم من التحويلات التلقائية الجارية عند المساواة فإنه من المستحسن التعود على أجراء عمليات تحويل النوع بشكل طاهر كما في: $I = INT(X)$ وذلك لتجنب الأخطاء غير المقصودة التي يصعب تعليلها عند تنفيذ البرنامج.

الجدول التالي يحتوي على بعض دوال التحويل المكتبية الشائعة في لغة FORATRAN90

Function	Meaning	Arg. type	Return Type
INT(x)	truncate to integer part x	REAL	INTEGER
NINT(x)	round nearest integer to x	REAL	INTEGER
FLOOR(x)	greatest integer less than or equal to x	REAL	INTEGER
FRACTION(x)	the fractional part of x	REAL	REAL
REAL(x)	convert x to REAL	INTEGER	REAL

أمثلة:

$$INT(-3.5) \Rightarrow -3$$

$$NINT(3.5) \Rightarrow 4$$

$$NINT(-3.4) \Rightarrow -3$$

$$FLOOR(3.6) \Rightarrow 3$$

$$FLOOR(-3.5) \Rightarrow -4$$

$$FRACTION\ (12.3) \Rightarrow 0.3$$

$$REAL\ (-10) \Rightarrow -10.0$$

2. الدوال الحسابية (Mathematical Functions):

تحتوي مكتبة لغة FORATRAN90 العديد من الدوال الحسابية المعروفة والجدول التالي يحتوي على بعض الدوال الشائعة الاستعمال في هذه اللغة:

Return Type	Arg. Type	mathematical formula	Meaning	Function F90
INTEGER	INTEGER	$\|x\|$	القيمة المطلقة	$ABS(X)$
REAL	REAL			
REAL	REAL	e^x	الدالة الأسية	$EXP(X)$
REAL	REAL	$\ln x$	اللوغارتيم الطبيعي	$ALOG(X)$ Or $LOG(X)$
REAL	REAL	$\log x$	اللوغارتيم الاعتيادي	$ALOG10(X)$ Or $LOG10(X)$
REAL	REAL	$\sqrt{x}$	الجذر التربيعي	$SQRT(X)$
REAL	REAL	$\sin x$	جيب الزاوية	$SIN(X)$
REAL	REAL	$\cos x$	جيب تمام الزاوية	$COS(X)$
REAL	REAL	$\tan x$	ظل الزاوية	$TAN(X)$
REAL	REAL	$\tan^{-1} x$	الظل المعكوس للزاوية	$ATAN(X)$
REAL	REAL	$\tanh x$	الظل الزائدي للمقطع	$TANH(X)$
REAL	REAL	$\sin^{-1} x$	جيب الزاوية المعكوس	$ASIN(X)$
REAL	REAL	$\cos^{-1} x$	جيب تمام المعكوس	$ACOS(X)$
REAL	REAL	$\sinh x$	جيب زائدي	$SINH(X)$
REAL	REAL	$\cosh x$	جيب تمام زائدي	$COSH(X)$
INTEGER	INTEGER		$Maximum\ of$ $x_1, x_2, \dots x_n$	MAX $(x_1, x_2, \dots x_n)$
REAL	REAL			
INTEGER	INTEGER		$Minimum\ of$ $x_1, x_2, \dots x_n$	MIN $(x_1, x_2, \dots x_n)$
REAL	REAL			

ملاحظـــــات:

1. يجب أن تكون الزوايا في الدوال المكتبية بالزوايا النصف قطرية.
2. لكي نتجنب الخلط بين الكميات الكسرية والكميات الصحيحة في التعبير الحسابي نستخدم دالة التحويل REAL.

العبارة الحسابية (Arithmetic Statement):

هي عبارة تستخدم لحساب قيمة تعبير حسابي ثم إعطاء هذه القيمة إلى متغير (فإذا كان لهذا المتغير قيمة سابقة فأنها تمحى وتحل محلها هذه القيمة الجديدة أي قيمة التعبير الحسابي). والصورة العامة للعبارة الحسابية هي: تعبير حسابي = أسم متغير (Variable Name = Arithmetic Expression).

وعلامة (=) هي ليست المساواة الموجودة في الرياضيات وأنما هي عملية أحلال بالبرمجة أي حل الطرف الأيمن (التعبير الحسابي) محل الطرف الأيسر (اسم المتغير) فمثلاً العبارة الحسابية A=B+C تفقد A قيمتها السابقة لتحل محلها قيمة B+C أما قيمة B,C فأنها لا تتغير.

وفي حالة ما إذا أختلف نوع المتغير عن نوع التعبير الحسابي فأن قيمة التعبير الحسابي تحسب أولاً حسب نوعه ثم تعدل هذه القيمة إلى صورة النوع الآخر قبل إعطائها للمتغير. فمثلاً إذا كان المتغير حقيقياً والتعبير صحيحاً فأن كل العمليات الحسابية في التعبير تجري بالأعداد الصحيحة ثم تحول النتيجة وهي عدد صحيح إلى الصورة الحقيقية قبل أعطائها للمتغير الموجود على يسار العبارة الحسابية وكذلك العكس فإذا كان المتغير صحيحاً والتعبير حقيقياً فإن قيمة التعبير الحقيقية المحسوبة تحول أولاً إلى قيمة صحيحة بحذف الجزء الكسري قبل إعطاء هذه القيمة للمتغير الصحيح على يسار العبارة.

نؤكد هنا على الصورة العامة للعبارة الحسابية حيث الطرف الأيمن عبارة عن تعبير حسابي أو متغير معرف سابقاً أو ثابت والطرف الأيسر عبارة عن اسم متغير واحد فقط. فمثلاً المعادلة الرياضية $X - Y = Z + T$ لا يمكن أن تكون عبارة حسابية وذلك لأن الطرف الأيسر فيها ليس متغيرًا واحداً وإنما هو تعبير حسابي.

عبارات الإدخال والإخراج (Input – Output Statements):

طرق إدخال البيانات (Input Data Methods):

هنالك طريقتان لإدخال البيانات هما:

1. **الطريقة المباشرة (Direct Mothod):** وتتم بإدخال البيانات مباشرة إلى الحاسبة من خلال العبارات الحسابية حيث يكون الطرف الأيمن في العبارة الحسابية ثابتاً. وعلى سبيل المثال

$$A = 2.5,\ M = 7\ ,\quad Z = 825.75/40.0$$

ويستخدم هذا النوع من الإدخال عندما تكون البيانات قليلة ولا يحبذ استخدامها حين الحاجة إلى عدد كبير من

البيانات أو في حالة تبديل البيانات بسبب استعمال البرنامج لأغراض أخرى.

2. **الإدخال باستخدام عبارة القراءة (Read Statement):** والصورة العامة لعبارة القراءة هي:

Read(m, n)List

m : يمثل رقم وحدة الإدخال المستخدمة ويعبر عنه بالرمز (*) ويعتمد على نوع الحاسبة .

n : رقم صحيح يدل على رقم عبارة الصيغة (Format) أو توضع * (نجمة) تعني (Free Format) أي صيغة حرة.

$List$: مجموعة المتغيرات المراد أدخالها مفصولة عن بعضها بفارزة (,).

مثال: $\mathrm{Read}(*,*)\mathrm{A, B, M}$

وتعني إدخال ثلاث متغيرات هما $\mathrm{A, B, M}$ باستخدام الصيغة الحرة (*). ويمكن كتابة عبارة القراءة بالصورة الأتية: $\mathrm{Read}\ *, \mathrm{A, B, M}$

مثال:

$Read(*,10)A, B, M$

$10\ \ Format\ (2F10.3,16)$

وتعني إدخال ثلاث متغيرات هما $\mathrm{A, B, M}$ باستخدام الصيغة (Format) المعبر عنها في عبارة رقم 10.

ويمكن كتابة عبارة القراءة بالصورة الأتية: $\mathrm{Read}(*, FMT = (2\mathrm{F10.3,16}))\mathrm{A, B, M}$

طرق إخراج النتائج (Output Results Methods):

هنالك عبارتان تستخدمان لإخراج النتائج هما:

1. **باستخدام عبارة الكتابة (Write Statement):** والصورة العامة لعبارة الكتابة هي:

Write(m, n)list

m : يمثل رقم وحدة إخراج النتائج المستخدمة ويعبر عنه بالرمز (*) ويعتمد على نوع الحاسبة .

n : رقم صحيح يدل على رقم عبارة الصيغة (Format) أو توضع * (نجمة) تعني (Free Format) أي صيغة حرة.

$List$: مجموعة أسماء المتغيرات المراد أخراجها مفصولة عن بعضها بفارزة (,).

مثال: $\mathrm{Write}(*,*)\mathrm{I,C}$

وتعني إخراج نتائج قيم المتغيرات I, C باستخدام الصيغة الحرة (*). والتي تطبع النتائج بطريقة التدوين اليائي

مثال:

$$Write(*,"(16,F10.3)")I,C$$

وتعني إخراج قيم متغيرات هما I, C باستخدام الصيغة (Format).

2. **باستخدام عبارة الطباعة (Print Statement):** والصورة العامة لعبارة الطباعة هي:

$$\mathbf{Print}\ *,\mathbf{List}$$

$List$: مجموعة أسماء المتغيرات المراد أخراجها مفصولة عن بعضها بفارزة (,).

مثال: $\mathrm{Print}\ *,\mathrm{A,B,C}$

وتعني إخراج نتائج قيم المتغيرات A, B, C باستخدام الصيغة الحرة (*).

مثال:

$$Print(3F8.3),A,B,C$$

وتعني إخراج قيم متغيرات هما A, B, C باستخدام الصيغة (Format).

ملاحظـــات:

1. في عبارة القراءة Read Statement يفضل استخدام الصيغة الحرة Format حتى يقوم المستخدم بأدخال البيانات بحرية تامة دون التقييد بأي صيغة Format.
2. في عبارة الكتابة Write Statement يفضل استخدام رقم عبارة الصيغة الحرة Format حتى يسهل على المستخدم إخراج النتائج بالشكل المطلوب (تستخدم في نسخ لغة FORATRAN القديمة)
3. في لغة FORTRAN90 يفضل استخدام عبارة Print في إخراج النتائج بشكليها (أستخدام صيغة حرة أو صيغة Format).
4. يتضح مما سبق أن البرنامج هو عبارة عن إدخال بيانات ومعالجة البيانات وإخراج النتائج.

عبارات الصيغة (Format statements):

هنالك عدة أنواع من الصيغ هي:

1. **صيغة الثوابت العددية الصحيحة (I Format):** تستعمل هذه الصيغة مع الأعداد الصحيحة (أي تستعمل مع متغيرات الفاصلة الثابتة) سواء في إدخال البيانات أو إخراج النتائج. والصورة العامة لها هي:

nIw

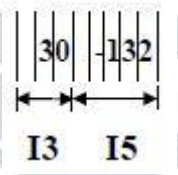
n : عدد الحقول في سجل الإدخال والأخراج.

I : رمز صيغة الأعداد الصحيحة.

w : عدد الأعمدة المحجوزة لكل متغير (حقل) متضمن الأشارة.

مثال: $PRINT(13,15), M, N$

فأذا كانت قيم كل من N, M هي $M = 30$, $N = -132$ فسوف يكون الإدخال كالاتي (تثقيب البيانات).



مثال: $PRINT(215), J, K$ وحيث $J = -12$ & $K = 159$

2. **صيغة الثوابت العددية الحقيقية (F Format):** تستعمل هذه الصيغة مع الأعداد الحقيقية (الكسرية) (أي تستعمل مع متغيرات الفاصلة السائبة) سواء في إدخال البيانات أو إخراج النتائج. والصورة العامة لها هي:

nfw. d

n : عدد الحقول في سجل الإدخال والأخراج.

F : رمز صيغة الأعداد الكسرية.

w : عدد الأعمدة الكلي في كل حقل.

d : عدد الأعمدة الى اليمين الفاصلة العشرية.

مثال: $PRINT(215), J, K$ وحيث $J = -12$ & $K = 159$

مثال: $PRINT$"(F9.3,F8.4)",X,Y وحيث $X = 12.213$ & $Y = -5.17$

```
|||12.213|-5.17|||
```

مثال: $PRINT$"(F10.5,2F7.4)",A,B,C وحيث $A = 296.28,\ B = 27.18$ & $C = -3.54$

```
||296.28|||27.18||-3.54|||
```

مثال: $PRINT$"(F10.3)",TOTAL وحيث $TOTAL = 2802.4$

```
||2802.4|||
```

3. **صيغة الأعداد الكسرية الأسية (E Format):** تستعمل هذه الصيغة مع الأعداد الكسرية الأسية والتي تكون قيمتها كبيرة جداً أو صغيرة جداً والصورة العامة لها هي:

nEw. d

n : عدد الحقول في سجل الإدخال والأخراج.

E : رمز صيغة الأعداد الكسرية والأسية وتطبع في الإدخال والإخراج.

w : عدد الأعمدة الكلي في كل حقل.

d : ويمثل عدد الأعمدة المخصصة لأرقام العدد فقط والتي محصورة بين الفاصلة العشرية والرمز E.

بشرط $(w - d \geq 7)$

مثال: $PRINT$"(E12.3)",AREA وحيث $AREA = 918000$

```
|||0.918E+06|||
```

مثال: $PRINT$"(E12.4,E10.3)",A,B وحيث $A = 251.832$ & $B = 0.0082468$

```
||0.2518E+03|0.825E-02||
```

ملاحظـــة: إذا كان الجزء الكسري في عبارة الصيغة $Ew.d$ أقل من الجزء الكسري الممثل للعدد المطلوب طبعه فإن الحاسب يقوم بتقريب الكسر إلى أقرب رقم.

مثال: $PRINT"(E13.6,E14.6)",H,S$ وحيث $H = 0.0000001357 \;\&\; S = 235.7978$

ملاحظـــة: إذا كان الجزء الكسري في عبارة الصيغة $Ew.d$ أكثر من الجزء الكسري الممثل للعدد المطلوب طبعه فإن الحاسب يقوم بتقريب أصفار على يمين العدد.

4. **صيغة A (A Format):** تستعمل هذه الصيغة لطباعة الثوابت الرمزية والصورة العامة لها هي:

$$\mathrm{A}xx$$

A : رمز الصيغة.

xx : عدد المواقع الموجودة في حقل هذه الصيغة وهي اختيارية (Optional).

مثال: $PRINT"(A4,13)",Y,IN$ وحيث $Y = ZAID, IN = 96$

مثال: $PRINT"(A)","FORTRAN90"$

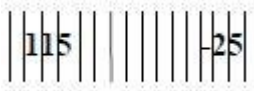
5. **صيغة فسحة أو إهمال الحقول (X Format):** تستعمل هذه الصيغة لترك فراغات بين الحقول والصورة العامة لها هي:

nX

n : عدد الأعمدة التي تترك فارغة.

X : صيغة فسحة الحقول.

مثال: $PRINT"(14,5X,16)",I,J$ وحيث $I = 115, J = -25$



مثال: $PRINT"(F12.6,4X,F6.2)",C,P$ وحيث $C = 927123456, P = 191.24$

927.123456 191.24

6. **صيغة تحديد الحقل T (T Format):** تستعمل هذه الصيغة لإلزام الكمبيوتر بالطباعة في عمود يحدد رقمه والصورة العامة لها هي:

Tn

n : رقم العمود المراد الطباعة فيه.

T : صيغة تحديد الحقل.

مثال: $PRINT"(A,T10,12)","FORTRAN90"$

FORTRAN 90

7. **صيغة القيود المتعدد (Multiply Record Format):** تستعمل هذه الصيغة عند إدخال أو إخراج عدد من السطور حيث يستعمل الخط المائل (Slash) (/) لإنهاء السطر.

مثال: $PRINT"(2F7.3/215)",C,D,I,J$ وحيث $C = 22.35, D = 91.62, I = 753$ & $J =$ 8024

22.35 91.62
753 8024

ملاحظـــــات:

1. يمكن استعمال عبارة صيغة واحدة تستخدم للقراءة والطباعة مثل:

$READ(*,10)I,J$

$WRITE\ (*,10)I,J$

10 $FORMAT\ (215)$

2. توضع فارزة لفرز صيغة عن صيغة أخرى في عبارة الصيغة.

3. اذا كان الحجز في صيغة FORMAT أقل من عدد الأعمدة أو المواقع فأن الطبع يظهر**** بقدر الحجز.

أمثلـــــــــة:

- حول كل علاقة رياضية مما يلي إلى عبارة واحدة مقابلة لها بلغة فورتران؟

$1.\ \emptyset = \sqrt{|m-n|}$ $2.\ X = \frac{\alpha^3.10^{22}}{\ln\left|\cos\frac{\emptyset}{3}\right|}\left|e^{\sin\emptyset}\right|$

$3.\ Y = \ln(|\sec x + \sin x|)$ $4.\ \theta = \sqrt[3]{\frac{e^{2x}}{\sec^2 x.\csc 2x}}$

الحل:

1. THETA=SQRT(ABS(REAL(M-N)))
2. X=ALFA**3*1.0E22*ABS(EXP(SIN(FI))) / LOG(ABS(COS(FI/3.)))
3. Y=LOG(ABS(1.0/COS(X)+SIN(X)))
4. THETA=(EXP(2*X)/((1./COS(X))**2*1/SIN(2*X)))**(1./3.)

- فيما يلي مجموعة من العبارات FORTRAN90 المطلوب بيان أي من هذه العبارات صحيحة وأيها غير صحيحة وبالنسبة للعبارات غير الصحيحة وضح الخطأ؟

$1.\ B = SQRT(X + 3E3)$

$2.\ Y = 3.2X + A ** 3/-4.0$

$3.\ X - Y = ALOG(T + Q)$

$4.\ J = J + 1$

$5.\ X\ Y = 12.5$

$6.\ XI = JI + K - 4$

$7.\ GAMA = A = (B + C) ** 2$

$8.\ PI = 3.14159$

$9.\ 12.5 = X - A$

$10.\ J = SIN(a + \cos(x))/4.0$

الحل:

1. عبارة صحيحة.
2. عبارة غير صحيحة لا يوجد رمز لعملية حسابية بين 3.2 والمتغيرX وكذلك الرمزان / و – متجاوران مباشرة.
3. عبارة غير صحيحة – الطرف الأيسر تعبير حسابي وليس متغيرًا واحداً.
4. عبارة صحيحة.
5. عبارة غير صحيحة – لاحتواء الطرف الأيسر على فراغ فصل بين مكونات اسم المتغير.
6. عبارة صحيحة.
7. عبارة غير صحيحة – لاحتواء العبارة على علامتين يساوي (أحلال).
8. عبارة صحيحة.
9. عبارة غير صحيحة – الطرف الأيسر ثابت وليس اسم متغير.
10. عبارة صحيحة.

واجب بيتي (Home Work):

1. حول كل علاقة رياضية مما يلي إلى عبارة واحدة مقابلة لها بلغة فورتران؟

1. $T = ln\left|\sin\frac{X}{3}\right| + \frac{1}{e^{2X}+4}$
2. $C = \sqrt{|COS(\alpha - n.b)|} + \ln|m-n|$
3. $H = 1 + X + X^2/2! + X^3/3! + X^4/4!$
4. $Y = \frac{1}{2}r^2(\theta - \sin\theta) + \frac{4}{3}\pi r^3$
5. $\sigma = \sqrt{\frac{y(x-\bar{x})^2}{n} - \frac{yn}{n^2}}$
6. $\beta = \frac{x^{3/2}(1+\tan^2 x)}{10^{-5}+\alpha x^4}$

2. فيما يلي مجموعة من العبارات FORTRAN90 المطلوب بيان أي من هذه العبارات صحيحة وأيها غير صحيحة وبالنسبة للعبارات غير الصحيحة وضح الخطأ؟

1. $Z2 = A * -B + C ** 4$
2. $X = SQRT(SQRT(X))$
3. $I + 1 = I$
4. $R = 16.9X + AB$
5. $SQRT(49.0) = 7.0$
6. $V - 3.96 = X ** 1.67$
7. $K12345 = I ** J$
8. $W = SQRT(YZ + 4E3 = 12.5)$
9. $-A = 5.0 * D - F$
10. $R2 = EXP(SIN9(X/2.0)$

3. ماهي القيمة المتوقعة لكل من SQ, IA, RN, I,X,Y بعد تنفيذ البرنامج التالي:

```
PROGRAM HOME_WORK
IMPLICIT NONE
REAL ::I,X,Y
INTEGER :: N=6,J=4,SQ=,RN,IA
SQ=6.8+N/4.
IA=4.2+N/8
RN=SQ+IA/6
X=N/REAL(J)
Y=1/3*X-1.2
PRINT*,SQ,IA,RN,I,X,Y
END PROGRAM HOME_WORK
```

Chapter Two

Control Constructs in FORTRAN90

المقدمة (Introduction)

لقد كان يعرف مثل هذا الفصل في نماذج FORTRAN القديمة مثل FORTRAN77 باسم عبارات السيطرة وكانت تشمل عبارات الانتقال (Go to Statements) وعبارات الانتقاء (If Statements).

اما النموذج الأخير FORTRAN90 فقد دخل عالم اللغات المتقدمة والتي تعرف باللغات البنيوية المبنية من وحدات منفصلة Structural Programming وبهذا التغير الجذري للغة اصبح البرنامج عبارة عن وحدات بنائية تتكون كل وحدة من تركيبة تسمى Block. وتقسم هذه التركيبات الى تركيبة الانتقاء If Construct وتركيبة الاختيار Case Construct ويفضل إقران أي تركيبة باسم وهذا الاسم يخضع لقواعد أسماء المتغيرات في FORTRAN90 من حيث الشكل وفي العادة تقترن بداية التركيبة بهذا الاسم كما يظهر في نهاية التركيبة تماماً كما في اسم البرنامج.

تركيبة الانتقاء (If Construct)

هناك ثلاثة أنواع من تركيبات الانتقاء في لغة FORTRAn90 هي:

1. **تركيبة (If…Then Construct):** وهي ابسط شكل لتركيبة الانتقاء والصيغة العامة لها هي:

$$IF(Logical\ expression)THEN$$
$$\left.\begin{array}{l}\ldots\ldots.\\ \ldots\ldots.\\ \ldots\ldots.\end{array}\right\} block\ of\ statements$$
$$END\ IF$$

وتعني هذه التركيبة اذا كان التعبير المنطقي صواب True أي كان الشرط متحققاً فأن مجموعة عبارات فورتران يتم تنفيذها واذا لم يتحقق الشرط فأن العبارات لا تتنفذ. وفي أي من الحالتين فان العبارة التالية التي تنفذ هي العبارة التي تلي عبارة END IF.

مثال: اكتب برنامج لإيجاد القيمة الصغرى لقيمتين معلومتين؟

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

$$IF(Logical\ expression)THEN$$
$$\left.\begin{matrix}\dots\dots\\ \dots\dots\\ \dots\dots\end{matrix}\right\} block\ of\ statements\ 1$$
$$ELSE$$
$$\left.\begin{matrix}\dots\dots\\ \dots\dots\\ \dots\dots\end{matrix}\right\} block\ of\ statements\ 2$$
$$END\ IF$$

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 مجموعة عبارات فورتران T وفي أي من الحالتين فأن العبارة التالية في التنفيذ هي العبارة التي تلي عبارة END IF.

مثال: اكتب 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
```

1. تركيبة IF العامة (General from of IF Construct): وهي تركيبة الأنتقاء العامة والصيغة العامة لهذه العبارة هي:

$$IF(Logical\ expression)THEN$$
$$\left.\begin{matrix}\dots\dots\\ \dots\dots\\ \dots\dots\end{matrix}\right\} block\ of\ statements\ 1 \quad T1$$
$$ELSE\ F(Logical\ expression)THEN$$
$$\left.\begin{matrix}\dots\dots\\ \dots\dots\\ \dots\dots\end{matrix}\right\} block\ of\ statements\ 2 \quad T2$$
$$ELSE\ F(Logical\ expression)THEN$$
$$\left.\begin{matrix}\dots\dots\\ \dots\dots\\ \dots\dots\end{matrix}\right\} block\ of\ statements\ 3 \quad T3$$
$$ELSE$$
$$\left.\begin{matrix}\dots\dots\\ \dots\dots\\ \dots\dots\end{matrix}\right\} block\ of\ statements\ 4$$
$$END\ IF$$

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

مثال: اكتب برنامج 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 ان 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 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 لتنفيذ رقم **n ,** وكمثال على عبارة الانتقال غير المشروط : **GO TO 30**

ملاحظة: 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 أيضا بعبارة **GO TO** الحسابية (Computed Go To Statement). والصيغة العامة لهذه العبارة هي: $\mathbf{GO\ TO}\ (n_1, n_2, n_3, \dots, n_m), I$

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

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

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 وأذا كانت $I = 2$ فأن التحويل سينتقل للعبارة رقم n_2 وهكذا. وكمثال على عبارة الانتقال غير المشروط : $\mathbf{GO\ TO}\ (10, 20, 30), N$ وهذا يعني: اذا كان $N = 1$ سينتقل التنفيذ الى عبارة رقم 10. و إذا كان $N = 2$ سينتقل التنفيذ الى عبارة رقم 20. و 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): هنالك نوعان هما:

1. عبارة IF الحسابية (Arithmetic IF Statement):

تأخذ عبارة IF الحسابية الشكل العام التالي: $\mathbf{IF(A)n_1, n_2, n_3}$

حيث :**A** : أما متغير أو تعبير حسابي يراد استعماله كشرط يتوقف على قيمته تنفيذ أحدى العبارات التي أرقامها $\mathbf{n_1, n_2, n_3}$.

$\mathbf{n_1, n_2, n_3}$:أرقام عبارات التنفيذ.

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

1. قيمة A سالبة ينتقل التنفيذ الى جملة رقم $\mathbf{n_1}$.
2. قيمة A صفر ينتقل التنفيذ الى جملة رقم $\mathbf{n_2}$.
3. قيمة A موجبة ينتقل التنفيذ الى جملة رقم $\mathbf{n_3}$.

مثال: $IF(X-4.0)10,20,30$

الحل: هذا يعني:

1. إذا كانت قيمة التعبير $(X-4.0)$ سالبة يحول التنفيذ الى عبارة رقم 10.
2. إذا كانت قيمة التعبير $(X-4.0)$ صفر يحول التنفيذ الى عبارة رقم 20.
3. إذا كانت قيمة التعبير $(X-4.0)$ موجبة يحول التنفيذ الى عبارة رقم 30.

مثال: $IF(K-100)10,20,10$

الحل: هذا يعني تحويل التنفيذ الى العبارة رقم 20 اذا كانت $K=100$ ويحول التنفيذ الى عبارة رقم 10 اذا كانت $K\neq 100$.

ملاحظة: 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. وعليك الاستعاضة عنها عند كتابتك لبرنامج يحمل تركيبة IF.

2. عبارة IF المنطقية (Logical IF Statement):

تأخذ عبارة IF الحسابية الشكل العام التالي: $\mathbf{IF(B)\ S}$

حيث :**B** : تعبير منطقي قيمته *True* او *False*.

B : جملة تنفيذية (مثل *GO TO* عبارة حسابية أو عبارة إدخال أو أخراج أو توقف....).

ملاحظات على عمل عبارة *IF* المنطقية:

- لا يجوز أن تكون (*S*) عبارة (*IF*) المنطقية أو عبارة (*DO*) التكرارية أو عبارة الإنهاء (*END*).
- في حالة (*B*) صواب يتم تطبيق (*S*) وفي حالة (*B*) خطأ لا يتم تطبيق (*S*) ويستمر البرنامج في تطبيق العبارة التالية: $IF\ (A > B) GO\ TO\ 20$

مثال:

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

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

```
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 FORTRAN90 وهذه التركيبة تشكل بديلاً لعبارة GO TO الحسابية في لغة FORTRAN90 القديمة وكذلك بديلاً لتركيبة الأنتقاء (IF Construct) ولكنها تتميز عنها وضوحها. 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 انطبقت قيمة case selector 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 إحدى 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 إحدى الجمل للطباعة بناءً على قيمة السرعة (SPEED) المذكورة 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 قيمة المتغير (SPEED) من 50 الى 89 فإن الجملة "LEGAL" تطبع وأذا كانت قيمة المتغير (SPEED) من 10

الى 49 فإن الجملة "LEGAL" تطبع "SLOW" وعدا ذلك من القيم سواء أكبر من 120 أو أقل من 10 فتطبع الجملة :

"DANGEROUS; TOO SLOW OR TOO FAST"

مثال:

```
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") أو ("YELLOW") أو ("GREEN") وأما ماعدا ذلك فيتم تنفيذ الجملة الموجودة في الفراغ (CASE DEFAULT).

مثال:

```
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 من التركيبة (X = -Y) هو الذي يجري تنفذه. 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.

مثال:

```
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 (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 المتغيرMONTH مع أي من القيم المدرجة في هذه الجملة كي ينفذ هذا الفرع. لاحظ أن مجال 7:8 أستخدم بهذا الشكل بسبب تعاقب 7 أو 8 وكان يصح أن يكتب المجال على شكل 8,7 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.

مثال:

أكتب برنامج لقراءة قيمة X ثم احسب Y من التعبيرين؟

$$Y = 5X^2 + 0.5X + 0.95 \quad IF\ X \leq 2.1$$
$$Y = 7X^2 + 0.7X + 0.53 \quad IF\ X > 2.1$$

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
```

مثال: أكتب برنامج لقراءة لإيجاد قيمة X من التعابير التالية:

$$X = \sqrt{A^2 + B^2 + C^2} \quad IN\ CASE\ J = 1$$
$$X = |A^2 + 2B - C| \quad IN\ CASE\ J = 2$$
$$X = \ln(A + B + C) \quad IN\ CASE\ J = 3$$

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 لان المتغير العددي المستخدم في عبارة SELECT CASE يجب أن يكون متغير صحيح INTEGER.

واجب بيتي (Home Work):

1. أكتب برنامجاً لقراءة قيمة المتغيرين n , x وحساب وطبع قيمة Y المعرفة:

$$Y = x + \cos x + \tan^{-1}\sqrt{x} \qquad if\ n < 0$$
$$Y = (x - \sin x)^2 - \sin 2x \qquad if\ n = 0$$
$$Y = (x + e^{2x} + \sin(x - 1.5))^2 \qquad if\ n > 0$$

2. 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 أخر. وأن يطبع البرنامج أيضاً عبارة ERROR أذا كانت قيمة المتغير C عدا ذلك.
3. أكتب برنامجاً لحساب مجموع المتسلسلة التالية:

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

4. 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 أن القيم (P = 3 , Q = 5 , R = 2 , T = 10.2 , K = 8)

5. إذا فرضنا أن 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))
```

Chapter Three

DO Construct in FORTRAN 90

المقدمة (Introduction)

في كل الأمثلة السابقة كانت تنفذ أي عبارة مرة واحدة ولكن في كثير من الأحيان نحتاج الى تكرار مجموعة من العبارات لعدد من المرات ولتحقيق هذا الغرض نستخدم تركيبة الدوران أو التكرار DO (DO Construct) وتسمح تركيبة DO بتكرار أي جملة أو مجموعة جمل عدداً معيناً من المرات أو تترك عدد تكرار الجمل مفتوحاً كي تنهيه جملة ظرفية (EXIT , CYCLE). وتستعمل تركيبة DO لإيجاد قيم المتواليات والمصفوفات وقيم عمليات الجمع والضرب المتكررة وأي عملية تحتاج إلى عدد من التكرارات.

هناك نوعين من تركيبة التكرار DO Construct في لغة FORTRAN90 هي:

1. **تركيبة الدوران المحددة (Loop Control With a DO Construct):** هذا النوع من تركيبة الدوران تكون محددة التكرار أي عدد مرات التكرار تكون محددة مسبقاً. والشكل العام لتركيبة DO المحددة كما يلي:

$$DO\ counter = initial, final, increment$$
$$\left.\begin{matrix}\dots\\ \dots\\ \dots\end{matrix}\right\} block\ of\ statements$$
$$END\ DO$$

حيث:

$counter$: متغير عددي (يفضل متغير صحيح للحلقات الدورية) يمثل عداد الحلقة التكرارية ويسمى المؤشر أيضاً.

$initial$: القيمة الابتدائية للعداد وهي أما أن تكون ثابت أو متغير عددي معرف أو تعبير رياضي.

$final$: القيمة النهائية للعداد وهي أما أن تكون ثابت أو متغير عددي معرف أو تعبير رياضي.

$increment$: مقدار الزيادة للعداد وهي أما أن تكون ثابت أو متغير عددي معرف أو تعبير رياضي وهي تضاف في كل مرة للقيمة الابتدائية للعداد ($initial$) حتى يصل إلى القيمة النهائية للعداد ($final$).

مثال:

```
DO Counter = 1, 10,2
PRINT *, Counter
END DO
```

أي أن قيمة عداد الحلقة الابتدائية هو 1 والقيمة النهائية 10 وقيمة الزيادة المنتظمة هي 2 وفي هذا المثال تتكرر جملة (PRINT *, Counter) خمس مرات ويطبع الأرقام الفردية من 1 الى 10.

مثال:

```
K = 10
DO Counter = K, K*K , 2
PRINT *, Counter
END DO
```

مثال:

```
DO Counter = 100, 1 , - 1
PRINT *, Counter
END DO
```

أن هذا الجزء من البرنامج يطبع الأرقام من 100 إلى 1 ويوضح أيضاً أن تسلسل قيمة عداد الحلقة لا يشترط أن يكون متصاعداً فإذا كان مقدار الزيادة سالباً فأن القيم التي يأخذها عداد الحلقة تكون متناقصة.

ملاحظات مهمة لاستخدام تركيبة DO المحددة:

1. إذا لم يذكر increment (مقدار الزيادة) يفهم انه يساوي 1. وعلى سبيل المثال: (DO I = 1, N) يعني هذا المثال أن الدليل I يأخذ القيمة الابتدائية 1 والقيمة النهائية N بزيادة مقدارها 1 في كل تكرار.
2. لا يجوز أن يكون مقدار الزيادة (increment) يساوي صفر.
3. لا يجوز أعادة تعريف أي من متغيرات تركيبة التكرار داخل حلقة التكرار ومثال ذلك:

```
DO I = 1 , 20
I = I+1 ←———— لا يجوز
X(I) = Y(I) + Z(I)
END DO
```

4. عداد الحلقة ويسمى أحياناً متغير الدوران هو متغير عددي (صحيح أو حقيقي) ولا ينصح باستخدام الأعداد الحقيقية لمتغير الدوران.

5. تركيبة التكرار لا تنفذ في الحالات التالية:

i) إذا كان مقدار الزيادة أكبر من صفر والقيمة الابتدائية أكبر من القيمة النهائية أي:

$$initial > final, inc. > 0$$

وعلى سبيل المثال: *DO I = 20 , 10 , 2*

ii) إذا كان:

$$initial < final, inc. < 0$$

وعلى سبيل المثال: *DO I = -5 , 10 , -1*

6. عدد مرات تكرار تركيبة التكرار يمكن إيجادها من العلاقة الآتية:

$$NO. = INT\left(\frac{final - initial + increment}{increment}\right)$$

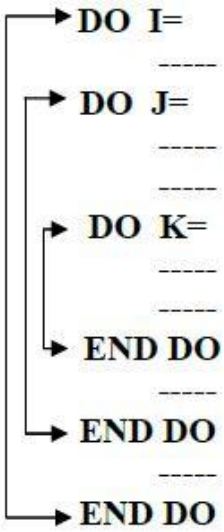
مثال: أكتب برنامج لإيجاد قيمة المفكوك N؟

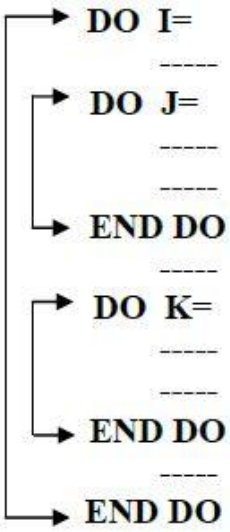
الحل:

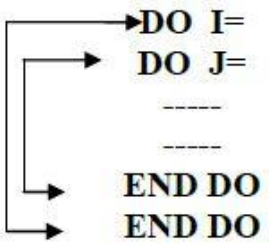
```
Program Factorial
Implicit none
Integer :: N, I, Fact
Read*,N
Fact =1
Do I = 1, N
Fact=Fact * I
End Do
Print*, N , "!=", Fact
End Program Factorial
```

حلقات التكرار المتداخلة (Nested Do Loops): قد تحتوي أحدى حلقات التكرار على حلقة تكرار أخرى أو أكثر وتسمى الحلقات التكرارية في هذه الحالة بالحلقات المتداخلة والقواعد المستخدمة للحلقات المتداخلة هي نفس القواعد المستخدمة لحلقة تكرار واحدة. ومع ذلك يمكن مراعاة النقاط الآتية:

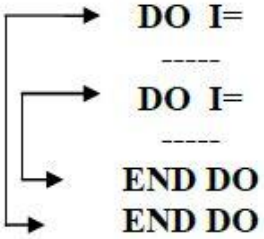
i. يجب أن يختلف عداد الحلقات الداخلية عن عداد الحلقات الخارجية.
ii. يجب أن تكون الحلقة التكرارية موجودة بكاملها داخل مدى الحلقة التكرارية الخارجية أي غير متقاطعة.







حلقات متداخلة مسموح بها



حلقات تكرارية متداخلة غير مسموح بها لأن لها نفس العداد I

مثال: أكتب برنامجاً لإيجاد مجموع مفكوك الأرقام الزوجية المحصورة بين 1,19؟

$$Sum = 2! + 4! + 6! + 8! + \cdots + 18!$$

الحل:

```
PROGRAM SUM_FACTORIAL
IMPLICIT NONE
INTEGER :: I, J, SUM , FACT
SUM = 0
DO I = 2, 18, 2
FACT = 1
DO J = 1, I
FACT = FACT* J
END DO
SUM = SUM + FACT
END DO
PRINT *, 'SUM =', SUM
END PROGRAM SUM_FACTORIAL
```

2. **تركيبة الدوران غير المحددة (Loop Control With a DO Construct):** هذا النوع من تركيبة الدوران يكون فيه التكرار غير محدد أي عدد مرات الدوران فيه مفتوحاً بحيث تتحكم فيه جمل ظرفية من داخله. والشكل العام لتركيبة DO غير المحددة كما يلي:

DO
...
$blockof\ statements$
$IF(logical_{expression})\ EXIT$
...
$IF(logical_{expression})\ CYCLE$
...
$blockof\ statements$
$END\ DO$

حيث:

$EXIT$: جملة ظرفية أذا حدث أن نفذت فإن الدوران ينتهي وينتقل الكمبيوتر إلى تنفيذ البرنامج بعد ترك التكرار (أي تنفيذ العبارات بعد جملة END DO).

$CYCLE$: جملة ظرفية أذا حدث أن نفذت داخل التكرار فإنها تقفز عن مجموعة الجمل اللاحقة لها وتبدأ دورة جديدة.

ملاحظة: أن هاتين الجملتين (EXIT , CYCLE) ظهرت في لغة FORTRAN90 (أي لم تكن موجودة في نسخ فورتران القديمة) وهي تغني عن بعض استخدامات GOTO في نسخ فورتران القديمة. وكما ذكرنا سابقاً فإنه يجب التخلص من GOTO قدر المستطاع.

مثال: أكتب برنامج لطباعة قوى العدد 3 أي $3^0 + 3^1 + 3^2 + 3^3 +$, ... ويتوقف الدوران عندما تتجاوز قيمة 3^n مقدار 2000.

الحل:

```
PROGRAM SOME_POWER_OF_3
IMPLICIT NONE
INTEGER :: POWER_OF_3
POWER_OF_3 =1                    ! THE ZERO POWER_OF_3
DO
```

```
PRINT *, POWER_OF_3
POWER_OF_3 = 3 * POWER_OF_3
IF (POWER_OF_3 >= 2000 ) EXIT
END DO
END PROGRAM SOME_POWER_OF_3
```

في هذا المثال نلاحظ أولاً خلو تركيبة التكرار من جملة تحديد عدد مرات الدوران (Loop Control) في بدايته. وهذا يعني أن عدد مرات الدوران يكون مفتوحاً.

لاحظ الجملة الظرفية (IF (POWER_OF_3 >= 2000) EXIT) حيث يتم تنفيذ الجملة الظرفية EXIT عند تحقيق الشرط (التعبير المنطقي POWER_OF_3 >= 2000) وبالطبع فإنه إذا نفذت جملة EXIT فإن التنفيذ ينتقل الى الجملة التي تلي جملة END DO وهي نهاية البرنامج.

ومن الأمثلة العملية على الدوران المفتوح هو أن تتحكم أحدى معطيات الإدخال في إنهاء الدوران. وغالباً ما تنتهي سلسلة من المعطيات الداخلة برقم مقصود كما في المثال التالي:

```
PROGRAM AVERAGE
IMPLICIT NONE
INTEGER :: NUMBER , NUMBER_OF_NUMBERS , SUM
SUM =0
NUMBER_OF_NUMBERS = 0
DO
READ *, NUMBER
IF (NUMBER < 0 ) EXIT
PRINT *, "Input data :", NUMBER
SUM = SUM + NUMBER
NUMBER_OF_NUMBERS = NUMBER_OF_NUMBERS + 1
END DO
PRINT *, " The average of the numbers is " , REAL(SUM) / NUMBER_OF_NUMBERS
END PROGRAM AVERAGE
```

يقوم هذا البرنامج بإدخال أرقام صحيحة موجبة ثم يقوم بجمعها وإيجاد معدلها. أن أي رقم سالب في سلسلة الأرقام المدخلة يعني نهاية البرنامج وهذا تتحكم به الجملة الشرطية IF (NUMBER < 0) EXIT.

ولاستعراض جملة CYCLE نأخذ المثال التالي:

```
PROGRAM ODD_NUMBERS
IMPLICIT NONE
INTEGER :: NUMBER , NUMBER_OF_ODD_NUMBERS
NUMBER_OF_ODD_NUMBERS = 0
DO
READ *, NUMBER
PRINT *, "Input data :", NUMBER
IF (NUMBER < 0 ) THEN
EXIT
ELSE IF (MODULO(NUMBER, 2) == 0) THEN
CYCLE
ELSE
NUMBER_OF_ODD_NUMBERS = NUMBER_OF_ODD_NUMBERS + 1
END IF
END DO
PRINT *, " The number of odd numbers is",
 NUMBER_OF_ODD_NUMBERS
END PROGRAM AVERAGE
```

وفي هذا البرنامج نقوم بإدخال أرقام معينة وتستخدم هنا تركيبة (IF- THEN) الشرطية فعند إدخال أول رقم سالب فسوف تنتهي الحلقة التكرارية من خلال العبارة :IF (NUMBER < 0) THEN EXIT واذا لم يتحقق هذا الشرط فيتم اختبار الرقم المدخل لمعرفة ما أذا كان هذا الرقم فردياً وتتم عملية الاختبار هذه بواسطة الدالة المكتبية MODULO(NUMBER, 2) وتكون قيمة الدالة المكتبية هي قيمة باقي قسمة NUMBER /2 وبالطبع تكون صفراً للأعداد الزوجية. لاحظ أنه عند انطباق هذا الشرط يقوم البرنامج بتنفيذ جملة CYCLE التي تنقل تركيبة الدوران إلى دورة جديدة.

مثال: أكتب برنامجاً لإيجاد قيمة Y من العلاقة الأتية:

$$Y = \sum_{a=1}^{5} \sum_{x=3}^{12} \frac{X^a(X-a)}{a(X+a)}$$

الحل:

```
PROGRAM VALUE_OF_Y
IMPLICIT NONE
REAL :: A , X , Y
Y = 0.0
DO A = 1., 5.
```

```
DO X = 3., 12.
Y = Y + X**A*(X-A) /(A*(X+A))
END DO
END DO
PRINT *, "Y = " ,Y
END PROGRAM VALUE_OF_Y
```

مثال: أكتب برنامجاً لقراءة درجات مئة طالب $S1, S2, S3, S4, \dots, Si, \dots, S100$ ثم أحسب:

1. الدرجة المتوسطة $AVG = \left(\sum_{i=1}^{100} Si\right)/100$
2. الفرق بين درجة كل طالب والدرجة المتوسطة. $Di = Si - AVG,\ i = 1,2,3,\dots,100$

الحل:

```
PROGRAM DEGREES_OF_STUDENTS
IMPLICIT NONE
REAL :: SUM = 0.0 , S(100) , DEGREE(100) , AVERAGE
INTEGER :: I
DO I = 1, 100
READ *, S(I)
SUM = SUM + S(I)
END DO
AVERAGE = SUM / 100.
PRINT "(T5,A, F8.3)", "AVERAGE =", AVERAGE
DO I = 1, 100
DEGREE(I) = S(I) – AVERAGE
PRINT *, DEGREE(I)
END DO
END PROGRAM DEGREES_OF_STUDENTS
```

ملاحظة: من الأخطاء الشائعة التي يقع فيها بعض الطلبة في البرنامج السابق وضع العبارة التي تعطى SUM قيمته الابتدائية داخل حلقة التكرار DO ومثال على ذلك:

```
DO I = 1, 100
READ *, S(I)
SUM = 0.0
SUM = SUM + S(I)
END DO
```

وهذا خطأ لأنه لا يؤدي إلى جمع قيم S(I) كلها بإضافتها واحدة بعد الأخرى إلى SUM حيث أنه بعد إضافة أي من قيم S(I) الى SUM وقبل إضافة S(I) التي تليها نعيد قيمة SUM الى قيمتها الابتدائية

SUM = 0.0 مرة أخرى حين نزيد قيمة I ونبدأ في تنفيذ الخطوات تحت عبارة DO مرة جديد ولذلك فيجب أن توضع العبارة SUM = 0.0 قبل عبارة DO حتى لا تنفذ هذه العبارة ألا مرة واحدة فقط.

واجب بيتي (Home Work):

1. أوجد قيمة J عند تنفيذ البرنامج الآتي:

```
PROGRAM VALUE_OF_J
IMPLICIT NONE
INTEGER :: M = 2 , N = 3 , J = 0 , I, L , K
DO I = 1, 2
DO K = 2, 3
M = M + 3
END DO
DO L = 1, 3
N = N + 4
END DO
J = J + M + N
END DO
PRINT " (5X,A, I6) " , "J=", J
END PROGRAM VALUE_OF_J
```

2. ما هي النتيجة المتوقعة للبرنامج التالي ، تتبع خطوات إيجادها بجدول.

```
PROGRAM VALUE_OF_SUM
IMPLICIT NONE
REAL :: SUM = 0.0
INTEGER :: I, J
DO I = 1, 5
DO J = 1, I
SUM = SUM + 1.5
END DO
END DO
PRINT "(2X, A , F8.2)", "SUM=", SUM
END PROGRAM VALUE_OF_SUM
```

المتواليات (المتسلسلات) (Series):

المتوالية: هي عبارة عن سلسلة من الحدود المتوالية والتي أما أن تزداد زيادة منتظمة أو تتناقص تناقص منتظم. وعلى سبيل المثال: $S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots + \frac{1}{30}$

$$S = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$$

والمتوالية تكون على نوعين هما:

1. **المتوالية المحددة:** وهي المتوالية التي تكون منتهية بحد معين ٠ ومثال ذلك:

$$Y = \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \frac{1}{8} + \cdots + \frac{1}{20}$$

ملاحظات:

i. نتعرف على المتوالية بأنها محددة أما من خلال شكل المتوالية أو من خلال منطوق السؤال بأن يذكر عبارة (لغاية 10 حدود أو لغاية 20 حد وهكذا).

ii. يفضل استخدام عبارة DO المحددة في المتواليات المحددة.

المتوالية غير المحددة: وهي تلك المتوالية التي تكون منتهية بعدة نقاط تدل على أنها غير منتهية الحدود. ومثال ذلك: $S = 1 + \frac{1}{3} - \frac{1}{5} + \frac{1}{7} - \frac{1}{9} + \cdots$

ملاحظات:

i. نتعرف على المتوالية بأنها غير محددة أما من خلال شكل المتوالية أو من خلال منطوق السؤال بأن يذكر عبارة (جد قيمة المتوالية لجميع الحدود التي تزيد فيها قيمة الحد عن $5*10^{-4}$ أو عبارة جد قيمة المتوالية أدناه لغاية قيمة الحد أقل من $5*10^{-4}$).

ii. تحل المتوالية غير المحددة بأستخدام عبارة DO غير المحددة (الدوران المفتوح).

أمثلة محلولة على المتواليات المحددة:

1. أكتب برنامجاً لحساب قيمة SUM من مجموع حدود المتسلسلة التالية:

$$SUM = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots + \frac{1}{30}$$

الحل:

```
PROGRAM VALUE_OF_SUM
IMPLICIT NONE
REAL :: SUM = 0.0
INTEGER :: I
DO I = 1, 30
SUM = SUM + 1 / I
END DO
PRINT *, 'SUM =' , SUM
END PROGRAM VALUE_OF_SUM
```

2. أكتب برنامجاً لحساب قيمة SUM من المتوالية أدناه ولغاية 25 حداً؟

$$SUM = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$$

الحل:

```
PROGRAM VALUE_OF_SUM
IMPLICIT NONE
REAL :: SUM = 0.0
INTEGER :: I
DO I = 1, 25
SUM = SUM + (-1)**(I+1) * 1. / (2* I – 1)
END DO
PRINT *, 'SUM =' , SUM
END PROGRAM VALUE_OF_SUM
```

3. أكتب برنامجاً لحساب قيمة π التقريبية من المتوالية أدناه ولغاية 20 حداً؟

$$\frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \cdots$$

الحل:

```
PROGRAM VALUE_OF_APPROXIMATE_PI
IMPLICIT NONE
REAL :: SUM = 0.0 , PI
INTEGER :: I
DO I = 1, 20
SUM = SUM + 1. / I**2
END DO
PI = SQRT (6*SUM)
PRINT *, 'PI=' , PI
END PROGRAM VALUE_OF_ APPROXIMATE_PI
```

4. أكتب برنامجاً لحساب قيمة Z من المتوالية أدناه ولغاية 10 حداً؟

$$Z = 1 - \frac{1}{3!} + \frac{1}{5!} - \frac{1}{7!} + \frac{1}{9!} - \cdots$$

الحل:

```
PROGRAM VALUE_OF_Z
IMPLICIT NONE
REAL :: Z = 0.0
INTEGER :: I , J , FACT
DO I = 1, 10
FACT = 1
DO J = 1, 2*I-1
FACT = FACT * J
END DO
Z = Z + (-1)**(I+1) * 1. / FACT
END DO
PRINT *, 'Z=' , Z
END PROGRAM VALUE_OF_Z
```

5. أكتب برنامجاً لحساب قيمة Y من المتوالية أدناه ولغاية 10 حداً؟

$$4 + Y^2 = 2X + \frac{4X}{45} - \frac{6X}{42} + \frac{8X}{39} - \frac{10X}{36} + \cdots$$

الحل:

```
PROGRAM VALUE_OF_Y
IMPLICIT NONE
REAL :: X , Y , SUM
INTEGER :: I
READ *, X
SUM = 2 * X
DO I = 1, 9
SUM = SUM + (2*I+2) * X / (48 – 3*I)
END DO
Y = SQRT (SUM - 4)
PRINT *, 'Y=' , Y
END PROGRAM VALUE_OF_Y
```

واجب بيتي (Home Work):

أكتب برنامجاً لحساب قيمة Y بموجب الشروط التالية:

أ. في حالة $-2 \leq x < 4$ يستخدم المتوالية التالية لعشرة حدود:

$$\frac{Y}{3} = X - \frac{X^2}{3!} + \frac{X^4}{5!} - \frac{X^6}{7!} + \frac{X^8}{9!} - \cdots$$

ب. للحالات الأخرى $Y^2 = |X^2 - 8X|$

يقوم البرنامج بطباعة قيمة X في السطر الثاني والعمود الرابع وقيمة Y في السطر الثالث والعمود السادس وبعناوين وصيغ مناسبة.

أمثلة محلولة على المتواليات غير المحددة:

1. أكتب برنامجاً لإيجاد قيمة SUM لغاية قيمة الحد أقل من 0.01 ثم أطبع عدد تلك الحدود:

$$SUM = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$$

الحل:

```
PROGRAM VALUE_OF_SUM
IMPLICIT NONE
REAL :: SUM = 0.0 , TERM
INTEGER :: I = 1
DO
TERM =-1**(I+1)*( 1. / (2*I-1))
```

```
IF ( ABS(TERM) < 1.E-2) EXIT
SUM = SUM + TERM
I = I + 1
END DO
PRINT * , "SUM =", SUM , " NO. OF TERMS = ", I-1
END PROGRAM VALUE_OF_SUM
```

2. أكتب برنامجاً لحساب قيمة Z من المتوالية أدناه ولغاية قيمة الحد الأقل من 10^{-4}؟

$$Z = 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \cdots$$

الحل:

```
PROGRAM VALUE_OF_Z
IMPLICIT NONE
REAL :: Z = 0.0 , TERM
INTEGER :: I = 1 , J , FACT
DO
FACT = 1
DO J = 1, I
FACT = FACT * J
END DO
TERM =1. / FACT
IF ( ABS(TERM) < 1.E-4) EXIT
Z = Z + TERM
I = I + 1
END DO
PRINT * , "Z = ", Z
END PROGRAM VALUE_OF_Z
```

3. أكتب برنامجاً لحساب قيمة π التقريبية من المتوالية أدناه لجميع الحدود التي تزيد فيها قيمة الحد عن $1 * 10^{-5}$وطبع عدد تلك الحدود؟

$$\frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \cdots$$

الحل:

```
PROGRAM VALUE_OF_APPROXIMATE_PI
```

```
IMPLICIT NONE
REAL :: SUM = 0.0 , TERM , PI
INTEGER :: I = 1
DO
TERM =1. / I**2
IF (ABS(TERM) <= 1.E-5) EXIT
SUM = SUM + TERM
I = I + 1
END DO
20 PI = SQRT (6*SUM)
PRINT * , "PI =", PI , " NO. OF TERMS = ", I-1
END PROGRAM VALUE_OF_APPROXIMATE_PI
```

4. أكتب برنامجاً يسأل عن قيمة X ثم يقوم بحساب وطبع قيمة Y لجميع الحدود التي تزيد القيمة المطلقة للحد عن $1 * 10^{-4}$؟

$$Y = \frac{\pi}{2} - \frac{3\sin X}{2!} + \frac{5\sin X}{4!} - \frac{7\sin X}{6!} + \frac{9\sin X}{8!} - \cdots$$

الحل:

```
PROGRAM VALUE_OF_Y
IMPLICIT NONE
REAL :: X , Y , FACT , TERM
INTEGER :: I = 1 , J
READ *, X
Y = 3.14159 / 2.
DO
FACT = 1.0
DO J = 1, 2*I
FACT = FACT * J
END DO
TERM =-1**I*(2*I+1)*SIN(X) / FACT
IF ( ABS(TERM) <= 1.E-4) EXIT
Y = Y + TERM
I = I + 1
END DO
PRINT * , "Y = " , Y
END PROGRAM VALUE_OF_Y
```

Chapter Four

ARRAYS in FORTRAN90

المتغيرات المؤشرة والمصفوفات Indexed Variables And Arrays

لا تخلو لغة من لغات البرمجة من استخدام المتغيرات المؤشرة (Indexed Variables) التي تضيف إمكانيات كبيرة لإجراء عمليات على مجموعة من المعطيات دفعة واحدة إضافة إلى أنه لا يمكن التعامل مع المصفوفات ذات الأبعاد المختلفة بعمليات كالضرب مثلاً ، إلا من خلال استعمال المتغيرات المؤشرة المرقمة. تعرف المصفوفة حسب لغات البرمجة بأنها مجموعة قيم لها صفات مشتركة (مثل تعريف عدد الطلبة في الصف الواحد) ويفضل خزن تلك القيم تحت اسم واحد يسمى المصفوفات أو المتغيرات المؤشرة وهي صورة أخرى للمتغيرات حيث تظهر في صورة مجموعة متكاملة لها صفات مشتركة كما أنها تحمل أسم واحد ويميز أي عنصر فيها عن طريق تحديد ترتيب وجوده في المجموعة. فالمجموعات التي يتم تحديد عناصرها بمؤشر واحد تسمى مصفوفات ذات البعد الواحد. أما المصفوفات التي تحدد عناصرها بمؤشرين فتسمى ذات البعدين وقد تكون للمصفوفات أكثر من مؤشرين ومثال ذلك: A(n) ,…., A(3), A(2) , A(1) بدلاً من: A1, A2, A3, …., An وذلك للمصفوفات الأحادية ذات البعد الواحد. أما بالنسبة للمصفوفات الثنائية ذات البعدين فنكتب المؤشرين للمتغير الواحد مفصولين بفارة بين قوسين مثل: A(1,1), A(1,2), A(1,3), …, A(1,n) بدلاً من: A11, A12, A13, …, A1n وهكذا...

ملاحظة: لم تكن المصفوفات أحدى المزايا الجديدة التي أضافتها لغة FORTRAN90 بل أنها كانت متوفرة في لغة FORTRAN77 وما قبلها، غير أن لغة FORTRAN90 أضافت ميزات بديعة تضيف سهولة للمبرمج وقوة للغة.

تعريف المصفوفات (Arrays declaration):

يجب تعريف المصفوفات قبل استخدامها. إذ أن لكل مصفوفة أسماً يخضع لقوانين بقية الأسماء في لغة FORTRAN90 ويستخدم لتعريف المصفوفات خاصية البعد DIMENSION ويتم التعريف كما يلي : نوع المعطيات في المصفوفة (عناصر المصفوفة) يتبعها DIMENSION وتحديد أبعاد المصفوفة (ويفصل بين الأبعاد بأستخدام الفارزة (,)) وشكل المصفوفة (مجال المميز أي عدد عناصر المصفوفة) ويكون بشكل (:) ومن ثم ذكر أسم المصفوفة كما في المثالين التاليين:

```
REAL , DIMENSION (1:9) :: X , Y
```

LOGICAL (-99:99) :: YESNO

ففي المثال الأول هناك مصفوفتين هما X و Y وعناصر هاتين المصفوفتين من نوع حقيقي REAL وكل مصفوفة ذات بعد واحد (One Dimension) وتتكون من 9 عناصر ويمكن الرجوع للعنصر الأول من المصفوفة X على الصورة X(1) ويمكن الرجوع للعنصر الأخير من المصفوفة X على الصورة X(9) ونفس الشيء للمصفوفة Y. أما المثال الثاني فهو الإعلان عن مصفوفة YESNO ذات البعد الواحد, وعناصر هذه المصفوفة من نوع Logical وعدد عناصر المصفوفة هو 199 عنصراً. العنصر الأول هو YESNO(-99) والعنصر الأخير هو YESNO(99).

ويمكن الإعلان عن قائمة بثمانية عشر أسماً، كل أسم منها مكّون من ثمانية أحرف كما يلي:

CHARACTER (LEN=8), DIMENSION(0:17)::CHAR_LIST

وفي المثال التالي: INTEGER , DIMENSION (0:2 , 3) :: TABLE_1

يتم الإعلان عن المصفوفة TABLE_1 ذات البعدين عناصرها من النوع الصحيح ومجال المميز (المؤشر) الأول لها يتكون من ثلاثة عناصر هما (0,1,2) أما مجال المميز الثاني (البعد الثاني) يتكون من ثلاثة عناصر هما (1,2,3) أي عدد عناصر المصفوفة هو 9 عناصر.

وفي المثال التالي: REAL , DIMENSION (3 , 4 , 5) :: TABLE_2

يتم الإعلان عن المصفوفة TABLE_2 ذات ثلاث ابعاد عناصرها من النوع الحقيقي ومجال المميز (المؤشر) الأول لها يتكون من ثلاثة عناصر هما (1,2,3) أما مجال المميز الثاني (البعد الثاني) يتكون من أربعة عناصر هما (1,2,3,4) وجال المميز الثالث (البعد الثالث) يتكون من خمسة عناصر هما (1,2,3,4,5).

وأما الجملة التالية فتعلن عن المصفوفة A ذات البعدين وعناصرها من النوع الحقيقي ولكن عدد العناصر في هذه المصفوفة ترك مفتوحاً (أي أن مجال المميز لكل بعد قد ترك غير محدد) ، ويتحدد عدد عناصر المصفوفة هذه عند تنفيذ جملة ALLOCATE داخل البرنامج.

REAL, DIMENSION (:) , ALLOCATE :: A

وكما لاحظنا فإن حجم بعض المصفوفات قد يترك مفتوحاً في بعض الأحيان وهذه الطريقة مفيدة جداً عندما يكون

حجم المعطيات غير معروف مقدماً.

ملاحظات عامة:

1. عندما يتطلب البرنامج استخدام مصفوفة عناصرها أكثر من (10) يجب تحديد حجم المصفوفة باستخدام خاصية الأبعاد DIMENSION السابقة الذكر وهي عبارة غير تنفيذية في لغة فورتران يتم فيها أخبار المترجم بحجز مواقع في ذاكرة الحاسبة لعناصر المصفوفة المستخدمة لذا يمكن عدم كتابتها والحجز باستخدام عبارات النوع فقط. وكما في الأمثلة الاتية:

```
REAL :: A(12) , B(20) ,C (40)
INTEGER :: X(3,4) , Y(4,5)
```

وتستخدم هذه الطريقة في حالة أذا كان الحجز لمصفوفات مختلفة الأبعاد أو يمكن تكرار كتابة عبارة الأبعاد لكل مصفوفة.

2. في لغة FORTRAN77 وما قبلها كان يجب على المبرمج أن يحدد دائماً حجم المصفوفة في داخل البرنامج وعندما يجهل المبرمج حجم المعطيات كان يلجأ الى تحديد حجم المصفوفة بأكبر قدر يعتقد أنه يلزم. والمشكلة في هذه الحالة تكون قصور ذاكرة الكمبيوتر عن استيعاب العديد من المصفوفات ذات الإحجام الكبيرة. وفي أحيان أخرى يخطأ تقدير المبرمج في تحديد حجم المصفوفة فلا يجد الحجم المختار في استيعاب معطيات أكبر من الحجم المحدد. أما عند استخدام المصفوفات المفتوحة الأحجام في لغة FORTRAN99 فأن حجم المصفوفة ترك ليتحدد أثناء التنفيذ. وكذلك فان هنالك جملة DEALLOCATE التي تلغي مصفوفة من داخل الذاكرة عند أنتهاء الحاجة اليها تاركة المجال لمصفوفات أخرى لاستعمال الذاكرة من خلال جملة ALLOCATE وبهذا فإنه أصبح بمقدور ذاكرة الكمبيوتر استيعاب برامج أكبر بكثير مما كانت مستوعبة عند البرمجة بلغة FORTRAN77.

3. يمكن أن تكون القيم في خاصية الأبعاد DIMENSION أي حجم المصفوفات عبارة عن متغيرات محددة بعبارة تحديد الثوابت PARAMETER وذلك لتعديل البرنامج بسهولة. وأمثلة ذلك:

```
INTEGER, PARAMETER :: SIZE=5, LOWER=3, UPPER = 5
INTEGER, PARAMETER :: SMALL = 10, LARGE = 15
REAL, DIMENSION(1:SIZE) :: x
INTEGER, DIMENSION(LOWER:UPPER, SMALL:LARGE) :: A, B
```

4. لقد مر معنا المصفوفات المفتوحة الأحجام ، وهي أنه أذا عرفت مصفوفة مفتوحة الحجم في داخل البرنامج الرئيس فأنه يجب تعريفها بأنها منصاعة لأي حجم ALLOCATABLE. وعند تنفيذ

البرنامج تتحدد الحاجة الى حجم معين للمصفوفة المفتوحة. إذ يمكن عندها حجز الحجم المطلوب داخل البرنامج بواسطة جملة ALLOCATLE.

5. يمكن مساواة مصفوفتين من نفس عدد العناصر والأبعاد فمثلاً أذا كانت المصفوفتان معرفتين كما يلي:

```
REAL :: A(10, 10, 5) , A(10, 10, 5)
```

فيمكن أن تساوي عناصر المصفوف A بعناصر المصفوف B وعلى الترتيب بواسطة أشارة المساواة على النحو A=B.

وكذلك يمكن مساواة أجزاء من مصفوفة بأجزاء من نفس المصفوفة أو من مصفوفة أخرى كما يلي:

```
C(1: 4, 1:3) = C(1:2 , 1:6)
```

وكذلك يمكن مساواة مصفوفة بعدد ثابت أو متغير بسيط، وبذلك تأخذ جميع عناصر المصفوفة قيمة هذا الثابت كما يلي: M(K+1 : N, K) =0 وبهذه المساواة تأخذ العناصر :

M(K+1, K)، M(K+2, K) ، M(K+3, K) = 0 ،...، M(N, K) جميعها القيمة صفر.

المصفوفات ذات البعد الواحد (One Dimensional Arrays):

المصفوفة ذات البعد الواحد التي تحتوي على عدد N من العناصر والتي تخزن في عدد من خلايا الذاكرة حيث يخزن كل عنصر في خلية وهذه الخلايا تحمل نفس الاسم. فمثلاً نفرض أنه لدينا مصفوفة أحادية A مكونة من عشرة عناصر فتكون صورتها الرياضية هي A1, A2, A3, ..., A10 وتمثل هذه العناصر في لغة فورتران على الصورة التالية: A(1) , A(2), A(3), ..., A(10) وتستخدم القواعد التي استخدمت لأسماء المتغيرات البسيطة لأسماء المصفوفات بمعنى أن أسم المصفوفة هو حرف أو كلمة أو مجموعة كلمات متكونة من (حروف وأرقام والرمز _) بشرط أن لا يتعدى طولها (31) رمزاً وأن لا يبدأ برقم.

وكمثال على مصفوفة أحادية نأخذ قائمة من أرقام امتحانية لعدد من طلاب في كلية ولنفرض أن لدينا 1000 طالب ولكل طالب رقم مكون من سبع خانات، ولتخزين هذه الأرقام في ذاكرة الكمبيوتر لابد من استخدام أسماء متغيرات لكل طالب.

$$STUDENT\_1 = 8724011$$
$$STUDENT\_2 = 8724012$$

$$STUDENT\_3 = 8724013$$

.
.
.

$$STUDENT\_1000 = 8725011$$

هذه الطريقة يمكن تحسينها باستخدام أسلوب المصفوفة الأحادية البعد

$$STUDENT(1) = 8724011$$
$$STUDENT(2) = 8724012$$
$$STUDENT(3) = 8724013$$

.
.
.

$$STUDENT(1000) = 8725011$$

وبهذه الطريقة يمكننا الرجوع إلى أي من أرقام الطلاب باستخدام أسم متغير واحد هو STUDENT وبإعطاء المؤشر رقماً فمثلاً يمكننا طباعة رقم الطالب العاشر بواسطة الجملة PRINT *, STUDENT(10)

ويمكن أن يكون المؤشر (المميز) متغير معرف مسبقاً بدلاً من ثابت مثل:

```
I = 10
PRINT *, STUDENT(I)
```

وبما أن تركيبة التكرار DO توفر أمكانية تغيير قيمة المؤشر ، فبإمكاننا قراءة وطباعة المصفوفة كاملة كما سيرد ذكره.

ملاحظات:

1. تمنح لغة FORTRAN90 إمكانية التعبير عن جزء من مصفوفة بطريقة بسيطة وعملية. فلو كان عندنا مصفوفة معرفة في بداية البرنامج على شكل INTEGER :: STUDENT(1000) فبإمكاننا الرجوع الى أول 200 عنصر من المصفوفة بواسطة STUDENT(1:200) وبذلك يمكننا طباعة أرقام أول مئتي طالب بواسطة PRINT "(I8/)", STUDENT(1:200).

   وبنفس الطريقة يمكننا مساواة العناصر رقم 2 حتى 5 من المصفوفة A بقيمة 1.0 كما يلي:

   A(2:5)=1.0 أي أن قيمة كل من العناصر A(2), A(3), A(4), A(5) هي 1.0.

   نلاحظ أننا نقوم بتحديد رقم أول عنصر ورقم آخر عنصر عندما نريد الرجوع الى جزء من المصفوفة. وبذلك تكون العناصر المعينة هي جميع العناصر المحصورة بين هذين العنصرين.

2. توفر لغة FORTRAN90 إمكانية الرجوع الى بعض العناصر المحصورة بين عنصرين أيضاً. ولهذا يجب تحديد رقم أول عنصر ورقم آخر عنصر وكذلك المسافة بين رقم كل عنصر ورقم العنصر الذي يليه. وفيما يأتي مثال يوضح ذلك التعبير A(2:10:2) يمثل جزء المصفوفة الحاوي للعناصر A(2), A(4), A(6), A(8), A(10). أي أن أول عنصر هو العنصر رقم 2 من المصفوفة A والعنصر الأخير هو العنصر رقم 10 والمسافة بين رقم عنصر والذي يليه 2. هذه الإمكانية لم تكن متوفرة في لغة فورتران من قبل ٠ وتنطبق نفس الطريقة على المصفوفات متعددة الأبعاد. ويجوز أن تكون المسافة بين رقم عنصر ورقم العنصر الذي يليه سالباً وبذلك يكون جزء المصفوفة المقصود منعكس الترتيب. ومثال على ذلك D(10:2:-2) يمثل العناصر D(2), D(4), D(6), D(8), D(10) مرتبة تنازلياً.

طرق قراءة وكتابة المصفوفات ذات البعد الواحد:

تتم قراءة وكتابة المصفوفة الأحادية البعد بعدة طرق منها:

1. قراءة وكتابة عنصر عنصر من عناصر المصفوفة.

مثال: قراءة وكتابة مصفوفة من خمسة عناصر:

```
READ *, A(1), A(2), A(3), A(4), A(5)
PRINT *, A(1), A(2), A(3), A(4), A(5)
```

2. قراءة وكتابة المصفوفة باستخدام الحلقة التكرارية DO.

مثال: قراءة وكتابة مصفوفة من خمسة عناصر:

```
DO I=1, 5
READ*, A(I)
PRINT*, A(I)
END DO
```

3. قراءة وكتابة المصفوفة باستخدام التكرار الضمني ومثال ذلك:

```
READ *, (A(1), I=1,5)
PRINT *, (A(1), I=1,5)
```

ملاحظات:

1. عند استخدام الشكل التالي في قراءة المصفوفة: READ *, (A(I), I=2, 9, 2)
سيتم قراءة العناصر A(8) ثم A(6) ثم A(4) ثم A(2) .
2. عند استخدام الشكل التالي في قراءة المصفوفات: READ *, (A(K), B(K), K=1, 100)

سيتم القراءة بالصورة التالية: A(1), B(1) ثم A(2), B(2) ثمA(3), B(3) ....

3. عند استخدام الشكل التالي في قراءة المصفوفات:

READ *, (A(I), I=1,100, (B(I), I=1, 10)

سيتم قراءة A(1) ثم A(2) ثم A(3) ثم ... A(100) .

بعدها سيتم قراءة B(1) ثم B (2) ثم B (3) ثم ... B (100) .

4. يمكن أجراء عمليات حسابية عادية (جمع ،طرح ، وضرب) بين مصفوفة وقيمة بسيطة (أو متغير بسيط). فالجملة الآتية: A=A+3 تجمع جميع عناصر المصفوفة A مع 3. وتقوم الجملة الآتية: A=A*0.4 بضرب جميع عناصر المصفوفة A مع 0.4.

امثله:

1. افرض أن A مصفوفة أحادية مكونة من 40 عنصراً: A1, A2, A3,…,A40

أكتب برنامجاً لقراءة قيم هذه العناصر وتعيين قيمة تسلسل أول عنصر سالب في هذه المجموعة واذا لم توجد قيم سالبة فأن البرنامج يصبع صفراً.

الحل:

```
PROGRAM ARRAY_1
IMPLICIT NONE
REAL ,DIMENSION (1:40) :: A
INTEGER :: I
READ*, (A(I), I=1,40)
DO I=1,40
IF (A(I) < 0.0) EXIT
PRINT *, 0
END DO
PRINT *, I
END PROGRAM ARRAY_1
```

2. أكتب برنامجاً لقراءة عناصر المصفوفة الأحادية X المكونة من عشرين عدداً حقيقياً وترتيب هذه العناصر ترتيباً تصاعدياً.

الحل:

```
PROGRAM ARRAY_2
```

```
IMPLICIT NONE
REAL , DIMENSION (20) :: X
REAL :: TEMP
INTEGER :: I , J
READ*, (X(I), I=1,20)
DO I=1,19
DO J= I+1 , 20
IF (X(I) > X(J)) THEN
TEMP = X(I)
X(I) = X(J)
X(J) = TEMP
END IF
END DO
END DO
DO I=1,20
PRINT "( F7.2)", X(I)
END DO
END PROGRAM ARRAY_2
```

3. أكتب برنامجاً لقراءة عناصر المصفوفة الأحادية A المكونة من عشرين عدداً حقيقياً ثم جد أصغر عنصر في المصفوفة.

الحل:

```
PROGRAM ARRAY_3
IMPLICIT NONE
REAL , DIMENSION (1:20) :: A
REAL :: SMALL
INTEGER :: I
READ*, (A(I), I=1,20)
SMALL = A(1)
DO I = 2, 20
IF (SMALL > A(I))THEN
SMALL = A(I)
END IF
END DO
PRINT "(T5 , A, F8.2)" , "THE SMALLEST NUMBER IS", SMALL
END PROGRAM ARRAY_3
```

واجب بيتي (Home Work):

أكتب برنامجاً لقراءة عناصر المصفوفة الأحادية X المكونة من عشرين عنصراً وحساب مجموع مربعاتها.

$$\sum_{i=1}^{20} X_i^2$$

المصفوفات ذات البعدين (Two Dimensional Arrays):

تعد المصفوفة ذات البعدين مجموعة مرتبة على شكل عدد من الصفوف (ROWS) وعدد من الأعمدة (COLUMNS) حيث أنه يلزم لتعيين أي عنصر في هذه المجموعة تحديد رقمين يمثلان رقم الصف ورقم العمود الذي يتواجد فيه العنصر. والعنصر $a_{i,j}$ في المصفوفة A والموجود في الصف رقم i والعمود رقم j يرمز له في فورتران بالرمز A(I,J) فمثلاً A(3,4) هو العنصر الموجود في الصف الثالث وفي العمود الرابع.

طرق قراءة وكتابة المصفوفات ذات البعدين: هناك نوعان من القراءة في المصفوفات ذات البعدين هي:

1. **قراءة المصفوفة أفقياً:** وتتم هذه القراءة بحيث تكون الأسطر حلقة خارجية والأعمدة حلقة داخلية أي (سطر سطر) ٠

مثال: إذا كانت لدينا المصفوفة التالية:

$$A = \begin{vmatrix} 5 & 3 & 1 \\ 2 & 6 & 7 \\ 0 & 1 & 8 \\ 4 & 0 & -2 \end{vmatrix}$$

تتم قراءة لمصفوفة أفقياً بأحد الأساليب التالية:

a. أسلوب DO:

```
DO I = 1,4
 DO J = 1,3
 READ *, A(I,J)
 END DO
```

حيث يتم أدخال البيانات كما يلي:

```
5      Enter
3      Enter
1      Enter
2      Enter
```

وهكذا

ملاحظة:

لا ينصح باستعمال أسلوب DO في قراءة وطباعة المصفوفات وذلك لأنه يجب الضغط على مفتاح RETURN بعد كل قيمة إدخال وأيضاً في الطباعة حيث يطبع كل عنصر تحت الآخر مهماً تكن صيغة الـــ(FORMAT).

b. أسلوب DO والحلقة:

$DO\ I = 1,4$
$READ *, (A(I,J), J = 1,3)$
$END\ DO$

$5,3,1\ Enter$
$2,6,7\ Enter$
$0,1,8\ Enter$
$4,0,-2\ Enter$

وهنا يتم إدخال البيانات كالآتي (سطر ٠٠ سطر):

c. أسلوب الحلقة الداخلية (التكرار الضمني):

$READ *, ((A(I,J), J = 1,3), I = 1,4)$

ويتم الإدخال أما بالأسلوب السابق أو كالآتي: $5,3,1,2,6,7,0,1,8,4,0,-2 \quad Enter$

ملاحظة:

أسلوب الحلقة الداخلية يعتبر أفضل الأساليب لإمكانية إدخال العناصر كيفما تشاء والطبع كيفما تشاء.

2. **قراءة المصفوفة عمودياً:** وتتم هذه القراءة بحيث تكون الأعمدة حلقة خارجية والأسطر حلقة داخلية أي (عمود عمود)

فمثلاً المثال السابق تتم القراءة عمودياً بنفس الأساليب السابقة وكالاتي:

a. أسلوب DO:

$DO\ J = 1,3$
$DO\ I = 1,4$
$READ *, A(I,J)$
$END\ DO$

حيث يتم أدخال البيانات كما يلي:
5 Enter
2 Enter
0 Enter
4 Enter
3 Enter وهكذا....

b. أسلوب DO والحلقة:

$DO\ J = 1,3$	$5,2,0,4 \quad Enter$
$READ *, (A(I,J), J = 1,4)$	$3,6,1,0 \quad Enter$
$END\ DO$	$1,7,8,-2 \quad Enter$

c. أسلوب الحلقة الداخلية (التكرار الضمني):

$$READ *, ((A(I,J), I = 1,4), J = 1,3)$$

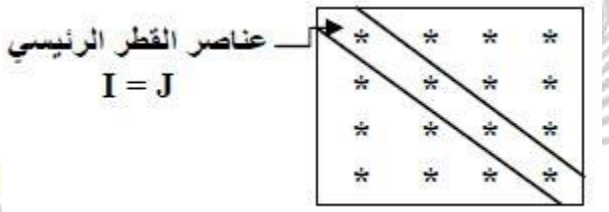
ويتم الإدخال أما بالأسلوب السابق أو كالآتي: $5,2,0,4,3,6,1,0,1,7,8,-2 \quad Enter$

المصفوفة المربعة (Square Matrix):

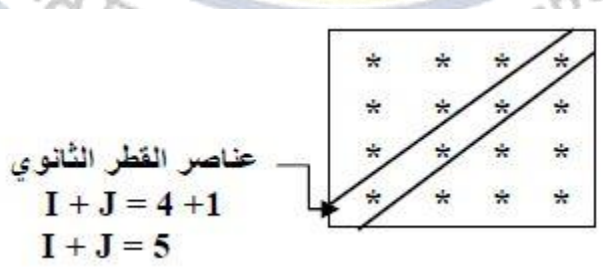
مصفوفة ثنائية البعد عدد الأسطر فيها مساوي لعدد الأعمدة وحجم المصفوفة أما يسمى بعدد الأسطر أو عدد الأعمدة.

مميزات المصفوفة المربعة:

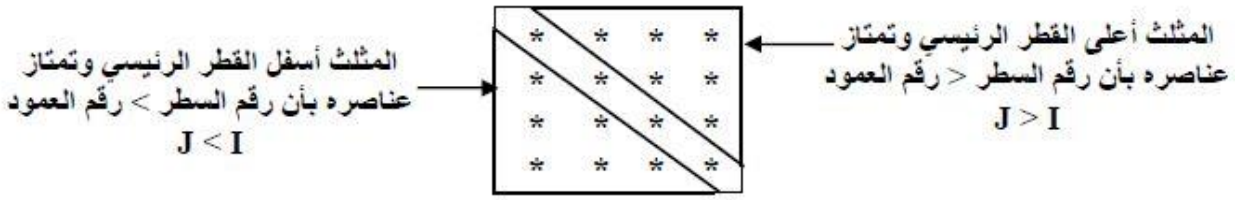
1. للمصفوفة المربعة قطر رئيسي يتميز بأن عناصره فيها رقم السطر = رقم العمود (I = J).



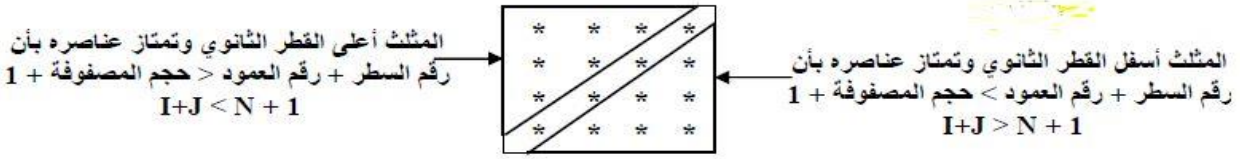
2. للمصفوفة المربعة قطر ثانوي يتميز بأن عناصره فيها:
(رقم السطر + رقم العمود = حجم المصفوفة +1) (I+J=N+1).



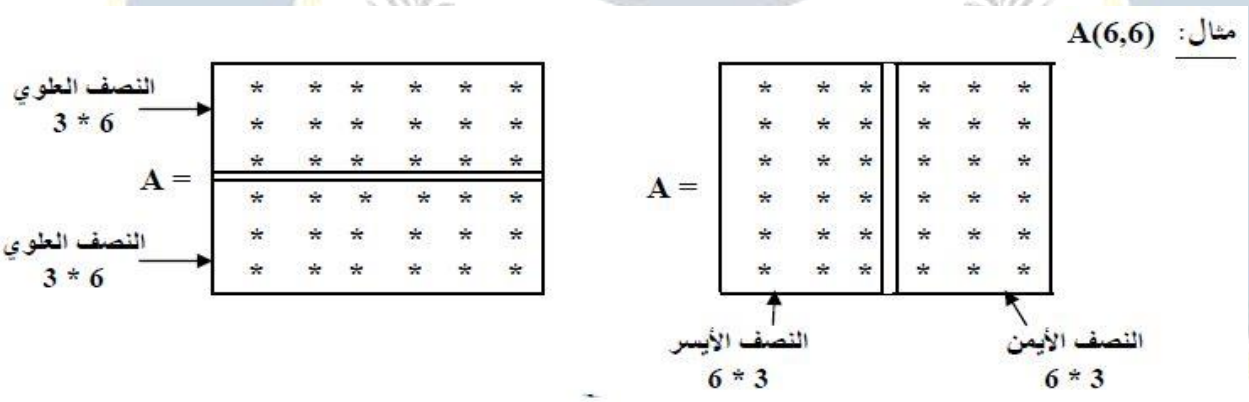
3. للمصفوفة المربعة عناصر مثلث تقع فوق القطر الرئيسي وعناصر مثلث تقع تحت القطر الرئيسي تتصف بما يلي:



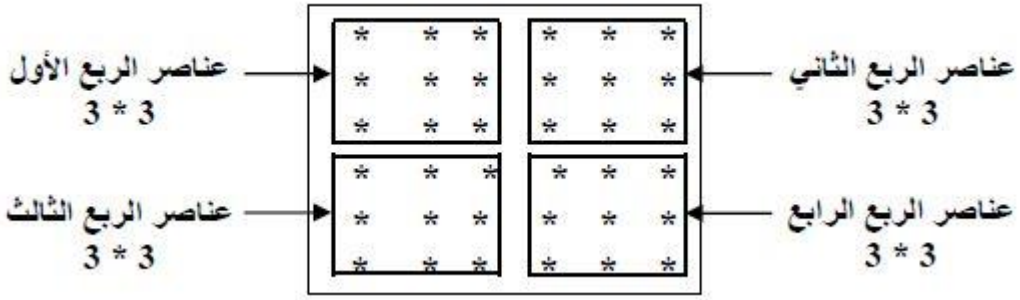
4. للمصفوفة المربعة عناصر مثلث تقع فوق القطر الثانوي وعناصر مثلث تقع تحت القطر الثانوي تتصف بما يلي:



5. للمصفوفة المربعة الزوجية الحجم (عدد الأسطر والأعمدة عدد زوجي) نصف أيسر ونصف أيمن ونصف مصفوفة علوي ونصف مصفوفة سفلي.

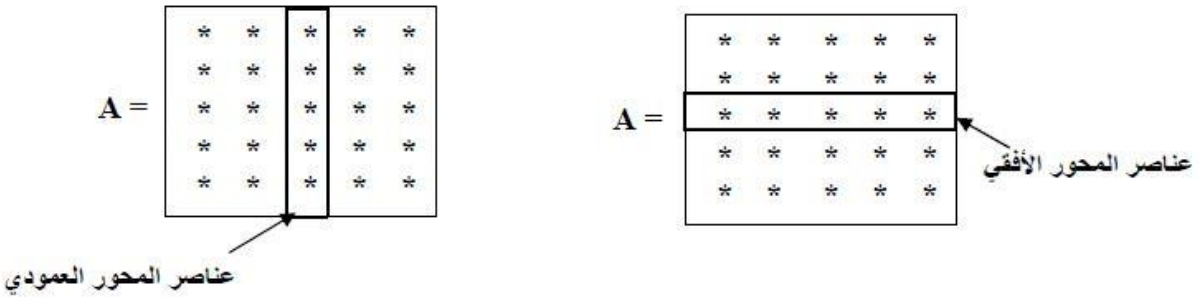


6. للمصفوفة المربعة الزوجية الحجم (عدد الأسطر والأعمدة عدد زوجي) لها ربع أول وربع ثاني وربع ثالث وربع رابع كما يلي:



7. المصفوفة المربعة الفردية الحجم (عدد الأسطر والأعمدة عدد فردي) لها محور أفقي (عناصر السطر الوسطي) ولها محور عمودي (عناصر العمود الوسطي).

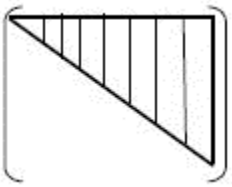
مثال: A(5,5)



8. المصفوفة المربعة التي كافة قيم عناصرها صفر تسمى مصفوفة صفرية (ZERO MATRIX).

قراءة أجزاء المصفوفة:

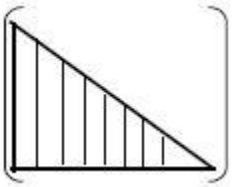
مثال: أكتب جزء البرنامج الخاص بقراءة الجزء المظلل من المصفوفة A(N,N) والذي يمثل عناصر القطر الرئيسي وما فوق القطر الرئيسي.



الحل:

```
READ *, ((A(I,J), J = I, N) , I = 1, N)
```

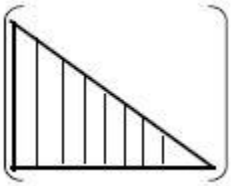
مثال: أكتب جزء البرنامج الخاص بقراءة الجزء المظلل من المصفوفة A(N,N) والذي يمثل عناصر القطر الرئيسي وما تحت القطر الرئيسي.

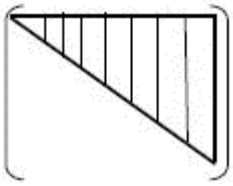


الحل:

```
READ *, ((A(I,J), J = I, N) , I = 1, N)
```

مثال: أكتب جزء البرنامج الخاص بقراءة الجزء المظلل من المصفوفة A(N,N) والذي يمثل عناصر القطر الثانوي وما فوقه والجزء المظلل الذي يمثل عناصر القطر الثانوي وما تحته.



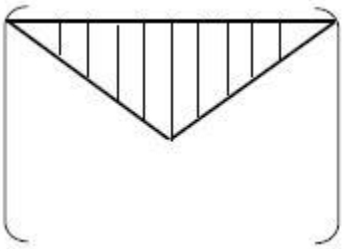


الحل:

```
READ *, ((A(I,J), J = 1, N+1-I) , I = 1, N)

READ *, ((A(I,J), J = N+1-I, N) , I = 1, N)
```

مثال: أكتب جزء البرنامج الخاص بقراءة الجزء المظلل من المصفوفة A(N,N).



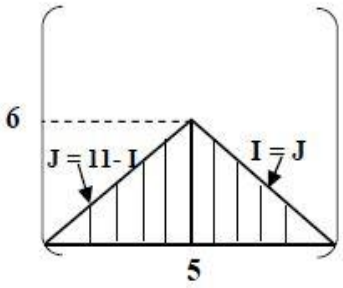
الحل:

```
READ *, (( A(I,J) , J = I, N+1-I) , I = 1, m*)
```

* $m = N/2 + 1$ أذا N هي عدد فردي

* $m = N/2$ أذا N هي عدد زوجي

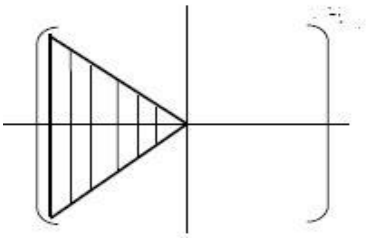
مثال: أكتب جزء البرنامج الخاص بقراءة الجزء المظلل من المصفوفة A(10,10).



الحل:

```
READ *, ((A(I,J), J = 11-I, I) , I = 6, 10)
```

واجب بيتي (Home Work):



أكتب جزء البرنامج الخاص بقراءة الجزء المظلل من المصفوفة A(N,N) عمودياً.

(الجواب يكون على سطر واحد فقط)

أمثلـــة:

1. نفرض أن المصفوفة *A(9,9)* والمطلوب كتابة برنامج:

i. يحسب عناصر المصفوفة *A* وفقاً للعلاقة: $A_{i,j} = \frac{1}{i+j-1}$

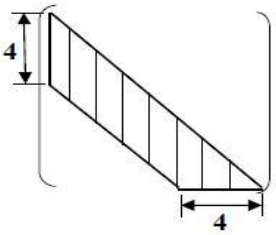
ii. يوجد مجموع عناصر القطر الرئيسي.

iii. يطبع العناصر الواقعة فوق القطر الرئيسي بحيث تكون عناصر كل صف على سطر واحد.

الحل:

```
PROGRAM SQUARE_MATRIX
IMPLICIT NONE
REAL , DIMENSION (9, 9) :: A
REAL :: SUMD
INTEGER :: I , J
DO I = 1, 9
DO J = 1, 9
A(I,J) = 1.0 / (I + J – 1)
END DO
END DO
DO I = 1, 9
PRINT "(9F7.2/)", ( A(I,J) , J = 1, 9)
END DO
SUMD = 0.0
DO I = 1, 9
SUMD = SUMD + A(I,I)
END DO
PRINT *, "SUMD=", SUMD
DO I = 1, 8
PRINT *, (A(I,J) , J= I +1 , 9)
END DO
END PROGRAM SQUARE_MATRIX
```

2. أكتب جزء البرنامج الخاص بقراءة الجزء المظلل من المصفوفة *A(10,10)* والمحصور بين القطر الرئيسي وما تحته بعرض (4) أفقياً.



الحل:

```
READ *, (( A(I,J) , J = 1, I) , I = 1, 4 ) , (( A(I,J) , J = I -3 , I) , I = 5, 10)
```

3. أكتب برنامجاً يقوم بضرب المصفوفة X(3,4) والمصفوفة Y(4,3) ثم اطبع المصفوفة Z الناتجة من عملية الضرب.

$$X=\begin{bmatrix}2 & 1 & 3 & 1\\ 4 & 0 & 2 & 3\\ 1 & 2 & 1 & 2\end{bmatrix} \qquad Y=\begin{bmatrix}1 & 0 & 4\\ 2 & 3 & 5\\ 1 & 2 & 2\\ 3 & 1 & 3\end{bmatrix}$$

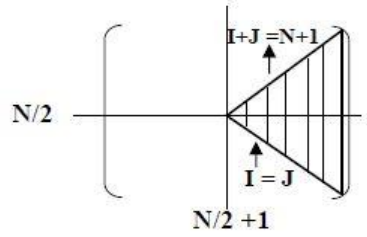
الحل:

$$Z=[X]*[Y]=\begin{bmatrix}10 & 10 & 22\\ * & * & *\\ * & * & *\end{bmatrix}$$

$$2*1+1*2+3*1+1*3=10$$
$$2*0+1*3+3*2+1*1=10$$
$$2*4+1*5+3*2+1*3=22$$

```
PROGRAM MATRIX_MULTIPLICATION
IMPLICIT NONE
REAL :: X(3,4) , Y(4,3) , Z(3,3)
INTEGER :: I , J , K
READ *, ((X(I,J) , J = 1, 4) , I = 1, 3)
READ *, ((Y(I,J) , J = 1, 3) , I = 1, 4)
DO I = 1, 3
DO J = 1, 3
Z(I,J) = 0.0
DO K = 1, 4
Z(I,J) = Z(I,J) + X(I,K) * Y(K,J)
END DO
END DO
END DO
PRINT "(3F5.0/)", (( Z(I,J) , J = 1,3) , I = 1,3)
END PROGRAM MATRIX_MULTIPLICATION
```

4. أكتب جزء البرنامج الخاص بقراءة الجزء المظلل من المصفوفة A(N,N) عمودياً.



الحل:

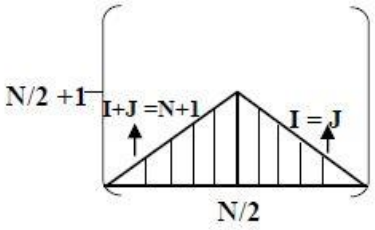
```
READ *, ((A(I,J) , I = N+1-J, J) , J = N/2 +1, N )
```

واجب بيتي (Home Work):

1. أكتب برنامج لقراءة المصفوفة A المكونة من 12 عنصر وقلب عناصرها بحيث يكون عنصر تسلسل 12 محل العنصر تسلسل 1 وتسلسل 11 محل 2 وهكذا.
2. أفرض أن D مصفوفة مكونة من ستة صفوف وتسعة أعمدة ، أكتب برنامج لتنفيذ التعليمات التالية:
   i. ضع مجموع عناصر العمود الثالث في SUM.
   ii. ضع مكان كل عنصر في الصف الثالث مجموع العنصرين المقابلين في الصفين الأول والثاني.
   iii. أجعل جميع عناصر الصف الرابع أصفاراً.
   iv. أطبع المصفوفة الناتجة.
3. افرض أن A مصفوفة مكونة من سبعة صفوف وتسعة أعمدة A(7,9) أكتب برنامجاً يقرأ قيم عناصر هذه المصفوفة ، ثم جد:
   i. عدد العناصر الموجبة ومجموع قيمها.
   ii. عدد العناصر السالبة ومجموع قيمها.
   iii. عدد العناصر الصفرية.
4. أكتب برنامجاً يقرأ عناصر المصفوفة المربعة A(N,N) حيث (N ≤ 10), ثم يقوم بما يلي:
   i. إبدال عناصر السطر الثاني مع عناصر السطر الخامس.
   ii. إبدال عناصر العمود الأول مع عناصر العمود الرابع.
   iii. إبدال عناصر النصف العلوي مع عناصر النصف السفلي.
   iv. إبدال عناصر النصف العمودي الأيمن مع النصف الأيسر.
   v. إبدال عناصر الربع الأول مع عناصر الربع الرابع.

vi. إبدال عناصر الربع الثاني مع عناصر الربع الثالث.

vii. إبدال عناصر القطر الرئيسي مع عناصر القطر الثانوي.

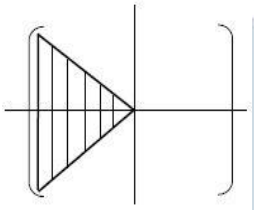


5. أكتب جزء البرنامج الخاص بقراءة الجزء المظلل للمصفوفة A(N,N).

6. المطلوب حساب قيمة Z من العبارة التالية:

$$Z = 41.298\sqrt{1+x^2} + x^{1/2}e^{ax} + \frac{e^{ax} - e^{-ax}}{2}\sin(x+a)$$

وذلك لقيم X من 1 الى 20 وبزيادة 0.1 في كل مرة ولقيم A من 0.1 الى 0.8 وبزيادة 0.05 لكل مرة وعلى أن تحسب قيمة Z لكافة احتمالات A,X وأن تطبع النتائج على شكل مصفوفة مؤلفة من 200 سطر تمثل قيم X و15 عموداً تمثل قيم A.



7. أكتب جزء البرنامج الخاص بقراءة الجزء المظلل من المصفوفة A(10,10) أفقياً.

8. أفرض أن كل من A, B, C مصفوفات مربعة تحتوي على عدد N من الصفوف و N من الأعمدة. حيث (N ≤ 6) ونفرض أن كلاً من X,Y مصفوفات أحادية تحتوي على عدد N من العناصر. أكتب برنامج يقرأ قيمة N وقيم عناصر كل من المصفوفتين A, B والمصفوفة الأحادية X ثم يقوم بحساب:

i. عناصر المصفوفة الأحادية Y والذي ينتج من ضرب المصفوفة A والمصفوفة X (Y=A*X) حيث:

$$Y_i = \sum_{j=1}^{N} A_{ij}x_j, \quad i = 1,2,3,4,\ldots,N$$

ii. عناصر المصفوفة C والتي تنتج من ضرب المصفوفتين A, B (C = A*B) حيث:

$$C_{ij} = \sum_{k=1}^{N} A_{ik}B_{kj}\,, \quad i = 1,2,3,4,\ldots N \qquad j = 1,2,3,4,\ldots,N$$

Chapter Five

SUBPROGRAMS in FORTRAN 90

المقدمة (Introduction)

أثناء كتابة بعض البرامج نحتاج لتكرار عملية معينة أكثر من مرة في مواقع مختلفة من نفس البرنامج، وقد تستلزم هذه العملية كتابة مجموعة من العبارات ربما مع مجرد تغيير بعض القيم أو تغيير أسماء بعض المتغيرات وبتكرار كتابة هذه العبارات يكون البرنامج طويلاً ويستلزم حجز مواضع للتخزين في وحدة الذاكرة الرئيسية. لاختصار خطوات البرنامج وتحسين كفاءته وتوفير مواضع التخزين فأننا نقوم بفصل هذه العبارات التي تنفذ خطوات هذه العملية المتكررة خارج البرنامج الرئيسي Main Program ونسميها برنامجاً فرعياً Subprogram ولا تخلو لغة برمجة واحدة من البرامج الفرعية سواء كانت هذه اللغة من اللغات البدائية أو اللغات المتطورة والمركبة حتى أن لغة التجميع ASSEMBLY تستعمل البرامج الفرعية. والفائدة الكبرى لاستخدام البرامج الفرعية هو التمكن من تجزئة البرامج الكبيرة إلى لبنات بنائية (برامج فرعية) تقوم كل لبنة (برنامج فرعي) بالقيام بمهمة محدودة ونتيجة للقصر النسبي لهذه البرامج الفرعية فإن أمكانية فحصها والتأكد من عملها كلاً على حدة يجعل مهمة البرمجة أكثر سهولة وأتقاناً. وبعد التأكد من عمل البرامج الفرعية تصبح مهمة المبرمج هي بناء البرنامج بها وبترتيب استخدامها بالشكل الصحيح. بالإضافة الى الفائدة التي ينالها المبرمج من استخدام البرامج الفرعية فإن أي قارئ آخر للبرنامج سيشعر بسهولة فهم البرنامج، لأن البرنامج الرئيسي يتكون من عبارات استدعاء فقط لهذه البرامج الفرعية ويعطى البرنامج الفرعي أسماً مميزاً يستدعى بموجبه كلما احتجنا تكرار هذه العملية في البرنامج الرئيسي ، وعند كتابة البرنامج الفرعي نذكر مع أسمه أسماء بعض المتغيرات (المتغيرات الشكلية) والتي تظهر (أما كمدخلات أو مخرجات) عند تنفيذ البرنامج الفرعي، وحين نشير في البرنامج الرئيسي إلى أسم البرنامج الفرعي نشير كذلك إلى القيم الفعلية أو الأسماء الفعلية التي نريدها لمتغيرات البرنامج الفرعي. والبرامج الفرعية هي برامج اختيارية، أي أن البرنامج الرئيسي قد يحتوي أو لا يحتوي على برامج فرعية.

ملاحظات:

1. يمكن تشبيه العلاقة بين (البرنامج الرئيسي) الذي ينتدب أبناءه (البرامج الفرعية) للقيام بكل الحسابات التفصيلية ثم أعطاء نتائجها الجاهزة إلى الأب بمجرد الاستدعاء.

2. نلاحظ في هذا الفصل أن لغة FORTRAN90 قد خطت خطوات واسعة عن لغة FORTRAN77 في استخدام البرامج الفرعية تماماً كما هو الحال في التركيبات الأخرى. وبكل ثقة يمكننا القول أن لغة FORTRAN90 هي لغة جديدة وحديثة تختلف تماماً عن نماذج لغة FORTRAN القديمة فلقد حوت لغة FORTRAN90 جميع الأدوات الموجودة في اللغات المتطورة الأخرى. بل قد زادت أدوات أخرى أكثر ملائمة وفعالية.

أنواع البرامج الفرعية (Subprogram Types):

تنقسم البرامج الفرعية عامة الى قسمين رئيسيين:

أولاً: البرامج الفرعية للدوال (البرامج الفرعية الدالية) Function Subprograms

ثانياً: البرامج الفرعية الروتينية (البرامج المساعدة) Subroutine Subprograms

البرامج الفرعية للدوال (البرامج الفرعية الدالية) Function Subprograms

وتنقسم البرامج الفرعية للدوال (حسب نوع الدالة) إلى ثلاثة أنواع:

1. البرنامج الفرعي لدالة مكتبية (Library Functions).
2. البرنامج الفرعي لدالة داخلية (عبارة الدالة) (Statement Function).
3. البرنامج الفرعي لدالة خارجية (External Function Subprogram).

البرنامج الفرعي لدالة مكتبية (Library Functions):

هذا النوع يشمل البرامج التي تحسب قيم كثير من الدوال الرياضية القياسية التي تستخدم كثيراً مثل:

SIN ( ) , ATAN ( ) , SQRT ( ) , ABS ( ) , ALOG ( ) ,MODULO (A,B) , ….

وقد سبق الحديث عن استخدام هذا النوع من الدوال، وتحتوي مكتبة الحاسوب على عدة برامج (فرعية) معدة لحساب قيمة أي من هذه الدوال القياسية حين نذكر أسمها في البرنامج الرئيسي.

البرنامج الفرعي لدالة داخلية (عبارة الدالة) (Statement Function):

وتستخدم إذا كانت الدالة المراد تكرار استخدامها يمكن تعريفها بعبارة واحدة فقط وليس بعدة عبارات.

توضع هذه العبارة الواحدة في نفس البرنامج الرئيسي وعادة في مطلعه. والصيغة العامة لعبارة الدالة هي:

$$\mathbf{Name(a, b, c, \ldots, n) = Expression}$$

Name : أسم الدالة (وتنطبق عليه شروط تكوين أسم المتغير).

$\mathbf{a, b, c, \ldots, n}$: المتغيرات الشكلية أو الصورية.

Expression: تعبير حسابي يحتوي على المتغيرات الشكلية.

ملاحظات:

1. عبارة الدالة لا تستخدم في بعض مترجمات لغة FORTRAN.
2. يمكن أن يشتمل التعبير الحسابي على متغيرات أو ثوابت أو دوال أخرى بالإضافة إلى أسماء المتغيرات الشكلية للدالة.
3. يجب أن لا يظهر في قائمة المتغيرات الشكلية (داخل القوسين بعد أسم الدالة) أي ثوابت أو تعابير أو عناصر مجموعة مرتبة (مصفوفة)، وإنما يجب أن يظهر أسماء متغيرات فقط، بينما المتغير الفعلي قد يكون ثابتاً أو متغيراً أو أسم مجموعة مرتبة أو عنصراً أو تعبيراً ، إلا أن نوع كل متغير فعلي يجب أن يكون نفس نوع المتغير الشكلي المقابل له (كأن يكون صحيحاً أو حقيقياً).
4. يجب أن يتساوى عدد المتغيرات الفعلية مع عدد المتغيرات الشكلية ، وأن تكتب المتغيرات الفعلية بنفس ترتيب المتغيرات الشكلية المقابلة لها.

فيما يلي أمثلة لبعض العبارات الحسابية لاستخدام الدالة ROOT والتي تعرف بعبارة الدالة:

```
ROOT (A, B, C) = (-B + SQRT(B*B – 4.0*A*C)) / (2.0*A)
X1 = ROOT (12.4 , Y-Q, H) + 12.5 * SIN(Y)
VALUE = ROOT (T(I) , T(I+1), 0.087) **3
G = H2 – ROOT (2.45, A, B)
```

فمثلاً في العبارة الأولى يحل الثابت 12.4 محل المتغير الشكلي (A) ويحل التعبير (Y-Q) محل المتغير (B) وكذلك يحل محل المتغير الفعلي (H) محل المتغير الفعلي (C). وهكذا بالنسبة لبقية العبارات.

فيما يلي بعض العبارات لدوال غير صحيحة للأسباب المذكورة إزاء كل منه:

عبارات الدوال غير صحيحة	سبب الخطأ
TOTAL (A(I), B) = A(I) *B	لا يجوز المتغير الشكلي متغير مؤشر.
8SUM (I, J, K) = I / J – K	أسم الدالة يجب أن يبدأ بحرف أبجدي وليس رقم.
RES (X, 2) = X *X	المتغيرات الشكلية يجب أن تكون أسماء متغيرات وليس رقم.

مثال: أكتب العبارة التالية بلغة فورتران وبين المتغيرات الفعلية والمتغيرات الشكلية:

$$G(X) = \frac{1}{X^5\left(e^{\frac{1.43}{TX}} - 1\right)}$$

الحل:

```
G(X) = 1.0 / (X**5 *(EXP(1.43 / (T*X)) – 1.0))
```

حيث أن: X: متغير شكلي. و T: متغير فعلي ولو كان شكلي لكانت الدالة G(X,T).

مثال: أوجد ناتج البرنامج التالي الذي يستخدم عبارة دالة (Statement Function)؟

```
PROGRAM STATEMENT_FUNCTION
IMPLICIT NONE
REAL :: F, X, Y, A, B, C
F(X,Y) = A*X+B*Y
X = 2.0 ; Y = 3.0
A = 4.0 ; B = 5.0
C = F(6.0 , 7.0)
PRINT *, 'C =' , C
END PROGRAM STATEMENT_FUNCTION
```

الجواب: C=59.000

البرنامج الفرعي لدالة خارجية (External Function Subprogram):

إذا كانت الدالة المراد تكرار استخدامها ، ويتطلب حساب قيمة الدالة في المرة الواحدة عدة خطوات أي تنفيذ عدة عبارات مثل حساب قيمة دالة مفكوك N(N!) مثلاً قد نحتاج في نفس البرنامج حساب !(N-M) و

N! و M! فبدلاً من كتابة العبارات التي تحسب N! ثم أعادة كتابة نفس العبارات التي تحسب M! ثم أعادة نفس العبارات التي تحسب (N-M)! بدلاً من ذلك نكتب برنامجاً فرعياً Subprogram لدالة تسمى مثلاً FACT(K) تحسب مفكوك K حيث K متغير شكلي وعند الحاجة إلى استخدام المتغيرات الفعلية N وM و (N-M) في البرنامج الرئيسي يكفي ان تستدعي الدالة FACT بذكر أسمها كما يلي: FACT(N), FACT(M), FACT(N-M) والصيغة العامة للبرنامج الفرعي الدالي هي:

$$FUNCTIONname(a_1, a_2, \dots a_2)RESULT(Output)$$

$$IMPLICIT\ NONE$$

$$t.s.$$

عبارات البرنامج .....
.....
.....

$$ENDFUNCTION\ name$$

حيث:

$name$: أسم البرنامج الفرعي أي أسم الدالة.

$a_1, a_2, \dots a_2$: المتغيرات الشكلية (الوسيطة).

$Output$: متغير يعطى قيمة داخل الدالة وتكون قيمته هذه هي القيمة التي يحسبها البرنامج.

$t.s.$: عبارات النوع لتوصيف وتحديد أنواع المتغيرات الشكلية، وكذلك أي متغيرات أخرى تظهر في هذا البرنامج الفرعي.

END : تعني نهاية البرنامج الفرعي الدالي.

ملاحظات:

1. جملة $RESULT(Output)$ هي جملة اختيارية أي يمكن كتابتها أو عدم كتابتها وفي حالة عدم كتابتها فأن أسم البرنامج الفرعي أي أسم الدالة هي التي يحسب قيمتها داخل البرنامج الفرعي.

2. في كل مرة تستدعي فيها الدالة أي البرنامج الفرعي يجب أن تنفذ على الأقل عبارة واحدة تعطي قيمة لأسم الدالة $name$ او اسم المتغير $Output$ وإلا فأن قيمة الدالة تكون غير معرفة، وعادةً تكون هذه العبارة عبارة حسابية في صورة $name = e$ حيث e : تعبير واسم الدالة $name$ يجب إلا يظهر في الطرف الأيسر.
3. لا تختلف طريقة استدعاء الدالة الخارجية في البرنامج الرئيسي عن استدعاء أي دالة أخرى، أي يمكن للدالة أن تظهر في وسط أي تعبير رياضي أو أي جملة كتابية ، على أن يراعى عند تنفيذ الدالة أن تكون العوامل عند استدعاء الدالة تتشابه من حيث العدد والنوع للعوامل الشكلية (الصورية) التي عرفت للدالة عند كتابتها وكذلك أن يكون نوع القيمة التي تعود لها الدالة ، أي نوع المتغير الموجود في جملة $RESULT$ أو اسم الدالة مناسباً للتعبير الرياضي الذي توجد به الدالة.
4. إذا أحد المتغيرات الشكلية إسماً لمصفوفة فيجب أن يحدد نوعها وعدد عناصرها في عبارة نوع في البرنامج الفرعي.

مثال: اكتب برنامجاً يقرأ قيمتي M,N حيث أن (N>M) ويحسب قيمة كل من Y,Z ويحسب البرنامج الفرعي المفكوك بواسطة دالة خارجية. حيث

$$Y = \frac{N!}{M!\,(N-M)!}$$

$$Z = 1 + M + \frac{M^2}{2!} + \frac{M^3}{3!} + \ldots + \frac{M^N}{N!}$$

الحل:

```
PROGRAM EXTERNAL_FUNCTION_1
IMPLICIT NONE
REAL :: Y , Z = 1.0
INTEGER :: I , FACT, N, M
READ *, N , M
Y = FACT (N) / ( FACT (M) * FACT (N-M) )
PRINT *, 'Y =' , Y
DO I = 1, N
Z = Z + M**I / FACT (I)
END DO
PRINT *, 'Z =' , Z
```

```
END PROGRAM EXTERNAL_FUNCTION_1
FUNCTION FACT (K)
IMPLICIT NONE
INTEGER :: I , FACT , K
FACT = 1
DO I = 1, K
FACT = FACT * I
END DO
END FUNCTION FACT
```

البرامج الفرعية الروتينية (البرامج المساعدة) Subroutine Subprograms:

نحتاج أحياناً لكتابة برنامج فرعي يعيد أكثر من قيمة واحدة أو يعيد مجموعة مرتبة من القيم أو يقوم بأجراء عملية ما، في مثل هذه الحالات يمكن أن نستخدم ما يسمى بالبرنامج الروتيني الفرعي بالمواصفات التالية:

1. القيم التي يعيدها تعطى للمتغيرات الفعلية وليس لأسم البرنامج الفرعي، فهذا الاسم لا تعطى له قيمة وأنما هو مجرد أسم لتعريف البرنامج.
2. عند مناداة (أي استخدام) البرنامج الفرعي الروتيني يجب أن ينادي بعبارة نداء مستقلة (Call Statement) توضع في البرنامج الرئيسي.
3. يبدأ البرنامج الفرعي الروتيني بعبارة تعريف تشتمل على كلمة (Subroutine) يعقبها أسم البرنامج ثم بين قوسين نضع العوامل الشكلية أو الصورية.

الصيغة العامة للبرنامج الفرعي الروتيني هي:

$$SUBROUTINE\ name(a_1, a_2, \ldots a_2)$$

$$IMPLICIT\ NONE$$

$$t.s.$$

$$\left.\begin{matrix}\ldots\ldots\\ \ldots\ldots\\ \ldots\ldots\end{matrix}\right\}$$ عبارات البرنامج

$$END\ SUBROUTINE\ name$$

حيث:

$name$: أسم البرنامج الفرعي (وهو كأسم أي متغير).

$a_1, a_2, \ldots a_2$: أسماء متغيرات الشكلية أو الصورية.

$Output$: متغير يعطى قيمة داخل الدالة وتكون قيمته هذه هي القيمة التي يحسبها البرنامج.

$t.s.$: عبارات النوع لتوصيف وتحديد أنواع مدخلات البرنامج ومخرجاته وكذلك أنواع أي متغيرات أخرى تظهر في هذا البرنامج الفرعي.

END : تعني نهاية البرنامج الفرعي الروتيني.

وكمثال على البرنامج الفرعي الروتيني نأخذ البرنامج الروتيني SWAP لمبادلة القيم بين أي متغيرين.

```
SUBROUTINE SWAP(X, Y)
IMPLICIT NONE
REAL :: X,Y,Z
Z = X
X = Y
Y = Z
ENDSUBROUTINESWAP
```

عبارة الاستدعاء (Call Statement):

تستعمل عبارة الاستدعاء في البرنامج الرئيسي عندما يرغب المبرمج في استدعاء برنامج روتيني فرعي لتنفيذ عمليات حسابية معينة يكون نتيجتها أكثر من قيمة واحدة ٠ وتكون الصيغة العامة لعبارة الاستدعاء هي:

$$Call\ name\ (a_1, a_2, \ldots, a_n)$$

حيث:

$name$: أسم البرنامج الفرعي الروتيني المطلوب استدعاءه.

$a_1, a_2, \ldots a_2$: المتغيرات الفعلية التي ستعطي للبرنامج الروتيني الفرعي وهذه قد تكون أسماء متغيرات أو أسماء مجموعات مرتبة أو أسماء عناصر مجموعات مرتبة ويجوز أستعمال تعابير أو ثوابت.

وعبارة الاستدعاء تنقل التحكم من البرنامج الرئيسي إلى الروتين الفرعي وكذلك تستبدل المتغيرات الشكلية في البرنامج الروتيني الفرعي بالمتغيرات الفعلية.

يمكن استخدام البرنامج الفرعي الروتيني SWAP لترتيب ثلاثة متغيرات تصاعدياً كما يلي:

```
PROGRAM SORT
IMPLICIT NONE
REAL :: A, B, C, Z
READ * , A, B, C! Read the numbers
! Sort the numbers
IF (A > B) THEN
CALL SWAP(A,B)
END IF
IF (A > C) THEN
CALL SWAP(A,C)
END IF
IF (B > C) THEN
CALL SWAP(B,C)
END IF
! Print the numbers
PRINT *, " The numbers, in ascending order, are: "
PRINT * , A, B, C
END PROGRAM SORT
SUBROUTINE SWAP(X, Y)
IMPLICIT NONE
REAL :: X,Y,Z
Z = X
X = Y
Y = Z
ENDSUBROUTINE SWAP
```

نلاحظ أن الجمل الثلاث التي كانت تتكرر داخل البرنامج الفرعي الروتيني SWAP قد استبدلت في البرنامج الرئيسي بعبارة الاستدعاء:

```
CALL SWAP(A,B)
CALL SWAP(A,C)
CALL SWAP(B,C)
```

ولاحظ هنا أن عبارة الاستدعاء تتكون من الكلمة CALL متبوعة بأسم الروتين يليه أقواس تحوي بعض العوامل. كما نلاحظ أن جملة الاستدعاء هذه قد اختصرت ثلاث جمل بجملة واحدة داخل البرنامج الرئيسي

وأن استخدام كلمة SWAP التي تعني المبادلة تدل بوضوح على طريقة عمل البرنامج الروتيني وباستخدام SWAP القارئ تفاصيل تنفيذ هذه المبادلة التي قد تكون غير مهمة للقارئ عند قراءة البرنامج الرئيسي أما أذا أراد قارئ البرنامج تفاصيل تنفيذ المبادلة فبإمكانه الرجوع إلى الروتين لمعرفة ذلك.

ملاحظات:

1. اسم الروتين الفرعي هو مجرد اسم لتعريف البرنامج ويجب أن لا يظهر في أي عبارة أخرى داخل البرنامج الفرعي.
2. المتغير الشكلي المستخدم كمخرج يجب ظهوره على يسار عبارة حسابية داخل البرنامج الفرعي أي أن البرنامج الفرعي الروتيني يجد قيمة هذا المتغير.
3. يمكن وضع الروتين بنفس الملف الذي يوجد به البرنامج الرئيس وذلك بعد جملة END PROGRAM SORT أو في ملف آخر منفصل يتم ربطه مع البرنامج الرئيس بواسطة المترجم Compiler أو الرابط Linker.

عبارة الاحتواء Contains (Contains Statement):

أن البرامج الفرعية التي سبق ذكرها يمكن لها أن تكون داخلية وخارجية. وعلى سبيل المثال نأخذ البرنامج SORT مع الروتين SWAP وعل الشكل المذكور آنفاً فإن SWAP هو روتين خارجي وذلك لأنه كتب بعد البرنامج الرئيس وليس في داخله (ينتهي البرنامج الرئيس بجملة END PROGRAM). وللعلم فإنه يجوز للروتين الخارجي أن يكتب مباشرة بعد البرنامج الرئيس وفي نفس الملف أو أن يكتب في ملف آخر.

بإمكاننا أن نجعل الروتين SWAP روتيناً داخلياً في داخل البرنامج الرئيسي ، أي قبل جملة END PROGRAM. وهنا يجب استخدام عبارة CONTAINS داخل البرنامج الرئيسي وقبل البرنامج الروتيني ،فمثلاً نقوم بإعادة كتابة البرنامج SORT باستخدام ثلاثة روتينات داخلية ، أحد هذه الروتينات يقوم بقراءة المعطيات والآخر يرتب قيم المعطيات والثالث يقوم بطباعتها أي أننا نكون بذلك قد قسمنا البرنامج داخلياً الى أجزائه الرئيسية. لذا تسبق هذه الروتينات عبارة CONTAINS التي تعني أن البرنامج الرئيسي يحتوي على الروتينات الداخلية التي تعقب هذه الجملة. وهنا نعيد كتابة البرنامج SORT باستخدام الروتينات الداخلية، باستعمال عبارة الاحتواء:

```
PROGRAM SORT
IMPLICIT NONE
REAL :: A, B, C
CALL READ_PART
CALL SORT_PART
CALL PRINT_PART
CONTAINS
SUBROUTINEREAD_PART
READ *, A, B, C
END SUBROUTINEREAD_PART
SUBROUTINE SORT_PART
IF (A > B) THEN
CALL SWAP(A,B)
END IF
IF (A > C) THEN
CALL SWAP(A,C)
END IF
IF (B > C) THEN
CALL SWAP(B,C)
END IF
END SUBROUTINE SORT_PART
SUBROUTINE PRINT_PART
PRINT *, " The numbers, in ascending order, are:"
PRINT *, A, B, C
END SUBROUTINE PRINT_PART
END PROGRAM SORT
SUBROUTINE SWAP(X, Y)
IMPLICIT NONE
REAL :: X,Y,Z
Z = X
X = Y
Y = Z
ENDSUBROUTINE SWAP
```

ملاحظات على برنامج PROGRAM SORT أعلاه:

1. يبين البرنامج أعلاه طريقة استدعاء الروتينات الداخلية في البرنامج الرئيس ونلاحظ أن هذه الروتينات استدعيت دون أعطاء عوامل (متغيرات فعلية) وكذلك أن الروتينات الداخلية لا تحتاج الى عوامل (متغيرات شكلية) والسبب في ذلك أنها تشترك مع البرنامج الرئيسي في جميع المتغيرات المعلنة في

البرنامج. غير أن عدم استعمال عوامل في الروتينات الفرعية ليس قاعدة ، فبالإمكان استخدام روتينات داخلية بعوامل (كما سيمر لاحقاً).

2. نلاحظ أن الروتين SWAP لا يزال روتيناً خارجياً. وهناك أسباب لإبقاء هذا الروتين خارجياً. أحد هذه الأسباب هو أنه لو كان SWAP داخلياً لكان يجب أن يحتويه الروتين الداخلي SORT_PART غير انه لا يمكن لروتين داخلي أن يحتوي روتيناً داخلياً آخر. والسبب الآخر هو أن الروتين SWAP هو من الروتينات المفيدة والمتكررة الاستعمال وباستعماله كروتين خارجي بالإمكان كتابته في ملف منفصل تستفيد منه البرامج المختلفة وذلك أفضل من أبقائه خاصاً لبرنامج واحد.

ملاحظة:

لا يمكن للبرامج الفرعية الداخلية (داخل البرنامج الرئيسي) أن تحتوي في داخلها على برامج فرعية أخرى. ولكن يمكن للبرامج الفرعية الخارجية (خارج البرنامج الرئيسي) أن تحتوي في داخلها على برامج فرعية أخرى بشرط أن تسبقها عبارة CONTAINS في البرنامج الفرعي. كما موضح في الصيغة العامة لكتابة البرامج في لغة FORTRAN 90 (البرنامج الرئيسي والبرامج الفرعية الداخلية أو الخارجية).

main program:

```
PROGRAM program-name
IMPLICIT NONE
[declaration statements]
[executable statements]
CONTAINS
[SUBROUTINE]
or
[FUNCTION]
END PROGRAM program-name
```

```
SUBROUTINE name[( arguments )]
[declaration statements]
[executable statements]
[CONTAINS
internal subprograms]
END [SUBROUTINE [name]]
```

or
FUNCTION name([arguments])
[declaration statements]
[executable statements]
[CONTAINS
internal subprograms]
END [FUNCTION [name]]

ملاحظات حول العوامل الصورية (المتغيرات الشكلية) والعوامل الحقيقية (المتغيرات الفعلية) في FORTRAN90:

1. توافق العوامل الشكلية (الصورية) مع العوامل الفعلية (الحقيقية):

لقد كانت لغة FORATRAN77 تتطلب الموافقة التامة ما بين العوامل الصورية (المذكورة عند تعريف البرنامج الفرعي) والعوامل الحقيقية عند استدعاء البرنامج الفرعي ولكن لغة FORATRAN90 باستحداث العوامل الخيارية والعوامل المحددة بأسماء والعوامل العامة فإن قواعد المطابقة لم تعد بالبساطة السابقة في لغة FORATRAN77. ولكن يجب تطابق العوامل الحقيقية والعوامل الصورية من حيث العدد والنوع، غير أن هناك أنواعاً جديدة من العوامل لم تكن متوفرة في لغة FORATRAN77 أسمها العوامل الخيارية التي يمكن تجاهلها عند استدعاء البرنامج. وفي الأحوال العادية لا زال الترتيب في التطابق ما بين العوامل الحقيقية والعوامل الصورية قائماً غير أن هنالك نوعاً جديداً من العوامل مرتبطاً بأسماء يمكن له تجاوز ذلك الترتيب، والمقصود في الترتيب هنا أن يتطابق العامل الشكلي الأول مع العامل الحقيقي الأول والثاني مع الثاني وهكذا.

2. تمرير العوامل:

كما هو معروف يمتلك البرنامج الفرعي (دالة أو روتين) متغيرات داخلية خاصة به تماماً مثل المتغيرات الخاصة بالبرنامج الرئيس وإذا فرضنا وجود متغير باسم X في البرنامج الرئيس وكان هناك متغير آخر بنفس الأسم في البرنامج الفرعي (الخارجي) فإن هذين المتغيرين منفصلان ولا علاقة بينهما.

3. العوامل المرجعية:

وتعني مطابقة العامل الحقيقي (المتغير الفعلي) مع العامل الصوري (المتغير الشكلي) المستخدم في البرنامج الفرعي من حيث مكان التخزين أي أن كلا العاملين على الرغم من اختلاف أسمائهما يعودان

الى موقع تخزين واحد. ومثال ذلك أذا كان لدينا متغير أسمه X في البرنامج الرئيس ومرر هذا المتغير الى البرنامج الفرعي على شكل عامل مرجعي ، وكان هذا المتغير X يطابق متغيراً صورياً R فإن أي تغير في قيمة R داخل البرنامج الفرعي تنعكس على قيمة X في البرنامج الرئيس ، لأ ّ ما يحتلان مخزناً موحداً ومشتركاً في الذاكرة.

وفيما يلي مثال على العامل المرجعي (وهو نفس الروتين SWAP) الذي استخدمناه سابقاً.

ولنفرض أنه تم استدعاء هذا الروتين بواسطة الجملة ((CALL SWAP(A, B) هنا المتغيران الحقيقيان A و B هما المتغيران المطابقان للعاملين الصوريين X و Y في أول سطر الروتين.

```
SUBROUTINE SWAP(X, Y)
IMPLICIT NONE
REAL :: X,Y,Z
Z = X
X = Y
Y = Z
ENDSUBROUTINE SWAP
```

يتم الرجوع هنا الى العامل الحقيقي عند أي استخدام للعامل الصوري المطابق فعند تنفيذ الجملة Z=X وتكون العملية مطابقة لعملية Z=A لان X و A يرجعان إلى نفس مكان التخزين في ذاكرة الكمبيوتر وهكذا بالنسبة لبقية الجمل المذكورة في الروتين.

4. العوامل القيمية:

يختلف هذا النوع من العوامل عن العوامل المرجعية السابقة في كون العامل الحقيقي والعامل الصوري لا يرجعان الى نفس المكان في الذاكرة وبهذا فإنه يمكن لهذا العامل (القيمي) أن يكون ثابتاً (الثابت لا يمكن له أن يتغير). وبذلك يجب الحرص على عدم استخدام العامل القيمي في الجانب الأيسر من أي تعبير رياضي (وجود متغير في الجهة اليسرى من تعبير رياضي يعني محاولتنا تخزين معلومات في موقع في داخل الذاكرة يرمز اليه بأسم المتغير). وكمثال على العوامل القيمية نأخذ الدالة التالية:

```
FUNCTION SERIES (M, N, X, P) RESULT (SUM)
IMPLICIT NONE
INTEGER :: M, N, I
REAL :: X, P, SUM
SUM = 0.
DO I = M, N
```

```
SUM = SUM +X + I*P
END DO
END FUNCTION SERIES
```

وتتم عملية استدعاء هذه الدالة في البرنامج الرئيس على شكل:

```
PRINT *, SERIES (200, 600,100.0 , 0.2)
```

كما نلاحظ فإن العوامل الحقيقية هي من النوع الثابت ونلاحظ أيضاً أن أياً من المتغيرات الصورية المطابقة لهذه العوامل لم تظهر في الطرف الأيسر من التعابير الرياضية المذكورة في الدالة. وهكذا فإن العوامل الحقيقية كان هدفها أعطاء قيم ابتدائية للعوامل الصورية.

5. العوامل المرتبطة بأسماء العوامل الصورية:

لقد سبق وأن ذكرنا أنه يجب التطابق ما بين العوامل الصورية والعوامل الحقيقية من حيث العدد والنوع وذلك على الترتيب. فالعامل الصوري الأول يتطابق مع العامل الحقيقي الأول وهكذا. غير أنه يمكن تجاوز هذه القاعدة في ترتيب العوامل أذا ربطنا كل عامل حقيقي بأسم العامل الصوري الذي ينطبق معه. و كمثال على ذلك يمكن استدعاء الدالة المذكورة في الملاحظة السابقة على شكل:

```
PRINT *, SERIES (P=0.1, M=200, N=600 , X=100.0)
```

تؤدي هذه الجملة نفس عمل الجملة المذكورة في الملاحظة السابقة. غير أننا هنا أستعضنا عن ترتيب العوامل باستخدامها بشكل عشوائي ولكن كل عامل حقيقي يرتبط بأسم العامل الصوري المطابق له من خلال أشارة المساواة.

كما أن هنالك مرونة أخرى في استخدام العوامل تتخلص في جعل بعض هذه العوامل مرتبة والبعض الآخر مرتبطاً بأسماء العوامل مثل:

```
PRINT *, SERIES (200, 600 , P=0.1, X=100.0)
```

حيث يتطابق ترتيب العاملين الأولين مع ترتيب نظيريهما من العوامل الصورية وأما الآخران فيرتبطان بأسماء العوامل الصورية المطابقة ولا يتفقان معها من حيث الترتيب.

6. العوامل الخيارية:

لا تشترط لغة FORTRAN90 انطباق العوامل الصورية مع العوامل الحقيقية من حيث العدد دائماً ولكن في مثل هذه الحالة يجب اتخاذ إجراءات معينة للإشارة الى مثل تلك الإمكانية حيث أن الأساس هو أن ينطبق عدد العوامل في الحالتين. وهنا نعيد كتابة الدالة SERIES مع جعل العامل الصوري أختيارياً أي أن هناك وجود أمكانية للاستغناء عن العامل الحقيقي المطابق لهذا العامل.

```
FUNCTION SERIES (M, N, X, P) RESULT (SUM)
IMPLICIT NONE
INTEGER, OPTIONAL :: M
INTEGER :: N, I , TEMP_M
REAL :: X, P, SUM
IF (PRESENT(M)) THEN
TEMP_M = M
ELSE
TEMP_M = 0
END IF
SUM = 0.
DO I = TEMP_M, N
SUM = SUM +X + I*P
END DO
END FUNCTION SERIES
```

نلاحظ هنا ان الدالة المكتبية PRESENT هي الأداة التي تدلنا على وجود عامل حقيقي مطابق للعامل الصوري M. وفي هذا المثال نقوم بفحص وجود مثل هذا العامل، فإذا كان موجوداً فإن المتغير الداخلي (الخاص بالدالة) TEMP_M يتلقى قيمة هذا العامل، وأما أذا تغيب العامل المطابق لـــ M في جملة الاستدعاء فان TEMP_M تأخذ القيمة صفراً ٠ وبذلك فإن أياً من الجمل التالية تصلح لاستدعاء الدالة SERIES.

لاحظ الاختلاف في عدد العوامل الحقيقية عندما يغيب العامل المطابق لـــ M. كما لاحظ اختلاف ترتيب العوامل عند ربط بعض العوامل الحقيقية بأسماء العوامل الصورية كما ورد في الملاحظة السابقة.

```
PRINT *, SERIES (0, 600 , 0.2,100.0)
PRINT *, SERIES (0, 600 , P=0.1, X=100.0)
PRINT *, SERIES (N=600 ,P=0.1, X=100.0)
PRINT *, SERIES (M=0,N=600 , P=0.1, X=100.0)
```

7. قصد استخدام العوامل:

وهذه أيضاً ميزة جديدة تضيفها لغة FORTRAN90 إذ بالإمكان أن يعبر المبرمج عن نيته في جعل بعض العوامل قابلة للتغيير في داخل البرنامج. تقوم كلمة INTENT (IN) أو INTENT (OUT) أو INTENT (INOUT) بهذا الغرض كما نلاحظ في المثال التالي:

```
FUNCTION SERIES (M, N, X, P) RESULT (SUM)
IMPLICIT NONE
INTEGER, INTENT (IN) , OPTIONAL :: M
INTEGER :: SUM
INTEGER :: I , TEMP_M
------
END FUNCTION SERIES
```

في هذا المثال أعدنا كتابة جزء من الدالة SERIES ليتبين فيها كيفية أبراز النية في استخدام العوامل. عندما تكون النية على أساس أن العامل IN فهذا يعني أنه لا يجوز إعطاء قيمة لهذا العامل داخل الدالة. أما أذا كان OUT فالمقصود أنه يجب أن يعطى العامل الحقيقي في داخل الدالة قيمة (أي أن يظهر في الجهة اليسرى من تعبير رياضي). وقد تكون النية INOUT أي أن المتغير الصوري يتلقى قيمة ابتدائية من العامل الحقيقي عند استدعاء الدالة. كما أن العامل الحقيقي يتلقى قيمة داخل الدالة ٠ وبالطبع فإنه لا ينصح باستخدام عوامل بنية OUT في الدوال لأن الغرض المعتاد للدوال أن تنتج قيمة واحدة ٠أن أبراز النوايا في طريقة استعمال العوامل تمنح المترجم (Compiler) إمكانية اكتشاف أخطاء غير مقصودة في الدالة (بدليل مخالفتها النية المعلنة) وبذلك فإن المترجم يحذر المبرمج من وجود هذه التناقضات.

8. البرامج الفرعية كعوامل:

يجوز الإعلان عن برنامج فرعي كعامل صوري في برنامج فرعي. كما يجب أن يكون العامل الحقيقي المطابق لهذا العامل الصوري برنامجاً فرعياً ويجب أن يكون هذا العامل برنامجاً فرعياً داخلياً ولا يجوز أن يكون خارجياً.

أمثلـــــة:

1. ماذا تتوقع نتيجة البرنامج التالي؟

```
PROGRAM SUBPROGRAM_1
IMPLICIT NONE
REAL :: F , X , A, B, C
```

```
INTEGER :: I , J , K
F(X) = X**2 + 2
A = 3.
B = 2.
I = 6
DO J = 1, I
C = ( F(A-B) – F(A+B) ) / 2.
K = C * SQRT ( ABS(C) ) + 0.5
A = A + 2.
B = B – 2.
IF ( K > 0 ) EXIT
PRINT"(2F6.2 , 2X , F7.3 , I4)", A, B, C, K
END DO
ENDPROGRAM SUBPROGRAM_1
```

الجواب:

العبارة الأولى هي عبارة دالة وبعدها تعريف القيم الابتدائية: I = 6, B = 2., A = 3.

	B	A	K	C	J
$F(A-B) = F(1) = 3$					
$F(A+B) = F(5) = 27$	0.	5.	-41	-12.	1
$C = (3-27)\ /\ 2\ = -24/2 = -12$	-2.	7.	0	0.	2
$K = -12 * \sqrt{12} + 0.5 = -41$	-4.	9.	148	28.	3

```
 5.00   0.00   -12.000   -41
 7.00   -2.00    0.000     0
```

2. أكتب برنامج لإيجاد قيمة Y من المتسلسلة التالية:

$$Y = \frac{1!}{2*3!} - \frac{3!}{3*6!} + \frac{5!}{4*9!} - \frac{7!}{5*12!} + \frac{9!}{6*15!} - \ldots$$

أستعمل برنامج فرعي دالي لحساب المفكوك، ويطبع الناتج عندما تقل قيمة الحد عن 10^{-4} وكالاتي:

العمود الثاني ↓

السطر الثالث → Y = صيغة مناسبة

الجواب:

```
PROGRAM SUBPROGRAM_2
IMPLICIT NONE
REAL :: Y = 0.0 , TERM
INTEGER :: I = 1 , FACT
DO
TERM = (-1) **(I+1)* FACT (2*I-1) /REAL((I+1) * FACT (3*I))
IF ( ABS(TERM) < 1.E-4 ) EXIT
Y = Y + TERM
I = I +1
END DO
PRINT "(// , T2 , A , F8.3)", 'Y =', Y
END PROGRAM SUBPROGRAM_2
FUNCTION FACT (K)
IMPLICIT NONE
INTEGER :: I , FACT , K
FACT = 1
DO I = 1, K
FACT = FACT * I
END DO
END FUNCTION FACT
```

3. لغرض أيجاد قيمة التكامل $\int_a^b f(X)dx$ وبأستخدام طريقة شبه المنحرف أكتب برنامج يقوم بما يلي:

a) يعرف الدالة الداخلية المذكورة: $F(x) = \frac{e^{x^2}}{x+2}\ln|2-x|$

b) يعرف قيمة N, b, a وحساب Δx من الصيغة الأتية: $\Delta x = \frac{b-a}{N}$

c) يحسب البرنامج قيمة التكامل Y من الصيغة الأتية:

$$Y = \frac{\Delta x}{2}\left(F(a) = F(b) + 2\sum_{i=1}^{N-1} F(xi)\right), \qquad x_i = a + i\Delta x \text{ حيث}$$

d) يطبع البرنامج قيمة b, a في السطر الأول ومقربة إلى أقرب عدد صحيح ومن العمود الرابع، ثم يطبع قيمة N بعد خمس فراغات من طباعة قيم b, a.

e) يطبع قيمة التكامل Y وبالتدوين اليائي (E-Format) في السطر الثالث ومن العمود السابع وبالعنوان التالي: قيمة The value of integration is = Y

الجواب:

```
PROGRAM FUNCTION_STATEMENT
IMPLICIT NONE
REAL :: F , X , A , B , DX , Y , SUM
INTEGER :: I , N
REAL , DIMENSION (100) :: XI
F (X) = EXP(X**2) / (X+2) * LOG(ABS(2-X))
READ *, A, B, N
DX = (B-A) / N
SUM= 0.0
DO I = 1, N-1
XI(I) = A + I * DX
SUM = SUM + F(XI(I))
END DO
Y = DX / 2. * (F(a) + F(b) + 2*SUM)
PRINT "(T4 , 2F6.0 , 5X , I4)", A, B, N
PRINT "(/ , 6X , A , E12.3)", 'The Value Of Integration is = ' ,Y
END PROGRAM FUNCTION_STATEMENT
```

4. أكتب برنامجاً يقرأ قيمتي M, N حيث ان $(N > M)$ ويحسب قيمة كل من Y, Z ويحسب البرنامج قيمة المفكوك بأستخدام برنامج فرعي روتيني خارجي:

$$Y = \frac{N!}{M!(N-M)!}$$

$$z = 1 + M + \frac{M^2}{2!} + \frac{M^3}{3!} + \ldots + \frac{M^N}{N!}$$

الجواب:

```
PROGRAM SUBPROGRAM_3
IMPLICIT NONE
REAL :: Y , Z = 1.0
INTEGER :: I ,N, M,FN, FM ,FNM , FI
READ *, N , M
CALL FACTOR ( N, FN )
CALL FACTOR ( M, FM )
```

```
CALL FACTOR ( N-M, FNM )
Y = FN / ( FM * FNM )
PRINT *, "Y =" , Y
DO I = 1, N
CALL FACTOR ( I, FI )
Z = Z + M**I / FI
END DO
PRINT *, 'Z =' , Z
ENDPROGRAM SUBPROGRAM_3
SUBROUTINE FACTOR (K, FACT)
IMPLICIT NONE
INTEGER :: I , FACT , K
FACT = 1
DO I = 1, K
FACT = FACT * I
END DO
END SUBROUTINE FACTOR
```

واجب بيتي (Home Work):

1. أكتب برنامج لقراءة X وحساب قيمة S من المتسلسلة التالية:

$$S^2 = \frac{X^2}{2*1} - \frac{X^5}{4*3!} + \frac{X^8}{6*6!} - \frac{X^{11}}{8*9!} + \ldots$$

أستعمل برنامج فرعي دالي لحساب المفكوك، وأطبع المتسلسلة عندما يصبح الفرق بين أي حدين متتالين أقل من $(2.5 * 10^{-3})$ وبثلاث مراتب بعد الفارزة وبالعنوان التالي:

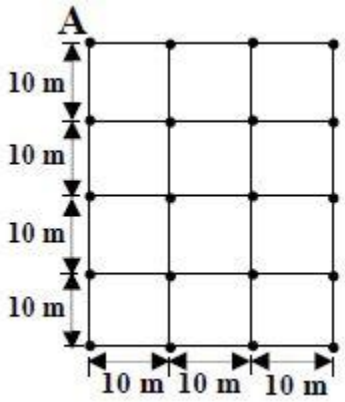
العمود الخامس ↓

السطر الثالث ⟵ The Value of Series S = الجواب

2. أكتب برنامجاً لإيجاد مربع قيمة العنصر بأكبر قيمة مطلقة من عناصر الصف رقم K في مصفوفة مربعة تشمل علىN صف وN عمود، أستخدم برنامج فرعي روتيني لإيجاد أكبر عنصر بقيمة مطلقة.
3. أكتب برنامجاً فرعياً روتينياً لإيجاد الجزء الصحيح INTPRT تعني (INTEGER PART) والجزء الكسري FRCPRT يعني (FRACTUR PART) لعدد حقيقي A ثم أختبر صحة

البرنامج بكتابة برنامج رئيسي يقرأ ثلاثة أعداد X,Y,Z ويوجد لكل منها الجزء الصحيح والجزء الكسري وذلك بالاستعانة بالبرنامج الفرعي، مثلاً إذا كان العدد 12.25 فأن الجزء الكسري هو 25. والجزء الصحيح هو12.

4. في مساحة مستطيلة أخذت مناسيب النقاط المبينة في الشكل أدناه أكتب برنامجاً واحداً لتنفيذ ما يلي:
   a) قراءة هذه المناسيب.
   b) أستدعاء برنامج فرعي لاستخراج النقطة ذات المنسوب الأعلى.
   c) حساب البعد الأفقي والبعد العمودي لهذه النقطة عن النقطة.
   d) طبع الناتج.



Chapter Six

Derived Types and Modules in FORTRAN90

أنواع المشتقة (Derived Types):

لقد مر معنا أنواع المعطيات في لغة FORTRAN (صحيحة ،حقيقية ، منطقية ، مركبة ورمزية) وتجيز لغة FORTRAN90 أضافة أنواع جديدة يجري تعريفها من قبل المبرمج وهذه الأنواع يجب أن تكون من نوع تركيبي او بنيوي Structured أضافة أنواع جديدة يجري تعريفها من قبل المبرمج وهذه الأنواع يجب أن تكون من نوع. تجري عملية تعريف نوع جديد بأستخدام كلمة Type في البداية وكلمة End Type في النهاية وما بين هاتين الكلمتين ينحصر تعريف مكونات التركيب التي تجري بطريقة التعريف العادية. والشكل العام للتعريف هو:

$$TYPE[::]type-name$$
$$component-definitions$$
$$END\ TYPE[type-name]$$

وللتصريح بمتغيرات تحمل نفس النوع نستخدم التعبير $TYPE(type-name) :: var1, var2, \ldots$

لنفترض أننا نريد تعريف تركيب باسم Line ومكونات هذا التركيب هي Line_Number و Text المكون الأول من النوع الصحيح والمكون الثاني من نوع رمزي تجري عملية تعريف هذا التركيب كما يلي:

```
TYPE LINE
INTEGER :: LINE_NUMBER
CHARACTER (LEN= LINE_LENGTH) :: TEXT
END TYPE LINE
```

ولتعريف متغير من نوع تركيبي من نوع LINE وليكن اسم هذا المتغير NEW_LINE نكتب:

```
TYPE (LINE)::NEW_LINE
```

ولتعريف متغير BASIC_PROGRAM كمصفوفة عناصرها من نوع التركيب LINE ويجري التعريف كما يلي:

```
TYPE (LINE) , DIMENSION (MAX_LINES) :: BASIC_PROGRAM
```

أن الطريق البديل لهذا التعريف ، هو أن تعرف BASIC_PROGRAM كمصفوفتين أحداهما من نوع صحيح لتمثيل رقم السطر والأخرى من نوع رمزي يمثل جمل البرنامج.

لو أستخدمنا الطريقة البديلة ، لأحتجنا عند رغبتنا بطباعة سطر من البرنامج لاستخدام جملة كهذه:

PRINT "(I5,X,A)" , LINE_NUMBER(L) , TEXT(L)

أما أذا أستخدمنا التركيب فإن طريقة الطباعة تكون:

PRINT "(I5,X,A)" , LINE(L)

وهي طريقة أنسب من الأولى.

وكمثال آخر لتعريف أي شخص يشمل التعريف (الأسم والعمر ورقم الهوية) بالشكل التالي:

```
TYPE PERSON
CHARACTER ( LEN = 10 ) :: NAME
REAL :: AGE
INTEGER :: ID
END TYPE PERSON
```

ولتعريف المتغيرين ALI , AHMED نكتب : TYPE (Person) :: ALI , AHMED

ولتعريف مصفوفة تسمى People تحوي على 10 أشخاص بنفس التعريف السابق نكتب:

```
TYPE ( Person ), DIMENSION ( 10 ) :: people
```

ولتعريف سجل لأشخاص يحوي أرقام تلفوناتهم ، بحيث يحتوي الرقم على رمز المدينة ورقم التلفون ، وكذلك يحتوي سجل كل شخص على أسم الدولة والمدينة وكذلك الرمز البريدي. أن أفضل طريقة لهذا التعريف أن نأخذ تعريف رقم التلفون وتعريف العنوان كلاً على حدة وأن نعرف كلاً منها كتركيب لوحده كما يلي:

```
TYPE PHONE_TYPE
INTEGER :: AREA_CODE , NUMBER
END TYPE PHONE_TYPE
TYPE ADDRESS_TYPE
INTEGER :: NUMBER
CHARACTER (LEN= 30) :: STREET , CITY
CHARACTER (LEN= 20) :: STATE
END TYPE ADDRESS _TYPE
```

ومن ثم دمج هذين التعريفين في تعريف واحد بالإضافة الى بعض المكونات الأخرى كما يلي:

```
TYPE PERSON_TYPE
CHARACTER (LEN= 40) :: NAME
TYPE (ADDRESS_TYPE) :: ADDRESS
TYPE (PHONE_TYPE) :: PHONE
CHARACTER (LEN= 100) :: REMARKS
END TYPE PERSON_TYPE
```

وأخيرًا يمكن تعريف المتغير AHMED كمايلي:

```
TYPE (PERSON_TYPE) :: AHMED
```

المقطوعات (Modules):

المقطوعة هي مجموعة من المتغيرات والبرامج الفرعية ، ويستطيع أي برنامج رئيسي أو فرعي أن يرجع اليها حسب حاجته ، وبذلك تكون هذه المتغيرات والبرامج الفرعية عامة لأي برنامج فرعي ، والشكل العام للمقطوعة هو:

```
MODULE module-name
IMPLICIT NONE
[specification part]
CONTAINS
[internal functions/subroutines]
END MODULE module-name
```

ملاحظات :

1. أن المقطوعة MODULE تشبه البرنامج الرئيسي الا أنه لا تحتوي على عبارات تنفيذية.
2. أن التعريف بالمتغيرات داخل المقطوعة ووجود البرامج الفرعية الداخلية هو أختياري.
3. يمكن للمقطوعات MODULES أن تحتوي في داخلها على برامج فرعية داخلية أو لا تحتوي.
4. لاستعمال المقطوعة داخل البرنامج الرئيسي نستخدم الأمرUSE بعد العبارة الأولى في البرنامج الرئيسي.

مثال على MODULE لا يحتوي على برامج فرعية هو MODULE CONSTANTS:

```
MODULE CONSTANTS
IMPLICIT NONE
REAL, PARAMETER :: PI=3.1415926
REAL, PARAMETER :: g = 980
```

```
INTEGER :: Counter
END MODULE CONSTANTS
```

مثال على MODULE لا يحتوي على جزء التعريف بالمتغيرات:

```
MODULE SUMAVERAGE
CONTAINS
REAL FUNCTION Sum(a, b,c)
IMPLICIT NONE
REAL, INTENT(IN) :: a, b, c
Sum = a + b + c
END FUNCTION Sum
REAL FUNCTION Average(a, b, c)
IMPLICIT NONE
REAL, INTENT(IN) :: a, b, c
Average = Sum(a,b,c)/3.0
END FUNCTION Average
END MODULE SumAverage
```

مثال على MODULE لا يحتوي على جزء التعريف بالمتغيرات وبرامج فرعية داخلية وكتابة برنامج رئيسي:

```
PROGRAM MAIN
USE MyUtils
IMPLICIT NONE
REAL :: A, B
READ*, A
B = Pi
CALL SWAP( A, B )
PRINT*, A, B
END PROGRAM MAIN
```

Whilethe module(again saved in a separatefile)is

```
MODULE MyUtils
IMPLICIT NONE
REAL, PARAMETER :: Pi = 3.1415927
```

```
CONTAINS
SUBROUTINE SWAP( X, Y )
REAL Temp, X, Y
Temp = X
X = Y
Y = Temp
END SUBROUTINE SWAP
END MODULE MyUtils
```

لنفترض أن لدينا قائمة بأرقام بطاقات ضائعة LOSTCARD وهناك متغير يحمل عدد هذه البطاقات NUMBER_OF_LOST_CARDS بإمكاننا الآن تعريف المقطوعة التالية:

```
MODULE LOST_CARD_MODULE
INTEGER , DIMENSION (:) , ALLOCATABLE :: LOST_CARD
INTEGER :: NUMBER_OF_LOST_CARDS
END MODULE LOST_CARD_MODULE
```

من الملاحظ أن المقطوعة محصورة بين الثنائية END MODULE و MODULE وأن هذه المقطوعة تحمل الاسم LOST_CARD_MODULE.

ويمكن لروتين مثل SEARCH أن يستعمل هذه المتغيرات وذلك بأستخدام جملة التعريف USE كما يلي:

```
SUBROUTINE SEARCH (CARD_NUMBER , FOUND)
USE LOST_CARD_MODULE
…..
DO I = 1 , NUMBER_OF_LOST_CARDS
IF (CARD_NUMBER == LOST_CARD(I) ) THEN
…..
```

وأذا استخدم أي روتين آخر نفس المقطوعة ، فإنه يشترك مع هذا الروتين باستخدام المتغيرات الموجودة في هذه المقطوعة ، وبذلك يمكن أحد الروتينات أن يعطي قيماً لهذه المتغيرات ويقوم روتين آخر بالاستفادة من هذه القيم.

ملاحظة :هناك شكلين لأستخدام العبارة USE في البرنامج الرئيسي هي:

```
USE module-name
USE module-name, ONLY: name_1, name_2, ..., name_n
```

مثال

```
MODULE SomeConstants
  IMPLICIT NONE
  REAL, PARAMETER :: PI = 3.1415926
  REAL, PARAMETER :: g = 980
  INTEGER         :: Counter
END MODULE SomeConstants
```

```
PROGRAM Main
  USE SomeConstants
  IMPLICIT NONE
    ..........
END PROGRAM Main
```

```
MODULE DoSomething
  USE SomeConstants, ONLY : g, Counter
  IMPLICIT NONE
CONTAINS
  SUBROUTINE Something(…)
    ……
  END SUBROUTINE Something
END MODULE DoSomething
```

PI is not available

ملاحظة:

يمكن كتابة MODULE بنفس الملف للبرنامج الرئيسي وذلك بكتابة الـــ MODULE في البداية ثم كتابة البرنامج الرئيسي كما في المثال التالي:

```
MODULE CONSTANTS
IMPLICIT NONE
REAL, PARAMETER :: PI=3.1415926 , e = 2.7182818
END MODULE CONSTANTS
PROGRAM USECONSTANT
USECONSTANTS
IMPLICIT NONE
real :: radius , area
radius = 10.5
AREA = pi * radius **2
print *, "area = " , area
ENDPROGRAM USECONSTANT
```

كما يمكن كتابة كلاً من البرنامج الرئيسي والمقطوعة MODULE في ملف منفصل كما في المثال التالي:

- أطبع MODULE باسم constmod.f90
- والبرنامج الرئيسي باسم const.f90

- ثم ترجم MODULE ثم ترجم البرنامج الرئيسي وبعدها أعمل الربط (LINKING) ثم التنفيذ.

```
module constants
implicit none
real, parameter :: pi = 3.1415926536
real, parameter :: e = 2.7182818285
contains
subroutine show_consts
print*, "pi = ", pi
print*," e = ", e
end subroutine show_consts
end module constants
```

```
program module_example
use constants
implicit none
real :: twopi
twopi = 2 * pi
call show_consts()
print*, “twopi = “, twopi
end program module_example
```