

تمارين البرمجة بفورتران

السنة الأولى
ماستر فيزياء
2022-2021

Becer Zoubir

قسم الفيزياء
كلية العلوم الدقيقة
جامعة الشهيد حمه لخضر بالوادي

كتابة هذه البرامج وتنفيذها تحتاج الى محرر نصوص ومترجم. توجد العديد من الادوات التي تساعدك على كتابة البرامج بشكل منظم والعديد منها تحوي مكتبات اللغة ومترجمات compilers وما عليك فقط هو كتابة البرنامج ثم ترجمته وتنفيذه وتتميز هذه الادوات بسهولة اكتشاف الأخطاء. على سبيل المثال نذكر بعض هذه الادوات التي يمكن استخدامها لكتابة البرامج الظاهرة في هذه التمرينات:

Programers's Notepad +gfortran+MinGW Shell, geany+gfortran, Plato, Silver Front, Code Blocks+gfortran, .NetBeens+gcc, VsCode+gfortran, Atom+gfortran, LFortran+Jupyter Notebook

يمكن كذلك استخدام مترجمات سحابية توفرها بعض المواقع للعديد من لغات البرمجة على الخط يمكن استعمالها مع الهواتف الذكية والحواسيب على حد سواء. لا تتطلب هذه المترجمات تنصيب اي مترجم اذ يكفي فقط فتح متصفح على هذه المواقع ثم كتابة البرنامج المرغوب ومن ثم ترجمته وتنفيذه. نعرض هنا بعض الامثلة والتي اجدها مفيدة جدا:

- <https://www.tutorialspoint.com>
- <https://onecompiler.com/fortran>
- <https://www.mycompiler.io>
- <https://www.onlinegdb.com>
- <https://rextester.com>
- <https://ideone.com/l/fortran>

تمرين 1

```
1 program First_Test
2 ! My first program
3 J = 1
4 K = 3
5 L = 2*J + K
6 J = 3*J + 2*L
7 K = K + 2
8 L = J + K + L
9 print*, J, K, L
10 stop
11 end program First_Test
```

نفذ هذا البرنامج في ذهنك وأكتب وضيفة كل سطر بين 2 و 10.

الحل

عند تنفيذ البرنامج يقوم الحاسوب بما يلي:

السطر 2: يتخطى الحاسوب هذا السطر لأنه تعليق.

السطر 3: يخزن الحاسوب القيمة 1 في مكان في الذاكرة اسمه J. يعني اسناد القيمة 1 الى المتغير J: $J \leftarrow 1$

السطر 4: يخزن الحاسوب القيمة 3 في مكان في الذاكرة اسمه K. يعني اسناد القيمة 3 الى المتغير K: $K \leftarrow 3$

السطر 5: يقوم الحاسوب بحساب قيمة العبارة $2*J + K$ ثم يخزن النتيجة في مكان في الذاكرة اسمه L. بمعنى: $2*J + K = 2*1 + 3 = 5$ ثم يخزن النتيجة 5 في L: $L \leftarrow 5$

السطر 6: يقوم الحاسوب بحساب قيمة العبارة $3*J + 2*L$ باستخدام القيم الحالية ل J و L ثم يستبدل القيمة الحالية ل J بالنتائج. بمعنى: $3*J + 2*L = 3*1 + 2*5 = 13$

ثم يسمح الحاسوب القيمة الحالية ل J ويستبدلها بـ 13: $J \leftarrow 13$

السطر 7: يقوم الحاسوب بزيادة القيمة الحالية لـ K بمقدار 2 : $K \leftarrow 5$

السطر 8: يقوم الحاسوب بجمع القيم الحالية لـ J و K و L ثم يستبدل القيمة الحالية لـ L بالنتائج. بمعنى:
 $J + K + L = 13 + 5 + 5 = 23$

ثم يسمح الحاسوب للقيمة الحالية لـ L ويستبدلها بـ 23 : $J \leftarrow 23$

السطر 9: يطبع الحاسوب في الشاشة القيم الحالية لـ J و K و L بالشكل:

13 5 23

السطر 10: تأمر التعليمة Stop الحاسوب بالتوقف.

تمرين 2

```
1  ! Test of sum and product of numbers
2  I=19
3  J=5
4  K=3
5  MS=I+J+K
6  MP=I*J*K
7  print*, I, J, K, MS, MP
8  STOP
9  END
```

إشرح خطوات تنفيذ البرنامج المقابل وحدد ما يعرضه في الشاشة بعد التنفيذ.

الحل

عند تنفيذ البرنامج يقوم الحاسوب بما يلي:

السطر 1: يتخطى الحاسوب هذا السطر لأنه تعليق بسبب وجود علامة التعجب !.

السطر 2: يخزن الحاسوب القيمة 19 في مكان في الذاكرة اسمه I. يعني اسناد القيمة 19 الى المتغير I : $I \leftarrow 19$

السطر 3: يخزن الحاسوب القيمة 5 في مكان في الذاكرة اسمه J. يعني اسناد القيمة 5 الى المتغير J : $J \leftarrow 5$

السطر 4: يخزن الحاسوب القيمة 3 في مكان في الذاكرة اسمه K : $K \leftarrow 3$

السطر 5: يقوم الحاسوب بحساب قيمة العبارة $I + J + K$ على يمين علامة التساوي ثم يخزن النتيجة 27 في مكان في الذاكرة اسمه MS. بمعنى:
 $I + J + K = 27$ ثم يخزن النتيجة 27 في MS : $MS \leftarrow 27$

السطر 6: يقوم الحاسوب بحساب قيمة الجداء $I * J * K$ باستخدام القيم الحالية لـ I و J و K ثم يسند (يخزن) النتيجة 285 للمتغير MP. بمعنى:
 $MP \leftarrow 285$

السطر 7: يطبع الحاسوب في الشاشة القيم المخزنة في المتغيرات I و J و K و MS و MP بهذا الترتيب كما يلي:

19 5 3 27 285

وهذا لمرة واحدة لانه تم اسخدام التعليمة print*، لمرة واحدة في هذا البرنامج.

السطر 8: تأمر التعليمة Stop الحاسوب بالتوقف. لكن وبسبب وجود التعليمة END بعدها يمكن الاستغناء عنها هنا والاكتفاء بـ END فقط.

تمرين 3

اشرح الغرض من هذا البرنامج.

```
1 program Test
2 read*,a,b,c,d,e
3 s=a+b+c+d+e
4 print*,a,b,c,d,e
5 print*,s
6 end program Test
```

تأمر التعليمة `read*,` بهذه الصياغة البرنامج بقراءة بيانات يختارها المستخدم من لوحة المفاتيح ثم تخزينها في المتغيرات الظاهرة بعدها حسب ترتيب ظهورها. هنا `a` ثم `b` ثم `c` ثم `d` ثم `e`. الميزة الأساسية لاستخدام التعليمة `read*,` هي تعميم البرنامج وجعله مستقلا عن بيانات الإدخال وبذلك يمكن استخدامه لأي قيم ادخال يرغب بها المستخدم.

تمرين 4

اشرح الغرض من هذا البرنامج.

```
1 program Area
2 read*,B,H
3 Area=(B*H)/2.0
4 print*,"B=",B,"H=",H,"Area=",Area
5 end program Area
```

تمرين 5

حدد الغرض من البرنامج المقابل.

```
1 program GuessMyAim
2 integer :: I
3 I=1
4 99 write(*,*) I
5 I=I + 1
6 go to 99
7 end program GuessMyAim
```

تأمر التعليمة `go to` الحاسوب بالذهاب الى سطر معلم برقم معين (مثل السطر الرابع المعلم بالرقم 99 في هذا البرنامج) ثم مواصلة تنفيذ التعليمات انطلاقا منه نحو بقية السطور الموالية سطرا بسطر.

تمرين 6

إشرح خطوات تنفيذ البرنامج المقابل وحدد ما يعرضه في الشاشة بعد التنفيذ.

```
1 program GuessWhatIprint
2 N=1
3 31 K=N*N
4 print*,N,K
5 N=N+2
6 if(N<10) go to 31
7 end program GuessWhatIprint
```

لاحظ ان الجملة `N=N+2` لاتعني التساوي بل تعني خذ القيمة الحالية المحفوظة في المتغير `N` وأضف لها 2 ثم خزن النتيجة في المتغير `N`. بعبارة اخرى ان القيمة الجديدة لـ `N` اصبحت تساوي اخر قيمة لـ `N` مضافا لها 2.

الحل

عند تنفيذ البرنامج يقوم الحاسوب بما يلي:

السطر 2: يخزن الحاسوب القيمة 1 في مكان في الذاكرة اسمه `N`. يعني اسناد القيمة 1 الى المتغير `N`: `N ← 1`.

السطر 3: يقوم الحاسوب بضرب قيمة `N` في نفسها واسناد النتيجة الى المتغير `K`: `K ← 1`.

السطر 4: يقوم الحاسوب بطبع القيم الحالية لـ `N` و `K` في الشاشة:

1 1

السطر 5: يقوم الحاسوب بجمع 2 على القيمة الحالية لـ N وتخزين نتيجة الجمع (3) في المكان N (بهذا تُمسح القيمة السابقة لـ N وتستبدل بـ 3) ونعبر عن هذه العملية كما يلي: $N \leftarrow 3$

السطر 6: يجرى السؤال التالي: هل N اقل من 10 ؟
وحيث أن $N=3$ ، فإن الجواب نعم، أي أن الشرط بين القوسين صحيح لذلك يأتمر الحاسوب بالأمر go to ليذهب الى السطر المعلم برقم 31 أي الى السطر الثالث من البرنامج. ويستعد لتنفيذ الجملة المكتوبة فيه وفي السطور الموالية له.

السطر 3: يقوم الحاسوب بتربيع القيمة الحالية لـ N ثم تخزين النتيجة (9) في المكان K: $K \leftarrow 9$

السطر 4: يقوم الحاسوب بطبع القيم الحالية لـ N و K:

3 9

السطر 5: يقوم الحاسوب بجمع 2 على القيمة الحالية لـ N لتصبح 5: $N \leftarrow 5$

السطر 6: يعود الحاسوب الى السطر المعلم برقم 31 بسبب ان الجواب عن السؤال "هل N اقل من 10 ؟" هو نعم.

السطر 3: يربع الحاسوب القيمة الحالية لـ N ثم تخزن النتيجة (25) في المتغير K: $K \leftarrow 25$

السطر 4: يطبع الحاسوب القيم الحالية المخزونة في N و K:

5 25

السطر 5: يقوم الحاسوب برفع القيمة الحالية لـ N بمقدار 2 لتصبح 7: $N \leftarrow 7$

السطر 6: يعود الحاسوب الى السطر الثالث بسبب ان $N=7 < 10$

السطر 3: يربع الحاسوب القيمة الحالية لـ N و يخزن النتيجة في K: $K \leftarrow 49$

السطر 4: يطبع الحاسوب قيم N و K:

7 49

السطر 5: يزيد الحاسوب قيمة N بمقدار 2: $N \leftarrow 9$

السطر 6: يعود الحاسوب الى السطر الثالث بسبب ان $N=9 < 10$

السطر 3: تستبدل قيمة K الحالية بمربع N: $K \leftarrow 81$

السطر 4: تطبع قيم N و K:

9 81

السطر 5: تزداد قيمة N بمقدار 2: $N \leftarrow 11$

السطر 6: ينفذ الحاسوب الجمل التالية في البرنامج حيث ان الجواب عن السؤال "هل $N=11 < 10$ " هو لا هذه المرة. أي أن الشرط بين القوسين غير محقق وبالتالي لا تنفذ التعليمة go to.

السؤال 7: يتوقف الحاسوب عن التنفيذ بسبب وجود عبارة End program.

وبالتالي تأخذ نتيجة الطبع الاجمالي الشكل التالي:

```
1      1
3      9
5     25
7     49
9     81
```

ومنه يتضح الغرض من البرنامج هو طباعة الاعداد الفردية من 1 الى 9 ومربعاتها جنباً الى جنب بحيث يظهر كل عدد فردي ومربعه في كل سطر.

تمرين 7

```
1  program checkme
2  ! there is/are something(s) wrong
   with this program
3  ! find out what and correct it
4  real :: a, b, c
5  a = -10.0
6  b = 23.5
7  c = a*b
8  d = sqrt(a)
9
10 print *, 'What is wrong with this?'
11 print *, 'Nothing now..'
12 print *, c is, c
13 write(*,*) "d = ",d
14 !
15 end program checkme
```

توجد مجموعة من الاخطاء في هذا البرنامج قم بتصحيحها.

تمرين 8

```
1  program whatIprint
2  integer :: i,n=0
3  do i=1,6
4  n=n+2
5  enddo
6  write(*,*) "n = ",n
7  end program whatIprint
```

ماهي النتيجة التي يعرضها البرنامج المقابل.

تمرين 9

اكتب برنامجاً فرترون يطلب من المستخدم ادراج ثلاثة اعداد حقيقية A, B, C ثم يعرض الحلول الحقيقية للمعادلة من الدرجة الثانية : $Ax^2 + Bx + C = 0$.

! لا تستخدم متغيرات منطقية. يجب ان يعالج البرنامج حالة ادخال جميع القيم معدومة وكذلك حالة عدم توفر حلول.

.1

```

1  program test
2  integer :: i
3      do i = 1, 5
4          if ( i == 3 ) cycle
5              WRITE (*,*) i
6          enddo
7  write (*,*) 'End of loop!'
8  end program test

```

.2

```

1  program test
2  integer :: i
3      do i = 1, 5
4          if ( i == 3 ) exit
5              WRITE (*,*) i
6          enddo
7  write (*,*) 'End of loop!'
8  end program test

```

حدد في الحالات التالية خرج البرنامج.

إذا تم تنفيذ التعليمة **cycle** داخل حلقة معينة فان البرنامج يتخطى تنفيذ الدورة الحالية ويتقدم مؤشر الحلقة بخطوة واحدة ثم يستأنف تنفيذ الحلقة. !

إذا تم تنفيذ التعليمة **exit** داخل حلقة معينة فان البرنامج يتوقف عن تنفيذ الحلقة وينتقل البرنامج الى تنفيذ التعليمات ابتداء من اول سطر بعد الحلقة مباشرة. !

.1

```

1
2
3
5
End of loop!

```

.2

```

1
2
End of loop!

```

حدد في الحالات التالية عدد مرات (دوران) الحلقة.

```

1. do index = 5, 10
2. do j = 7, 10, -1
3. do index = 1, 10, 10
4. do loop_counter = -2, 10, 2
5. do time = -5, -10, -1
6. do i = -10, -7, -3

```

.2

```

loop1: do i = 1, 10
  loop2: do j = i, 10
    loop3: do k = i, j
      ...
    end do loop3
  end do loop2
end do loop1

```

.3

```

loopx: do i = 1, 10
  ...
  loopy: do j = 1, 10
    ...
  end do loopx
end do loopy

```

حدد في الحالات التالية ما اذا كانت تعليمات فورترن صحيحة او خاطئة . في حالة الخطأ حدد السبب.

.1

```

Loop1: do i = 1, 10
  loop2: do j = 1, 10
    loop3: do i = i, j
      ...
    end do loop3
  end do loop2
end do loop1

```

```

1 program test
2 character(len=8) :: a, b, c
3 a = 'ABCDEFGHIJ'
4 b = '12345678'
5 c = a(5:7)
6 b(7:8) = a(2:6)
7 end program test

```

```

1 program test
2 character(len=10) :: a
3 character(len=8) :: b, c
4 a = 'ABCDEFGHIJ'
5 b = '12345678'
6 c = a(1:3) // b(4:5) // a(6:8)
7 end program test

```

حدد ما ستحويه المتغيرات a,b,c عند نهاية تنفيذ البرامج المقابلة.

إذا كانت العبارة المسندة الى متغير حرفي أطول من طوله المصرح به تسند العبارة مع اسناد فراغات على يمينه مكان الاماكن المتبقية الشاغرة. في الحالة العكسية تلغى الاحرف الفائضة عن الطول المصرح به.

يمكن دمج سلسلتي حروف معا في سلسلة واحدة عن طريق المؤثر `//` وتعرف العملية باسم **concatenation**.

```

1. do irange = -32768, 32767
2. do j = 100, 1, -10
3. do kount = 2, 3, 4
4. do index = -4, -7
5. do i = -10, 10, 10
6. do i = 10, -2, 0
7. do

```

حدد في الحالات التالية عدد مرات (دوران) الحلقة.

```

1  INTEGER :: Class
2
3  SELECT CASE (Class)
4      CASE (1)
5          WRITE(*,*) 'Freshman'
6      CASE (2)
7          WRITE(*,*) 'Sophomore'
8      CASE (3)
9          WRITE(*,*) 'Junior'
10     CASE (4)
11         WRITE(*,*) 'Senior'
12     CASE DEFAULT
13         WRITE(*,*) "Hmmm, I don't
            know"
14 END SELECT
15 WRITE(*,*) 'Done'

```

اشرح خطوات هذا القسم من برنامج معين.

حدد نتائج الدوال التالية مستعينا بالجدول السفلي.

INT (2.9995)	NINT (2.9995)
CEILING (2.9995)	FLOOR (2.9995)
INT (-2.9995)	NINT (-2.9995)
CEILING (-2.9995)	FLOOR (-2.9995)

دوال تغيير الصنف

اسم الدالة	صنف العمدة	صنف النتيجة	المعنى
INT(X)	REAL	INTEGER	تأخذ الجزء الصحيح لـ X
NINT(X)	REAL	INTEGER	تدوير العدد لاقرب عدد صحيح
CEILING(X)	REAL	INTEGER	تدوير العدد لاقرب عدد صحيح في الاتجاه الموجب
FLOOR(X)	REAL	INTEGER	تدوير العدد لاقرب عدد صحيح في الاتجاه السالب
REAL(I)	INTEGER	REAL	تحويل العدد الصحيح الى حقيقي

```

1  module constants
2      real (kind(1.0d0)), parameter :: pi
        =3.141592653589793d0, h
        =6.62606891d-34
3  end module
4
5  module print_pi_mod
6  contains
7      subroutine print_pi
8          use constants
9          write(*,*) 'pi is ', pi
10     end subroutine
11 end module
12
13 program hbar
14     use constants
15     use print_pi_mod
16     write(*,*) 'hbar is ', h/(2*pi)
17     call print_pi
18 end

```

إن أبسط طريقة لمشاركة البيانات في Fortran 90 تتم عن طريق تعريف قسم برمجي لاستعمال التركيبة `module [name] ... end module` حيث [name] اسم اختياري، ولكنه فريد من نوعه. يمكنك تعريف واستخدام عدة اقسام برمجية من هذا الصنف حسب الحاجة. تتم مشاركة المتغيرات المعرفة في هذا القسم عن طريق استدعاء القسم بالتعليمة `use name` في اي قسم وظيفي اخر لكن قبل التعليمة `implicit none`.

```

1 program test
2
3 real(kind(1.0d0)),dimension(1:10) :: E
   , A, SIG,Tau
4 integer:: n,i
5 open(unit=10,file='data.dat',status='
   old')
6 write(*,'(a$)') 'Give the number n of
   cross section need to be read from
   the file: '
7   read(*,*) n
8   do i=1,n
9     read(10,*) E(i),A(i),SIG(i) !
       here A is not used
10    Tau(i)=exp(SIG(i)*E(i))
11    write(*,*),i, E(i),SIG(i),Tau(
       i)
12  enddo
13  close(10)
14
15 end program test

```

حدد الهدف من البرنامج المقابل.

! لاحظ ان المتغير A تقرأ قيمه من الملف ولكن غير مستخدم في الحساب. لان التعليمة `read(10,*)` تقرأ سطرا بسطر البيانات من ملف يحوي ثلاثة اعمدة:

data.dat

1.000E+01	1.503E-04	1.382E-03
1.200E+01	1.804E-04	1.657E-03
1.400E+01	2.104E-04	1.932E-03
1.600E+01	2.405E-04	2.206E-03
1.800E+01	2.705E-04	2.481E-03
2.000E+01	3.005E-04	2.755E-03
2.200E+01	3.305E-04	3.028E-03
2.400E+01	3.605E-04	3.301E-03
2.600E+01	3.904E-04	3.574E-03
2.800E+01	4.204E-04	3.847E-03