الباب الثالث

التكرار والتحكم

Loops and Control

3–1 مقدمة

استخدمنا في الحل الخوارزمي المصطلح (طالما – أنجز) ، وهو مصطلح أفادنا كثيرًا في تكرار عملية معينة دون أن نكرر الجملة الخاصة بتلك العمليـــــة. كما استخدمـــنا في الحل الخـــوارزمي المـــصطلح (لو – إذًا – وإلاّ) للتحكـم في سير العمليات ، وذلك بتوقيع الحالات المنطقية التي يتم تحويل التنفيذ بناء على صحتها من عدمه .

في هذا الباب ندرس الجمل التي تقوم بتلك الوظائف التي يتحكم بواسطتها المبرمج في سير البرنامج ، سواء من حيث تكرار العمل أو توجيهه من جـــزء إلى آخر في البرنامج .

3–2 جملة WHIE - DO

تقابل هذه الجملة المصطلح (طالما – أنجز) حيث تعني كلمـــة WHILE بينما أو طالما ، وكلمة DO تفيد طلباً بأداء عمل معين .

ويمكن التعبير عن جملة WHILE بالشكل اللغوي التالي :

حيث العبارة المنطقية إما أن تكون صائبة TRUE ، وفي هذه الحالة يتم إنجاز الجملة أو الجمل التي تلي كلمة DO والمحددة بين كلمة BEGIN وكلمة END، أو تكون خاطئة FALSE ليتحول التنفيذ إلى ما بعد الجملة . فإذا كانت الجملة مركبة فتكون النهاية هي END ، وإذا كانت جملة واحدة ، فينتهي المدى عند نهاية الجملة ؛ أي إن الحالة الأولى تكون على الشكل :

WHILE (عبارة منطقية) DO

BEGIN

جملة أو أكثر

END;

مع ملاحظة عدم ضرورة تحديد BEGIN و END في حالة تنفيذ جملة واحدة .

مثال (1–2–3)

حول الخوارزمية في المثال (1–2–1) إلى برنامج بلغة باسكال .

هذه الخوارزمية تقوم بوضع جدول المسافات بالبوصة وما يقابلها بالسنتمتر ، وبالتحديد من بوصة واحدة إلى **100** بوصة بزيادة بوصة واحدة في كل دورة .

بالرجوع إلى الشكل (1-2-1) واستخدام المتغير inch بـــدلاً مـــن س ، والمتغير cm بدلاً من ص ، نحصـــل مباشرة عـــلى البرنامـــج المبين في الشـــكل (1-2-3) .

لاحظ أن هذا البرنامج الذي أسميناه table (أي جدول) يطبـــع علـــى الشاشة النتائج المتكونة من 100 سطر ابتداء بالسطر :

1.0 inch = 2.54 cm

إلى السطر الأخير :

100.0 inch = 254.00 cm

شكل (1-2-3) : برنامج لتكوين جدول لتحويل البوصة إلى السنتمتر .

```
PROGRAM table;
{ a program to convert inch to cm }
VAR inch, cm : REAL;
BEGIN
   inch := 1;
   WHILE (inch<=100) DO
   BEGIN
      cm :=2.54*inch;
      WRITELN(inch:6:1, 'inch =  ',cm:6:2,'cm');
      inch:=inch+1;
   END;
READLN;
END.
```

كما يلاحظ في البرنامج استخدام المؤثر <= (وتعني أقل من أو تساوي) أي أن الشرط المنطقي هو :

طالما inch أقل من أو تساوي 100 يجب الاستمرار في الـــدوران وتنفيـــذ الجمل الثلاثة داخل دورة طالما .

بصورة عامة فإن المؤثرات المنطقية التي استخدمناها في الخوارزميات لها المعـــنى نفسه في لغة باسكال .

لاحظ أيضا استعمال جملة READLN في نهاية البرنامج بعد إتمام دورة طالما وطباعة جميع النتائج . والغرض من ذلك هو إبقاء شاشة المخرجات وعدم الرجوع إلى شاشة تربو باسكال ، حتى تقوم بالضغط على المفتاح enter أو غيره .

مثال (2-2-3)

اكتب برنامجاً لحساب متوسط درجات عدد من الطلبة التي يتم إدخالها عن طريق لوحة المفاتيح ، بحيث توضع لآخر درجة قيمة سالبة كعلامة لنهاية قائمة الدرجات .

يبين البرنامج بالشكل (2-2-3) البرنامج المطلوب في المثال ، حيث استخدمنا المتغير grade ليدل على الدرجة ، و المتغير count يرمز لعدد الطلبة ، أما total فهو مجموع الدرجات و av متوسط هذه الدرجات .

شكل (2-2-3) : برنامج حساب متوسط درجات الطلبة .

```
PROGRAM average;
VAR grade : REAL;
    count : LONGINT;
    total , av : REAL;
BEGIN
     count := 0;  total := 0;
     WRITE('enter grade-->');
     READ(grade);
     WHILE(grade>0) DO
     BEGIN
          count := count +1;
          total := total + grade;
          WRITE(' enter grade-->');
```

```
        READ(grade);
    END;
    av := total/count;
    WRITELN('average=', av: 5:2);
END.
```

3-3 المؤثرات المنطقية

تعتبر العبارة (3 < 5) عبارة منطقية وهي طبعاً صائبة TRUE ، كما أن العبارة (4 > 3) تعتبر أيضاً عبارة منطقية ولكنها خاطئة FALSE . ويمكن ربط هاتين العبارتين بالمؤثرات AND ، على الشكل :

(3 > 4) AND (3 < 5)

أي :

(3 > 4) و (3 < 5)

وهي عبارة خاطئة لأن شرطها الأيسر (4 > 3) خاطئ ، كما يمكن استخدام المؤثر المنطقي OR (أي أو) على النحو :

لو (العمر > 60) أو (العمر < 15) أنجز ...

فالعبارة بين القوسين تعتبر منطقية وهي صحيحة (صائبة) إذا كان أحد شرطيها صحيحًا . أي أن :

إذا كانت A ، B عبارتين منطقيتين فإن :

A AND B

تعتبر عبارة منطقية وهي صائبة فقط عندما تكون كل من A و B صائبة.

أما العبارة :

A OR B

فهي صائبة إذا كانت إحدى العبارتين A أو B أو كلاهما صائبة .

أما المؤثر NOT فهو ينفي العبارة المنطقية ، فعندما نقول العبارة

NOT A

تكون هذه العبارة صائبة إذا كانت A خاطئة ، والعكس صحيح .

مثال (3-3-1)

ماذا يحدث عند تنفيذ البرنامج المبين بالشكل (3-3-1) ؟

استخدمنا في هذا البرنامج المتغير correct ، وهو من النـــوع البـــولي (أو بمصطلح آخر النوع المنطقي) ، وقد عينا له العبارة المنطقية :

correct := (x > 15) AND (x < 100)

فإذا كانت هذه العبارة صائبة ، أي أن قيمة x المدخلة أقل من 100 وأكبر من 15 ، فإن التنفيذ يتحول إلى الجمل داخل مدى WHILE ، حيث يتم حســاب y وتطبع قيمتها ، أما إذا خرجت x عن الفترة المحددة فإن correct تصبح لها القيمة الخاطئة FALSE ، ولايتم تنفيذ مدى WHILE ، ولكن يتحول مباشرة إلى جملـــة طباعة العبارة 'data out of range' أي أن البيانات خارج النطـــاق المطلـــوب ، ويتوقف .

شكل (3-3-1) : استخدم المتغير المنطقي .

```
PROGRAM average;
VAR grade : REAL;
    count : LONGINT;
    total , av : REAL;
BEGIN
     count := 0;  total := 0;
     WRITE('enter grade-->');
     READ(grade);
     WHILE(grade>0) DO
     BEGIN
          count := count +1;
          total := total + grade;
          WRITE(' enter grade-->');
          READ(grade);
     END;
     av := total/count;
     WRITELN('average=', av: 5:2);
END.
```

3–4 جملة الشرط (IF - THEN - ELSE)

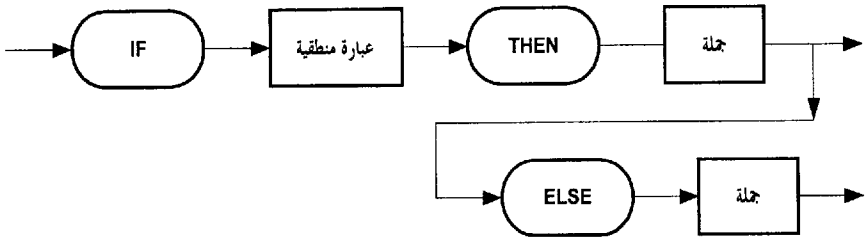
لقد استخدمنا في الحل الخوارزمي المصطلح :

لو - إذًا - وإلا ؟

لإنجاز بعض العمليات المشروطة بأداة الشرط لو . في لغة باسكال نستخدم الأداة (IF) بدلاً من (لو) ، و (THEN) بدلاً من (إذًا) ، وELSE بدلاً من (وإلا) ، وذلك على النحو المبين بالشكل 1–4–3 .

لاحظ في هذا الشكل أن ELSE والجمل التي تليها هي اختيارية وليست ضرورية . كما يجب أن نراعي حصر الجمل بين BEGIN و END (سواء بعد THEN أو بعد ELSE) إذا زاد عددها عن الجملة الواحدة .

شكل (3–4–1) : تركيبة جملة IF .



مثال (1–4–3)

ترجم الخوارزمية في المثال (1–6–1) إلى لغة باسكال .

هذه الخوارزمية كُتِبَتْ لإيجاد أعلى قيمة ضمن 10 قيم موجبة . ولترجمتها إلى باسكال ، نستخدم مثلا المتغير value بدلاً من ر ، والمتغير vmax بدلاً من ق والمتغير count بدلاً من ع . أما بقية البرنامج فهي مترجمة مباشرة .

من مصطلحات الحل الخـــوارزمي إلى مصـــطلحات باسكال . وشـــكل (2-4-3) يبين البرنامج المطلوب .

شكل (2-4-3) : برنامج إيجاد أعلى قيمة بين 10 قيم موجبة .

```
PROGRAM findmax;
VAR value, vmax : REAL;
    count : INTEGER;
BEGIN
     vmax := 0;
     count := 1;
     WHILE( count <= 10) DO
     BEGIN
          WRITE('entr a value-->');
          READ(value);
          IF(value>vmax) THEN
                    vmax := value;
                    count := count +1;
     END;
     WRITELN('highest value=',vmax:10:2);
END.
```

في هذا البرنامج ، لاحظ استخدام جملة IF على النحو :

IF value > vmax THEN

vmax: = value;

حيث توجد لدينا جملة واحدة كجواب شرط لأداة الشرط IF ، وبالتـــالي لم تكن هناك ضرورة لاستخدام BEGIN و END . كما نلاحظ عـــدم وجـــود ELSE لأنها اختيارية ، وليست هناك حاجة إليها في هذا البرنـــامج لأن عمليـــة استبدال قيمة vmax لاتتم إلا إذا كانت هذه القيمة أقل من value .

مثال (2-4-3) :

المطلوب كتابة برنامج للعبة تخمين عدد مجهول لدى اللاعب ، (وليكن هذا العدد مثلا 47) بحيث يطلب البرنامج من اللاعب إدخال عدد كتخمين للعدد الصحيح ، فإن كان أقل من العدد الصحيح ، ظهرت كلمة higher أي أعلى، و إذا كان أعلى من العدد الصحيح ظهرت كلمة lower أي أقل . ويقوم اللاعب بإدخال التخمين الجديد بناء على إحدى هاتين المعلومتين ، حتى إذا وصل إلى العدد المطلوب ، ظهرت كلمة (good) مع طباعة عدد المحاولات .

يبين شكل (3-4-3) البرنامج المطلوب في هذا المثال ، ويوضح هذا البرنامج إمكانية أن يكون العدد المدخل أكبر من المطلوب (أي 47) وفي هذه الحالة يكتب الكلمة (lower) ، أو أن يكون أقل من 47 وعندها يكتب الكلمة (higher) . أما عندما يكون التخمين صحيحاً فلا يتنفذ مدى WHILE ويكتب عدد المحاولات ويتوقف .

شكل (3-4-3) : برنامج لعبة تخمين .

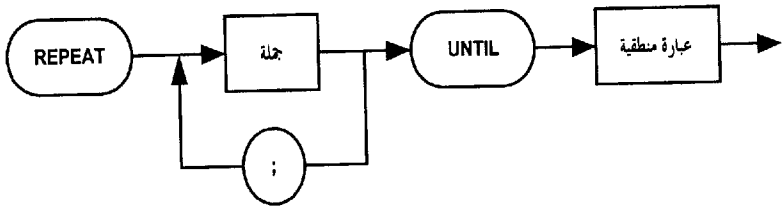
```
PROGRAM guess;
VAR k , trials, g, r : INTEGER;
BEGIN
     FOR k:=1 TO 40 DO WRITE('*');
     WRITELN;
     WRITELN(' Welcome to the guessing     game!!!');
     FOR k:= 1 TO 40 DO WRITE('*');
     WRITELN;
     trials := 1;
     r := 47;
     WRITE('enter your guess-->');
```

```
     READLN(g);
WHILE g<> r DO
     BEGIN
          WRITE('You have guessed ', g );
          IF( g > r ) THEN
             WRITE('It is wrong.
                Enter a lower guess--> ')
          ELSE
              WRITE(' It is wrong.
                Enter higher guess-> ');
          trials:= trials + 1;
          READLN(g);
     END;
     WRITELN('good!  It took you '
                    , trials,' trials');


END.
```

3–5 التكرار المشروط (جملة REPEAT)

تعتبر هذه الجملة شبيهة في تركيبها و مفعولها لجملـــة طالمـــا (WHILE)، ولكن بصورة مختلفة قد نجدها أحياناً أكثر ملاءمة . وهي تأخذ الصورة :



وتعني تكرار (repeating) تنفيذ الجمل حـــتى (until) تحقـــق العبـــارة المنطقية. وبالـــرجوع إلى المثال (1–4–3) الذي استخدمنا فيه جملة WHILE – DO بالشكل الآتي :

count: = 1

```
DO    WHILE   count < = 10
                BEGIN
                ....

                ....
count: = count + 1
                END
```

فإن استخدام REPEAT لهذه الحالة يكون على النحو :

```
count: = 1;
REPEAT
....

....
count: = count + 1
UNTIL
count > 10;
```

لاحظ في هذا المثال أن العبارة المنطقية بعد WHILE نقيض العبارة المنطقية بعد REPEAT . و السبب في ذلك يرجع إلى أن الأولى هي شرط تنفيذ الجمل المحددة ، بينما العبارة الثانية هي شرط توقف هذا التنفيذ . كما نلاحظ أن الجمل في دورة REPEAT تنفذ مرة واحدة على الأقل ، بينما في دورة WHILE قد لا تنفذ على الإطلاق .

مثال (1–5–3)

اكتب برنامجًا لتحويل المسافات :

5, 10, 15 , ... , 100

من نظام البوصة إلى السنتيمتر .

يبين البرنامج بالشكل(1-5-3) البرنامج المطلوب ، وفيه نلاحظ استعمال جملة REPEAT التي تعمل على تكرار تنفيذ جمل الدورة حتى يصبح المتغير inch أكبر من 100.

شكل (1–5–3) : برنامج تحويل من بوصة إلى سنتيمتر .

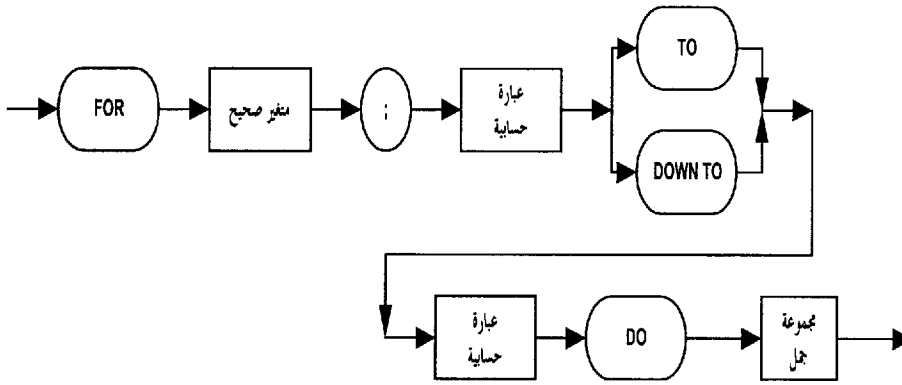
```
PROGRAM fromINCHtoCM;
VAR inch, cm: REAL;
BEGIN
     Inch := 5;
     REPEAT

          cm := 2.54*inch;
          WRITELN(inch:3:0,' inch= ',
                        cm:6:2,' cm');
          inch := inch +5;
     UNTIL
          inch > 100;
END.
```

3–6 التكرار المحدد (FOR - DO)

يستخدم التكرار المحدد (أي المحدد في عدد دوراته) بصورة شـــائعة في البرمجة ، ربما لأنه أكثر سهولة في استعماله من أي أسلوب آخر. والصورة العامـة لهذا التكرار يوضحها شكل (1–6–3) .

شكل (1–6–3) : تركيبة جملة FOR .



نلاحظ من هذا الشكل أن الجملة تبدأ بكلمة FOR متبوعة بمتغير صحيح ورمز التعين = : وعلى يمينه الرقم أو عبارة حسابية ، وهو الرقم الذي يتعين للمتغير الصحيح في بداية الدورة loop ، حيث يبدأ في الازدياد إذا استخدمنا كلمة TO أو في التناقص إذا استخدمنا كلمة DOWNTO ، ومقدار الزيادة أو النقصان هو الواحد الصحيح ، حتى تصل قيمة المتغير إلى الرقم الناتج من التعبير الحسابي الذي يلي كلمة TO أو DOWNTO. أما الجمل التي تأتي بعد DO فيتم تنفيذها عدداً من المرات (الدورات) حتى يتعدى المتغير الصحيح القيمة النهائية المحددة ، ولهذا يسمى هذا المتغير بالعداد . وعند الانتهاء من دورة (- FOR DO) ، لا يحتفظ العداد بقيمته بل يصبح غير معرف (undefined) .

لاحظ ضرورة وضع مجموعة الجمل (إذا زاد عددها عن الواحد) بين BEGIN و END .

مثال (1–6–3) :

اكتب البرنامج الذي يقوم بطباعة العمود التالي:

```
*
*
*
*
*
```

وذلك باستخدام جملة for .

يبين الشكل (2-6-3) البرنامج المطلوب. وفيه نلاحظ أن عداد دورة for هو المتغير I وهو يبدأ من القيمة 1 إلى القيمة 5 ، وهي عـــدد الأســـطر في العمـــود المطلوب رسمه .

شكل (2–6–3) .

```
PROGRAM stars;
VAR i : INTEGER;
BEGIN
     FOR i := 1 TO 5 DO
         WRITELN('*');
END.
```

لاحظ في هذا البرنامج أن العداد يتزايد نظرا لاستخدام TO ، وإذا أردنـــاه أن يتناقص نستعمل DOWNTO كما في المثال التالي:

مثال(2-6-3)

اكتب برنامجًا يقوم بكتابة الأرقام من 100 إلى 90 تنازلياً .

يبين البرنامج بالشكل (3-6-3) البرنامج المطلوب.

شكل (3–6–3) :

```
PROGRAM count;
VAR k : INTEGER;
BEGIN
    FOR k:= 100 DOWNTO 90 DO
        WRITELN(k);
END.
```

في هذا البرنامج عداد الدورة هو k وهـــو كما هو واضح يبدأ من القيمة 100 وينتهي عند القيمة 90 .

3–7 التكرار المركب

يجوز أن تحتوي جملة التكـــرار WHILE أو REPEAT أو FOR علـــى جملة تكرار أخرى أو أكثر ، ونسمى مثل هذه الحـــالات بـــالتكرار المركـــب (Nested Structure) .

مثال (1–7–3)

اكتب برنامجًا يحتوي عـــلى تكرار من أجل طباعة جدول الضرب للأعـــــداد من 1 إلى 9 .

هذا البرنامج مبين بالشكل(1-7-3) وفيه نلاحظ وجـــــود حلقـــة loop باستعمال جملة FOR بمتغير (عداد) هو m وبداخلها حلقة أخـــرى باســـتخدام جملة FOR أيضاً وبعداد n . تسمى الحلقة الأولى بالحلقة الخارجية والثانية بالحلقـــة الداخلية ، وفي مثل هذه الحالة نلاحظ أن العداد في الحلقة الداخلية يتغير بمعـــدل أسرع من عداد الحلقة الخارجية ، وهذا شيء طبيعي لأن الحلقة الداخليـــة يتـــم تنفيذها بالكامل في كل خطوة زيادة أو نقصان لعداد الحلقة الخارجية .

شكل (3-7-1) : برنامج جدول الضرب .

```
PROGRAM multiply;
VAR m, n, p : INTEGER;
BEGIN
     FOR m:= 1 TO 9 DO
     BEGIN
          FOR n:=1 TO 9 DO
          BEGIN
               p:= m*n;
               WRITE(p:3);
          END;
     WRITELN;
     END;
     END.
```

كملاحظة جانبية ، نشير هنا إلى أهمية جملة WRITELN في هذا البرنـــامج، وذلك لبداية سطر جديد بعد الانتهاء من كل دورة داخلية .

3-8 التفريع باستخدام جملة GOTO

رغم أن هذه الجملة ، غير مرغوب في استخدامها نظراً لتعارضها مع فكرة البرمجة الهيكلية ، إذ يجد قارئ البرنامج الذي يحتوي على جملة التفرع **GOTO** شيئاً من الصعوبة في تتبع وفهم البرنامج... إلا أن معرفة استخدام هذه الجملـــة ضرورية لاكتمال استيعاب لغة باسكال ، و أيضاً لوجود حالات تضطرنـــا إلى استخدامها .

وتستخدم هذه الجملة على النحو .

GOTO n

حيث n أي عدد موجب صحيح ، ويسمى عنوان label. ويشير إلى رقم الجملة التي يتحول إليها التنفيذ بناء على هذه التعليمة . ومن الضروري تحديد أن هذا الرقم هو عنوان في بداية البرنامج ، وبالتحديد قبل جملة VAR. فمثلاً الجملة:

```
LABEL 25;
```

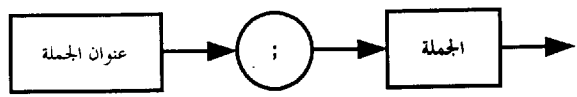
تشير إلى أن 25 هو رقم جملة في البرنامج . ويوضع الرقم مع الجملة المشار إليها و مباشرة و قبلها وتفصل بينهما الشارحة (:) مثل

```
25:   WRITELN ( X );
```

وبذلك يمكن التحويل إلى هذه الجملة من نقطة في البرنامج باستخدام الجملة .

```
GOTO 25;
```

وبصورة عامة فإن الجملة يمكن أن يسبقها رقم يسمى عنوان الجملة Statement label على أن يفصل بينهما الرمز (:) على النحو التالي :



مع ضرورة تحديد عنوان الجملة في جملة Label .

9- 3 تمارين

في كتابة برامج التمارين (1 ، 2 ، 3، 4 ، 5) استخدم 3 طرق ، وذلك باستعمال :

- جملة WHILE - DO
- جملة REPEAT - UNTIL
- جملة FOR - DO

لاحظ أن هذه البرامج مطلوب حلها الخوارزمي في تمارين (7-1)

1- المطلوب كتابة برنامج لتحويل درجات الحرارة من النظام المئوي C إلى نظام الفهرنهايت F ، ابتداء من 1 إلى 100 درجة مئوية . تطبع النتائج في جدول واضح . لاحظ أن :

$$f = (9/5)\, c + 32$$

2- اكتب برنامجاً لإيجاد مجموع الأعداد الفردية

1 ، 3 ، 5 ، ..... ، 50

3- اكتب برنامجاً لإيجاد أصغر درجة ضمن 50 درجة ، علماً بأن أكبر درجة ممكنة هي 100 .

4- اكتب برنامجاً لإيجاد متوسط درجات عدد من الطلبة حيث يتم إدخال الدرجات عن طريق لوحة المفاتيح ، وفي حالة وجود درجة سالبة فذلك يعني نهاية الإدخال ، وهي ليست درجة حقيقة .

5- في تمرين (4) ، افترض وجود طلبة وطالبات ، والمطلوب إيجـــاد متوســـط الطلبة ، ومتوسط الطالبات . يمكن قراءة الجنس مع الدرجة باستخدام المتغير الصحيح S ويساوي 1 للطالب ، و 2 للطالبة.

6- اكتب برنامجاً لحساب ضريبة الدخل طبقاً للخوارزمية المطلوبة في تمريـــن(8) من مجموعة تمارين (7-1) . تطبع النتائج في جدول واضح مثل :

number	income	tax
1	xxx.xxx	xxx.xxx
2	xxx.xxx	xxx.xxx
3	xxx.xxx	xxx.xxx

.....

7- اكتب برنامجاً للخوارزمية المطلوبة في تمريـــن (9) مـــن مجموعـــة تمـــارين (7-1). يمكن استخدام الرموز التالية في الطباعة :

F تقدير ضعيف

P تقدير مقبول

G تقدير جيد

A تقدير ممتاز

8- لدينا مجموعة من البيانات تحتوي على :

number , mf , fbc

حيث number يرمز لرقم الشخص ، mf لجنسه ('M' للذكر و 'F' للأنثى) وfbc عدد صحيح يبين الحالة الصحية . وحيث إن عدد الأشخاص غـــير معلوم ، وضع آخر سطر من هذه البيانات بحيث يحتوي على رقم ســـالب مقابل number . اكتب برنامجاً لعمل الإحصائيات التالية :

- عدد الأشخاص الكلي .
- عدد الذكور الكلي و عدد الإناث الكلي .

- عدد الذكور الذين يزيد fbc لديهم عن 300 .
- النسبة المئوية لعدد الإناث الذين يزيد fbc لديهم عن 300 نسبة إلى عدد الإناث الكلي .

9- استخدم التكرار المركب في كتابة برنامج يطبع الشكل الآتي :

```
*
****
*********
**********
```

أي أن عدد الشكل (*) في السطر الأول = 1 ، وفي السطر الثاني = 4، وفي السطر الثالث = 9 ، وفي السطر الرابع = 16 .

* * *