

تمهيد

كلمة **Fortran** هي اختصار في الأصل إلى جملة **Formula Translation** وقد نشأت هذه اللغة لغة فورتران عام ١٩٥٧م على يد فريق من المبرمجين في شركة **IBM** باقتراح أحد المهندسين جون باكسون (**John W. Backus**) وتم إنشاء هذه اللغة من أجل استخدامها في التحليلات العددية والحوسبة العلمية وكذلك للأغراض الهندسية وتعتبر هذه اللغة هي أولى لغات البرمجة ذات المستوى العالي (**High Level**) كما تتميز هذه اللغة بالبساطة والإيجاز ويتم تنفيذ الأوامر بها تبعاً لتسلسل منطقي ومنذ عام ١٩٥٧ حتى الآن قد صدر العديد من الإصدارات لهذه اللغة وهي (66 - 70 - 77 - 90 - 95 - 2008 - 2013 - 2015) وحتى الآن لازالت تدرس في العديد من الجامعات وفي هذه الأوراق القليلة سوف نتعرف معاً على الأوامر الأساسية للغة فورتران وطريقة كتابتها مع بعض الأمثلة البسيطة التي تساعد على فهم الأوامر بشكل أفضل بالإضافة إلى أمثلة قصيرة للتطبيق على ما سوف نتناوله بعد كل جزء ، وهذا الكتاب عبارة عن مقدمة بسيطة للمبتدئين في التعامل مع لغة فورتران وليس للمحترفين ، وما أصابني في هذا الكتاب من توفيق فهو من الله عز وجل وما أصابني من تقصير فمن نفسي أسأل الله أن يجعل هذا العمل خالصاً لوجهه الكريم.

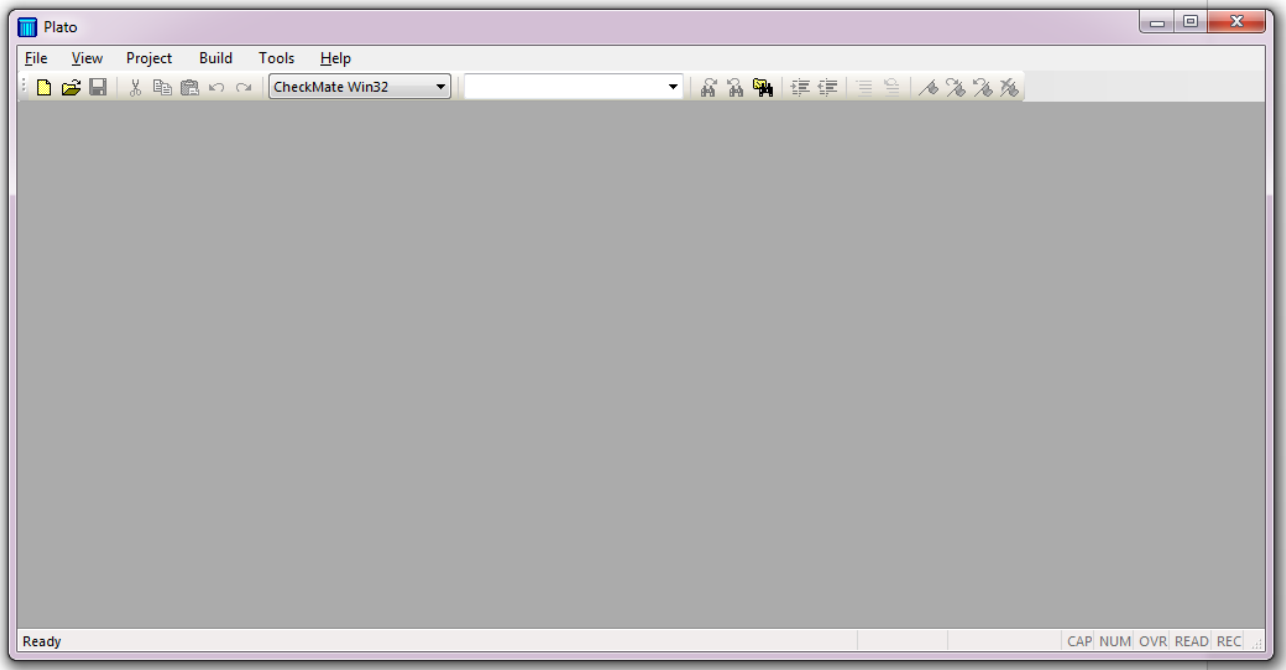
الفصل الأول

مقدمة

(Introduction)

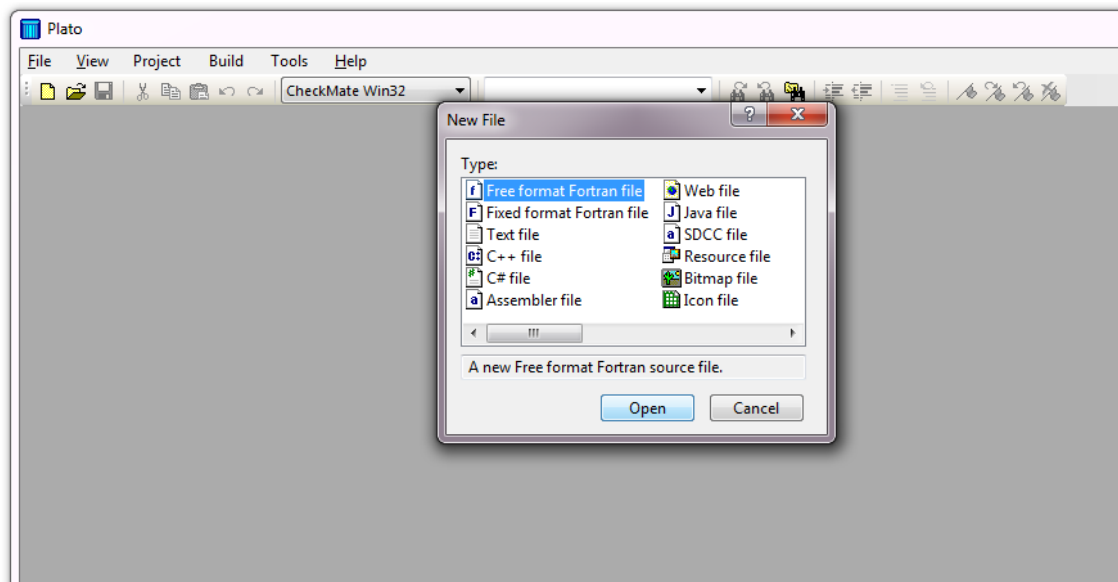
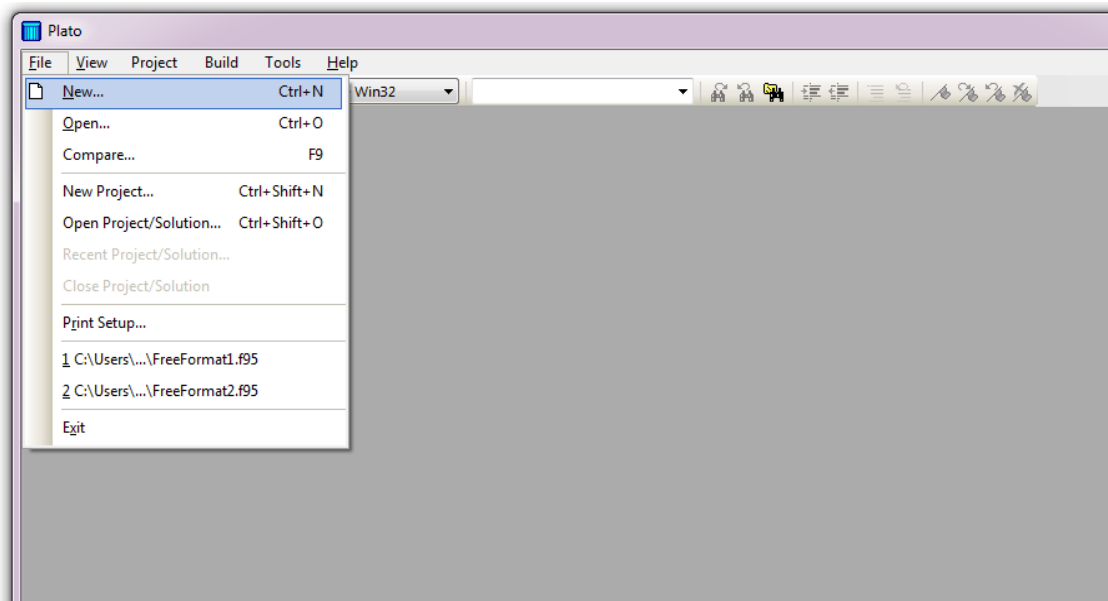
البيئة البرمجية

أصبحت البرمجة أكثر سهولة مع استخدام الواجهات الرسومية وذلك لوجود العديد من البرامج التي تساعدك في كتابة الأكواد بشكل منظم والعديد من البرامج التي تحتوى على مكتبات اللغة والمترجم وأصبح كل ما عليك هو كتابة الأكواد ثم مراجعة البرنامج وتشغيله وتتميز هذه البرامج بسهولة اكتشاف الأخطاء ومن أكثر البرامج المستخدمة في كتابة لغة Fortran برنامج **Code Blocks** و **Silver Front** و **Plato** وهى برامج ذات واجهة بسيطة سهلة الاستخدام وفى هذا الكتاب سوف نستخدم برنامج plato حيث أنه أكثرها بساطة فى الشكل والإستخدام وهذه هى صورة واجهة البرنامج:



بدأ مشروع جديد

لكى تبدأ مشروع جديد عليك اتباع الخطوات التالية من قائمة File اختر New سوف يظهر لك صندوق اختر منه Free Format Fortan File كما هو موضح بالصورة التالية:

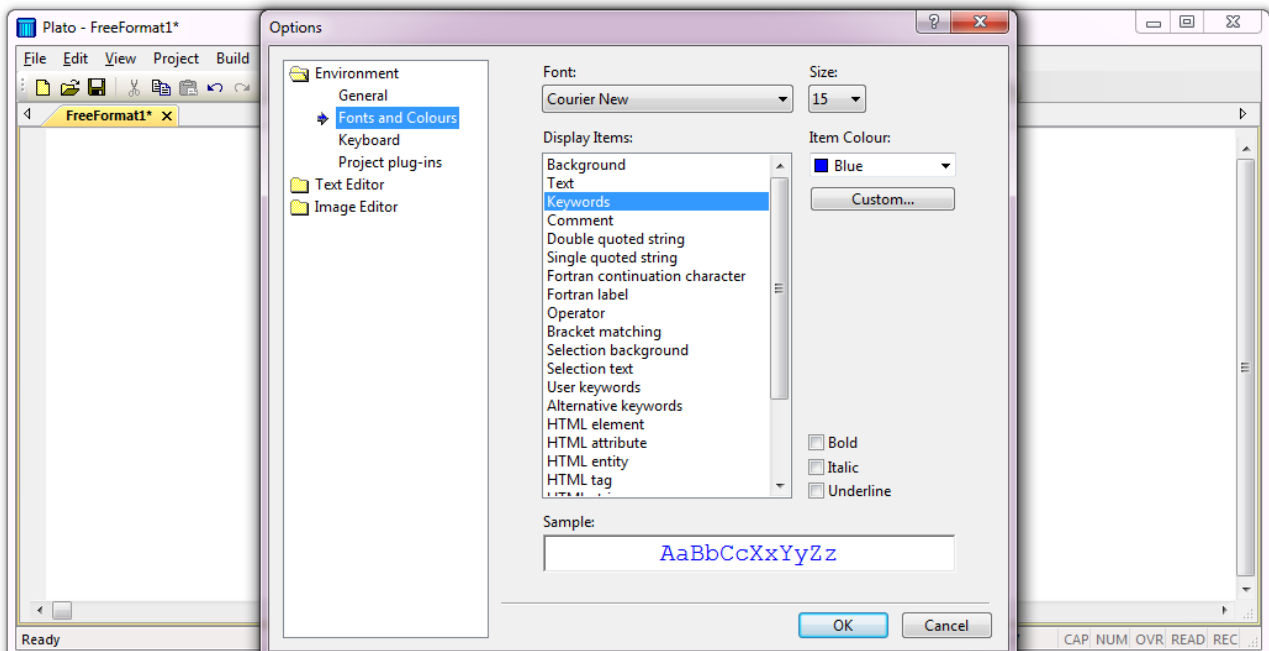
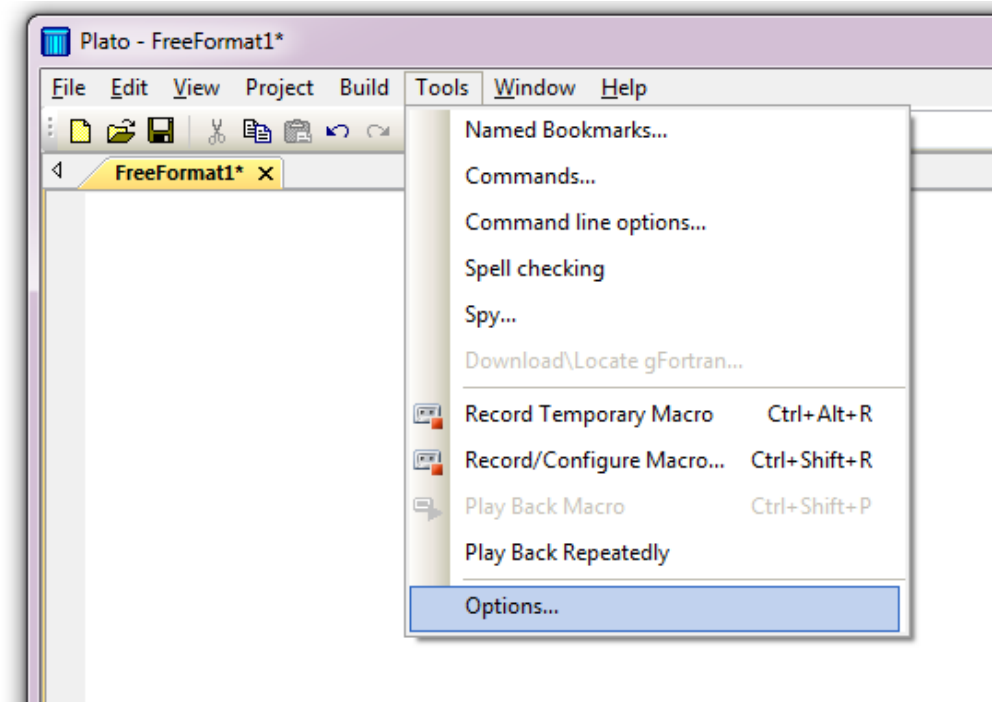


نموذج لبرنامج صغير يعرض رسالة ترحيب ! Hello World

A screenshot of the Plato IDE showing a Fortran program. The window title is 'Plato - FreeFormat1*'. The menu bar includes 'File', 'Edit', 'View', 'Project', 'Build', 'Tools', 'Window', and 'Help'. The toolbar shows various icons, and the dropdown menu is set to 'CheckMate Win32'. The main editor area contains the following Fortran code:

```
Program Hello_World
Implicit None
Write(*,*) 'Hello World !'
pause
EndProgram Hello_World
```

قد يكون الخط صغير بعض الشيء فى البرنامج ويمكن تكبيره من خلال الخطوات التالية:



البنية الأساسية للبرنامج

يتكون كود البرنامج في لغة فورتران من ثلاثة أجزاء رئيسية.

1-The Declaration Section.

في هذا الجزء يتم كتابة الجملة الابتدائية التي تحتوي على اسم البرنامج بالإضافة إلى الإعلان عن المتغيرات والثوابت .

2-The Execution Section.

في هذا الجزء يتم كتابة الأوامر والأكواد التي سيقوم البرنامج بتنفيذها .

3-The Termination Section.

في هذا الجزء يكون البرنامج قد انتهى من تنفيذ الأوامر ويخبر المترجم بأن البرنامج قد اكتمل ، وينتهي هذا الجزء بالجملة الخاتمة للبرنامج .

الشكل العام للبرنامج

```
!Declaration section
program calculator
implicit none
integer :: x,y,z
real :: a,b,c
character :: my_name

!Execution section

يتم كتابة الأوامر في هذا الجزء من البرنامج!

!Termination Section

Endprogram calculator
```

لاحظ الآتى:

١- يتم كتابة البرنامج بين السطريين

Program **name**

.....

.....

.....

End program **name**

حيث **name** هو اسم البرنامج

٢- تستخدم علامة التعجب (!) فى كتابة التعليقات وهذه التعليقات لا يلتفت اليها المترجم عند ترجمة البرنامج ولكنها تساعد المبرمج فى كتابة بعض الملاحظات وتنظيم العمل داخل البرنامج لمعرفة وظيفة كل جزء من البرنامج عند إعادة فتح الكود للتعديل عليه أو الإضافة كما أنه يساعد المبرمجين الآخرين على فهم بنية البرنامج ووظيفة كل دالة أو جزء.

Ex.

! Welcome to your first program

مرحباً بك فى أول برنامج لك !

أنواع المتغيرات (Types of Data)

| TYPE | الاستخدام |
|-----------|---|
| INTEGER | تستخدم مع الأرقام الصحيحة. |
| REAL | تستخدم مع الأرقام العشرية. |
| CHARACTER | تستخدم مع الحروف والأسماء. |
| LOGICAL | تستخدم مع البيانات التي لها احتمالين (TRUE - FALSE)، قليلة الاستخدام. |

Examples of Data :

INTEGER [1-2-3-4-5-56-789-234-56789]

REAL [2.5-3.56-5.009-1.0]

CHARACTER [m-s-n-U-D-A-a-ahmed-may]

شروط اختيار أسماء المتغيرات :

- ١- يبدأ بحرف ولا يبدأ برقم.
- ٢- يمكن أن يحتوى على أرقام.
- ٣- لا يمكن استخدام الرموز إلا الرمز `underscore`.
- ٤- الحروف الكبيرة والصغيرة متساوية حيث (`ahmed=AHMED`) يعبران عن متغير واحد .
- ٥- لا يزيد اسم المتغير عن ٣٢ حرف.

الإعلان عن المتغيرات (Declaring Variables)

يتم الإعلان عن المتغيرات من خلال الصيغة الآتية :

Type :: Variable

Type: نوع المتغير (integer-real-character-logical)

Variable: اسم المتغير (num1-factor-sum-....-num2)

Examples:

Integer :: num1

Integer :: num1,num2

Real :: x,y,z

Character :: a,b,c

Character(len=7) :: my_name

Character(7) :: my_name

لاحظ:

عند تعريف متغير من النوع **character** يجب تحديد طول المتغير وتستخدم كلمة **Len** لتدل على طول المتغير ويمكن الإستغناء عنها وكتابة الرقم مباشرةً أى أن المثالين الأخيرين متساويين فى المعنى .

معلومة هامة ::

فى حالة استخدام متغيرات لم يتم تعريفها من قبل يقوم البرنامج بتعريفها كمتغيرات من النوع integer وذلك اذا كان اسم المتغير يبدأ بأحد الحروف الآتية (i-j-k-l-m-n) وفى ما عدا ذلك يقوم بتعريفها كمتغيرات من النوع real .

ولإلغاء هذه الخاصية نقوم بكتابة الجملة الآتية :

Implicit None

بعد جملة البدء

Program **name**

الإعلان عن الثوابت (Declaring Constants)

يتم الإعلان عن الثوابت من خلال الصيغة الآتية :

Type, parameter :: Variable=value

Examples:

Integer,parameter :: sum=0

Real,parameter :: pi=3.1415

Character(len=8),parameter :: myname='Mohamed'

لاحظ: وضع النص بين علامتى تنصيص '-----'

العمليات الحسابية (Arithmetic operations)

فى هذا الجزء سوف نتعرف على الرموز المستخدمة للتعبير عن العمليات الحسابية داخل البرنامج وهى موضحة بالجدول التالى :

| الرمز | الإستخدام | مثال |
|-------|--------------------|------------------------------------|
| + | الجمع | $12 + 13 = 25$, $2.0 + 3.0 = 5.0$ |
| - | الطرح | $25 - 12 = 13$ |
| / | القسمة | $90/3 = 30$ |
| * | الضرب | $5*5=25$, $4*16 = 64$ |
| ** | الاس | $2**5 = 32$ |
| = | الناتج | |
| e | تعبير عن ١٠ اس رقم | $2e5=2 \times 10^5=200000$ |

ترتيب العمليات الحسابية داخل البرنامج

| Precedence Level | Operator | Operation |
|------------------|----------|-----------------------------|
| 1 st | - | unary - |
| 2 nd | ** | exponentiation |
| 3 rd | * / | multiplication and division |
| 4 th | + - | addition and subtraction |

١- الأس

٢- الضرب والقسمة

٣- الجمع والطرح

يمكن استخدام الأقواس فى ترتيب العمليات الحسابية كالتالى :

$$\text{Sum} = ((a+b)**3) * ((25+d)/2)$$

عند القيام بعمليات القسمة يختلف نوع الخرج طبقاً للقواعد التالية :

$$\frac{\text{Integer}}{\text{Integer}} = \text{Integer} \quad \text{ex.} \quad \frac{3}{2} = 1$$

$$\frac{\text{Real}}{\text{Real}} = \text{Real} \quad \text{ex.} \quad \frac{3.0}{2.0} = 1.5$$

$$\frac{\text{Real}}{\text{Integer}} = \text{Real} \quad \text{ex.} \quad \frac{3.0}{2} = 1.5$$

يجب مراعاة هذه القواعد عند كتابة الكود حتى نحصل على قيم سليمة .

المعاملات المنطقية (Logical operator)

| | |
|----------------------|--|
| And | النتيجة صحيحة إذا كان كلا الشرطين صحيحين |
| If (t>37 .and. t<40) | |
| Or | النتيجة صحيحة إذا كان أي من الشرطين صحيحين |
| If (t>37 .or. t=37) | |
| Not | إذا كان خطأ فهو صحيح وإذا كان صحيح فهو خطأ |
| If (t=37 .not. t=40) | |

الدوال (Functions)

فى هذا الجزء سوف نتعرف على بعض الدوال المستخدمة فى لغة Fortran والتي هى جزء من مكتبة اللغة.

| الدالة | الاستخدام | مثال |
|------------|--------------------------|------------------|
| Sqrt(x) | لحساب الجذر التربيعى ل X | Sqrt(4) = 2 |
| ABS(x) | لحساب القيمة المطلقة لx | ABS(-50) = 50 |
| Sin(x) | حساب دالة الجيب | Sin($\pi/2$)=1 |
| Cos(x) | حساب دالة جيب التمام | Cos(0)=1 |
| Tan(x) | حساب دالة الظل | Tan(0)=0 |
| ASin(x) | معكوس دالة الجيب | Asin(1)= $\pi/2$ |
| ACos(x) | معكوس دالة جيب التمام | Acos(1)=1 |
| ATan(x) | معكوس دالة الظل | Atan(0)=0 |
| EXP(x) | لحساب e^x | EXP(2)=7.3890560 |
| Log(x) | اللوغاريتم الطبيعي لx | Log(10)=1 |
| Log10(x) | لوغاريتم الأساس ١٠ | |
| Mod(a,b) | باقى قسمة a على b | Mod(5,2)=1 |
| Max(a,b) | أيهما أكبر | Max(10,5)=10 |
| Min(a,b) | أيهما أصغر | Min(10,5)=5 |
| Ichar(C) | الرقم المقابل للحرف C | Ichar(C)=67 |
| Achar (67) | الحرف المقابل للرقم 67 | Achar(67)=C |

ملحوظة:

عند استخدام الدوال المثلثية فى لغة Fortran يجب ادخال الزوايا على شكل radian وليس Degree .

دوال التحويل (Conversion Function)

هي دوال تستخدم لتغيير نوع القيمة integer أو real .

Real(integer value)

تقوم هذه الدالة بتحويل الأرقام الصحيحة إلى أرقام عشرية بإضافة العلامة العشرية حتى إذا كان الرقم صفر.

$$\text{Real}(45)=45.0$$

Int(real value)

تقوم هذه الدالة بتحويل الأرقام العشرية إلى أرقام صحيحة بإزالة العلامة العشرية وأى أرقام بعدها.

$$\text{Int}(25.5)=25$$

Nint(real value)

تقوم هذه الدالة بتحويل الرقم العشري إلى صحيح مع التقريب.

$$\text{Nint}(24.6)=25$$

Ceiling(real value)

تقوم هذه الدالة بتحويل الرقم العشري لرقم صحيح أكبر منه أو يساويه.

$$\text{Ceiling}(24.5)=25$$

Floor(real value)

تقوم هذه الدالة بتحويل الرقم العشري لرقم صحيح أصغر منه أو يساويه.

$$\text{Floor}(24.3)=24$$

جمل الإدخال والإخراج (I/O Statements)

Print*,

يستخدم هذا الأمر في طباعة البيانات على الشاشة .

Ex.

```
Print*, 'Welcome to my program .'
```

```
Welcome to My Program .
```

Write(*,*)

يستخدم هذا الأمر في طباعة البيانات على الشاشة أو داخل ملف .

Ex.

```
Write(*,*) 'Hello World !'
```

```
'Hello World !'
```

Read(*,*)

يستخدم هذا الأمر في قراءة البيانات من المستخدم أو من ملف .

Ex.

```
Print*, 'Enter your name'
```

```
Read(*,*) name
```

حيث name هو متغير من النوع character .

Write (*,*) & Read (*,*)

فى جمل الإدخال والإخراج يوجد علامتين داخل القوس لكل منهما
وظيفة يجب التعرف عليها:

العلامة الأولى : وتدل على أن وحدة الإدخال الأساسية هى لوحة المفاتيح
(Key board) ويمكن استبدالها برقم للقراءة من ملف معين يحمل نفس الرقم
وسوف نتطرق إلى هذا فى الجزء الخاص بالملفات .

Ex.

Write (10,*) , Read (9,*)

حيث ١٠ هو رقم الملف الذى سيتم الكتابة فيه و٩ رقم الملف الذى يتم
القراءة منه

العلامة الثانية : وتعنى أن النص سوف يتم عرضه بتنسيق تلقائي على الشاشة
ويمكن من خلالها التحكم فى شكل النص المعروض وطريقة عرضه وسوف يتم
مناقشة ذلك لاحقاً فى الجزء الخاص بالتنسيق (Format) .

الفصل الثانی

الجملة الشرطية

(Condition Statements)

الجمل الشرطية (Condition Statements)

تستخدم الجمل الشرطية في تنفيذ الأوامر المشروطة وفي هذه الحالة يجب أن تتحقق الشروط أولاً ليقوم البرنامج بتنفيذ الأوامر وفي حالة لم تتحقق الشروط يقوم البرنامج بتنفيذ مجموعة أخرى من الأوامر.

وقبل التطرق إلى الجمل الشرطية يجب أولاً التعرف على البعض الرموز المستخدمة معها لكتابة الشروط وهي موضحة بالجدول التالي:

| Relational Operation | Relational Operator (normal) | Relational Operator (alternate) |
|-----------------------|------------------------------|---------------------------------|
| Greater than | > | .gt. |
| Greater than or equal | >= | .ge. |
| Less than | < | .lt. |
| Less than or equal | <= | .le. |
| Equal to | == | .eq. |
| Not equal to | /= | .ne. |

يمكن استخدام الرموز أو الإختصارات مثل :

$30 > 20$ \longrightarrow $30 .gt. 20$

$34 /= 20$ \longrightarrow $34 .ne. 20$

IF Statement

```
IF (condition) then
```

```
Statement.....
```

Ex. End IF

في هذه الطريقة من جمل If يتم تطبيق مجموعة من الجمل في حالة تحقق شرط معين.

```
If (GAME_LIVES==0) then
```

```
Write(*,*)'Game Over'
```

```
Write(*,*)'Try again'
```

```
End if
```

Simple Form of IF

```
IF (condition) Statement
```

في حالة تنفيذ أمر واحد يمكن كتابت الجملة في سطر واحد مع حذف then

Ex.

```
If (gamelives==0) Write(*,*)'Game Over'
```

IF Then Else

```
IF (condition) then
```

```
Statement1
```

```
Else
```

```
Statement2
```

```
End IF
```

يستخدم هذا الشكل لتنفيذ أمر معين في حالة تحقق الشرط فإذا لم يتحقق يتم تنفيذ مجموعة أخرى من الأوامر.

Ex.

```
If (GAME_LIVES > 0) then
Write(*,*)'Still Alive,Keep Going'
Else
Write(*,*)'Game Over,Try again'
End if
```

IF Then Else IF

```
IF (condition1) then
Statement1
Else IF (condition2) then
Statement2
Else
Statement3
End IF
```

في هذا الجزء يتم وضع مجموعة من الشروط إذا تحقق أحدها يتم تنفيذ الأوامر المتعلقة به.

Ex.

```
If (GAME_LIVES > 0) then
Write(*,*)'Still Alive,Keep Going'
Else If (GAME_LIVES < 0) then
Write(*,*)'Game Over,Try again'
Else
Write(*,*)'Extra life granted'
End if
```

Select Case Statement

Select case (variable)

Case (selector-1)

Statement 1

Case (selector-2)

Statement 2

Case (selector-3)

Statement 3

Case default

Statement

End select

يعتبر هذا الأمر أكثر حرية من الأمر IF حيث يتيح مجموعة كبيره من المتغيرات لشرط واحد يتم تنفيذ الأمر المتعلق بالشرط الذى يتحقق دون الحاجة لكتابة جمل if كثيرة .

Ex.

```
select case (monthnumber)
case (1,3,5,7,8,10,12)
daysinmonth = 31
case (2)
  if (mod(year,4)==0) then
    daysinmonth = 29
  else
    daysinmonth = 28
  end if
case (4,6,9,11)
```

```
daysinmonth = 30
case default
write(*,*) "Error, month number not valid."
end select
```

البرنامج السابق يقوم بطباعة عدد أيام الشهر وفي حالة ادخال رقم شهر خطأ تظهر رسالة خطأ فمثلا اذا ادخل ١٣ تظهر رسالة خطأ حيث عدد شهور السنة ١٢ أى تكون الأرقام (١،٢،٣،.....،١٠،١١،١٢).

ملاحظة:

عند كتابة selectors داخل الأقواس يمكن كتابتها بأكثر من طريقة فى حالة مطابقة الأوامر على أكثر من selector كما فى المثال السابق

```
case (4,6,9,11)
```

أو فى حالة الأرقام المتتالية يمكن كتابة الأرقام من ١ إلى ١٠ كالتالى:

```
case (1:10)
```

الفصل الثالث

الجملة التكرارية

(Loops)

الجمل التكرارية (Loops)

تستخدم أوامر التكرار في تنفيذ بعض الأوامر عدد معين من المرات أو حتى يتحقق شرط معين وبذلك نتفادي تكرار كتابة الأوامر مئات المرات مما يجعل البرنامج أسرع في العمل وأقل في مساحة التخزين، كما أن لها استخدامات أخرى في قراءة البيانات في المصفوفات والتي سوف نتعرف عليها لاحقاً .

أولاً: (Do Loop)

Do variable=start,stop,step

Statements.....

End do

| Variable | اسم العداد أو متغير من النوع integer |
|------------|--------------------------------------|
| Start | رقم البداية |
| Stop | رقم النهاية |
| Step | الخطوة أو مقدار الزيادة أو النقصان |
| statements | الأوامر المراد تكرارها |

ملحوظة: في حالة عدم وضع قيمة للخطوة (step) يتم فرضها بواحد (1).

Ex.

المطلوب عمل برنامج لطباعة الأرقام من 1 إلى 100

```
Program printnum
Implicit none
Integer :: n
Do n=1,100
Print*,n
End do
End program printnum
```


ثانياً: (While loop)

في هذا النوع من أوامر التكرار يتم التحقق من الشرط أولاً فإذا كان True يقوم البرنامج بتنفيذ الأوامر وإذا كان False ينتقل البرنامج لتنفيذ الأوامر التي بعد End Do .

Form 1

Do while (logical expression)

Statements.....

End do

Ex.

المطلوب في المثال التالي جمع الأرقام من 1 إلى 50 .

```
program sum_num
implicit none
integer :: n,sum
n=0
sum=0
do while (n<=50)
sum = sum + n
n=n+1
end do
print*,sum
pause
end program sum_num
```

لاحظ: في هذا البرنامج تم استخدام كلمة pause ، عند عمل run للبرنامج ستجد أن البرنامج قد اختفي بشكل سريع دون أن تتمكن من رؤية النتائج وعند استعمال كلمة pause تظل شاشة البرنامج موجودة وتتمكن من مشاهدة النتائج ولن تختفي حتى تضغط على Enter من الكي بورد.

Form 2

يتم تكرار الأوامر حتى يتحقق الشرط فيتم الخروج من دائرة التكرار.

Do

IF (logical expression) Exit

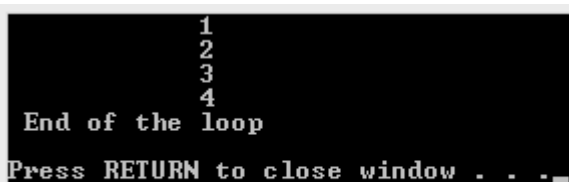
Statements.....

End do

Ex.

الكود التالي يوضح برنامج يقوم بطباعة الأرقام بداية من ١ ويتوقف عندما يصل إلى ٥ .

```
program exit_loop
implicit none
integer :: i
i=1
do
  if (i==5) exit
write(*,*) i
i = i + 1
end do
write(*,*) 'End of the loop'
pause
end program exit_loop
```



```
1
2
3
4
End of the loop
Press RETURN to close window . . . _
```

نتيجة تشغيل البرنامج

لاحظ : تم استخدام كلمة Exit للخروج من التكرار.

Cycle

تستخدم كلمة cycle للعودة إلى بداية التكرار.

Ex.

في هذا المثال سوف يقوم البرنامج بطباعة الأرقام من ١ إلى ٥ ولكنه سيتجنب طباعة رقم ٣ وذلك بسبب كلمة cycle أمام الشرط.

```
program cycle_loop
implicit none
integer :: i
do i=1,5
  if (i==3) cycle
  write(*,*) i
end do
pause
end program cycle_loop
```

```
1
2
4
5
***Pause:
Enter system command or press ENTER key to restart: _
```

Nested Loop

عبارة عن جمل تكرارية متداخلة يتم استخدامها في بعض البرامج مثل جدول الضرب وغيره وفي حالة استخدامها يفضل إعطاء اسم لكل دورة تكرارية كما هو موضح بالمثال التالي:

Ex.

```
program nested_loop
implicit none
integer :: i,j,product
first_loop :do i=1,3
  second_loop :do j=1,3
    product=i*j
    write(*,*) i,'*',j,'=',product
  end do second_loop
end do first_loop|
pause
end program nested_loop
```

الفصل الرابع

التنسيق

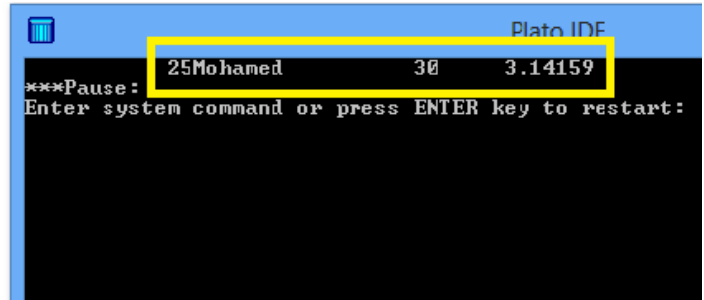
(Format)

التنسيق (Format)

لقد ذكرنا سابقاً أن في الجزء الخاص بالعلامات الموجودة في أقواس جمل الإدخال والإخراج أن العلامة الثانية تعنى عرض النص بالتنسيق الأساسي للبرنامج .

لاحظ البرنامج التالي وطريقة عرض البيانات فيه.

```
program for_mat
implicit none
integer :: i=25,y=30
real :: pi=3.141592654
character(7) :: name="Mohamed"
write(*,*)i,name,y,pi
pause
end program for_mat
```



```
Plato IDE
25Mohamed 30 3.14159
***Pause:
Enter system command or press ENTER key to restart:
```

الملاحظات:

- 1- تلاصق الكلمات مع الأحرف في 25Mohamed
- 2- وجود مسافة كبيرة بين Mohamed و 30 وكذلك بين 30 و 3.14159
- 3- تم عرض عدد كبير من الأرقام بعد العلامة العشرية 3.14159

في حالة استخدام التنسيق سوف نتمكن من :

- 1- ضبط المسافات الرأسية والأفقية.
 - 2- التحكم في عدد الأرقام بعد العلامة العشرية.
 - 3- التحكم في عدد الحروف والأرقام المراد عرضها على الشاشة.
- ولإستخدام التنسيق يجب التعرف على بعض الرموز التي سوف نستخدمها ودلالاتها قبل التعرف على طريقة كتابة التنسيق والتي يلخصها الجدول التالي:

| | |
|---|--|
| c | رقم العمود الرأسي. |
| d | عدد الخانات (الأرقام) بعد العلامة العشرية في المتغير من نوع real |
| m | أقل عدد من الأرقام يجب عرضه. |
| w | عدد الخانات المستخدم للعرض. |
| r | عدد مرات تكرار التنسيق. |
| n | عدد المسافات . |
| / | مسافة رأسية. |

توضيح عدد خانات العرض w

مثال: Mohamed تشغل ٧ خانات اذا كان w =5 سوف يتم عرض Moham

الصيغ المستخدمة في التنسيق

| | |
|-----------|--------------|
| Integer | rlw or rlw.m |
| Real | rFw.d |
| Logical | Rlw |
| Character | rA or rAw |
| space | nX |
| Tab | Tn |

طرق كتابة التنسيق

يوجد عدة طرق لكتابة التنسيق في لغة فورتران ولكننا سنكتفي بذكر الطريقتين الأكثر استخداماً.

الطريقة الأولى:

```
Write (*, 100) .....  
100 Format (.....)
```

في هذه الطريقة يتم استبدال * الثانية برقم ثم كتابة الرقم في سطر جديد وأمامه كلمة Format ويتم وضع التنسيق بين () ويفصل بينها ، .

Ex.

```
Write (*,100) i ,x
```

```
100 format (1x ,i6,f10.2)
```

معنى التنسيق 1x مسافة واحده ثم i6 متغير من نوع integer سوف يتم عرضه في ٦ خانات ثم f10.2 متغير من نوع real سوف يتم عرضه في ١٠ خانات ولا يزيد عدد الأرقام بعد العلامة العشرية عن ٢ .

الطريقة الثانية:

```
Write (*, '(...format...)' ) .....
```

في هذه الطريقة يتم وضع التنسيق داخل علامتي '()' داخل جملة write بدلاً من العلامة * الثانية .

Ex.

```
Write (*, '(1x ,i6,f10.2)') i ,x
```

وهذا نفس المثال السابق عند كتابته بهذه الطريقة.

لاحظ:

في الطريقة الثانية يجب كتابة التنسيق في كل Write بينما في الطريقة الأولى يمكن كتابة جملة تنسيق واحدة لأكثر من جملة write .
من الممكن كتابة نص داخل جملة فورمات ليتم عرضه على الشاشة أو كتابته داخل ملف.

Ex.

Integer :: i

Real :: x

Write (*, 100) i,x

100 Format ('Welcome to',1x ,i6,f10.2)

ملاحظات:

- العلامة العشرية (.) والسالب (-) يشغل كل منهما خانة عند العرض على الشاشة.

مثال:

إذا كانت $w=6$ سيتم عرض **-25,12469** بهذا الشكل **-25,12** بحيث تشغل 6 خانة فقط.

- عند استعمال m إذا كان عدد الأرقام أقل من m يتم وضع أصفار.

مثال:

إذا كانت $m=5$ سيتم عرض **23** بهذه الصورة **00023**

مثال توضيحي ١

`integer :: num1=42, num2=123, num3=4567`

`write (*,'(i2)') num1`

1

تعرض بهذا الشكل "42"

`write (*,'(i2,i3)') num1, num2`

2

تعرض بهذا الشكل "42123"

`write (*,'(i2,i4)') num1, num2`

3

تعرض بهذا الشكل "42 123"

`write (*,'(i5,i5,i5)') num1, num2, num3`

4

تعرض بهذا الشكل "42 123 4567"

`write (*,'(i6.4)') num1`

5

تعرض بهذا الشكل "0042"

مثال توضيحي ٢

`real :: var1=4.5, var2=12.0, var3=2145.5713`

`write (*,'(f3.1)') var1`

تعرض بهذا الشكل "4.5"

`write (*,'(f5.2,f8.3)') var1, var2`

تعرض بهذا الشكل "4.50 12.000"

`write (*,'(f10.4,f10.4,f10.4)') var1, var2, var3`

تعرض بهذا الشكل "4.5000 12.0000 2145.5713"

مثال توضيحي ٣

```
character(len=11) :: msg = "Hello World"
```

```
write (*,'(a11)') msg
```

تعرض بهذا الشكل "Hello World"

```
write (*,'(a)') msg
```

تعرض بهذا الشكل "Hello World"

```
write (*,'(a9,2x,a)') msg, "Goodbye cruel world"
```

تعرض بهذا الشكل "Hello Wor Goodbye cruel world"

لاحظ عدم وجود اختلاف بين الجملة الأولى والثانية.

مثال توضيحي ٤

```
program test
implicit none
integer::index=12
real::time=12 , amplitude= 50 , phase=24 , depth=100
write (*,100) index,time,depth,amplitude,phase
100 format(t20,'results for test number',i3,///,&
1x,'time=',f7.0,/,&
1x'depth=',f7.1,'meters',/,&
1x,'amplitude=',f8.2,/,&
1x,'phase=',f7.1)
pause
end program test
```

```
results for test number 12

time=    12.
depth= 100.0meters
amplitude=   50.00
phase=   24.0
***Pause:
Enter system command or press ENTER key to restart:
```

لا حظ: علامة & تستخدم فى وصل الأسطر وتستخدم إذا كان الأمر كبير ولا يمكن كتابته فى سطر واحد.

الفصل الخامس
العمليات على الملفات
(File Operation)

العمليات على الملفات (File Operation)

تمكننا لغة الفورتران من التعامل مع الملفات عن طريق فتح ملف موجود مسبقاً أو إنشاء ملف جديد لحفظ بعض البيانات وذلك من خلال بعض الأوامر والموجودة في الجدول التالي:

| | |
|-----------|------------------------------------|
| Open | يستخدم لفتح ملف . |
| Close | يستخدم لإغلاق الملف الحالي. |
| Rewind | يستخدم للعودة لبداية الملف الحالي. |
| Backspace | يستخدم للرجوع سطر واحد داخل الملف. |

سوف نتناول في الشرح الأمر Open حيث يعتبر هذا الأمر هو الأكثر استخداماً مقارنة بالأوامر الأخرى والتي يندر استخدامها.

الصورة العامة لكتابة الأمر Open

```
Open (unit=unit number, file='file name',  
      Status='file status', action=file action,  
      lostat=status variable)
```

Unit: عبارة عن رقم صحيح يتم اعطائه للملف ويمكن أن يكون أي رقم عدا (٥، ٦) وذلك لأنهما يعبران عن أجهزة الإدخال الأساسية في الحاسب الآلى.

File: اسم الملف المراد فتحه أو انشاءه منتهى بالإمتداد مثل inputs.txt

Status: حالة الملف وتكون واحده من الآتى:

[New-Old-Replace-Scratch-Unknown]

Action: تحديد نوع الملف هل هو للكتابة أو للقراءة أو للإثنين.

[Read-write-readwrite]

lstate: يتم اعطاءها رقم يعبر عن رسالة خطأ حيث يتم عرض هذا الرقم عند وجود مشكلة بالبرنامج وهو دائماً رقم موجب .

Ex.

```
Open (unit=10, file='inputs.txt', status='new', action=write, iostate=5)
```

مثال تطبيقي

فى هذه المثال سوف نقوم بكتابة برنامج يقوم بقراءة بعض السطور من ملف خارجى ثم يقوم بكتابة رقم السطر والسطر الأصلي فى ملف جديد.

Solution

```
program linenumbers
! الإعلان عن المتغيرات !
implicit none
integer :: i, rdopst, wropst, rdst
character(30) :: rdfile, wrfile
character(132) :: line
! كتابة اسم البرنامج على الشاشة !
write (*,*) "Line Number Example"
! طلب اسم ملف الإدخال من المستخدم !
do
```

```

write (*,'(a)', advance="no") "Input File Name: "
! قراءة اسم ملف الإدخال
read (*,*) rdfile
! فتح ملف الإدخال
! في حالة وجود خطأ في فتح الملف يتم عرض رسالة خطأ
! وفي هذه الحالة يتم الخروج من التكرار
open(12, file=rdfile, status="old", &
action="read", position="rewind", &
iostat=rdopst )
if (rdopst==0) exit
write (*,'(a/,a)') "Unable to open input file.",&
"Please reenter"
end do
! طلب اسم ملف الإخراج من المستخدم
do
write (*,'(a)', advance="no") "Output File Name: "
! قراءة اسم ملف الإخراج
read (*,*) wrfile
! فتح ملف الإخراج
! في حالة وجود خطأ في فتح الملف يتم عرض رسالة خطأ
! وفي هذه الحالة يتم الخروج من التكرار
open(14, file=wrfile, status="replace", &
action="write", position="rewind", &
iostat=wropst )
if (wropst==0) exit
write (*,'(a, a/,a)') "Unable to open ", &

```



```

"output file.", "Please reenter"
end do
!-----
  i= 1
do
! قراءة السطر من ملف الإدخال
read (12, '(a)', iostat=rdst) line
! عند الوصول إلى نهاية الملف يتم الخروج من المر التكرارى
if (rdst >0) stop "read error"
if (rdst <0) exit
! كتابة رقم السطر والسطر الأصى
write (14, '(i10,2x,a)') i, line
i = i + 1
end do
! إغلاق الملفات وانهاء البرنامج
close(12)
close(14)
end program linenumbers

```

الفصل السادس

المصفوفات

(Arrays)

المصفوفات (Arrays)

تعتبر المصفوفات من أهم الأوامر البرمجية فى أى لغة برمجة وذلك لإستخداماتها المتعددة والمهمة وسوف نتعرف فى هذا الجزء على نوعيين من المصفوفات المصفوفة الأحادية والمصفوفة ثنائية البعد.

أولاً: مصفوفة الوحدة

يمكن اعتبار مصفوفة الوحدة كعمود واحد له عنوان رئيسي A وكل قيمة داخل العمود لها رقم يدل عليها مثل القيمة الأوله فى المصفوفه التى اسمها A يعبر عنها بمتغير اسمه $A(1)$ وكذلك جميع القيم حتى آخر قيمة فى المصفوفة وذلك اعتماداً على عدد قيم المصفوفة التى تم إدخالها فى البداية عند تعريف المصفوفة.

| | A |
|----|---------|
| 1 | <value> |
| 2 | <value> |
| 3 | <value> |
| 4 | <value> |
| 5 | <value> |
| 6 | <value> |
| 7 | <value> |
| 8 | <value> |
| 9 | <value> |
| 10 | <value> |

تعريف المصفوفة:

```
type, dimension(extent) :: name1
```

Type: نوع البيانات داخل المصفوفة.

Dimention: عدد خانات المصفوفة وهو عبارة عن رقم صحيح.

Nam1: اسم المصفوفة مثل A .

Ex.

```
integer, dimension(1000) :: nums1
```

```
integer, dimension(-5:5) :: ranges
```

ويطلق على هذا النوع من المصفوفات بالمصفوفة الاستاتيكية حيث تم اعطائها قيمة عند تعريفها ويوجد نوع آخر من المصفوفات وهو المصفوفة الديناميكية ويتم تعريفها بهذا الشكل:

```
type, dimension(:), allocatable :: name
```

```
integer :: status variable
```

```
allocate(name(dimension), stat= status variable)
```

وفى المصفوفة الديناميكية يتم تقدير بعد المصفوفة أثناء تنفيذ البرنامج.

Ex.

```
integer, dimension(:), allocatable :: nums2
```

```
integer :: allst
```

```
allocate(nums2(1000), stat=allst)
```

تعيين قيم المصفوفة

يمكن تعيين قيم المصفوفة بأكثر من طريقة فإذا كانت المصفوفة تتكون من عدد قليل من القيم ولنفرض أربعة قيم يمكن تعيينهم بالشكل التالي:

```
real, dimension(4) :: costs=(/10.0, 15.0, 20.0, 25.0/)
```

حيث يتم وضع القيم بالترتيب داخل قوسيين بهذا الشكل (/...../) حيث يفصل بين القيم بعلامة (,).

أما الطريقة الثانية لإدخال قيم المصفوفة عن طريق قراءتها من المستخدم ويكون باستخدام جملة `read(*,*)` كالتالي:

```
integer, dimension(4) :: A
```

```
read(*,*) A(1)
```

```
read(*,*) A(2)
```

```
read(*,*) A(3)
```

```
read(*,*) A(4)
```

ولكن في حالة ادخال عدد كبير من القيم مثلاً ٥٠ قيمة فالطريقة السابقة لا تصلح ويجب استخدام الأوامر التكرارية في هذه الحالة كما بالشكل التالي:

```
integer, dimension(50) :: A
```

```
integer::i
```

```
do i=1,50
```

```
write(*,*) 'Enter a New Entry'
```

```
read(*,*) A(i)
```

```
end do
```

ثانياً: المصفوفة ثنائية البعد

| Array Name | index | 1 | 2 |
|------------|----------|---------|---------|
| | 1 | <value> | <value> |
| | 2 | <value> | <value> |
| | 3 | ... | ... |
| | | ... | ... |
| | | <value> | <value> |
| | <i>n</i> | <value> | <value> |

تتكون هذه المصفوفة من مجموعة من الأعمدة والصفوف مرتب بها القيم كما هو موضح بالشكل التالي:

ويكون لكل قيمة ترتيب مثل $A(1,1)$, $A(1,2)$, $A(2,2)$ وهكذا حيث يعبر الرقم الأول عن رقم الصف والرقم الثاني عن رقم العمود كما هو موضح بالشكل التالي:

| Array Name | index | 1 | 2 |
|------------|----------|-------------------|-------------------|
| | 1 | arr(1,1) | arr(1,2) |
| | 2 | arr(2,1) | arr(2,2) |
| | 3 | arr(3,1) | arr(3,2) |
| | | ... | ... |
| | | ... | ... |
| | <i>n</i> | arr(<i>n</i> ,1) | arr(<i>n</i> ,2) |

تعريف المصفوفة:

```
type, dimension(extent,extent) :: name1
```

وكذلك مثل مصفوفة الوحدة يمكن للمصفوفة ان تعرف بشكل استاتيكي أ،
بشكل ديناميكي كالتالى:

```
type, dimension(:,:),allocatable :: name
```

```
integer :: status variable
```

```
allocate(name (dimension,dimension), stat=status variable)
```

Ex.

```
integer, dimension(:,:), allocatable :: nums2
```

```
integer :: allstat
```

```
allocate(nums2(100,100), stat=allstat)
```

تعيين قيم المصفوفة

يمكن تعيين بعض القيم فى المصفوفة بالشكل التالى:

```
table1(1,1) = 121.3
```

```
table1(10,5) = 98.125
```

كما يمكن استخدام الأوامر التكرارية فى إدخال قيم المصفوفة مثل:

```
do i = 1, 10
  do j = 1, 10
    tmptable(i,j) = 0.0
  end do
end do
```

في هذا المثال تم تعيين كل قيم
المصفوفة بقيمة تساوي صفر.

Ex.

```
integer, dimension(10,10):: nums2
```

```
Do i=1, 10
```

```
    Do j= 1, 10
```

```
        Write(*,*) ' Enter element ',i,',',j
```

```
        Read(*,*)nums2 (i,j)
```

```
    End do
```

```
End do
```

في هذا المثال تم تعريف مصفوفة (10،10) وتم ادخال القيم من المستخدم باستخدام
Nested Loop مع كتابة رقم العنصر في كل مره يطلب البرنامج إدخاله.

الفصل السابع
البرامج الفرعية
(Subprograms)

البرامج الفرعية (Subprograms)

جميع الأمثلة السابق ذكرها في عذا الكتاب كانت عبارة عن برامج صغيرة وبسيطة ولكن عند التعامل مع البرامج الكبيرة يصبح الأمر أكثر صعوبة ويحتاج إلى الكثير من التقسيم لذلك يلجأ المبرمج إلى استخدام البرامج الفرعية أو بمعنى آخر تقسيم البرنامج إلى برامج أو أجزاء مصغرة يقوم كل جزء في البرنامج بوظيفة معينة تتعاون جميعها من أجل تنفيذ الوظيفة الأساسية للبرنامج ويوجد نوعين من هذه البرامج المصغرة سوف نتعرف عليهما في هذا الجزء وهما:

Subroutine -

Function -

الصورة العامة للبرنامج

كل من function و subroutine يمكن كتابتهما داخل أو خارج البرنامج أي بعد أو قبل جملة End Program كما هو موضح في الصورة التالية وسوف نتطرق لكل أمر بالتفصيل على حده.

```
program <name>
```

```
<declarations>
```

```
<program statements>
```

```
contains
```

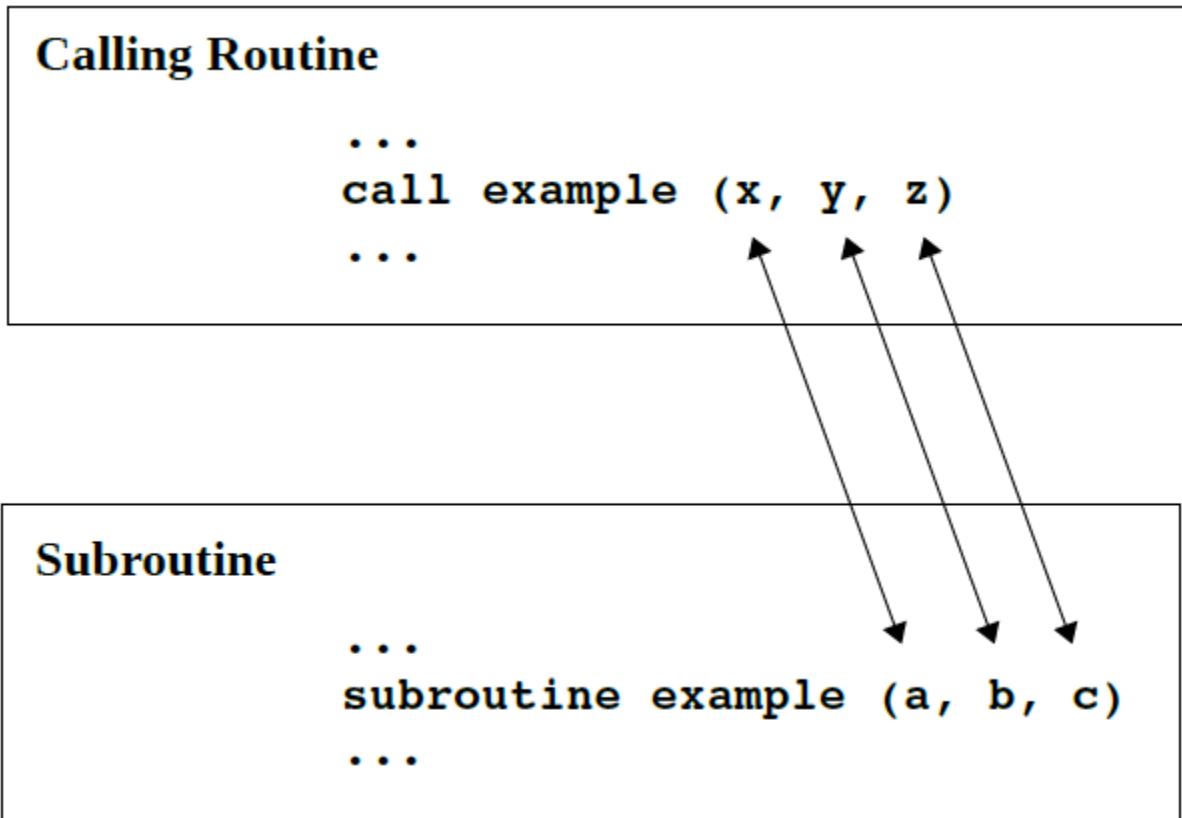
```
<internal functions or subroutines>
```

```
end program <name>
```

```
<external functions or subroutines>
```

متغيرات البرامج المصغرة

قبل استخدام البرامج المصغرة يجب أولاً تعريفها قبل استدعائها في البرنامج فكما ذكرنا سابقاً أنه يمكن تقسيم البرنامج باستخدامها ليقوم كل جزء بوظيفة معينة ثم يقوم كل جزء أو برنامج مصغر بإرسال نتائج تنفيذه إلى الجزء الرئيسي من البرنامج حيث يتم استخدام هذه النتائج لإخراج الناتج الرئيسي وعلى هذا يجب أن ترتبط المتغيرات في البرنامج المصغر مع المتغيرات في الجزء الأساسي في البرنامج وتوضح الصورة التالية بعض المتغيرات في الجزء الأساسي وهي x, y, z والتي تم التعبير عنها داخل البرنامج المصغر بـ a, b, c مع العلم أنه يجب كتابتها بنفس الترتيب فـ a يعود على x وهكذا وكذلك يمكن استخدام نفس الرموز كمتغيرات في البرنامج المصغر.



وتسمى المتغيرات في البرنامج المصغر بالمتغيرات الوهمية حيث يتم استخدامها لحساب جزء معين من البرنامج ثم تنتهي وظيفتها بخروج النتائج للبرنامج الأساسي.

أولاً : Subroutine

هي عبارة عن برنامج مصغر يقوم باستقبال بعض المعلومات كمدخلات يقوم على أساسها بالعودة للبرنامج الأساسي بنتيجة أو مجموعة من النتائج ويتم كتابته بهذا الشكل:

```
subroutine <name> ( <arguments> )  
<declarations>
```

```
<body of subroutine>
```

```
return  
end subroutine <name>
```

Name : هو اسم البرنامج المصغر والذي يستخدم لإستدعاء البرنامج المصغر داخل البرنامج الأساسي.

Arguments : المتغيرات الوهمية المستخدمة داخل البرنامج المصغر.

Decleration : الإعلان عن المتغيرات داخل البرنامج المصغر.

Body of subroutine : جسم البرنامج .

ملاحظات :

- يتم استدعاء الـ Subroutine داخل البرنامج عن طريق جملة Call

Call Subroutine name (arguments)

- عند كتابة البرنامج المصغر داخل البرنامج أي قبل جملة End program يتم اسباق الكود بكلمة Contains كما هو موضح في المثالي التالي:

مثال توضيحي ١

```
program subExample
implicit none
real :: x1=4.0, y1=5.0, z1=6.0, sum1, ave1
call sumAve(x1, y1, z1, sum1, ave1)
write (*,'(a,f5.1,3x,a,f5.1)') "Sum=", sum1, &
"Average=", ave1
contains
subroutine sumAve (a, b, c, sm, av)
real, intent(in) :: a, b, c
real, intent(out) :: sm, av
sm = a + b + c
av = sm / 3.0
return
end subroutine sumAve
end program subExampI
```

المثال التالي يمثل برنامج لحساب مجموع ومتوسط ثلاثة أعداد تم الإستعانه فيه ببرنامج مصغر يقوم بحساب المجموع والمتوسط ثم يقوم بأرسال النتائج إلى البرنامج الأساسي ليتم عرضها على المستخدم في تنسيق معين .

لاحظ :

عند الإعلان عن المتغيرات في الـ subroutine تم استخدام كلمة intent والتي لها ثلاث قيم وهي (in – out – inout) ويتم استخدامها لتحديد نوع المتغير هل هو دخل أم خرج أم دخل وخرج حيث في البرنامج الأرقام الثلاثة a,b,c عبارة عن مدخلات للبرنامج و كل من av, sm مخرجات تعبر عن المجموع والمتوسط .

مثال تطبيقي ١

في هذه المثال سوف نستخدم subroutine تدعى Swap من أجل تبديل قيم كل من المتغيريين a,b .

```
program calling_func
implicit none
  real :: a, b
  a = 2.0
  b = 3.0
  Print *, "Before calling swap"
  Print *, "a = ", a
  Print *, "b = ", b
call swap(a, b)
  Print *, "After calling swap"
  Print *, "a = ", a
  Print *, "b = ", b
end program calling_func
subroutine swap(x, y)
implicit none
  real :: x, y, temp
  temp = x
  x = y
  y = temp
end subroutine swap
```

لاحظ عند عمل run للبرنامج تم تبديل قيم كل من a , b

```
Before calling swap
a = 2.00000
b = 3.00000
After calling swap
a = 3.00000
b = 2.00000
Press RETURN to close window...
```

مثال تطبيقي ٢

في هذا المثال سوف يتم حساب المميز عن طريق subroutine تسمى intent_example ثم يتم العودة بالقيمة إلى البرنامج لطباعتها.

```
program calling_func
implicit none
  real :: x, y, z, disc
  x= 1.0
  y = 5.0
  z = 2.0
  call intent_example(x, y, z, disc)
  Print *, "The value of the discriminant is"
  Print *, disc
end program calling_func
subroutine intent_example (a, b, c, d)
implicit none
  ! dummy arguments
  real, intent (in) :: a
  real, intent (in) :: b
  real, intent (in) :: c
  real, intent (out) :: d
  d = b * b - 4.0 * a * c
end subroutine intent_example
```

لاحظ أن المتغيرات a,b,c تم تعيينها كدخول في البرنامج المصغر عن طريق جملة intent وكذلك d تم تعيينها كخرج من البرنامج.

عند عمل run للبرنامج يتم عرض البرنامج بالشكل التالي:

```
The value of the discriminant is
17.0000
Press RETURN to close window...
```


ثانياً: Function

هى عبارة عن حالة خاصة من Subroutine حيث يكون لها خرج واحد أو ناتج واحد وتستخدم لإجراء نفس العمليات على قيم مختلفة لنفس المتغير أ، مجموعة من المتغيرات.

تعريف الـFunction

عند تعريفها تأخذ أحد انواع البيانات ويتم تعريفها كما هو موضح بالصورة التالية:

```
<type> function <name> ( <arguments> )  
<declarations>  
  
    <body of function>  
  
    <name> = expression  
    return  
end function <name>
```

وهى لا تختلف فى تعريف الـsubroutine إلا فى أنها يتم تعريفها بأحد أنواع البيانات باعتبار أن لها قيمة واحدة أو خرج واحد حيث يتم تعريف كلاً من الدالة والمتغيرات الوهمية.

ملاحظات:

- عند كتابة الكود تجاهل علامات < و >
- يمكن تعريف الدالة داخلها بهذه الطريقة

```
function area_of_circle (r)  
implicit none  
    real :: area_of_circle  
    real :: r  
end function area_of_circle
```

أو في البداية بهذه الطريقة

```
Real function area_of_circle (r)
implicit none
  real :: r
end function area_of_circle
```

حيث في كلا المثالين `area_of_circle` هي عبارة عن دالة من النوع `real` في متغير واحد وهو نصف القطر `r`

مثال تطبيقي ١

في هذا المثال سوف يقوم البرنامج بحساب مساحة دائرة معلوم نصف قطرها.

```
program calling_func
  real :: a
  a = area_of_circle(2.0)
  Print *, "The area of a circle with radius 2.0 is"
  Print *, a
end program calling_func
! this function computes the area of a circle with radius r
function area_of_circle (r)
! function result
implicit none
  ! dummy arguments
  real :: area_of_circle
  ! local variables
  real :: r
  real :: pi
  pi = 4 * atan (1.0)
  area_of_circle = pi * r**2
end function area_of_circle
```

نتيجة تشغيل البرنامج

```
The area of a circle with radius 2.0 is
12.5664
Press RETURN to close window...
```

الوحدات (Modules)

عند كتابة البرامج الكبيرة والتي تحتوى على عدد كبير من ال-Subroutines أ، ال-Functions تستخدم الوحدات (Modules) لتحزيم البرامج المصغرة أى أنها تكون بداخلها كما أن المتغيرات التى يتم تعريفها داخل الوحدة (Module) يمكن استخدامها فى أى جزء من البرنامج أ، فى أى Subroutine داخل البرنامج وذلك على خلاف المتغيرات التى توجد فى كل برنامج مصغر والتي تستخدم فقط داخل البرامج المصغرة وتتميز الوحدات بالآتى:

- 1- يمكن استخدامها كملف مصدري يتم استخدام بياناته أو متغيراته فى كل أجزاء البرنامج.
- 2- تستخدم فى تجميع البرامج المصغرة بداخلها ويجب قبل كتابة أى (Function أو subroutine) داخل ال-Module أن يسبقها كلمة **Contains** .
- 3- يتم كتابة ال-Module قبل البرنامج الرئيسي .
- 4- يتم استدعاء ال-Module بكلمة **Use**

الصورة العامة لكتابة ال-Module

أولاً ال-Module

```
module name1
declarations
contains
subroutine and/or function definitions
end module name1
```

ثانياً البرنامج الرئيسي

```
Program name2
Use name1
.....
End Program name2
```

مثال تطبيقي ١

في المثال التالي سوف نقوم بحساب مساحة دائرة ورفع e لأس X

```
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 :: x, ePowerx, area, radius
  x = 2.0
  radius = 7.0
  ePowerx = e ** x
  area = pi * radius**2
call show_consts()
print*, "e raised to the power of 2.0 = ", ePowerx
print*, "Area of a circle with radius 7.0 = ", area
end program module_example
```

شرح البرنامج

فى البرنامج السابق تم استخدام Module لتعريف المتغيرات أو الثوابت الرئيسية ليتم استخدامها فى كل أجزاء البرنامج وهى Pi و e .

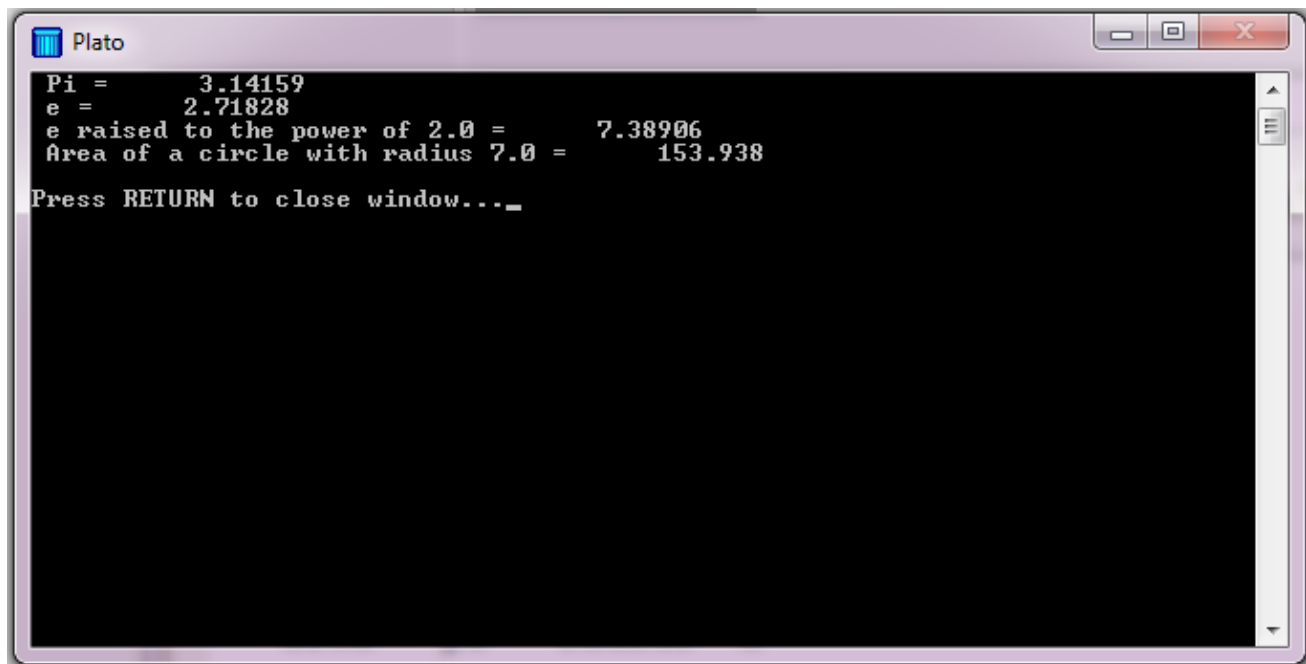
```
real, parameter :: pi = 3.1415926536
real, parameter :: e = 2.7182818285
```

ولإستخدام هذه الثوابت فى البرنامج الرئيسي تم استدعاء الـModule باستخدام كلمة use كما هو موضح.

use constants

وبذلك أمكن استخدام كل من Pi و e داخل البرنامج الرئيسي وفى حالة حذف جملة use من البرنامج الرئيسي لن تتم العمليات الحسابية وسيقوم البرنامج باعتبار كل من pi و e متغيرات لم يتم تعريفها وسوف يقوم بإظهار رسالة خطأ عند عمل Run للبرنامج وبذلك لن يعمل البرنامج.

وعند عمل Run للبرنامج السابق سوف تظهر النتيجة التالية على الشاشة.



```
Plato
Pi =      3.14159
e =      2.71828
e raised to the power of 2.0 =      7.38906
Area of a circle with radius 7.0 =      153.938
Press RETURN to close window..._
```

ملاحظات:

يمكن التحكم في المتغيرات والثوابت التي يتم تعريفها في الـ Module وتعريفها كأحد نوعين إما Private أو Public وفي حالة تعريفها كـ Private لا يمكن استخدامها خارج الـ Module أي أنها تكون متغيرات خاصة بالـ Module نفسه أما في حالة Public يمكن استخدامها في جميع أجزاء البرنامج ويتم تعريفها كـ Private كالتالي:

```
real, parameter,private :: pi = 3.1415926536
```

وفي البرنامج السابق إذا تم تعريف كل من Pi و e على أنها private سوف تظهر رسالة الخطأ التالية عند تشغيل البرنامج.

Output

```
: warning 298 - Variable E has been used without being given an initial value
: warning 298 - Variable PI has been used without being given an initial value
: error 283 - PI must appear in a type declaration because IMPLICIT NONE has been used
: error 283 - E must appear in a type declaration because IMPLICIT NONE has been used
```

وفي حالة عدم تحديد نوع المتغيرات سوف يقوم البرنامج باعتبارها Public .

تمارين محلولة

1. In a certain application, a closed system undergoes a thermodynamic process and it is required to calculate the work and heat transfer during this process. **Write** a Fortran program that asks the user to specify the type of the process and the corresponding inputs, then calculates w and q as following:

For isothermal process: $q=w=R.T_1.\ln(P_1P_2/)$

For isentropic process: $q=0$,& $w=c_v*(T_1-T_2)$

For isobaric process: $q=c_p*(T_2-T_1)$ & $w=R*(T_2-T_1)$

For isochoric process: $w=0$,& $q=c_v*(T_2-T_1)$

For polytropic process: $w=R_{n-1}*(T_1-T_2)$,& $q=c_v*(T_2-T_1)+w$

المطلوب : برنامج لحساب الشغل وكمية الحرارة لنظام مغلق حيث يقوم المستخدم بإدخال نوع الإجراء داخل النظام والحالة الخاصة به من ضغط ودرجة حرارة ثم يقوم البرنامج بحساب الشغل وكمية الحرارة من خلال القوانين السابقة.

Solution

```

program state
implicit none
الإعلان عن المتغيرات !
integer ::I,n
real ::t1,t2,p1,p2,q,w,cp,cv,R
إعلام المستخدم بأرقام كل حالة لسهولة الإدخال!
print*,'If the system isothermal enters 1'
print*,'If the system isentropic enters 2'
print*,'If the system isobaric enters 3'
print*,'If the system isochoric enters 4'
print*,'If the system polytropic enters 5'
read(*,*) I

```


تحديد نوع الإجراء أولاً من خلال رقمه!

ادخال البيانات المطلوبة للحسابات!

القيام بالحسابات!

```
select case (I)
```

```
case(1)
```

```
print*, 'Enter T1,P1,P2,R'
```

```
read(*,*)T1,P1,P2,R
```

```
w=R*T1*log(P1/P2)
```

```
q=w
```

```
case(2)
```

```
print*, 'Enter T1,T2,cv'
```

```
read(*,*)T1,T2,cv
```

```
q=0
```

```
w=cv*(T1-T2)
```

```
case(3)
```

```
print*, 'Enter T1,T2,R,cp'
```

```
read(*,*)T1,T2,R,cp
```

```
q=cp*(T2-T1)
```

```
w=R*(T2-T1)
```

```
case(4)
```

```
print*, 'Enter T1,T2,cv'
```

```
read(*,*)T1,T2,cv
```

```
w=0
```

```
q=cv*(T2-T1)
```

```
case(5)
```

```
print*, 'Enter T1,T2,cv,R,n'
```

```
read(*,*)T1,T2,cv,R,n
```

```
w=(R/(n-1))*(T1-T2)
```

```
q=cv*(T2-T1)+w
```

```
case default
```

قى حالة ادخال رقم غير معروف!

يظهر للمستخدم رسالة خطأ!

```
print*, 'Wrong Entry'
```

```
end select
```

طباعة الناتج النهائي للمستخدم!

```
Print*, 'Q = ',q, ' kj/kg', ' & w = ',w, ' kj/kg'
```

```
pause
```

```
end program state
```

2. Write a Fortran program that reads a Number then calculates its factorial as follows;

For $n=0$: $n!=1$

For $n>0$: $n!=n(n-1)(n-2)\dots\dots(3)(2)(1)$

المطلوب : حساب مضروب رقم يقوم المستخدم بإدخاله.

Solution

```
program fact
implicit none
integer::f,i,n
print*,'Enter a number to calculate it is factorial'
read(*,*)n
if (n==0) print*,'Factorial of ',n,'= 1 '
if (n>0)then
  f=n
  Do i=1,n-1
    f=i*f
  end do
end if
print*,'The factorial of',n,'=',f
pause
endprogram fact
```

3. A digital thermometer reads the temperature of human body, then displays the status (Normal, Slightly above Normal, Too High, Below Normal or Too Low). The following piece of code has been written for this task, would it work properly?

```
IF (temp==37) THEN
WRITE(*,*) 'Normal'
ELSE IF (temp>37) THEN
WRITE(*,*) 'Slightly Above Normal'
ELES IF (temp>39) THEN
WRITE(*,*) 'Too High'
ELSE IF (temp<37) THEN
WRITE(*,*) 'Slightly Below Normal'
ELES IF (temp<35) THEN
WRITE(*,*) 'Too Low'
END IF
```

المطلوب : إكتشاف الأخطاء فى الكود السابق ثم إعادة كتابة البرنامج مع تصحيح الأخطاء.

Solution

```
program temp
implicit none
integer::t
write(*,*) 'Enter your Temperature,please.'
read(*,*)t
if (t==37) then
    print*, 'Normal Temperature'
```

```
else if ((t>37).and.(t<39)) then
    print*, 'Slightly above Normal'
else if (t>39)then
    print*, 'Too High'
else if ((t<37).and.(t>35))then
    print*, 'Slightly below Normal'
else if (t<35)then
    print*, 'Too low'
Endif
pause
endprogram temp
```

الكود المعطى فى السؤال غير قابل للتطبيق بسبب احتواءه على اخطاء منطقية حيث فى حالة درجة الحرارة أكبر من ٣٩ يتحقق شرطين فى نفس الوقت الشرط لدرجة الحرارة أكبر من ٣٧ وكذلك لدرجة الحرارة أكبر من ٣٩ وفى هذه الحالة لا يستطيع البرنامج ادراك أى الأمرين صحيح ويجب كتابة النتيجة على أساسه ونفس المشكلة أيضاً فى حالة درجتى الحرارة ٣٥ و ٣٧ .

4. Write a Fortran program to display the sum and arithmetic average of the first 100 elements in the following series;

1, 2, 4, 8, 16

المطلوب : برنامج يقوم بحساب المجموع والمتوسط الحسابي لأول ١٠٠ رقم في السلسلة التالية ١،٢،٤،٨،١٦

Solution

```
program fact
implicit none
integer::i,j,n
real::sum
i=0
n=1
sum=0.0
  do j=1,100
    sum=sum+n
    n=2*n
    i=i+1
  enddo
  print*,'The Average = ',sum/i
pause
endprogram fact
```

5. Write a Fortran program that calculate the Arithmetic Mean, Geometric Mean, Root-Mean-Square (RMS) Average and Harmonic Mean for a set of numbers as follows;

$$\text{Arithemtic Mean} = \frac{1}{N} \sum_{i=1}^N x_i$$

$$\text{Geometric Mean} = \sqrt[N]{x_1 x_2 x_3 \dots x_N}$$

$$\text{RMS Average} = \sqrt{\frac{1}{N} \sum_{i=1}^N x_i^2}$$

$$\text{Harmonic Mean} = \frac{N}{\frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{x_3} + \dots + \frac{1}{x_N}}$$

المطلوب : حساب التعبيرات الرياضية الموضحة.

Solution

```

program calculations
implicit none
!declaring variables
integer::n,i,j,k,l,m
integer,dimension(100)::x
real::ar,ge,rms,hm,sum,pro,sum2,sum3
sum=0
pro=1
sum2=0
sum3=0
print*,'enter the number of values (n)'
read(*,*)n
do i=1,n
  print*,'Enter X',i
  read(*,*)x(i)
end do

```

```
!Arithmetic Mean
```

```
Do j=1,n  
    sum=sum+x(j)  
end do  
ar=sum/n
```

```
!Geometric Mean
```

```
Do k=1,n  
    pro=pro*x(k)  
end do  
ge=pro*(1/n)
```

```
!RMS Average
```

```
Do l=1,n  
    sum2=sum2+x(l)**2  
end do  
rms=(sum2/n)**0.5
```

```
!Harmonic Mean
```

```
Do m=1,n  
    sum3=sum3+(1/x(m))  
end do  
hm=n/sum3  
print*,'Arithmetic Mean =',ar  
print*,'Geometric Mean =',pro  
print*,'RMS Average =',rms  
print*,'Harmonic Mean =',hm  
pause  
end program calculations
```


6. The atmosphere surrounding the earth is divided into two layers (Troposphere and Stratosphere):

1-**Troposphere** (from 0 m to 11000 m): where air temperature T varies with height (as the altitude increases, the temperature decreases) according to:

$$T = T_o - \alpha z$$

In addition, the air pressure P and density ρ can be calculated according to:

$$P = P_o \left(\frac{T}{T_o} \right)^{\frac{g}{\alpha R}}$$
$$\rho = \frac{P}{RT}$$

Where:

T [K], [Pa] and ρ [kg/m³] are temperature, pressure and density of air at altitude z [m].

α = Temperature lapse rate (the rate of decrease of temp. with altitude) =0.0065K/m

R = Air constant =287J/kg K

g = Acceleration of gravity =9.81m/s²

T_o = Temperature of air at sea level =288.15 K

P_o = Pressure of air at sea level =101325 Pa

2- **Stratosphere** (from 11000 m to 20000 m): where temperature T remains constant as the altitude increases, and equals to =-56.5 °C=216.5 K

In addition, the air pressure P and density ρ can be calculated according to:

$$P = P_o \left(\frac{T_o - \alpha z_1}{T_o} \right)^{\frac{g}{\alpha R}} \times e^{-\left(\frac{g(z-z_1)}{R(T_o - \alpha z_1)} \right)}$$

$$\rho = \frac{P}{RT}$$

Where

T [K], [Pa] and ρ [kg/m³] are temperature, pressure and density of air at altitude z [m].

z_1 = Height of troposphere layer = 11000 m

Write a computer program to calculate T [°C],[bar] and ρ [kg/m³] at any altitude z [m] and mention at any layer this altitude lies.

Define α, T_o, P_o and z_1 as PARAMETER in variables declaration section.

المطلوب : كتابة برنامج يقوم فيه المستخدم بإدخال الارتفاع ويقوم البرنامج بتحديد في أى طبقة يكون هذا الارتفاع بالإضافة إلى حساب الضغط والدرجة الحرارة وكثافة الهواء عند هذه النقطة.

Solution

```

program fact
implicit none
تعريف المتغيرات والثوابت!
real,parameter::alpha=0.0065,r=287
real,parameter:: g=9.81,t0=288.15,p0=101325,z1=11000
real::z,roh,p,t

```

```

character(len=15)::place
print*, 'Enter The Hight, Please.'
read(*,*) z
تحديد الطبقة واجراء الحسابات!
if((z>0).and.(Z<11000))then
    t=t0
    p=p0*(t/t0)**(g/(alpha*r))
    roh=p/(r*t)
    place='Troposphere'
elseif ((z>=11000).and.(z<=20000))then
    t=216.5
    p=p0*((t0-alpha*z1)/t0)**(g/(alpha*r))* &
    exp((g*(z1-z))/(r*(t0-alpha*z1)))
    roh=p/(r*t)
    place='Stratosphere'
endif
طباعة النتائج!
print*, 'The layer is :', place
print*, 'The pressure = ', p, ' pa'
print*, 'The Temperature = ', t, ' K'
print*, 'The Density = ', roh, ' Kg/cubic Meter'
pause
endprogram fact

```

7. Write a program which read two matrices from two files and print their multiplication in a new file. (First make sure that the dimensions of the matrices are compatible)

المطلوب : كتابة برنامج يقوم بقراءة مصفوفتين من ملفين وطباعة حاصل ضربهما في ملف آخر.

ملاحظات

- عند ضرب المصفوفات يجب أن يتساوى عدد أعمدة المصفوفة الأولى مع عدد صفوف المصفوفة الثانية.
- المصفوفة الناتجة يكون أبعادها عدد صفوف المصفوفة الأولى وعدد أعمدة المصفوفة الثانية.

$$\begin{array}{c} \text{الناتج} \\ \boxed{A(5,6) * B(6,4) = C(5,4)} \\ \text{الشرط} \end{array}$$

Solution

```
program matrix_multiplication
  implicit none
  تعريف ثلاث مصفوفات دون اعطاءهم قيمة مبدئية !
  real, allocatable, dimension(:, :)::a, b, c
  integer::i, j, k
  integer::status
  integer::a_row, a_col, b_row, b_col
  real::sum
```

فتح الملف الذي يحتوى على المصفوفة الأولى !

```
open(unit=15,file='inp_a.txt',status='old',iostat=status)
if(status/=0)then
  write(*,*)'error opening file inp_a.txt'
  stop
end if
```

فتح الملف الذى يحتوى على المصفوفة الثانية !

```
open(unit=15,file='inp_b.txt',status='old',iostat=status)
if(status/=0)then
  write(*,*)'error opening file inp_b.txt'
  stop
end if
```

قراءة أبعاد المصفوفات من الملفات !

```
read(15,*)a_row,a_col
read(16,*)b_row,b_col
```

اختبار شروط ضرب المصفوفات وهو أن يكون عدد أعمدة الأول يساوى عدد صفوف الثانى !
فى حالة عدم توافق الأبعاد تظهر رسالة خطأ وينتهى البرنامج !

```
if(a_col /=b_row )then
  write(*,*)'incompatible matrix sizes'
  stop
end if
```

فى حالة توافق الأبعاد يتم تعيين أبعاد المصفوفات !

```
allocate(a(a_row,a_col),b(b_row,b_col),stat=status)
if(status /=0) then
  write(*,*)'Failed to allocate memory for matrices A,B'
  stop
end if
```

```
allocate(c(a_row,b_col),stat=status)
if(status /=0) then
    write(*,*)'Failed to allocate memory for matrices A,B'
    stop
end if
```

قراءة عناصر المصفوقتين !

```
read(15,*)((a(i,j),j=1,a_col),i=1,a_row)
read(16,*)((b(i,j),j=1,b_col),i=1,b_row)
```

ضرب المصفوفات باستخدام سلسلة من الأوامر التكرارية !

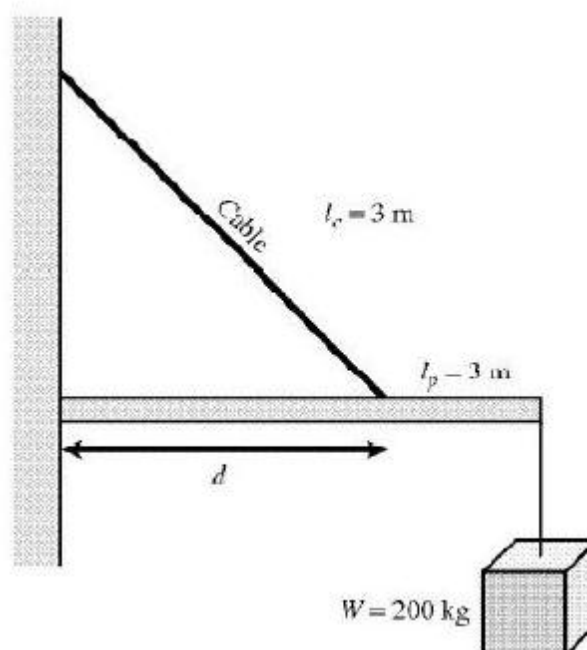
```
do i=1,a_row
    do k=1,b_col
        sum=0.0
        do j=1,a_col
            sum=sum+a(i,j)*b(j,k)
        end do
        c(i,k)=sum
    end do
end do
```

كتابة الناتج النهائي !

```
write(*,*) 'The Result is an array',a_row,'x',b_col
write (*,*) ((c(i,j),j=1,b_col),i=1,a_row)
end program matrix_multiplication
```

8. A 200-kilogram object is to be hung from the end of a rigid 3-meter horizontal pole of negligible weight, as shown. The pole is attached to a wall by a pivot and supported by a 3-meter cable, which is attached to the wall at a higher point. The tension force on this cable is given by;

$$T = \frac{W \cdot l_p \cdot l_c}{d \sqrt{l_p^2 - d^2}} \cdot g$$



Write a program to determine the distance d at which the cable should be attached in order to minimize the tension on the cable. To do this, the program should calculate the tension on the cable for $d = 0.5 \text{ m}$ to $d = 2.8 \text{ m}$ (with 0.1 m intervals), then locate the position that produces the minimum tension.

المطلوب : حساب المسافة d التي يكون عندها أقل شد وحساب الشد عند هذه المسافة.

Solution

```
PROGRAM calc_tension
IMPLICIT NONE
```

الإعلان عن المتغيرات !

```
REAL :: dist ! المسافة حتى نقطة الاتصال (m)
INTEGER :: i ! متغير العداد التكرارى
REAL :: lc = 3. ! طول الحبل (m)
REAL :: lp = 3. ! طول العمود (m)
REAL :: saved_dist ! مسافة التوصيل المحفوظة
REAL :: saved_tension = 999999. ! الشد المحفوظ
REAL :: tension ! الشد فى الحبل
REAL :: weight = 200. ! وزن الجسم (kg)
```

حساب الشد عند كل نقاط الإتصال بين ١ إلى ٧ قدم !

```
DO i = 5, 28
dist = REAL(i) / 10.
```

حساب الشد !

```
tension = weight * lc * lp / ( dist * SQRT(lp**2 -
dist**2) )
```

كتابة النتائج !

```
WRITE (*,*) 'dist = ', dist, ' tension = ', tension
```

اختبار أقل قيمة للشد !

```
IF ( tension < saved_tension ) THEN
saved_tension = tension
saved_dist = dist
```

```
END IF
```

```
END DO
```

كتابة أقل شد على الشاشة !

```
WRITE (*,*) 'Minimum at d = ', saved_dist, ' tension = ', saved_tension
```

```
END PROGRAM calc_tension
```


9. Write a program to calculate the volume of one Mole of an ideal gas as a function of pressure as its pressure varies from 1Kpa to 1001Kpa steps 100Kpa.

Where: $R = 8.314 \text{ kPa/mol.K}$

المطلوب : حساب حجم واحد مول من الغاز المثالي كدالة في الضغط عندما يتغير الضغط من ١ إلى ١٠٠١ كيلو باسكال مع مقدار تغير ١٠٠ كيلو باسكال.

Solution

```
PROGRAM ideal_gas1
IMPLICIT NONE
! الثوابت
REAL,PARAMETER :: R = 8.314
! المتغيرات
INTEGER :: i
REAL :: n = 1.0 ! عدد المولات (mol)
REAL :: p ! الضغط (kPa)
REAL :: t ! درجة الحرارة (K)
REAL :: v ! الحجم (L)
! إدخال درجة الحرارة بواسطة المستخدم
WRITE (*,*) 'Enter gas temperature in kelvins:'
READ (*,*) t
! حساب الحجم كدالة في الضغط
DO i = 1, 1001, 100
! تعيين الضغط أولاً
p = i
! حساب الحجم
```

```
v = n * R * t / p
```

كتابة الحجم عند الضغط المناظر !

```
WRITE (*,*) 'Pressure = ', p, ', Volume = ', v
```

```
END DO
```

```
END PROGRAM ideal_gas1
```

10. Write a program which solve a system of equations using Gauss-Jordan method.

المطلوب : حل نظام من المعادلات الخطية باستخدام طريقة جاوس جوردون عن طريق ادخال المعاملات من ملف خارجي والقيام بالحساب ثم طباعة المعاملات التي تم قراءتها والنتائج على الشاشة.

ولكن قبل البدء في حل هذا المثال يجب أولاً التعرف على طريقة جاوس جوردون وهي طريقة تستخدم لإيجاد حل مجموعة أو منظومة من المعادلات الخطية والمثال التالي يوضح طريقة استخدام هذه الطريقة مع مجموعة (منظومة) المعادلات التالية:

$$\begin{aligned} X_1 + 2X_2 + 3X_3 &= 9 \\ 2X_1 - X_2 + X_3 &= 8 \\ 3X_1 \quad \quad - X_3 &= 9 \end{aligned}$$

طريقة الحل

$$\left[\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 2 & -1 & 1 & 8 \\ 3 & 0 & -1 & 3 \end{array} \right]$$

$$\begin{array}{l} -2r_1 + r_2 \\ \longrightarrow \end{array} \left[\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 0 & -5 & -5 & -10 \\ 0 & -6 & -10 & -24 \end{array} \right]$$

$$\begin{array}{l} -3r_1 + r_3 \\ -1/5 \\ \longrightarrow \end{array} \left[\begin{array}{ccc|c} 1 & 2 & 3 & 9 \\ 0 & 1 & 1 & 2 \\ 0 & -6 & -10 & -24 \end{array} \right]$$

$$\begin{array}{l} -2r_2 + r_1 \\ \longrightarrow \\ 6r_2 + r_3 \end{array} \left[\begin{array}{ccc|c} 1 & 0 & 1 & 5 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & -4 & -12 \end{array} \right]$$

$$\frac{-1}{4}r_3 \Rightarrow \begin{bmatrix} 1 & 0 & 1 & : & 5 \\ 0 & 1 & 1 & : & 2 \\ 0 & 0 & 1 & : & 3 \end{bmatrix}$$

$$-r_3 + r_1 \Rightarrow \begin{bmatrix} 1 & 0 & 0 & : & 2 \\ 0 & 1 & 0 & : & -1 \\ 0 & 0 & 1 & : & 3 \end{bmatrix}$$

$$-r_3 + r_2$$

$$\therefore \begin{bmatrix} x_1 = 2 \\ x_2 = -1 \\ x_3 = 3 \end{bmatrix}$$

∴ مجموعة الحل

ملحوظة: في حالة وجود صفر على الوتر الرئيسي يتم تبديل صفيين وذلك لن يؤثر على النتيجة في شيء.

Solution

البرنامج الرئيسي !

```
program linear_eqn
```

```
implicit none
```

```
integer,parameter::max_size=10
```

```
real*8,dimension(max_size,max_size)::a
```

```
real*8,dimension(max_size)::b
```

```
character(len=20)::file_name
```

```
integer::i,j,n,istat,error
```

فتح اسم الملف الذي يحتوى على المعاملات !

```
write(*,*)'Enter the file name containing the coefficient'
```

```
read(*,*)file_name
```

فتح ملف الإدخال وقراءة المعاملات !

```
open(unit=10,filefile_name,status='old',iostat=istat)
```

فى حالة وجود خطأ فى فتح الملف يقوم البرنامج بعرض رسالة خطأ !

```
if(istat/=)then
```

```
write(*,*)'Failed to open the file :',file_name
```

```
stop
```

```
endif
```

```
read(10,*) n قراءة عدد المعادلات !
```

إذا كان عدد المعادلات أكبر من القيمة المفروضة يعرض البرنامج رسالة خطأ !

```
if (n>max_size) then
```

```
write(*,*) 'no. of equation greater than allowed',max_size
```

ويتم إنهاء البرنامج !

```
stop
```

```
endif
```

! وفي حالة أن المعادلات أقل من أو تساوى أقصى قيمة !

! يتم كتابة المعاملات التي تم إدخالها أولاً على الشاشة !

```
do i=1,n
  write(*,11) (a(i,j),j=1,n),b(i)
enddo
11 format (1x,7(f11.4,2x))
```

! ثم كتابة حل المعادلات

! باستدعاء النتائج التي حصلنا عليها من البرنامج المصغر الذي قام بحل المعادلات

```
call gauss(a,b,max_size,n,error)
if(error==1)then
  write(*,*)'zero pivoting encountered'
  stop
endif
```

! طباعة النتائج

```
write(*,*)'The solution of the equatons is : '
doi=1,n
write(*,12)i,b(i)
enddo
```

```
12 format(1x,'x(',12,')=',f16.6)
end program linear_eqn
```

! برنامج مصغر لحل مجموعة المعادلات

```
subroutine gauss(a,b,ndim,n,error)
implicit none
integer,intent(in)::ndim
real*8,intent(inout),dimension(ndim,ndim)::a
real*8,intent(inout),dimension(ndim)::b
```

```

integer,intent(in)::n
integer,intent(out)::error
! المتغيرات المحلية
real*8,parameter::epsilon=1.0e-6
real*8::temp
mainloop:do irow=1,n ! الكود التكرارى الخاص بالمعادلات
ipeak=irow
max_pivot:do jrow=irow+1,n
if(abs(a(jrow,irow))>abs(a(ipeak,irow)))then
ipeak=irow
endif
! التأكد من وجود صفر على المحور
if(abs(a(ipeak,irow))<epsilon)then
error=1
return
endif
! تغير ترتيب المعادلات
! وذلك فى حالة وجود أصفار على الوتر الرئيسي
swap_eqn:if(ipeak/=irow)then
! تبديل عناصر الصفوف فى حالة وجود صفر على الوتر
swap_loop:do kcol=1,n
temp=a(ipeak,kcol)
a(ipeak,kcol)=temp
enddo swap_loop
temp=b(ipeak)
b(ipeak)=b(irow)

```

```
b(irow)=temp
```

```
endif swap_eqn
```

! عملية الحذف للحصول على أصفار

```
eliminate:do jrow=1,n
```

```
if(jrow/=irow)then
```

```
factor=-a(jrow,irow)/a(irow,irow)
```

```
do kcol=1,n
```

```
a(jrow,kcol)=a(jrow,kcol)+factor*a(irow,kcol)
```

```
enddo
```

```
b(jrow)=b(jrow)+factor*b(irow)
```

```
endif
```

```
enddo eliminate
```

```
enddo mainloop
```

! القسمة على عناصر الوتر

```
divide:do irow=1,n
```

```
b(irow)=b(irow)/a(irow,irow)
```

```
a(irow,irow)=1.0
```

```
enddo divide
```

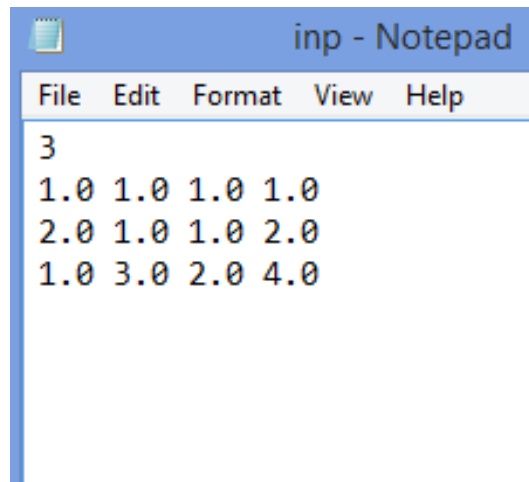
```
error=0 إذا لم تكن المعادلات لها حل وحيد !
```

```
return
```

```
end subroutine gauss
```

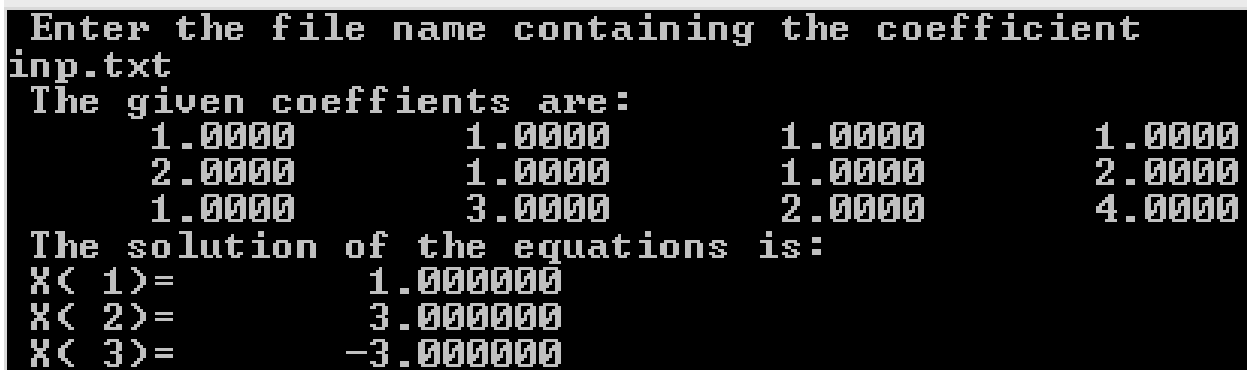

تحقيق الكود

في حالة أن المدخلات كما هي مبينة في الصورة التالية :



```
inp - Notepad
File Edit Format View Help
3
1.0 1.0 1.0 1.0
2.0 1.0 1.0 2.0
1.0 3.0 2.0 4.0
```

سوف تكون النتائج كما بالشكل التالي:



```
Enter the file name containing the coefficient
inp.txt
The given coefficients are:
  1.0000    1.0000    1.0000    1.0000
  2.0000    1.0000    1.0000    2.0000
  1.0000    3.0000    2.0000    4.0000
The solution of the equations is:
X< 1>=    1.000000
X< 2>=    3.000000
X< 3>=   -3.000000
```

ASCII Table

This table lists the American Standard Code for Information Interchange (ASCII) characters or symbols and their decimal numbers.

| Char. | Dec. |
|-------|------|
| | 32 |
| ! | 33 |
| " | 34 |
| # | 35 |
| \$ | 36 |
| % | 37 |
| & | 38 |
| ' | 39 |
| (| 40 |
|) | 41 |
| * | 42 |
| + | 43 |
| , | 44 |
| - | 45 |
| . | 46 |
| / | 47 |
| 0 | 48 |
| 1 | 49 |
| 2 | 50 |
| 3 | 51 |
| 4 | 52 |
| 5 | 53 |
| 6 | 54 |
| 7 | 55 |
| 8 | 56 |
| 9 | 57 |
| : | 58 |
| ; | 59 |
| < | 60 |
| = | 61 |
| > | 62 |
| ? | 63 |

| Char. | Dec. |
|-------|------|
| @ | 64 |
| A | 65 |
| B | 66 |
| C | 67 |
| D | 68 |
| E | 69 |
| F | 70 |
| G | 71 |
| H | 72 |
| I | 73 |
| J | 74 |
| K | 75 |
| L | 76 |
| M | 77 |
| N | 78 |
| O | 79 |
| P | 80 |
| Q | 81 |
| R | 82 |
| S | 83 |
| T | 84 |
| U | 85 |
| V | 86 |
| W | 87 |
| X | 88 |
| Y | 89 |
| Z | 90 |
| [| 91 |
| \ | 92 |
|] | 93 |
| ^ | 94 |
| _ | 95 |

| Char. | Dec. |
|-------|------|
| ` | 96 |
| a | 97 |
| b | 98 |
| c | 99 |
| d | 100 |
| e | 101 |
| f | 102 |
| g | 103 |
| h | 104 |
| i | 105 |
| j | 106 |
| k | 107 |
| l | 108 |
| m | 109 |
| n | 110 |
| o | 111 |
| p | 112 |
| q | 113 |
| r | 114 |
| s | 115 |
| t | 116 |
| u | 117 |
| v | 118 |
| w | 119 |
| x | 120 |
| y | 121 |
| z | 122 |
| { | 123 |
| | 124 |
| } | 125 |
| ~ | 126 |
| | 127 |

المراجع العلمية

- 1- Ed Jorgensen-Introduction to Programming using Fortran 95/2003/2008-Version 3.0.29- December, 2016
- 2- Stephen Chapman-Fortran 95_2003 for Scientists and Engineer -Third Edition
- 3- <http://www.tutorialspoint.com/fortran>
- 4- <https://en.wikipedia.org/wiki/Fortran>