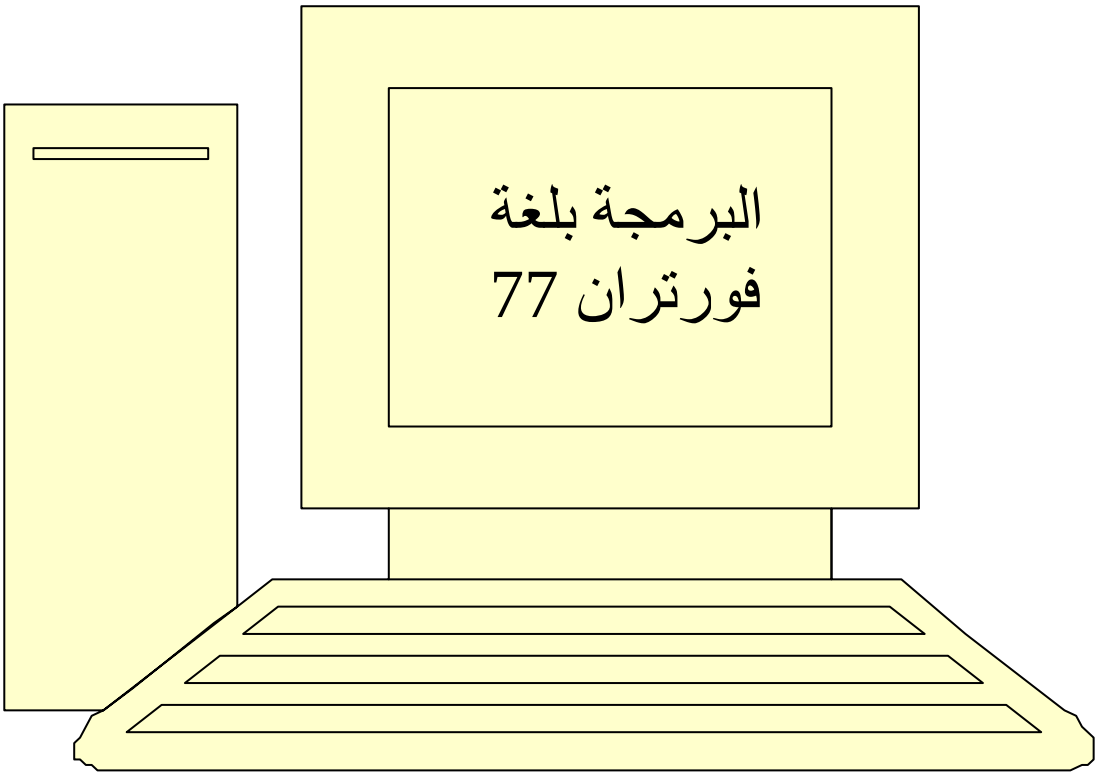


المصدر: فورتران 77 مدخل إلى برمجة الحاسبات، تأليف د.محمد زكي، ونبيل خليل.

الفصل الأول: مدخل إلى الحاسوب

الفصل الثاني: مبادئ لغة فورتران.

الفصل الثالث: جمل الشرط والتحكم.

الفصل الرابع: التكرار.

الفصل الخامس: جمل الإدخال والإخراج.

الفصل السادس: المتغيرات الموسومة.

الفصل السابع: البرامج الفرعية.

الفصل الثامن: الملفات.

الفصل الأول
مدخل إلى الحاسوب

الحاسبة: هي عبارة عن مجموعة من الأجهزة تعمل متكاملة مع بعضها لمعالجة مجموعة من البيانات الداخلة وفقا لبرنامج موضوع مسبقا للحصول على النتائج المطلوبة.

ومن أهم مميزات الحاسبة:

١. السرعة في انجاز العديد من العمليات الحسابية أو المنطقية خلال زمن قليل جدا.
٢. دقة نتائجها.
٣. القابلية على خزن المعلومات والبيانات واسترجاع هذه المعلومات عند الحاجة.
٤. قابليتها على العمل باستمرار ولفترة طويلة دون الإحساس بالتعب أو الملل.
٥. نظرا لأهميتها يمكن استخدامها في تطبيقات مختلفة في المجالات العلمية والإدارية والاقتصادية.

أنواع الحاسبات:

A. تصنيف الحاسبة حسب الحجم:

تختلف الحاسبات بعضها عن بعض في الكفاءة والأداء ويمكن اعتبار ان زيادة حجم الذاكرة تؤدي إلى زيادة سرعة وكفاءة الحاسبات وبهذا يمكن تقسيم الحاسبات حسب حجم الذاكرة إلى:

١. الحاسبات الصغيرة.
٢. الحاسبات المتوسطة.
٣. الحاسبات الكبيرة.

B. تصنيف الحاسبة حسب غرض الاستعمال:

1. الحاسبات خاصة الاستعمال (special purpose computers):

تستخدم هذه الحاسبات لأداء وظيفة محدده مثل الحاسبات المستخدمة في التحكم في العمليات الصناعية.

2. الحاسبات عامة الأغراض (general purpose computers):

تستخدم هذه الحاسبات في مختلف الميادين والمجالات حيث تمتلك المرونة الكافية لتأمين الكفاءة العالية في المجالات التجارية، العلمية، الطبية، الهندسية.

C. تصنيف الحاسبة حسب البيانات المستخدمة:

١. **الحاسبات الرقمية** (digital computers):

في هذا النوع من الحاسبات يتم تمثيل قيم المتغيرات والكميات بواسطة الأعداد وتمتاز بدقتها العالية ويمكن استخدامها في كافة المجالات التجارية والعلمية.

٢. **الحاسبات التناظرية** (analog computers):

يعالج هذا النوع من الحاسبات البيانات التي تتغير بين كل لحظة وأخرى مثل درجة الحرارة، الرطوبة، سرعة الرياح، وتعتبر هذه الحاسبات أسرع من الحاسبات الرقمية إلا ان دقتها اقل وتستخدم في المجالات العلمية.

٣. **الحاسبات الهجينة (hybrid computers)**:

يجمع هذا النوع من الحاسبات ميزات الحاسبات الرقمية والتناظرية حيث تمتاز بقدرتها على تخزين البيانات ودقتها ومن مساوئها تكلفتها العالية.

أجزاء الحاسبة:

تتألف الحاسبة من جزئين رئيسيين هما:

- الماديات (Hardware).
- البرمجيات (Software).

الماديات (Hardware): وهي المكونات المادية للحاسبة وتتألف من الوحدات التالية:

١. **وحدة الإدخال** (input unit):

تقوم هذه الوحدة بإدخال المعلومات والبرامج إلى الحاسبة ومن أهم هذه الوحدات هي:

أ. لوحة المفاتيح (key board).

ب. الفارة (mouse).

ت. الماسح الضوئي (scanner).

٢. **وحدة المعالجة المركزية** (central processing unit):

تشكل وحدة المعالجة المركزية الجزء الرئيسي للحاسبة وتتولى معالجة المعلومات وتنفيذها حيث تقوم بجميع العمليات الحسابية والمنطقية كما تقوم بالسيطرة على سير العمليات داخل الحاسبة وتتكون وحدة المعالجة المركزية من الأجزاء التالية:

أ. **وحدة الحساب والمنطق** (arithmetic and logical unit):

تقوم هذه الوحدة بجميع العمليات الحسابية (القسمة، الضرب، الجمع، الطرح) كما تقوم بالعمليات المنطقية.

ب. **وحدة السيطرة** (control unit):

تقوم وحدة السيطرة بتنسيق جميع الأنشطة داخل وحدة المعالجة المركزية وتنفيذ التعليمات حسب ورودها في البرنامج حيث تقوم بتفسير هذه التعليمات واصدار الأوامر إلى الأجزاء الأخرى للحاسبة لتنفيذ تلك الأوامر.

٣. **وحدة الذاكرة** (memory unit):

تقوم هذه الوحدة بخزن المعلومات والنتائج وتعرف الوسيلة المستخدمة لخزن المعلومات أو البيانات التي تتعامل معها الحاسبة بالذاكرة، وتقاس الذاكرة ب (Byte) وكل (Byte) مكون من (8 Bit).

وللحاسبة نوعان من الذاكرة:

أ. **الذاكرة الرئيسية** (main memory):

وتستخدم لتخزين البيانات والبرامج التي يراد تنفيذها والتي تتلاشى بمجرد الانتهاء من تنفيذ البرنامج وتتميز هذه الذاكرة بالسرعة العالية في تبادل المعلومات.

ب. **الذاكرة المساعدة** (auxiliary memory):

وهي وحدة ثانوية لخزن المعلومات والبرامج وتتميز هذه الذاكرة بان سعتها اكبر ولكنها اقل سرعة من الذاكرة الرئيسية وتستطيع أن تحتفظ بالمعلومات لمدة طويلة.

٤. **وحدة الإخراج** (output unit):

تقوم هذه الوحدة بإخراج المعلومات المخزونة في الذاكرة أو النتائج المستحصلة من العمليات الحسابية والمنطقية إلى الوسط الخارجي ومن أهم هذه الوحدات:

أ. الشاشة (monitor).

ب. الطابعة (printer).

البرمجيات (Software): وهي مجموعة البرامج التي تحتاجها الحاسبة أثناء العمل، والبرنامج هو مجموعة من الايعازات التي تكتب لحل مسالة ما وتكون هذه الايعازات مرتبة بأسلوب يتفق والطريقة التي تعمل بها الحاسبة.

نظام التشغيل: وهو عبارة عن مجموعة من البرامج والبيانات تعمل بصورة منتظمة ومترابطة لتشغيل الحاسوب وتسهيل التعامل معه وتشغيل البرامج المتوفرة لمعالجة البيانات بأفضل صورة ومن أهم أنظمة التشغيل:

- نظام التشغيل CPM
- نظام التشغيل MS-DOS
- نظام التشغيل WINDOWS

لغات البرمجة:

بشكل عام هناك نوعان من لغات البرمجة وهي اللغات العالية المستوى (high level languages) واللغات واطئة المستوى (low level languages)، وتتم كتابة البرامج للحاسبة باللغات العالية المستوى حيث تتفق مجموعة الأوامر الخاصة بها مع لغات وأفكار الإنسان. ان معظم اللغات العالية المستوى لغات لإغراض عامة ومن أهمها:

1. لغة فورتران (Fortran language): تستخدم للتطبيقات العلمية والرياضية وكلمة فورتران (Fortran) هي اختصار (formula translator).

2. لغة كوبول (Cobol language): وهي من أكثر اللغات شيوعا في مجال التطبيقات التجارية.

3. لغة باسكال (bascal language): وتستخدم للأغراض العلمية والتجارية.

4. لغة بيسك (basic language): وهي لغة سهلة التعلم وكلمة (basic) هي اختصار (beginner's all-purpose symbolic instruction code).

5. لغة (quick basic): وهي لغة أكثر تطورا من لغة (basic) وتمتاز بسهولة التحكم والتنفيذ.

حل المسائل بواسطة الحاسبة:

إن حل المسائل بواسطة الحاسبة يتم من خلال الخطوات التالية:

١. تحديد وتحليل المسألة.
٢. كتابة الخوارزمية.
٣. رسم المخطط الانسيابي.
٤. كتابة البرنامج.
٥. اختبار البرنامج.
٦. توثيق البرنامج.

١. تحديد وتحليل المسألة:

ويتم ذلك من خلال معرفة معطيات المسألة والغرض منها اي وضع تصور كامل للمسألة المراد حلها وكيفية الحل.

٢. الخوارزميات (algorithms):

وهي عبارة عن مجموعة من الخطوات المرتبة بشكل متسلسل لتنفيذ عمليات حسابية أو منطقية أو غيرها.

مثال: اكتب خوارزمية لحساب قيمة المعادلة التالية:

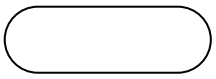
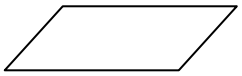
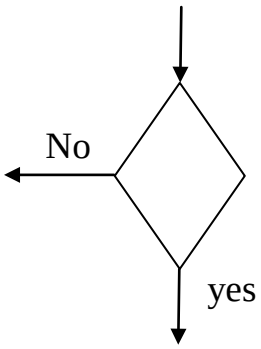
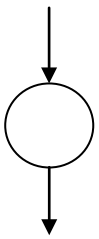
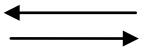
$$y = A + B$$

١. أبدأ
٢. ندخل قيمة A , B
٣. نجمع قيمة A مع قيمة B ونجعلها مساوية إلى y
٤. نطبع قيمة y
٥. توقف

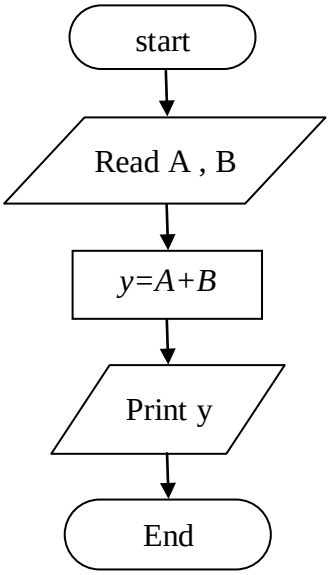
3. المخطط الانسيابي (flow chart):

هو عبارة عن مجموعة من الأشكال المترابطة التي تشير إلى العمليات الضرورية في البرنامج ويستعين المبرمج بهذه المخططات لتساعده في متابعة خطوات الحل من البداية إلى النهاية.

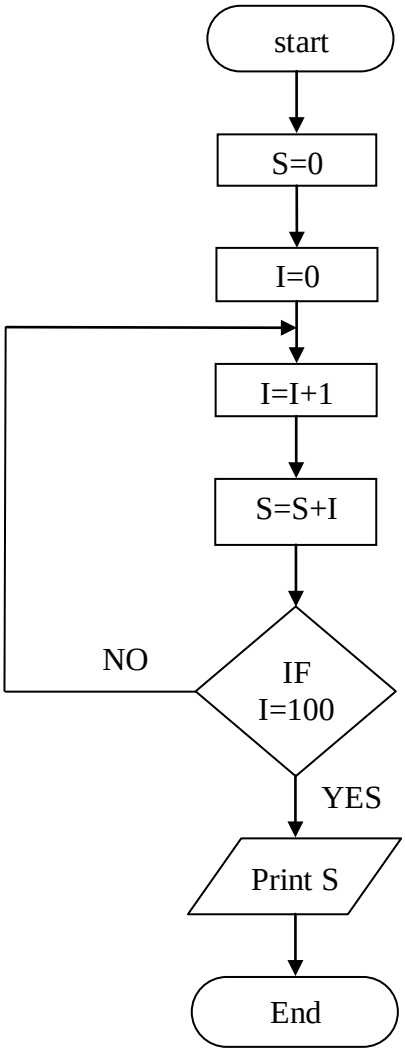
وتستخدم في المخططات الانسيابية مجموعة من الأشكال لتوضيح العمليات والايعازات ومن هذه الأشكال:

الشكل	العملية التي يدل عليها
	يستعمل هذا الرمز لبداية أو نهاية البرنامج (start , end)
	يستعمل هذا الرمز لجمل الإدخال والإخراج (input , out put)
	يستعمل هذا الرمز لتوضيح عمليات معالجة المعلومات ويشير إلى وجود عملية حسابية رياضية وتكتب العملية داخل الرمز
	يستعمل هذا الرمز عندما تكون الحاجة لاتخاذ قرار معين أو المفاضلة بين اختيارين أو أكثر
	يستعمل هذا الرمز للتوصيل
	يستعمل لتوضيح اتجاه سير تنفيذ العمليات في داخل المخطط

مثال: ارسم المخطط الانسيابي لحساب قيمة (y) حيث إن: $y = A + B$



مثال: أرسم المخطط الانسيابي لحساب مجموع الأعداد من (1) إلى (100):



H.W : ارسم المخطط الانسيابي لحساب قيمة كل من y , z من العلاقات التالية:

$$y = x^2 - \frac{1}{x}$$
$$z = y^3 + 6x$$

4. كتابة البرنامج:

بعد الانتهاء من المخطط الانسيابي نقوم بكتابة البرنامج وذلك بتحويل المخطط الانسيابي إلى مجموعة من الجمل التي تفهمها الحاسبة وهذه الجمل مجتمعة تسمى بالبرنامج.

5. اختبار البرنامج وتصحيح الأخطاء الموجودة فيه.

٦. **توثيق البرنامج:** بعد التأكد من عمل البرنامج بصورة صحيحة يتم توثيقه ليتسنى للآخرين الاستفادة منه.

الفصل الثاني

مبادئ لغة فورتران 77 (Fortran 77)

مقدمة:

تعد لغة فورتران من أقدم لغات البرمجة العليا الموجودة، وهي أول لغة عالمية عالية المستوى وجدت لبرمجة الحاسبة وباستخدام حروف اللغة الانكليزية والنظام العددي العشري. وتعد هذه اللغة من أكثر اللغات شيوعاً واستعمالاً للأغراض العلمية.

إذ أدى انتشار وتطور القطاعات الصناعية المختلفة إلى تحفيز الباحثين للبحث عن وسائل جديدة لدراسة وتطوير هذه الصناعات، وذلك من خلال تصميم برامج حاسوبية تعمل على تسهيل التحليل الرياضي والتصميم لمختلف النماذج الصناعية. وقد صممها مجموعة من الباحثين في شركة (IBM) ومن ثم طورت إلى اللغة المعروفة بلغة فورتران الثانية (FORTRAN II)، حيث أصبح بالإمكان كتابة برامج فرعية ضمن البرنامج الرئيس بعدما كان ذلك غير ممكن في لغة فورتران واحد ثم طورت لغة فورتران الثانية إلى اللغة المعروفة بلغة فورتران الرابعة (FORTRAN IV). وامتازت هذه اللغة الأخيرة عن سابقتها بإمكانية إجراء العمليات المنطقية (LOGICAL OPERATION) وعمليات الأعداد المعقدة، ثم تطورت إلى صورة فورتران 77.

إن كلمة فورتران هي اختصار لكلمتين هما كلمة (FORMULA) وتعني المعادلة أو الصيغة والكلمة الأخرى (TRANSLATION) وتعني ترجمة المعادلات أو الصيغ. وبالرغم من أن تصميم لغة فورتران كان فقط لغرض التطبيقات العلمية، لكنها قابلة للاستخدام لحل مسائل تجارية، وبالنظر لشعبية هذه اللغة فإنها لا زالت مستمرة حتى يومنا هذا، وعلى الرغم من ظهور لغات أخرى أكثر كفاءة إلا أن لغة فورتران 77 ظلت تحتل موقع الصدارة وخاصة في الاستخدامات العلمية. وهي كأي لغة أخرى تتألف من مجموعة من العبارات والإيعازات الموضوعة حسب قواعد معينة ينبغي الالتزام بها.

الرموز الأساسية في لغة فورتران 77:

تتكون لغة فورتران من أحرف لاتينية (صغيرة أو كبيرة) (A←Z) وأرقام (0←9) بالإضافة إلى الرموز الموضحة في الجدول:

الرمز	معناه	الرمز	معناه
'	علامة المتن	( )	قوسين
:	نقطتان رأسية	+	علامة الجمع
,	فارزة	−	علامة الطرح
.	فارزة عشرية	*	نجمة وعلامة الضرب
**	الرفع	/	علامة القسمة
=	المساواة		

الثوابت Constants:

كمية ثابتة لا تتغير خلال تنفيذ البرنامج، وهي على أنواع:

1- الثابت الصحيح Integer Constant:

هو العدد الذي يخلو من الفارزة العشرية وقد يُسبق بعلامة سالبة (−) أو موجبة (+) اختيارية، ويحجز الثابت الصحيح في ذاكرة الحاسبة أربعة بايتات (البايت: هي الخلية التي تتكون منها وحدة الذاكرة ذات ثمانية أرقام ثنائية (بت)) مثل:

400,-2,-85,160

2- الثابت الحقيقي Real Constant:

هو العدد الذي يحتوي على الفارزة العشرية، وقد يُسبق بعلامة سالبة (−) أو موجبة (+) اختيارية، مثل:

-4.625, 0.009, 2.5, -0.37666, -7.5

كما قد يظهر الثابت الحقيقي مع الأس وعندئذ يستعمل الحرف E للدلالة على الأساس 10 مرفوعاً إلى الأس، مثل:

2.5×10^{-5}=2.5E-5

$500=5\times10^{2}$=5E2

ويحجز الثابت الحقيقي ذو الدقة الاعتيادية أربعة بايتات في الذاكرة، ويكتب بما لا يزيد على سبعة أرقام للمحافظة على دقته.

أما الثابت الحقيقي ذو الدقة المضاعفة فيحجز ثمانية بايتات في الذاكرة، ومن ثم تزداد دقة الثابت، وباستعمال ما لا يزيد على 15 رقماً، ويستعمل الحرف D للدلالة عليه، مثل:

$5030=50.3\times10^{2}$=50.3D2

3- الثابت المركب Complex Constant:

يتكون من جزأين حقيقي وخيالي، ويكتب بالشكل الآتي:

A-B.i

إذ أن A يمثل الجزء الحقيقي و B.i يمثل الجزء الخيالي أو الجذر التربيعي لـ(1−). يكتب الثابت المركب داخل أقواس، مثل:

(1.1,2.2)=1.1+2.2i

يحجز الثابت المركب ثمانية بايتات متجاورة في الذاكرة، أربعة بايتات للجزء الحقيقي، وأربعة بايتات أخرى للجزء الخيالي، ويمكن زيادة عدد البايتات لزيادة الدقة.

4- الثابت المنطقي Logical Constant:

ويكون على حالتين خطأ (قيمة زائفة) (.false.) أو صواب (قيمة حقيقية) (.true.)، ويحجز بايتاً واحداً في الذاكرة.

5- الثابت الحرفي Character Constant:

لا يحوي الثابت الحرفي قيمة عددية بل هو عبارة عن حرف أو مجموعة من الحروف، وتستخدم علامة المتن لإحاطة حروف الثابت، مثل:

'ALI', 'Tikrit University'

ويعد الفراغ حرفاً ويحجز كل حرف بايتاً واحداً.

المتغيرات Variables:

المتغير هو اسم يرتبط بقيمة قد تتغير أو تبقى ثابتة أثناء تنفيذ البرنامج.

فعلى سبيل المثال مساحة الدائرة A:

$A=\pi\times R^2$

فإن π تمثل النسبة الثابتة، ومن ثم فإن النسبة الثابتة تمثل ثابتاً حقيقياً، أما R فتمثل نصف قطر الدائرة، وكلما تغيرت قيمة R تغيرت المساحة A، إذا كل من A وR متغير يستوعب قيمة عددية واحدة قد تتغير أثناء التنفيذ.

تبدأ المتغيرات من (A← Z) أو (a← z) وقد تلحق بأرقام، مثل:

Hour, X20, I, Mosul.

المتغيرات الغير مقبولة:

(لوجود .) I.K, (للابتداء برقم) 2Y , (لوجود –) Y7-2

1- المتغير الصحيح Integer Variable:

يكون المتغير صحيحاً إذا ابتدأ اسمه بالأحرف التالية: i, j, k, l, m, n، مثل: Kt, L2, Min.

وتحجز أربعة بايتات. يمكن استخدام عبارة Integer النوعية عند استخدام أحرف لا تبدأ (من i إلى n) أو حتى لتغيير عدد البايتات المحجوزة، مثل:

```
Integer *2 S
Integer x,y,z      or Integer *4 x,y,z
Integer *1 v1, v2
```

فالمتغيران v1 و v2 متغيران صحيحان، يحجز لكل منهما بايت واحد في الذاكرة.

2- المتغير الحقيقي Real Variable:

هي المتغيرات التي تبدأ بأي حرف عدا الحروف (من i إلى n)، المتغير الحقيقي ذو الدقة الاعتيادية يخزن أربعة بايتات، مثل:

```
ZED, area
Real L1
```

أو دقة مضاعفة

```
Real *8 M1, M2
```

تستخدم real لتحقيق هدفين، أولهما حجز (تحديد) عدد البايتات للمتغير (كما تم تحويله في المثال أعلاه إلى دقة مضاعفة)، وثانيهما لتحويل المتغيرات الصحيحة التي تبدأ (من i إلى n) إلى متغيرات حقيقية.

3- المتغير المركب Complex Variable:

وهي تتكون من جزأين حقيقي وخيالي ويعرف بالجملة:

```
Complex x,y,z
```

وتخزن في الحاسبة ثمانية بايتات، وللدقة المضاعفة:

```
Complex *16 x,y
```

وهي جملة نوعية تستخدم لتحديد صفة المتغير وعدد البايتات المحجوزة لكل متغير مركب، وبذلك تطابق عمل integer و real.

4- المتغير المنطقي Logical Variable:

يعرف في بداية البرنامج باستخدام العبارة النوعية:

```
Logical A,B
```

A وB متغيرات منطقية تخزن واحد بايت، لأن القيمة المخزونة إما أن تكون حقيقية (.true.) أو زائفة (.false.).

5- المتغير الحرفي Character Variable:

يتعامل مع الحروف بدلاً من الأعداد، يعتمد الحجز على عدد الحروف التي يتكون منها المتغير الحرفي.

```
Character A, B, C
Character A*1, B*1, C*1
```

أو

```
Character *1 A, B, C
Character *6 VAR*3, IN, NAME*20
```

هنا يأخذ المتغير VAR ثلاثة بايتات، والمتغير IN ست بايتات، والمتغير NAME عشرين بايتاً، إذ يمكن إعطاء حجز مختلف لكل متغير حرفي على حدة، وهي ميزة لا تجدها مع الجمل النوعية السابقة.

مثال:

```
Real K,L
K=5.3
L=3.2
S=L+K
Print *, 'S=',S
Stop
End
```

التعابير Expressions:

يعرف التعبير بأنه مجموعة ثوابت أو متغيرات تفصل فيما بينها عمليات معينة كالعمليات الحسابية تقود إلى الحصول على ناتج، وهي على أنواع:

- تعبير حسابي
- تعبير حرفي
- تعبير علاقي
- تعبير منطقي

سنشرح هنا التعبير الحسابي فقط، وسنؤجل شرح بقية التعابير إلى فصول لاحقة.

التعبير الحسابي Arithmetic expression:

يستخدم فيه العمليات الحسابية للفصل بين المتغيرات أو الثوابت للحصول على ناتج، والعمليات الحسابية هي: الإضافة (+) والطرح (−) والضرب (*) والقسمة (/) والرفع (**)، وتكتب من اليسار إلى اليمين وفي مستوى واحد، مثال:

x^3-5y+z

تكتب بالشكل التالي:

x**3-5*y+z

تسلسل تنفيذ التعبير الحسابي:

1) تنفذ عملية الرفع (**) أياً كان موقعها.
2) ثم تنفذ عملية الضرب (*) أو القسمة (/)، فلها أسبقية واحدة، إلا أن التنفيذ يبدأ بالعملية الأسبق من اليسار إلى اليمين.
3) ثم عملية الإضافة (+) أو الطرح (−)، فلها أسبقية واحدة، إلا أن التنفيذ يبدأ بالعملية الأسبق من اليسار إلى اليمين.

وعند استخدام الأقواس فإن تسلسل التنفيذ يبدأ بمحتويات الأقواس أولاً، وبالأسبقيات المعروفة نفسها.

مثال:

6+2**3/4
=6+8/4
=6+2
=8

مثال:

4*6**2/(2+3*4-12)
=4*6**2/(2+12-12)
=4*6**2/(14-12)
=4*6**2/2
=4*36/2
=144/2
=72

ملاحظات عامة حول استخدام التعبير الحسابي:

١) لا يجوز استخدام علامتي عمليتين حسابيتين متتاليتين من دون أن يفصل بينهما متغير أو ثابت، باستثناء عملية الرفع (**) التي تعد عملية واحدة، فلا يجوز كتابة (P*-Q)، بل يستعمل القوسان (P*(-Q)).

٢) يفضل توحيد المتغيرات والثوابت (m-5.0) ⇐ (m-5).

٣) قسمة ثابت صحيح على ثابت صحيح تعطي ناتجاً صحيحاً مثل (5/2=2)، فإذا أردت ناتجاً حقيقياً لناتج النسبة فيجب أن يكون أحد الثابتين أو كليهما حقيقياً:

5.0/2=5/2.0=5.0/2.0=2.5

٤) طريقة تنفيذ عملية الرفع داخل الحاسبة: إذا كان الأس صحيحاً (s**2)، تتم عملية الرفع بتكرار ضرب الأساس بنفسه عدداً من المرات يساوي قيمة الأس (s*s)، أما إذا كان الأس حقيقياً، فتتم عملية الرفع لوغاريتمياً، بضرب الأس بلوغاريتم الأساس ثم عكس الناتج لوغاريتمياً، وهنا يجب أن لا تكون قيمة الأساس سالبة، لأن لوغاريتم كمية سالبة غير معرف.

جمل الإحلال Assignment Statement:

في لغة فورتران تأخذ جملة الاحلال الصيغة التالية:

تعبير=متغير

إذ تتكون من طرفين يفصل بينهما عملية المساواة، الطرف الأيمن يتكون من تعبير (متغير أو ثابت أو مجموعة من الثوابت والمتغيرات تفصل فيما بينهما عمليات حسابية)، أما الطرف الأيسر فإنه يتكون من متغير واحد ولا يجوز أن يحتوي على أي عملية حسابية، كما لا يجوز أن يظهر التعبير في الطرف الأيسر محل المتغير، مثل:

B=Z*K-3.0*C

(N1='Name') هي جملة إحلال حرفية، لكن يجب قبلها استخدام جملة character N1.

المعداد Counter: جملة إحلال حسابي تقوم بالعد كلما مر عليها التنفيذ، مثال:

```
x=0.5
x=x+0.5
```

أي كأننا نقول x(new)=x(old)+0.5.

جملة إحلال مركبة: (وهي لها علاقة بالمتغيرات المركبة) وعند الإشارة إليها يجب تعريفها بجملة complex.

```
Z1=r1+q1i     & z2=r2+q2i
complex Z1,Z2
```

العمليات التي تجرى على المتغيرات المركبة:

- الإضافة المركبة $(r1+r2)+(q1+q2).i \Rightarrow k=Z1+Z2$
- الطرح المركب $(r1-r2)+(q1-q2).i \Rightarrow k=Z1-Z2$
- الضرب المركب $(r1.r2-q1.q2)+(r1.q2+q1.r2).i \Rightarrow k=Z1.Z2$

الدوال المكتبية Library Functions:

برنامج فرعي مستقل صمم لأداء عملية معينة من خلال مقدار يعطى للدالة، جدول (1.6) يوضح مجموعة من الدوال المكتبية الأساسية بأنواعها المختلفة.

SQRT: جذر تربيعي (وهي اسم الدالة) يحسب لقيمة y وهو الدليل الوسيط.

```
X=SQRT(5.5)
X=SQRT(y)
X1=SQRT(y+2.5)
```

يجب أن نشير هنا إلى تطابق الناتج مع نوع الاسم فإن كان صحيحاً كان الناتج صحيحاً وهكذا.

الدالة log10 تستخدم لحساب لوغاريتم عدد صحيح أما Alog10 فإنها تحسب اللوغاريتم العشري للعدد الحقيقي ذو دقة اعتيادية و Dlog10 فتقوم بحساب اللوغاريتم العشري لأي مقدار حقيقي ذو دقة مضاعفة،

مثال:

```
Double Precision x
X=0.625D3
Y=16.0
K=121
XR=DLOG10 (x)
YS=Alog10(y)
KT=LOG10(k)
```

عبارات الإدخال والإخراج:

جملة القراءة المباشرة read:

هناك إدخال مباشر عن طريق استخدام جملة الإحلال وكالآتي:

```
A=10
B=1.6
Z=A+B=11.6
```

إلا أننا في كثير من الأحيان نحتاج أن ندخل المتغيرات من وسط خارجي إلى الحاسبة كأن تكون لوحة مفاتيح أو ملف وكالآتي:

```
Read *, A
Read *, x,y,z
```

عند تنفيذ هذه الجملة فإن الحاسبة تطالب بالقيم من الوسط الخارجي، وتستعمل النجمة * للدلالة على القراءة المباشرة التي لا نحتاج فيها أي تعريف للقيمة المقروءة.

<u>قواعد عامة:</u>

١) عندما يكون هناك جملتان للقراءة فإن إدخال البيانات يجب أن يكون في سطرين.

```
Read * ,A,B ⇒ 5,6
Read *, I,J ⇒ 3,4
```

٢) إذا كان عدد البيانات في السطر الواحد أقل مما هو مطلوب في جملة القراءة عندئذ يذهب البرنامج للسطر الثاني.

```
Read *, x, y
11
3.5
```

إذا زادت البيانات يهمل الباقي أما إذا قلت فيظهر خطأ.

٣) يجب أن تتطابق نوعية البيانات مع نوعية المتغيرات المقروءة إلا في حالة المتغيرات الحقيقية والصحيحة فإن البرنامج يبدلها ذاتياً.

```
Read *, A, I ⇒ 1,500 ⇒ 1.0, 500
```

٤) يجوز استخدام الفراغات بدل الفوارز في قيم البيانات.

جملة الطباعة المباشرة print:

وهي جملة إخراج للنتائج من داخل الحاسبة إلى الوسط الخارجي

Print *, c ⇒ 12
Print *, 'z=',z ⇒ z=5
Print *, 'Name:', Name ⇒ Name: Mohammed

جملة التعليق Comment Statement:

هي جملة نشرح بها خطوات البرنامج، وهي جملة غير تنفيذية، وتستخدم الحرف C في بداية العمود الأول أو علامة *.

جملة النهاية END:

هي الجملة التي ينتهي بها البرنامج فهي لا تظهر إلا مرة واحدة في نهاية كل برنامج، والخطوات التي بعدها لا يمكن تنفيذها.

جملة التوقف STOP:

هي جملة تنفيذية توقف تنفيذ البرنامج متى وصل إليها التنفيذ وقد يتكرر ظهورها داخل البرنامج حسب الحاجة.

جملة اسم البرنامج PROGRAM:

تستعمل لإعطاء اسم للبرنامج، وهي غير تنفيذية، ولا يجوز استخدام اسم البرنامج كأحد المتغيرات داخل البرنامج.

Program name

إذ يمثل name اسم البرنامج

H.W:

١. اكتب برنامجاً لحساب المعادلة الآتية:

$C = (I + N)^{0.5}$,I=4.5 , N=6.3

بحيث يطبع الناتج بالشكل: C=

٢. اكتب برنامجاً لحساب المعادلة الآتية:

$y = x + x/z$, x=5 , z=2

بحيث يطبع الناتج بالشكل: y=

٣. اكتب برنامجاً لحساب المعادلة الآتية:

$$C = \frac{4 + \frac{17}{M}}{\sqrt{N}}$$, M=4 , N=5

بحيث يطبع الناتج بالشكل: C=.

٤. اكتب برنامجاً لحساب قيمة (Z) من المعادلة الآتية:

$$z = \frac{a}{\sqrt{a+b}} + \cos(y)\tan^{-1}(x/y)$$

إذا علمت ان (a , b) هما ثابتان حقيقيان ذو دقة مضاعفة و (x , y) هما ثابتان حقيقيان ذو دقة اعتيادية.

الفصل الثالث

جمل الشرط والسيطرة Control Statement

تستخدم لتغيير مسار تنفيذ البرنامج حسب مقتضيات الحل:

١ - جملة (أقصد غير المشروطة) (Unconditional Go to Statement):

تكتب بالصيغة الآتية:

Goto n

n: رقم الجملة التي يذهب إليها التنفيذ.

وتقوم هذه الجملة بنقل التنفيذ من الموقع الحالي إلى الموقع الذي يحدده رقم الجملة (n) كما ويستفاد منها أيضاً في عزل مجموعة من الجمل عند تحقق شرط معين.

```
   Print *, 'Input the Value of A'
10 Read *, A
   B=A**2
   print *, 'B=',B
   Go to 10
   Stop
   End
```

٢ - جملة (أقصد المشروطة) (Conditional Go to Statement):

تكتب بالصيغة الآتية:

Goto (n1,n2, …, nm), k

n1,n2 …nm: أرقام جمل تنفيذية داخل البرنامج.

k: ثابت أو متغير أو تعبير حسابي ويجب أن يكون صحيحاً ويمثل تسلسل الجملة التي ينتقل إليها تنفيذ البرنامج، وتكون قيمته محددة بعدد الجمل التنفيذية المشار إلى أرقامها بين القوسين، فمثلاً إذا كان عدد الجمل التنفيذية أربع فإن أعلى قيمة لـ k أربعة. والفارزة بين القوس و k اختيارية يمكن إهمالها.

مثال: اكتب برنامج بلغة (fortran) لحساب قيمة (y) حسب العلاقات الآتية:

$$y=\begin{cases} A+B & n=1 \\ AB & n=2 \\ A/B & n=3 \\ A^B & n=4 \end{cases}$$

```
    Print *, 'Input the Value of A,B, and n'
    Read *, A,B,n
    Goto (10,20,30,40),n
10  y=A+B
    Goto 50
20  y=A*B
    Goto 50
30  y=A/B
    Goto 50
40  y=A**B
50  print *, 'y=',y
     Stop
     End
```

ملاحظة للبرنامج السابق: يجب أن تكون n قيمة تتراوح من 1 إلى 4 فعند إدخال قيمة n=2 فإن البرنامج يتجه للخطوة رقم 20 وهكذا.

H.W: اكتب برنامج بلغة (fortran) لحساب وطبع قيمة (z) باستخدام عبارة **(أقصد المشروطة):**

$z = \sqrt{x^2 + y^2}$ IF j = 1
$z = 2x^2 + y^2$ IF j = 2
$z = 6x - y$ IF j = 3

العوامل العلاقية Relational Operators:

عوامل مفاضلة تستخدم لتبيان العلاقة بين كميتين. والجدول أدناه يبين العوامل العلاقية المستعملة في لغة فورتران:

>	أكبر من	.GT.
≥	أكبر من أو يساوي	.GE.
<	أصغر من	.LT.
≤	أصغر من أو يساوي	.LE.
=	يساوي	.EQ.
≠	لا يساوي	.NE.

النقطتان المحيطتان بكل رمز جزء من الرمز.

أمثلة:

$x>y \quad \rightarrow \quad x.GT.y$

$A+B \leq C^2 \quad \rightarrow \quad A+B.LE.C**2$

وتستخدم أيضاً للمتغيرات الحرفية، مثال:

Name.EQ.'Ali'

العوامل المنطقية Logical Operators:

العوامل المنطقية المستعملة في لغة فورتران:

الرمز	المعنى
.NOT.	لا
.AND.	و
.OR.	أو

عمل .NOT. : تغيير حالة المتغير المنطقي إلى الحالة المعاكسة، ويستخدم لمتغير واحد فقط.

A	.NOT.A
F	T
T	F

F: خطأ (false).

T: صح (True).

عمل AND: يكون صحيح فقط في حالة كون المتغيرين يحويان القيمة (True).

A	B	A.AND.B
F	F	F
F	T	F
T	F	F
T	T	T

عمل OR: يكون زائف او خاطئ في حالة كون المتغيران زائفان.

A	B	A.OR.B
F	F	F
F	T	T
T	F	T
T	T	T

أمثلة:

A.GT.B.AND.A.LT.G
True.AND.True=True

٣ - عبارة (إذا) المنطقية Logical IF Statement:

```
IF (e) S1
S2
```

إذ إن: (e) متغير علاقي أو منطقي.

(S1,S2): تمثلان جملتان تنفيذيتان داخل البرنامج.

فإذا كان الناتج (e) صح (true) نفذت الحاسبة الجملة التنفيذية S1 أما إذا كان ناتج (e) خطأ (false) فلا تنفذ الجملة التنفيذية S1، وبعد ذلك يتجه التنفيذ إلى العبارة S2.

مثال: اكتب برنامج بلغة (fortran) لحساب قيمة (y) باستخدام إذا المنطقية:

$$y=\begin{cases} x^2 & x\geq 0 \\ x+2 & x<0 \end{cases}$$

الحل:

```
Print *, 'Input the Value of x'
Read *, x
IF (x.GE. 0) y=x**2
IF (x.LT.0) y=x+2
Print *, 'y=',y
Stop
End
```

ويمكن حل المثال السابق باستخدام عبارة **Goto**.

```
   Print *, 'Input the Value of x'
   Read *, x
   IF (x.GE. 0) Goto 10
   IF (x.LT.0) Goto 20
10 y=x**2
   Goto 30
20 y=x+2
30 Print *, 'y=',y
   Stop
   End
```

مثال: اكتب برنامج بلغة (fortran) لحساب مجموع الأعداد من (1) إلى (100).

```
   Integer Sum
   N=0
   Sum=0
10 N=N+1
   Sum =Sum+N
   If (N.LT. 100) Go to 10
   Print *, 'Sum=',Sum
   Stop
   End
```

H.W: اكتب برنامج بلغة (fortran) لحساب مجموع الأعداد الزوجية من (1) إلى (100).

مثال: اكتب برنامج يطبع الأعداد من (1−10).

```
   N=0
10 N=N+1
   Print *, N
   If (N.LT.10) Go to 10
   Stop
   End
```

أما إذا أردنا الأعداد الزوجية فيكون العداد كالآتي:

```
   N=N+2
```

٤ - عبارة (إذا-إذن) (If-Then Statement):

```
If (e) Then
.
.
.
End If
```

إذا تحقق الشرط وكان ناتج (e) صحيحا (true) فان الجمل المحصورة بين Then و End If ستنفذ اما إذا لم يتحقق الشرط وكان ناتج (e) زائفا (false) فان الحاسبة ستترك هذا المقطع بجميع جمله ويتجه التنفيذ إلى أول جملة تنفيذية تأتي بعد جملة End If مباشرة.

هذا الأسلوب يتيح تنفيذ عدد من الجمل التنفيذية عند تحقق الشرط على خلاف If المنطقية والتي تتحقق جملة واحدة فقط.

مثال: اكتب برنامج بلغة (fortran) لإيجاد الأعداد الموجبة لعشرة أعداد صحيحة (x)، ثم يطبعها ويحسب عددها.

```
   Integer *2 x
   Integer *1 N,M
   M=0
   N=0
10 M=M+1
   Read *, x
   If (x.GT. 0) Then
   Print *, x
   N=N+1
   End If
   If (M.LT.10) Go to 10
   Print *, 'number of positive numbers=',N
   Stop
   End
```

مقطع 1

مقطع 2

٥ - مقطع (إذا-إذن-وإلا) If-Then-Else Block:

```
   IF (e) Then
   ……
   ……
   Else
   ……
   ……
   End If
```

الجزء الأول من المقطع ينفذ عندما يكون الشرط صحيحاً

الجزء الثاني من المقطع ينفذ عندما يكون الشرط خطأ

يتكون مقطع (إذا–إذن–وإلا) من جزأين، ينفذ الجزء الأول الذي يأتي بعد then عند تحقق الشرط في جملة If، وينفذ الجزء الثاني الذي يأتي بعد Else عند عدم تحقق الشرط.

مثال: اكتب برنامج بلغة (fortran) لقراءة قيمتي I و J ثم يطبع أيهما أكبر.

```
read *, I,J
If (I.GT.J) then
print *, 'I is the greater than J'
else
print *, 'J is the greater than I'
end If
stop
end
```

٦ جمل (إذا) المتداخلة:

يمكن استخدام أكثر من جملة مقارنة واحدة ضمن المقطع الواحد ويتم ذلك باستخدام جملة (وإلا –إذا) (Else If) وعلى النحو التالي:

```
If (e1) Then
.....
.....        } ينفذ هذا المقطع فقط عندما تكون e1 صحيحة
.....
Else If (e2) Then
.....
.....        } ينفذ هذا المقطع فقط عندما تكون e2 صحيحة
.....
Else If (e3) Then
.....
.....        } ينفذ هذا المقطع فقط عندما تكون e3 صحيحة
.....
End If
```

يجوز استعمال أي عدد من Else If، وعند عدم تحقق أي من هذه الشروط فإن التنفيذ سيتجه إلى أول جملة تنفيذية بعد End If.

مثال: إذا كانت (r) تمثل نصف قطر دائرة، اكتب برنامج بلغة (fortran) لقراءة (r) ثم لقراءة احداثيي نقطة ما تتمثل في (x , y) وبيان فيما إذا كانت هذه النقطة تقع داخل الدائرة أم على محيطها، أم خارجها، علماً أن الصفر يمثل مركز الدائرة.

```
Print*,'input value of r,x,y'
Read*, r,x,y
s=sqrt((x-0)**2+(y-0)**2)
If (r.gt.s) then
Print*,'the point inside the circle'
Else if (r.lt.s )then
Print*,'the point outside the circle'
Else if (r.eq.s) then
Print*,'the point on the circumference'
End if
Stop
End
```

H.W: أقرأ المعدل (M) المحصور بين (الصفر) و (99)، ثم اكتب برنامج بلغة (fortran) باستخدام جملة (**إذا المتداخلة**) لطبع الناتج على النحو التالي:

الدرجة	صفر – 49	59 – 50	69 – 60	79 – 70	89 – 80	99 – 90
التقدير	F	P	M	G	V.G	E

٧ **جملة (إذا) الحسابية (Arithmetic If Statement):**

If (e) n1,n2,n3

e: تعبير حسابي فقط، وهي بذلك تختلف عن باقي جمل If.

n1: رقم جملة يتجه إليه التنفيذ عندما يكون التعبير سالباً.

n2: رقم جملة يتجه إليه التنفيذ عندما يكون التعبير صفراً.

n3: رقم جملة يتجه إليه التنفيذ عندما يكون التعبير موجباً.

يكره استعمال جملة إذا الحسابية لأنها تضعف من انسيابية البرنامج وبنيته، لذا يفضل التقليل من استعمالها واستعمال الأنواع الأخرى من جمل المقارنة.

مثال: If (x-y) $10^{-},20^{0},30^{+}$

If (J) $5^{-},5^{0},7^{+}$

مثال: اكتب برنامج بلغة (fortran) لحساب قيمة (y) باستخدام (**إذا الحسابية**) من العلاقة التالية:

$$y=\begin{cases} x^2 & x\geq 0 \\ x+2 & x<0 \end{cases}$$

```
      Print *, 'Input the Value of x'
      Read *, x
      IF (x) 10, 20, 20
10    y=x+2
      Goto 30
20    y=x**2
30    Print *, 'y=',y
      Stop
      End
```

الفصل الرابع

التكرار

هو تكرار مجموعة من الجمل في البرنامج للحصول على نتيجة معينة، ويحتاج المبرمج إلى جملة المعداد وجملة المقارنة، وكالآتي:

مثال: اكتب برنامج بلغة (fortran) لإيجاد مجموع الأعداد الصحيحة من (1) إلى (100) باستخدام جملة معداد وجملة مقارنة:

```
      Integer sum
      N=0
      sum=0
10    N=N+1
      sum=sum+N
      if (N.LT.100) goto 10
      print *, sum
      stop
      end
```

إلا أنه من الممكن استخدام جملة تنفيذية مختصة بتكرار الجمل، وهي جملة (نفذ) (Do Statement).

جملة (نفذ) الانفرادية (Do Statement):

تأخذ جملة Do Statement الشكل الآتي:

Do n I=e1,e2,e3

إذ أن:

n: رقم جملة تحمله آخر جملة في دارة Do، ويكون صحيحاً.

I: متغير صحيح أو حقيقي ويمثل دليل الدارة.

e1: تعبير صحيح أو حقيقي أو ذو دقة مضاعفة، وناتج هذا التعبير يبدأ به دليل الدارة I.

e2: تعبير صحيح أو حقيقي أو ذو دقة مضاعفة، وناتج هذا التعبير يمثل الحد الأعلى للدارة.

e3: تعبير صحيح أو حقيقي أو ذو دقة مضاعفة، وناتج هذا التعبير يمثل مقدار الزيادة التي تزداد بها قيمة الدليل عند انتهاء كل جولة من جولات الدارة. عند اختفاء e3 تعد الزيادة الضمنية بمقدار 1.

والأمثلة الآتية توضح كيفية كتابة جملة Do:

```
Do 10 I=1,20,2
```

يبدأ الدليل بالقيمة 1 ثم 3 ،5 ،7 ..... إلى القيمة 19 ويتوقف لان القيمة التالية 21 وهي اكبر من الحد الأعلى e2.

وقد تكتب الدارة بالشكل الآتي:

```
Do 10 K=I+1,J
```

لاحظ القيمة الابتدائية عبارة عن تعبير حسابي.

كما يمكن أن تكتب بالشكل الآتي:

```
Do 100 C=10.0,2.0,-2.0
```

مثال: اكتب برنامج بلغة (fortran) لإيجاد مجموع الأعداد الصحيحة من (1) إلى (100) باستخدام جملة نفذ:

```
      Integer sum
      sum=0
      Do 10 N=1,100,1
10    sum=sum+N
      print *, sum
      stop
      end
```

لاحظ أن جملة Do حلت محل المعداد وجملة المقارنة، أما رقم 10 فهي آخر جملة تنفيذية في الدارة.

كيف يتم تنفيذ الدارة:

عند تنفيذ جملة Do تقوم الحاسبة بالإجراءات الآتية:

١. حساب ناتج التعبير e1 ليصبح القيمة m1، ثم حساب ناتج التعبير e2 ليصبح القيمة m2، فحساب ناتج التعبير e3 (مقدار الزيادة) ليصبح القيمة m3.

٢. تقوم الحاسبة باجراء التحويرات المناسبة لتوحيد النوعية التي تتطابق مع نوعية دليل الدارة، مثل:

```
Do 10 k=A,B,J
```

هنا تقوم الحاسبة بتحوير قيمتي A وB إلى النوع الصحيح.

٣. تحتسب القيمة 1 كمقدار للزيادة عند غياب m3.

٤. يتم حساب عدد جولات الدارة كالآتي:

```
MAX((INT(m2-m1+m3)/m3),0)
```

الدالة INT تستخلص القيمة الصحيحة من الناتج، أما الدالة MAX فتأخذ أكبر قيمة من القيمتين: الناتج الصحيح والصفر، وتكون هذه القيمة هي عدد الجولات، مثلاً:

```
Do 10 I= 1,50
m1=1, m2=50, m3=1
INT((50-1+1)/1)=INT(50)=50
MAX(50,0)=50
```

لذا يكون عدد جولات الدارة يساوي 50.

٥. إذا كان عدد الجولات يساوي صفراً لا تنفذ الحاسبة الدارة، ويتجه التنفيذ إلى أول جملة تلي نهاية الدارة. أما إذا كان عدد الجولات لا يساوي صفراً، فإن دليل الدارة يأخذ قيمة m1 ، وتنفذ محتويات الدارة، ثم ينقص عدد جولات الدارة بواحد، ويعاد تنفيذ عدد الجولات إلى الصفر، عندئذ يتجه التنفيذ خارج حدود الدارة.

جملة استمر (continue statement)

جملة لا دور لها وليس لها تأثير على سير البرنامج، مثل:

```
    Do 10 I=1,10              Do 10 I=1,10
    Sum=sum+x*I         10    Sum=sum+x*I
10  continue
```

إلا أننا نضطر إلى استخدامها عند الحاجة إليها عند استخدام جمل التحكم مثل جمل IF و GOTO و STOP وغيرها، كما في المثال الآتي:

```
    Do 10 I=1,80
    Read *, k
    IF (k.eq.I) goto 15
10  continue
15  ……
```

مثال: اكتب برنامج بلغة (fortran) لإيجاد العلاقة التالية:

$$S = \sum_{i=1}^{10} x^i$$

```
    Sum=0.0
    Read*, x
    Do 10 I=1,10
10  Sum=Sum+x**I
    Print *, 'Sum=', Sum
    Stop
    End
```

مثال: أكتب برنامج بلغة (fortran) لحساب وطباعة مربع الأرقام الصحيحة من 1 إلى 100.

```
      Do 10 I= 1,100
      J= I*I
10    print *, J
      Stop
      End
```

شروط الدارة:

١. لا يجوز أن يكون رقم جملة نهاية الدارة رقماً لجملة تحكم مثل جمل IF وGOTO وSTOP وDO. أي لا يجوز أن تنتهي الدارة بإحدى جمل التحكم المعروفة.

٢. إذا كانت معاملات الدارة e1 وe2 وe3 تعابير حسابية تحوي أسماء لمتغيرات فيجب أن تعرف قيم هذه المتغيرات قبل الدخول إلى الدارة.

٣. عند وجود مقطع IF داخل دارة DO فيجب أن يكون كامل المقطع محتوى في دارة DO، مثلا:

```
   DO 10 k=1,8
   IF (SQRT(x).gt.12.0) then
   ….
   ….
   Else
   …
   …
   ENDIF
10 continue
```

٤. عند وجود دارة DO داخل مقطع من مقاطع IF فيجب أن تكون الدارة بأكملها محتواة في مقطع IF، مثل:

```
      IF (A.lt.B) then
      Do 100 I=1,10
      …..
      …..
      …..
100    Continue
      Else
      Do 200 I= 1,20
      …..
```

```
      …..
      …..
200   Continue
      ENDIF
```

٥. يجوز الخروج من الدارة قبل إكمال جولاتها، باستعمال إحدى جمل التحكم، مثل:

```
      Do 10 I=1,10
      Read *, y
      X=float(I)
      IF(sqrt(y).gt.sqrt(x)) goto 20
      Print *, y
10    Continue
20    Stop
      End
```

٦. لا يجوز الدخول إلى الدارة من الخارج من غير المرور على بوابة الدارة (جملة DO)، وبالمقابل يجوز أن يكون لجملة DO رقم يتجه إليه التنفيذ من خلال جملة تحكم.

٧. يجوز استعمال دليل الدارة داخل الدارة، ولكن لا يجوز تغيير قيمته داخل الدارة، مثل:

```
      Integer S
      S=0
      N=10
      Do 10 I=1,N
      S=S+I
10    Continue
```

٨. يجوز استعمال الدليل نفسه في دارات متعاقبة، مثل:

```
      Do 150 k=1,10,2
      …..
      …..
150   Continue
      …..
      Do 200 k=200,400
      …..
200   Continue
```

الدارات المتداخلة:

قد تتداخل الدارات، وعندئذ يجب أن تكون الدارة أو الدارات الداخلية محتواة في الدارة الخارجية تماماً، وكمثال على ذلك:

```
      Do 10 I=1,2
        Do 20 J=1,3
      Print *, I,J
20    Continue
10    Continue
      Stop
      End
```

الدارة الداخلية

الدارة الخارجية

عند التنفيذ يتم انجاز الدارة الداخلية I ثم الدارة الخارجية، وما سيطبعه البرنامج سيكون:

I	J
1	1
1	2
1	3
2	1
2	2
2	3

- يشترط عند تعدد الدارات عدم تقاطعها، مثل:

```
      Do 10 I=1,10
        Do 20 J=5,7
10    Continue
20    Continue
```

- يجوز أن تنتهي الدارات المتداخلة في جملة واحدة.

```
      Do 10 I=1,5
      Do 10 J=2,7
      Do 10 k=1,3
      …..
      …..
      …..
10    Continue
```

- يفضل استخدام جملة نهاية منفصلة لكل دارة.

- لا يجوز الدخول إلى الدارات المتداخلة من الخارج، أو من دارة خارجية إلى أخرى داخلية.
- لا يجوز استعمال اسم مشترك للدليل بين الدارات المتداخلة، فلا يجوز:

```
      Do 10 I=1,3
      Do 20 I=1,10,2
      …..
      …..
20    Continue
10    Continue
```

- لا يجوز أن يكون اسم الدليل لدارة داخلية مشابهاً لأحد معاملات الدارة الخارجية، مثل:

```
      Do 10 I=N1,N2
        Do 20 N1=1,3
      …..
      …..
20    Continue
10    Continue
```

مثال: اكتب برنامج بلغة (fortran) لحساب قيمة (y) من العلاقة الآتية:

$$y = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots\cdots + \frac{x^n}{n!}$$

```
      Integer F
      Print *, 'Input the values of x,n'
      Read *, x,n
      y=1.0
      Do 10 I=1,n
      F=1
      Do 20 J=1,I
20     F=F*J
10    y = y + x**I/F
      Print *, 'y =', y
      Stop
      End
```

مثال: اكتب برنامج بلغة (fortran) لحساب وطبع قيمة (s) من العلاقة الآتية:

$$s = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots\cdots\cdots \frac{x^n}{n!}$$

```
      Print*,'input the value of x,n'
      Read*,x,n
      S=1
      P=1
      Do 10 I=2,n,2
      F=1
      Do 20 J=1,I
20    F=F*J
      P=P*(-1)
10    S=S+(P*x**I)/F
      Print*,'S=',S
      Stop
      End
```

مثال: يمكن حساب كمية الحرارة المتولدة في موصل كهربائي باستعمال المعادلة التالية:

$$H = \frac{I^2 RT}{42}$$

حيث ان:

H: كمية الحرارة المتولدة، R: مقاومة الموصل مقاسة بالاوم، T: الزمن المستغرق لإمرار تيار مقاسا بالثواني، I: التيار المار بالموصل مقاسا بالأمبير.

اكتب برنامج بلغة (fortran) لحساب قيمة (H) لقيم مختلفة من التيار من (1) إلى (10) أمبير وبزيادة نصف أمبير في كل مرة، افرض ان (T) تساوي (1) ثانية و (R) تبدأ بقيمة (10) اوم ومن ثم تزداد قيمتها (1) اوم لكل زيادة نصف أمبير في التيار.

```
      Real I
      T=1
      R=10
      Do 10 I=1,10,0.5
      H=(I**2*R*T)/42
      Print*,H
      R=R+1
10    Continue
      Stop
      End
```

H.W:

1. اكتب برنامج بلغة (fortran) لقراءة (30) عدد وحساب عدد القيم الموجبة (بضمنها الصفر) وعدد القيم السالبة في هذه المجموعة من الأعداد.

2. اكتب برنامج بلغة (fortran) لحساب مجموع الأعداد الفردية الصحيحة من (1) إلى (300)، ثم لحساب مجموع الأعداد الزوجية الصحيحة من (1) إلى (300) أيضاً، ثم جد المجموع الكلي للأعداد الصحيحة من (1) إلى (300).

3. اكتب برنامج بلغة (fortran) لحساب وطبع قيمة (**y**):

$$y = \sum_{x=1}^{8} \frac{1}{\sqrt{(x^2+1)}} + \frac{1}{x+2}$$

الفصل الخامس

جمل الإدخال والإخراج

كنا في السابق نقرأ المتغيرات ونطبعها بشكل مباشر دون تحديد حجم وعدد مواقع هذه المتغيرات والفراغات التي بينها، كذلك بالنسبة لصيغة الطباعة حيث كانت الحاسبة تتكفل بذلك، إذ كانت الإشارة * تعني الطريقة المباشرة للإدخال والإخراج من دون التقيد بصيغة معينة.

```
Read *, A,B,C
print *, A,B,C
```

لذلك نحتاج في كثير من الأحيان إلى تنظيم النتائج وفق تصور معين تتحدد فيه مواقع النتائج وحجمها إضافة إلى الفراغات التي تفصل بينها، وهذا ما تحققه جملة الصيغة.

جمل الصيغة في الطباعة Format statements

ويتحدد عملها في تعريف الوصف اللازم لكل متغير يراد قراءته أو طباعته ولها الوصف الآتي:

```
      Print 10, A, B,C
10    format (…….)
```

حيث يشير الرقم الصحيح بعد جملة الطباعة إلى رقم جملة الصياغة، أما ما بين القوسين فهي طريقة الوصف.

ملاحظة: جملة الصيغة غير تنفيذية فيجوز أن تظهر في بداية البرنامج وآخره، ويجوز أيضاً أن تشترك عدة جمل إدخال وإخراج بجملة صيغة واحدة شرط أن تتوافق المتغيرات المقروءة أو المطبوعة.

صيغة المتغير الصحيح (I):

يستعمل الرمز I للدلالة على ان الحجز هو لمتغير صحيح وتظهر بالشكل الآتي:

rIw.m

إذ تمثل:

r: مرات التكرار.

I: دليل الحجز لمتغير صحيح.

w: عدد المواقع المحجوزة لارقام العدد.

m: أقل عدد من أرقام قيمة المتغير الصحيح المطلوب طباعتها.

```
      A=10, B=20, C=30
      Print 10, A,B,C
10    format (I2,I2,I2)
```

1	0	2	0	3	0

```
     N1=15
     N2=30
     N3=125
    Print 8,N1,N2,N3
8   format (I3,I3,I4)
```

ستظهر القيم مطبوعة ابتدأ من الموقع الاول للسطر وعلى النحو الاتي

	1	5		3	0		1	2	5

بما ان الحجز متساوي لكل من N1,N2 لذا يمكن كتابة جملة الصيغة بالشكل التالي مستفيدين من دليل التكرار من دون ان يتغير شكل الناتج

```
8  format (2I3,I4)
```

كما يمكن ان نوحد احجام المتغيرات الثلاثة بالشكل التالي:

```
8  format (3I4)
```

سيظهر الناتج كما ياتي:

		1	5			3	0		1	2	5

صيغة المتغير الحقيقي:

1- صيغة الحجز (F):

ويستخدم للمتغير الحقيقي الاعتيادي

```
rFw.d
```

r: عدد مرات التكرار.

F: دليل الحجز للمتغير الحقيقي اعتيادي بدون أس.

W: عدد المواقع للمتغير الحقيقي الاعتيادي إضافة إلى الإشارة والفارزة والكسر.

D: عدد مواقع الكسر يمين الفارزة.

```
      A=5.3, B=-20.651, C=150.12
      Print 10, A,B,C
10    format (F4.1,F8.3,F7.2)
```

	5	.	3		-	2	0	.	6	5	1		1	5	0	.	1	2

دائماً يوضع فراغ يمين العدد للفصل بين عدد وآخر، لذلك يزداد عدد المواقع w متضمنة هذا الفراغ.

وباستعمال صيغة التكرار على سبيل المثال:

```
10   format (3F8.3)
```

			5	.	3	0	0		-	2	0	.	6	5	1		1	5	0	.	1	2	0

2- صيغة الحجز (E)

للمتغيرات الحقيقية ذات الأس

rEw.dEe

r: عدد صحيح يمثل دليل التكرار.

E: دليل الحجز لمتغير حقيقي ذي أس.

w: عدد صحيح يمثل حجم الحجز –عدد المواقع–، ويمثل مجموع عدد المواقع التي يحتاجها العدد مع ملحقاته.

d: عدد صحيح يمثل عدد مواقع الكسر –يمين الفارزة–.

e: عدد المواقع المحجوزة للأس، وعند تركها فإن الحاسبة ستحجز للأس ثلاثة مواقع، موقع لإشارة الأس، وموقعين للأس.

```
x=0.0032=0.32E-02
y=1.12E-10=0.112E-09
z=3.2567E13=0.32567E+14
Print 10, x,y,z
10   format (E9.2,E10.3,E12.5)
```

كيفية الحجز

	0	.	3	2	E	-	0	2		0	.	1	1	2	E	-	0	9		0	.	3	2	5	6	7	E	+	1	4

صيغة المتغير الحقيقي ذو الدقة المضاعفة (D)

تتعامل مع المتغيرات الحقيقية ذات الدقة المضاعفة وهي مثل E، وتزداد دقة الكسر لكي يصل إلى 15 مرتبة كسرية، وتعرف المتغيرات ذات الدقة المضاعفة باستعمال الجملة النوعية real 8* أو الجملة النوعية double precision، وصيغتها:

rDw.d

r: عدد صحيح يمثل عدد مرات التكرار.

D: دليل الحجز لمتغير حقيقي ذي أس.

w: عدد صحيح يمثل حجم الحجز –عدد المواقع–، ويمثل مجموع عدد المواقع التي يحتاجها العدد مع ملحقاته.

d: عدد صحيح يمثل عدد مواقع الكسر –يمين الفارزة–.

مثال: 0.123456789×10^{-3} D16.9

	0	.	1	2	3	4	5	6	7	8	9	D	-	0	3

صيغة الحجز للمتغيرات المركبة

وتستخدم صيغ الحجز I وF وE وD على أن تحجز موقعين لكل متغير مركب، مثلاً:

```
Complex z1,z2
z1=cmplx0,1D2,-0.2D3
z2=cmplx 0.5,-0.1
Print 100, z1,z2
100  format (2D10.2,2F6.1)
```

صيغة الحجز للمتغير الحرفي (A)

وتستخدم للمتغير الحرفي، وصيغتها:

rAw

r: عدد صحيح يمثل عدد مرات التكرار.

A: دليل الحجز لمتغير حرفي.

w: عدد صحيح يمثل عدد المواقع المحجوزة للمتغير الحرفي.

مثال:

```
Name='Ahmed'
Unv='Tikrit'
Print 10, Name,Unv
10   format (A6, A7)
```

	A	h	m	e	d		T	i	k	r	i	T

الصيغة (x):

وتستخدم لترك الفراغات بين قيمة وأخرى وصيغتها

nx

```
A=10
B=1.5
Print 10, A,B
10  format (3x,I2,2x,F3.1)
```

			1	0			1	.	5

الصيغة (T):

هذه الصيغة تحدد الموقع الذي يبدأ فيه طباعة الناتج وتكتب بالشكل الآتي:

Tc

T: حرف الصيغة.

c: ثابت صحيح يمثل الموقع الذي يبدأ منه طبع القيمة التالية.

للمثال أعلاه تكتب جملة الصياغة كالآتي:

```
10 format (T4,I2,T3,F3.1)
```

الصيغة (/) slash:

وتستخدم لنقل النتائج من سطر إلى آخر

/: الانتقال إلى السطر التالي.

//: الانتقال إلى السطر الثالث، وترك الثاني فارغاً.

مثلا ليكن:

```
N1=120
N2=1.6
N3='Ahmed'
Print 10, N1,N2,N3
10  format (1x,I3/1x,F3.1//1x,A)
```

النتائج:

السطر الأول 120 □

السطر الثاني 1.6 □

السطر الثالث فراغ

السطر الرابع Ahmed □

الفصل السادس

المتغيرات الموسومة

في السابق كان المتغير يحتفظ بقيمة واحدة أثناء تنفيذ البرنامج، وقد تتغير داخل البرنامج وتحل القيمة الجديدة محمل القيمة القديمة، إلا أن هذا غير كافي في بعض الأحيان عندما يراد منا على سبيل المثال ترتيب رقمين تصاعدياً فإننا نحتاج جملة مقارنة، ولكن ماذا عن ترتيب آلاف الأرقام تصاعدياً، هنا تظهر الحاجة إلى المتغيرات الموسومة.

المتغير الموسوم هو المتغير الذي يمثل مجموعة من القيم المتجانسة من حيث النوع. فمثلاً في حالة كون اسم المتغير الموسوم صحيحاً فإن القيم تكون صحيحة، وإذا كان حقيقياً يجب أن تكون القيم حقيقية، وهكذا بالنسبة لبقية الأنواع.

يكتب المتغير الموسوم في الحاسبة بهذه الصيغة:

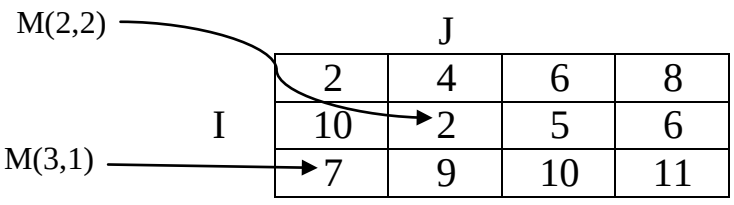
المتغير الموسوم x	قيمته
x(1)	10.1
x(2)	-5.0
x(3)	1.5

حيث أن:

x: اسم المتغير الموسوم.

(1) و(2) و(3): وسم المتغير أو تسلسله. ويمكن أن يكون الوسم رقم أو متغير أو تعبير أو متغير موسوم.

إن المتغيرات الموسومة ذات الوسم الواحد كما في المثال أعلاه هي منظومات أو مصفوفة ذات بعد واحد (one dimensional matrices) وهناك متغيرات موسومة ذات بعدين (Two dimensional matrices) وتظهر المصفوفة ذات البعدين M(I,J) بالشكل الآتي:



فيتم تحديد أي عدد بإعطاء الصف أولاً ثم العمود ثانياً.

أما المصفوفة ذات الثلاثة أبعاد (three dimensional matrices) M(I,J,K) فتمثل كالآتي:

I: رقم الصف.

J: رقم العمود.

K: رقم المستوي.

بفرض أننا نريد M(3,1,1)، القيمة هنا في الصف الثالث العمود الأول المستوي الأول

جملة البعد (Dimension statement)

تقوم جملة البعد بتعريف كل المتغيرات الموسومة وتحديد نوعها وعدد عناصرها، وتكتب بالأشكال التالية:

Dimension A(1:10)

حيث ان A مصفوفة أحادية لها عشرة عناصر تبتدئ بالتسلسل 1 وتنتهي بالتسلسل 10، ويجوز أيضا أن تكتب جملة البعد بالشكل الآتي:

Dimension A(10)

على افتراض إن التسلسل يبدأ بواحد

وهذه المصفوفة حقيقية لان الاسم حقيقي لذا فان كل عناصر المصفوفة A يجب ان تكون حقيقية

Dimension M(1:5,1:7) ذات بعدين

حيث ان: M مصفوفة ذات بعدين تتكون من خمسة صفوف وسبعة أعمدة أي تتكون من 35 عنصر موزعة على خمسة صفوف وسبعة أعمدة وهي مصفوفة صحيحة لان الاسم M اسم صحيح ويمكن كتابتها أيضا بالشكل الأتي:

Dimension M(5,7)

وتكتب عادة في بداية البرنامج، ويجوز أن يكتب أكثر من متغير موسوم في جملة بعد واحدة:

Dimension A(10), M(100,10), x(3,6,2)

استخدام جمل نوعية مثل Real و Integer و Complex لتعريف المتغيرات الموسومة وتحديد الحجز لها مثل:

Integer Z(10)
Real N(5)

إذ أنها تقوم مقام جملة البعد.

كما يمكن استخدام جملة البعد مع الجمل النوعية السابقة، مثل:

```
Dimension A(10)          Real *4 A(10)
Real *4 A        or      Real *4 A(1:10)
```

قراءة المصفوفة أحادية البعد:

يتم قراءة قيم عناصر مصفوفة أحادية البعد بأحد الأساليب التالية:

١. بتحديد التسلسل:

```
Dimension A(5)
Read *, A(1) , A(2), A(3), A(4), A(5)
```

لكن هذه القراءة مملة حين تكون هناك أعداد كبيرة من الأرقام.

٢. باستخدام جملة Do:

```
      Dimension A(5)
      Do 10 I=1,5
      Read *, A(I)
10    continue
```

إن البيانات هنا تحجز خمسة أسطر في كل سطر قيمة واحدة.

٣. باستخدام جملة Do ضمنية:

```
Dimension A(5)
Read *, (A(I),I=1,5)
```

إن القيم المقروءة هي A(1), A(2), …..A(5)، وتدخل القيم سطر بعد سطر أو في سطر واحد تفصل بينها الفارزة.

ويحبذ استخدام جملة الصيغة مع Do الضمنية:

```
      Dimension A(5)
      Read 10, (A(I),I=1,5)
10    Format (5F4.1)
```

٤. جملة البيانات Data statement

جملة البيانات هي إحدى جمل الإدخال التي تستخدم لإعطاء قيم أولية للمتغيرات وتوضع في أي موقع في البرنامج ولكن يحبذ أن توضع في البداية، وتأخذ الشكل الآتي:

```
Data A/2.5/          ⇒ A=2.5
or
Data x,y,z /2,3,4/
```

ولمصفوفة أحادية البعد تكتب بالشكل التالي:

```
Dimension A(5)
Data A/2,-7,1,8,-2/

or
Data A/2*1.5,3*2.5/
```

ملاحظة: عند إعطاء قيم أكثر أو أقل من القيم المحجوزة في عبارة القراءة يعطي البرنامج رسالة تبين وجود خطأ.

والتنفيذ سيكون بالشكل الآتي:

2	1.5
	1.5
3	2.5
	2.5
	2.5

ولمصفوفة ثنائية البعد

```
Real K(3,2)
Data K /3*1.5, 3*6.5/
```

العمود الأول	العمود الثاني
1.5	6.5
1.5	6.5
1.5	6.5

ويجوز استخدام Do ضمنية مع جملة البيانات

```
Integer A(10)
Data (A(I),I=1,10)/10*0/
```

لاحظ أننا صفرنا قيم المنظومة A.

ولتصفير الأعداد الفردية:

```
Data (A(I),I=1,10,2)/5*0/
```

طباعة المصفوفات أحادية البعد:

تتم الطباعة من خلال عدة أساليب:

١. بتحديد تسلسل العنصر المطلوب طباعته:

```
Print *, A(2), A(4)
```

٢. من خلال جملة Do

```
      Dimension k(10)
      Data k/……/
      Do 10 I=1,10
      Print *, k(I)
10    continue
      Stop
      End
```

٣. من خلال اسم المنظومة

```
      Dimension N(5)
      Data N/……/
      Print *, N
      Stop
      End
```

تطبع العناصر على سطر واحد ثم تنزل البقية للسطور التالية في حالة عدم كفاية السطر الحالي.

أو باستخدام جملة صيغة

```
      Print 10, k(I)
10    format (2x,I3)      سطر بعد سطر
```

٤. باستخدام Do الضمنية

```
      Dimension k(10)
      Data k/……/
      Print 10, (k(I), I=1,10)
10    format (2x,I3)
      Stop
      end
```

يمكن استخدام جملة الصيغة مباشرة في جملة الطباعة كما موضح أعلاه.

مثال: المصفوفة A تحتوي على عشرة عناصر هي:

$$A = [2 \quad 7 \quad 8 \quad -4 \quad -6 \quad 10 \quad 19 \quad 3 \quad 1 \quad -9]$$

اكتب برنامج بلغة (fortran) لقراءة عناصر المصفوفة الأحادية وإيجاد مجموعها:

```
      Dimension A(10)
      Data A/2, 7, 8, -4, -6, 10, 19, 3, 1, -9/
      Sum=0.0
      Do 10 I=1,10
10    Print *, A(I)
      Do 20 I=1,10
20    Sum=Sum+A(I)
      Print *, Sum
      Stop
      End
```

مثال: المصفوفة N تتكون من عشرة عناصر هي:

$$N = [1 \quad 2 \quad -3 \quad 6 \quad 4 \quad -2 \quad 8 \quad 15 \quad 10 \quad 3]$$

اكتب برنامج بلغة (fortran) لإيجاد أكبر عنصر

```
      Dimension N(10)
      Data N/1,2,-3,6,4,-2,8,15,10,3/
      Print *, (N(I), I=1,10)
      Nmax=N(1)
      Do 10 I=2,10
      If (Nmax.gt.N(I)) Goto 10
      Nmax=N(I)
10    continue
      Print *, 'Nmax=',Nmax
      Stop
      End
```

جملة المتغير الوسيط Parameter statement

وهي من الجمل الوصفية مثل جملة البعد، ولها فائدة كامنة في أنها تعرف قيمة ثابتة (غير قابلة للتغيير داخل البرنامج)، وتكتب كالآتي:

Parameter (symbol=constant expression)

إذ إن:

Symbol: اسم المتغير الوسيط.

Constant expression: تعبير يتكون من ثابت واحد أو مجموعة ثوابت تفصل العمليات الحسابية بينها.

فمثلاً: Parameter (N=40) يعني أن يأخذ المتغير الوسيط N القيمة 40 ويبقى محتفظاً بها دائماً خلال تنفيذ البرنامج.

Dimension Array (N) $\Rightarrow$ Dimension Array (40)

ويمكن تعريف أكثر من متغير وسيط واحد، مثال:

Parameter (K=2+3,M=70)

في المثال أعلاه تم تعريف أكثر من متغير وسيط واحد وهما M,K في جملة متغير وسيط واحد تفصل الفارزة بين متغير وآخر وقيمة K هي 5 ، M قيمتها هي 70

Parameter (A=3.5)
Parameter (B=A+1.5)

(قيمة B تساوي 5)

مثال: المتغيران الموسومان x و y يحويان 100 عنصر لكل منهما، اكتب برنامج بلغة (fortran) لحساب Z من العلاقة الآتية:

$$Z = \sum_{i=1}^{100} \frac{x(i) + y(i)}{i!}$$

```
      Parameter (N=100)
      Integer F
      Dimension x(N), y(N)
      Do 10 I= 1,N
      Read *, x(I),y(I)
10    continue
      Z=0.0
      Do 20 I= 1,N
      F=1
      Do 30 J= 1,I
30    F=F*J
      Z=Z+(x(I)+y(I))/F
20    continue
      Print *,'Z=',Z
      Stop
      End
```

مثال: اكتب برنامج بلغة (fortran) لقراءة عشرة أرقام في مصفوفة أحادية، ومن ثم قلب عناصرها بحيث يكون أول عنصر آخر عنصر والأخير يكون الأول.

```
      Parameter (N=10)
      Dimension A(N), B(N)
      Read *, (A(I),I=1,N)
      Do 10 I= 1,N
      A((N+1)-I)=B(I)
10    continue
      Print *, (A(I),I=1,N)
      Print *, (B(I),I=1,N)
      Stop
      End
```

H.W: اكتب برنامج بلغة (fortran) لترتيب عشرة أعداد في المصفوفة الأحادية A بصورة تصاعدية وأطبعها.

المصفوفات ذات البعدين

يتحدد العنصر في المنظومة ذات البعدين بتحديد صفه وعموده، وإن قراءة المصفوفة تتطلب تحديد تسلسلين على عكس المصفوفة ذات البعد الواحد التي تتطلب تسلسل واحد.

قراءة المصفوفة ذات البعدين

١. بتحديد تسلسل كل عنصر مقروء

```
      Dimension A(3,2)
      Integer A
      Read *, A(1,1) , A(1,2), A(2,1), A(2,2), A(3,1), A(3,2)
```

وكأنها ستة متغيرات مستقلة، وهنا القراءة صف صف أي صفية، أما إذا كانت القراءة عمودية فتكون كالآتي:

```
      Read *, A(1,1) , A(2,1), A(3,1), A(1,2), A(2,2), A(3,2)
```

مثال:

$$\begin{bmatrix} 3 & 4 \\ 10 & 6 \\ -2 & 5 \end{bmatrix}$$

ويكون إدخال البيانات كالآتي:

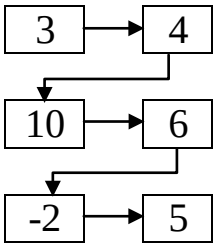
القراءة الصفية: 3,4,10,6,-2,5

القراءة العمودية: 3,10,-2,4,6,5

٢. من خلال جملة Do:

$$\begin{bmatrix} 3 & 4 \\ 10 & 6 \\ -2 & 5 \end{bmatrix}$$

القراءة الصفية:

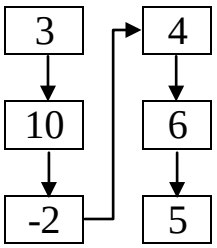



```
      Dimension A(3,2)
      Do 10 I=1,3
      Do 10 J=1,2
10    Read *, A(I,J)
```

القراءة العمودية:

```
      Dimension A(3,2)
      Do 10 J=1,2
      Do 10 I=1,3
10    Read *, A(I,J)
```

أما البيانات فتكون كالآتي:



٣. من خلال Do الضمنية

– القراءة الصفية

```
Dimension A(3,2)
Read 10, ((A(I,J),J=1,2),I=1,3)
10    Format (2I3)
```

يكون الإدخال كل قيمتين للسطر الواحد. الدارة الضمنية الخارجية الأولى للدليل I الخاص بالصفوف، والثانية للدليل J الدارة الضمنية الداخلية الخاص بالأعمدة، وستبدأ القراءة بقراءة الصف الأول ثم الثاني وهكذا.

– القراءة العمودية

ستكون القراءة هنا ثلاثة قيم لكل سطر

```
Dimension A(3,4)
Read 10, ((A(I,J),I=1,3),J=1,4)
10    Format (3I3)
```

إن جملة الصيغة (3I3) تدل على ثلاثة قيم في السطر الأول وهي تمثل العمود الأول وهكذا في السطر الثاني وتدل على العمود الثاني.

٤. باستخدام جملة البيانات Data

يمكن قراءة المصفوفة ذات البعدين بطريقتين، الصفية والعمودية، مثلاً المصفوفة A:

3	4
10	6
-2	5

– القراءة الصفية

```
Data A/3,4,10,6,-2,5/
```

– القراءة العمودية

```
Data A/3,10,-2,4,6,5/
```

طباعة المصفوفة ثنائية البعد:

يفضل طباعة المصفوفة صفياً لكي تظهر بالشكل المفهوم، وذلك لأن جهاز الطبع يطبع سطراً بعد آخر:

1- باستخدام اسم المنظومة:

```
Dimension A(3,2)
Data A/3,10,-2,4,6,5/
Print *, A
Stop
End
```

تكون الطباعة هنا حسب جملة DATA أي عمودية ولكن على سطر واحد في الطباعة، وإذا كانت جملة DATA بشكل صفي فإن الطباعة لهذه الحالة تكون مشابهة لها.

2- من خلال جملة DO الخارجية

```
Dimension A(3,2)
Data A/3,10,-2,4,6,5/
Do 10 I=1,3
Do 10 J=1,2
10   Print *, A (I,J)
Stop
End
```

الطباعة هنا ستكون عمودية، قيمة كل عنصر على سطر مستقل.

ملاحظة: لا يجوز استخدام جملة البيانات DATA بشكل صفي ومن ثم طباعتها مع الدارات وذلك لأن DATA تقرأ القيم عمود بعد عمود حسب الحجز للمصفوفة.

أما في حالة كتابة جملة صيغة للطباعة، وكالآتي:

```
Dimension M(3,2)
Data M/3,10,-2,4,6,5/          القراءة بشكل عمودي
Do 10 I=1,3
Do 10 J=1,2
10   Print 20, M (I,J)          الطباعة صفية
20   format(2I3)
Stop
End
```

3- من خلال دارتين ضمنيتين

```
      Integer A(3,2)
      Data A/3,10,-2,4,6,5/
      Print 10, ((A(I,J),J=1,2),I=1,3)        الطباعة صفية
10    format(2I4)
      Stop
      End
```

النتائج ستظهر بالشكل:

3 4

10 6

-2 5

مثال: اكتب برنامج بلغة (fortran) لقراءة المصفوفة N صفاً صفاً، ثم أطبع الناتج صفاً بعد صف:

$$N = \begin{bmatrix} 1 & 5 & 4 \\ 9 & 11 & 13 \end{bmatrix}$$

```
      Dimension N(2,3)
      read 10, ((N(I,J),J=1,3),I=1,2)
10    format(3I3)
      Print 20, ((N(I,J),J=1,3),I=1,2)
20    format(3I3)
      Stop
      End
```

مثال: اكتب برنامج بلغة (fortran) لقراءة مصفوفة (5×5) ومن ثم حساب مجموع عناصر المصفوفة.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

```
      Integer A(5,5)
      Data A/1,2,3,4,5,6,7,8,9,10,11,12,
     &13,14,15,16,17,18,19,20,21,22,23,24,25/
      print 10, ((A(I,J),J=1,5),I=1,5)   الطباعة صفية
10    format (5I5)
      sum=0.0
      Do 20 I=1,5
      Do 20 J=1,5
20    sum=sum+A(I,J)
      print *,'sum=',sum
      stop
      end
```

مثال: اكتب برنامج بلغة (fortran) لإيجاد مجموع عناصر القطر الرئيسي للمثال السابق.

القطر الرئيسي القطر الثانوي

1,1	1,2	1,3	1,4	1,5
2,1	2,2	2,3	2,4	2,5
3,1	3,2	3,3	3,4	3,5
4,1	4,2	4,3	4,4	4,5
5,1	5,2	5,3	5,4	5,5

```
      Integer A(5,5)
      Data A/1,2,3,4,5,6,7,8,9,10,11,12,
     &13,14,15,16,17,18,19,20,21,22,23,24,25/
      print 10, ((A(I,J),J=1,5),I=1,5)
10    format (5I5)
      sum=0.0
      Do 20 k=1,5
20    sum=sum+A(k,k)
      print *,'sum=',sum
      stop
      end
```

إذا أردنا جمع عناصر القطر الثانوي فالخطوة رقم 20 ستكون كالآتي:

```
20    sum=sum+A(k,6-k)
```

مثال: اكتب برنامج بلغة (fortran) لإيجاد مجموع العناصر الواقعة تحت القطر الرئيسي.

```
      Dimension N(5,5)
      Data N/1,2,3,4,5,6,7,8,9,10,11,12,
     &13,14,15,16,17,18,19,20,21,22,23,24,25/
      print 10, ((N(I,J),I=1,5),J=1,5)
10    format (5I5)
      sum=0.0
      Do 20 I=2,5
      Do 20 J=1,I-1
20    sum=sum+N(I,J)
      print *,'sum=',sum
      Stop
      End
```

H.W: اكتب برنامج لإيجاد مجموع العناصر الواقعة فوق القطر الرئيسي للمثال أعلاه.

مثال: اكتب برنامج بلغة (fortran) لقراءة الجزء الأعلى من مصفوفة A(4×4) صفاً صفاً، ثم كون الجزء الأسفل منها، علماً أن المصفوفة A متماثلة، أي أن A(I,J)=A(J,I)، ثم أطبع عناصر المصفوفة.

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

```
      Dimension A(4,4)
      read *, ((A(I,J),J=I,4),I=1,4)
      Do 10 I=1,4
      Do 10 J=1,4
10    A(J,I)=A(I,J)
      Print 20, ((A(I,J),J=1,4),I=1,4)
20    format(4(2x,F3.1))
      Stop
      End
```

مثال: اكتب برنامج بلغة (fortran) لقراءة أرقام مصفوفة A(4×4) وإيجاد مجموع عناصر الإطار الخارجي.

$$\begin{bmatrix} 3 & 5 & 2 & 6 \\ 8 & 7 & 1 & 9 \\ 2 & 4 & 6 & 8 \\ 1 & 5 & 7 & 2 \end{bmatrix} \qquad \begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a44 \end{bmatrix}$$

```
      Dimension A(4,4)
      read 10, ((A(I,J),J=1,4),I=1,4)
10    format (3x,4F8.3)
      sum=0.0
      Do 20 J=1,4
20    sum=sum+A(1,J)+A(4,J)
      Do 30 I=2,3
30    sum=sum+A(I,1)+A(I,4)
      print *,'sum=',sum
      stop
      end
```

مثال: اكتب برنامج بلغة (fortran) لقراءة المصفوفة الصحيحة A(4,4) ومن ثم تحويل عناصرها إلى مصفوفة صحيحة أحادية B.

```
      Dimension A(4,4),B(16)
      Integer A,B
      Data A/3,8,2,1,5,7,4,5,2,1,6,7,6,9,8,2/   قراءة عمودية
      k=0
      Do 10 I=1,4
      Do 10 J=1,4
      k=k+1
10    B(k)=A(I,J)
      print 20,((A(I,J),J=1,4),I=1,4)
20    format (3x,4I4)
      print 30,(B(k),k=1,16)
30    format (3x,16I4)
      Stop
      End
```

ستطبع المصفوفة B على سطر واحد.

يمكن استخدام دارة ضمنية مع جملة البيانات، وكالآتي:

Data ((A(I,J),J=1,4),I=1,4) /3,5,2,6,8,7,1,9,2,4,6,8,1,5,7,2/

وهنا تم تحويل القراءة العمودية لجملة البيانات إلى قراءة صفية.

مثال: اكتب برنامج بلغة (fortran) لقراءة مصفوفة أحادية B(16) وتحويل عناصرها إلى مصفوفة مربعة.

```
      Dimension A(4,4),B(16)
      Integer A,B
      Data B/3,8,2,1,5,7,4,5,2,1,6,7,6,9,8,2/
      k=0
      Do 10 I=1,4
      Do 10 J=1,4
      k=k+1
10    A(I,J)=B(k)
      print *,B
      print 30,((A(I,J),J=1,4),I=1,4)      طباعة صفية
30    format (3x,4I4)
      stop
      end
```

جمع وطرح المصفوفات:

تتم عملية الجمع بجمع العناصر المتقابلة في المصفوفتين ولابد من أن تكون المصفوفتين متماثلة.

$$A=\begin{bmatrix}2 & 1\\ 3 & 4\end{bmatrix} \quad ,B=\begin{bmatrix}1 & 5\\ 6 & 3\end{bmatrix}$$

$$A+B=\begin{bmatrix}2+1 & 1+5\\ 3+6 & 4+3\end{bmatrix}=\begin{bmatrix}3 & 6\\ 9 & 7\end{bmatrix}$$

يمكن إنجاز العملية السابقة بلغة فورتران، وكما موضح في المثال التالي:

مثال: اكتب برنامج لجمع المصفوفتين A و B وليكن A(2,2), B(2,2), C(2,2) حيث أن C ناتج الجمع.

```
      parameter (m=2,n=2)
      dimension A(m,n),B(m,n),C(m,n)
      data A/2,3,1,4/
      data B/1,6,5,3/
      Do 10 I= 1,m
      Do 10 J=1,n
10    C(I,J)=A(I,J)+B(I,J)
      print 20,((A(I,J),J=1,n),I=1,m)
20    format (3x,2f3.1)
      print 30,((B(I,J),J=1,n),I=1,m)
30    format (3x,2f3.1)
      print 40,((C(I,J),J=1,n),I=1,m)
40    format (3x,2f3.1)
      stop
      end
```

ويمكن القيام بعملية الطرح أيضاً باستبدال علامة الإضافة بالطرح.

H.W: المصفوفة A(4,4) والمصفوفة B(4,4) هما على النحو التالي:

اكتب برنامج بلغة (fortran) لقراءة المصفوفتين A,B ثم يقوم بحساب حاصل جمع المصفوفتين A,B.

ضرب المصفوفات:

كي تتم عملية ضرب مصفوفتين يجب أن يتماثل عدد أعمدة مصفوفة مع عدد صفوف المصفوفة الأخرى.

على سبيل المثال المصفوفتين A(3,2) و B(2,3).

$$A=\begin{bmatrix} a11 & a12 \\ a21 & a22 \\ a31 & a32 \end{bmatrix} \quad ,B=\begin{bmatrix} b11 & b12 & b13 \\ b21 & b22 & b23 \end{bmatrix}$$

نجد مجموع حاصل ضرب عناصر الصف الأول من المصفوفة A مع عناصر العمود الأول للمصفوفة B لنحصل بذلك على عنصر المصفوفة C الواقع في الصف الأول العمود الأول c_{11}، وكذلك بالمثل لعناصر الصف الأول لـA مع عناصر العمود الثاني لـB لنحصل على c_{12} وهكذا.

$c_{11}=a_{11}b_{11}+a_{12}b_{21}$
$c_{12}=a_{11}b_{12}+a_{12}b_{22}$
$c_{13}=a_{11}b_{13}+a_{12}b_{23}$
==========
$c_{21}=a_{21}b_{11}+a_{22}b_{21}$
$c_{22}=a_{21}b_{12}+a_{22}b_{22}$
$c_{23}=a_{21}b_{13}+a_{22}b_{23}$
==========
$c_{31}=a_{31}b_{11}+a_{32}b_{21}$
$c_{32}=a_{31}b_{12}+a_{32}b_{22}$
$c_{33}=a_{31}b_{13}+a_{32}b_{23}$

وبذلك نحصل على المصفوفة C

$$C=\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

بصورة عامة يمكن تمثيل ذلك كما يلي:

$$C(I,J)=\sum_{K=1} A(I,K)\times B(K,J)$$

مثال: اكتب برنامج لإيجاد حاصل ضرب المصفوفة الصحيحة A(3,2) والمصفوفة الصحيحة B(2,3) وليكن الناتج المصفوفة الصحيحة C.

```
      Integer A(3,2),B(2,3),C(3,3)
      data A/1,2,3,4,5,6/
      data B/7,8,9,10,11,12/
      Do 10 I= 1,3
      Do 10 J=1,3
      C(I,J)=0
      Do 10 K=1,2
10    C(I,J)=C(I,J)+A(I,K)*B(K,J)
      print 20,((C(I,J),J=1,3),I=1,3)
20    format (5x,3I4)
      stop
      end
```

H.W: المصفوفة A(3,3) والمصفوفة B(3,3) هما على النحو التالي:

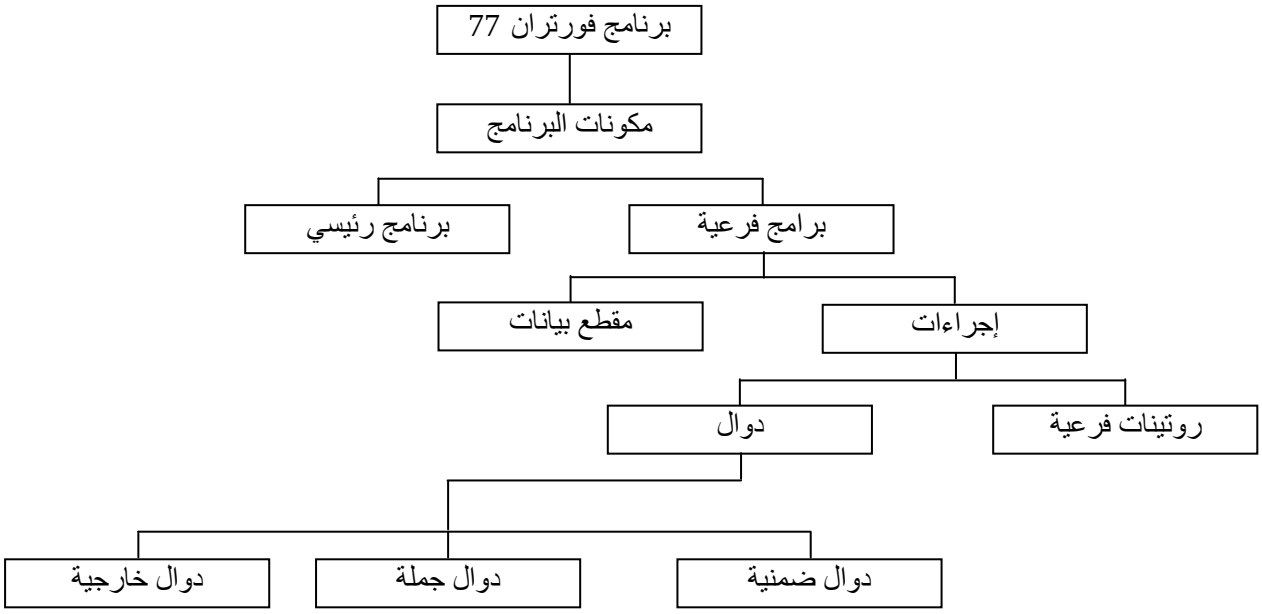
$$A = \begin{bmatrix} 7 & 5 & 8 \\ 9 & 4 & 7 \\ 5 & 8 & 9 \end{bmatrix}, \; B = \begin{bmatrix} 3 & 2 & 7 \\ 4 & 1 & 3 \\ 2 & 6 & 5 \end{bmatrix}$$

اكتب برنامج بلغة (fortran) لقراءة المصفوفتين A,B ثم احسب المصفوفة C حيث ان:

$$C = (A + B) * (A - B)$$

الفصل السابع : البرامج الفرعية

تعاملنا سابقا مع البرامج كوحدة واحدة يطلق عليها البرامج الرئيسية (main-programs) وقد تطرقنا لبعض البرامج الفرعية مثل الدوال المكتبية ((ABS(x),Sin(x)) والتي لا تنفذ وحدها وإنما من خلال البرنامج الرئيسي.



شكل (1) بنية البرنامج

- **البرنامج الرئيسي:** هو البرنامج الذي يمثل محور العملية البرمجية، ومن خلاله تستدعى بقية البرامج الفرعية. وينتهي بجملة Stop كما يمكن للبرنامج الرئيسي أن ويأخذ اسما معينا باستعمال جملة Program . ولا يجوز أن يستدعي البرنامج الرئيسي نفسه.
- **البرامج الفرعية:** تتكون من مقاطع البيانات Block data والإجراءات Procedures .
- **مقطع البيانات:** برنامج فرعي تنحصر وظيفته في إعطاء قيم أولية لمجموعة من المتغيرات الاعتيادية والمتغيرات الموسومة.
- **الأجراء:** برنامج فرعي يقوم بأداء مهمة معينة عند استدعائه من قبل البرنامج الرئيسي، والأجراء قد يكون دالة Function أو روتينا فرعيا Subroutine .
- **الدالة:** وتقسم إلى:-

١. دالة خارجية (External Function): وهي برنامج فرعي يكتب بمعزل عن البرنامج الرئيسي ويحوي جملا تنفيذية تقوم بحساب نتيجة واحدة يعيدها إلى البرنامج الرئيسي بعد استدعائه للدالة.
٢. دالة الجملة (Statement Function): جملة واحدة توضع داخل البرنامج الرئيسي للحصول على نتيجة واحدة وتسمى أيضا بالدالة الداخلية (internal Function).
٣. دالة ضمنية (Intrinsic Function): هي الدوال المكتبية المجهزة في المبرمج.

دالة الجملة: يطلق عليها أحيانا بالدالة الداخلية Internal Function وتظهر كجملة واحدة داخل البرنامج الرئيسي كجملة إحلال.

Name (a1,a2,……,an)=e

Name : اسم الدالة وهو متغير.

a1,a2 : اسم الدليل الوسيط.

e : تعبير فورتراني أو قد يحتوي ثوابت ودوال ضمنية وإشارة لدوال جملة سبق تعريفها وإشارة لدوال خارجية.

مثال: اكتب دالة جملة لحساب F من المعادلة التالية، ثم استدعي هذه الدالة ضمن برنامج رئيسي:

الحل: يجب تعريف دالة الجملة قبل أول جملة تنفيذية في البرنامج الرئيسي.

```
F(x)=(1.0+Exp(x))/(1.0-Exp(-x))
Read *,B
Z=F(B)
Y=B+F(1.5)
A=1.5*B/F(B+2.5)
Print*,B,z,y,A
Stop
End
```

ملاحظة: 1. يجب أن تتوحد المتغيرات والثوابت مع دليل الدالة (x) وهو متغير حقيقي.

٢. إن دالة الجملة تحتوى ضمن البرنامج الرئيسي ولا يمكن استدعائها من برنامج آخر، وتستدعى دالة الجملة من خلال اسمها.

مثال: اكتب دالة جملة لحساب: $F(x) = x^3 + 2x^2 - x + 10$

```
Integer x,y,y1,y2
F(x)=x**3+2*x**2-x+10
y=f(5)
y1=5*f(3)
y2=f(-2)
Print*,y,y1,y2
Stop
End
```

ملاحظة: يجوز كتابة دالة الجملة بعد real or Integer ، ولابد أن يماثل الدليل الوسيط للدالة اسم الدالة.

H.W: اكتب برنامج لطباعة عشر ارقام من(1 الى 10) للدليل الوسيط x للدالة f(x) ، إذ أن الدالة هي:

$$F(x) = x^3 + 2x^2 - x + 10$$

الدالة الضمنية: وتسمى أيضا بالدالة المبنية built-in function وتم التطرق سابقا إلى هذه الدوال والتي تسمى بالدوال المكتبية.

أمثلة:

```
K=INT(2.5)
N=max0 (15,20,5,30)
ML=Mod (15,7)
```

الدالة الخارجية :- هو عبارة عن برنامج فرعي يتكون من مجموعة من الجمل الفورترانية للقيام بمهمة معينة ، وهي كالبرنامج الفرعي يكتب بمعزل عن البرنامج الرئيسي:

```
t Function name (a1,a2,…….,an)
```

t: جملة نوعية لتحديد نوع الدالة الخارجية وهذه الجمل هي (Integer,Real,character)، ويجوز إهمال t وعندئذ تتحدد نوعية الدالة بنوعية اسمها.

Function : الأمر الذي يعلن بداية الدالة الخارجية.

Name : اسم الدالة وهو متغير وعن طريقه تعود النتيجة التي تحسبها الدالة.

ان صيغة البرنامج للدالة الخارجية يكتب كالأتي :

```
T function name(a1,a2,….,an)
                .
                .
          جمل فورترانية
                .
                .
Return
End
```

مثال: اكتب دالة خارجية لاستقبال عددين صحيحين وإعادة الأكبر منهما، استعمل هذه الدالة في برنامج رئيسي لإيجاد اكبر قيمة من أربعة متغيرات ، استخدم GRT اسما للدالة الخارجية.

```
Program main
Integer w,x,y,z,GRT
Read*,w,x,y,z
K1=GRT(w,x)
K2=GRT(y,z)
Print*,GRT(k1,k2)
Stop
End

Integer function GRT(n1,n2)
If (n1.GT.n2) then
GRT=n1
Else
GRT=n2
End if
Return
End
```

هنا اسم الدالة GRT هو الذي يعيد النتيجة للبرنامج بجملة إحلال.

مثال: اكتب برنامج لإيجاد مجموع عناصر مصفوفة أحادية باستخدام دالة خارجية:

```
Integer  s
Dimension  m(10)
Data m/……/
Print*, s(m)
Stop
End

Integer Function s(n)
Dimension n(10)
S=0
Do 10 I=1,10
10 s=s+n(i)
Return
End
```

ملاحظات مهمة:

١. اسم الدالة يجب ان يظهر ولو لمرة واحدة على الطرف الأيسر لجملة الإحلال فعن طريقه تنتقل النتيجة النهائية إلى البرنامج الرئيسي.

٢. الأدلة الوسيطة في الدالة الخارجية هي أدلة صورية يستعاض عنها بأدلة حقيقية في البرنامج الرئيسي على ان تماثل الأدلة الصورية عددا وتسلسلا ونوعية.

٣. يمكن أن يكون الدليل الوسيط الحقيقي عند الاستدعاء أيا مما يأتي :

- ثابتا.
- متغيرا اعتياديا.
- متغير موسوم مع الوسم.
- متغير موسوم مع ذكر الاسم فقط بلا وسم.
- اسما لدالة خارجية أخرى أو روتين فرعي.

٤. ظهور اسم الدالة الخارجية في الحالات الآتية يقود إلى نتيجة خاطئة وهي :-

- في الطرف الأيسر من جملة الإحلال.
- في جملة read .
- كدليل مؤشر لدارة do .

الروتين الفرعي Subroutine :-

هو برنامج فرعي كالدالة الخارجية ، يكتب بمعزل عن البرنامج الرئيسي لكنه يختلف عن الدالة الخارجية بالأمور التالية:

1 . إمكانية إعادة أكثر من قيمة للروتين الفرعي على عكس الدالة الخارجية التي تعيد قيمة واحدة.والأدلة الوسطية في الروتين الفرعي هي قيم داخلة وخارجة ويكتب كالآتي:

```
Subroutine   name (a1,a2,……,an)
or  subroutine name
```

Name : اسم الروتين الفرعي.

ai : الأدلة الوسطية الداخلة والخارجة.

2. ليس لنوعية اسم الروتين الفرعي دخل في إعادة القيم وإنما نوعية الأدلة الوسيطة هي التي تحدد القيم العائدة ولا تعاد النتائج للروتين الفرعي بجملة إحلال مثل الدالة الخارجية.

3. تستدعى الدالة الخارجية في البرنامج الرئيسي عن طريق جملة إحلال في الطرف اليمن بينما يستدعى الروتين الفرعي في جملة مستقلة هي جملة استدعاء.

```
Call name (a1,a2,……,an)
```

أما هيكل الروتين الفرعي فلا يختلف عن الدالة الخارجية الأ في البداية.

```
Subroutine  name(a1,a2,….,an)
                .
            جمل فورترانية
                .
Return
End
```

ملاحظة :

- إن الأدلة الوسيطة في الروتين الفرعي هي أدلة صورية تستبدل بأدلة حقيقية عند الاستدعاء بجملة call شرط أن تتجانس الأدلة الوسيطة في العدد والتسلسل والنوع.
- عندما يكون الدليل الوسيط اسما لمنظومة يجب أن يظهر اسم المنظومة بجملة تلي جملة subroutine لتعريف حجم ونوع المنظومة هذه.

مثال: اكتب روتينا فرعيا لحساب مجموع عناصر مصفوفة احادية Array(30) :

```
Dimension Array(30)
Read*,(Array(i),i=1,30)
Call sum(Array,30,s)
Print*,s
Stop
End

Subroutine sum(A,N,s)
Dimension A(n)
s=0.0
Do 10 i=1,n
10  s=s+A(I)
Return
End
```

H.W : اكتب روتينا فرعيا لحساب مساحة ومحيط دائرة ذات قطر R يستدعى ضمن برنامجا رئيسيا لحساب المساحة السطحية وحجم اسطوانة طولها L

الفصل الثامن: الملفات

في كثير من الأحيان نحتاج إلى أن نحفظ البيانات في أوساط أخرى غير الذاكرة ولمدة طويلة، وليست البيانات وحدها تحتاج إلى وسط خارجي لخزنها. البرامج أيضا تحتاج إلى مثل هذا الوسط فليس من المقبول أن نعيد إدخال البرنامج سطرا سطرا او كلمة كلمة في كل مرة نحتاج فيها إلى تنفيذه فذلك مضيعة للوقت وهدر في المجهود.

ان خزن البرنامج بكل تفاصيله في وسط غير الذاكرة واستدعاءه عند الحاجة هو الحل المطلوب والمقبول. هذا الوسط الذي يقوم بحفظ المعلومات ويعيدها عند الطلب هو الملف الذي تخزن فيه المعلومات.

التعامل مع الملفات:

وفرت لغة (fortran 77) جملا للتعامل مع الملفات، فهناك جمل لفتح الملفات وغلقها والاستعلام عن خواصها مثل جمل (open, close, inquire) وهناك جمل للقراءة من الملف أو الكتابة فيه مثل read و write وقد تستخدم هاتان الجملتان للقراءة أو الكتابة في ملفات داخلية. وهناك نوع ثالث من الجمل تعين المبرمج في تحديد موقع مؤشر الملف مثل جمل backspace و rewind و endfile

- طريقة فتح الملفات :-

```
Open (1,file='m.dat')
Open (2,file='n.dat')
```

- طريقة قراءة البيانات:

```
    Read (*,*) A,B,C
    Read (1,*) A,B,C
    Read (1,10) A,B,C
10 format (3F7.2)
```

الرقم (1): يرمز إلى رقم الملف الذي يقرأ البرنامج من خلاله قيم A,B,C .

الرقم (10): يرمز إلى جملة الصيغة.

- طريقة طباعة ألنتائج

```
      Write  (*,*) A
      Write  (2,*) A
      Write  (2,10) A
10    format ('A=',F8.1)
```