

الباب الثاني

من الخوارزميات إلى لغة باسكال

From Algorithms To Pascal Language

2-1 مقدمة

إن المصطلحات التي استخدمناها في كتابة الخوارزميات ما هي إلا كلمات اصطلاحنا على معانيها ووظائفها بحيث تصبح الخوارزمية مفهومة لمن اطلع على هذه المعاني ، ولكنها لن تفهم من قبل الحاسوب إلا إذا صممنا برنامجا خاصا يترجم هذه الكلمات للآلة . ونظرا لعدم توفر مثل هذا المترجم ، فإننا سنقوم بهذه المهمة و لكن بطريقة غير مباشرة ، حيث إننا سنترجم هذه الخوارزميات إلى لغة محددة و هي لغة باسكال ، التي تعود مصطلحاتها إلى اللغة الإنجليزية .

وقد جاء اسم اللغة من اسم العالم الرياضي بليز باسكال Blaise Pascal الذي عاش في القرن الثامن عشر و الذي كان مهتما بالرياضة الحسائية ، إلا أنه لم يصمم لغة (باسكال) بل إنها من تصميم و إنجاز نيكولوس ويرث Niklaus Wirth في بداية السبعينيات ، كما أشرنا سابقا .

ويوجد اليوم أكثر من صيغة واحدة لهذه اللغة ، وسوف نستخدم بالتحديد صيغة TURBO PASCAL ذات الانتشار الواسع لدى مستخدمي الأجهزة الشخصية ، نظرا لسهولة استخدامها وسرعة تنفيذها .

2-2 بنية البرنامج

يتكون البرنامج في لغة باسكال أساسا من ثلاثة أجزاء :

- 1- جزء تعريف البرنامج : وفيه نحدد اسم البرنامج و ملفي الإدخال والإخراج (input , output) .

2- جزء تعريف المتغيرات : وفيه يتم تعريف المتغيرات المستخدمة في البرنامج وأنواعها .

3- جزء محتوى على جعل البرنامج ونسميه الجزء التنفيذي .

وبين الشكل (1-2-2) هيكلًا مبسطًا للشكل اللغوي Syntax diagram لهذه الأجزاء الثلاثة ، ويلاحظ في هذا الشكل أننا نستخدم الشكل البيضاوي :



لل كلمات المحجوزة مثل PROGRAM و VAR و END .

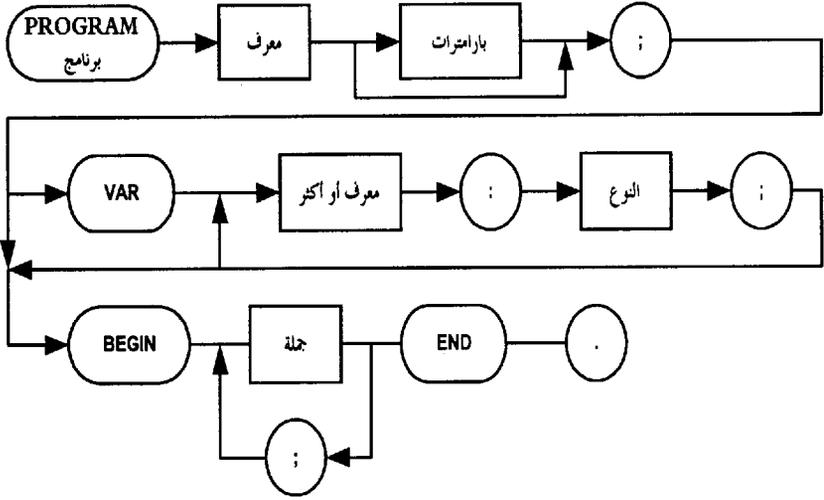
كما نستخدم الدائرة للرموز مثل الفاصلة المنقوطة والشارحة والقاطعة .

كما تبين الأسهم الاتجاه الذي يسير فيه ترتيب محتوى البرنامج . وفي حالة وجود دورة ، فإن ذلك يعني إمكانية تكرار ما بداخل هذه الدورات عدة مرات . فمثلا نستعمل الكلمة المحجوزة VAR لتحديد قائمة المعرفات (المتغيرات) ونوعها بحيث نستخدم كلمة VAR مرة واحدة لعدة متغيرات على أن يفصل بين تحديد كل متغير وآخر بالفاصلة المنقوطة ، مثل :

```
VAR X: REAL;  
Y : INTEGER;  
Z : CHAR;
```

كما نلاحظ من هذا الشكل أن الجزء الثاني والخاص بتعريف المتغيرات هو اختياري وليس ضروريا إذا كان البرنامج لا يحتوي على متغيرات . كما نشير هنا إلى أن مجموع الجزأين يسمى بالقالب Block . ونرجع الآن إلى الصندوق (بارامترات) الذي يأتي بعد معرف (أي اسم) البرنامج .

شكل (2-2-1): تخطيط مبسط لبرنامج بلغة باسكال .



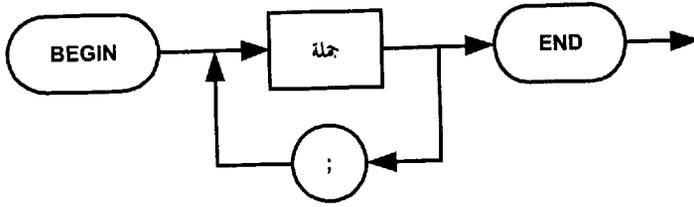
يتضح من الشكل أن هذا التحديد اختياري ، وفي حالة استخدامه ، فيكون على النحو (input , output) حيث ترمز input إلى ملف الإدخال ، وترمز output إلى ملف الإخراج . وسنرجع إلى هذا الموضوع عند مناقشة موضوع الملفات .

أما بالنسبة للجزء الأخير من برنامج لغة باسكال فإنه يمثل الجزء التنفيذي من البرنامج ، وهو يتكون من جملة واحدة على الأقل على أن يبدأ هذا الجزء بكلمة BEGIN وينتهي بكلمة END (انظر شكل 2-2-2) ، وتسمى هذه التركيبة بالجملة المركبة . لاحظ أخيراً أن القاطعة تحدد نهاية البرنامج terminator .

مثال (2-2-1)

يبين البرنامج بالشكل (2-2-3) تسلسل البرنامج في لغة باسكال بناء على الشكل اللغوي (2-2-1) . وهذا البرنامج ما هو إلا ترجمة للخوارزمية (1-1-1) ، التي تقوم بتحويل القياسات من وحدة البوصة إلى السنتيمتر .

شكل (2-2-2) : جملة مركبة (Compound Statement).



شكل (2-2-3) : برنامج بلغة باسكال .

```
PROGRAM convrtInchs To Cm;
VAR Inchs, Cents : REAL;
BEGIN
READLN (Inchs);
Cents := 2.54 * Inchs;
  WRITELN ( Inchs , Cents );
END.
```

في هذا البرنامج نلاحظ:

- 1- معرف البرنامج (أي اسم البرنامج) هو Conver Inchs To Cm .
- 2- تحديد Inchs و Cents كمتغيرين كسريين (أي حقيقيين)
- 3- الجزء التنفيذي يحتوي على 3 جمل هي :

اقراء Inchs
READLN (Inchs);

احسب $2.54 * Inchs$ وعين الناتج للمتغير Cents

Cents := 2.54*Inchs;

اكتب القيمتين : Inchs و Cents

WRITELN(x,y);

نلاحظ أن معرف البرنامج هو Convert Inchs To Cm ويعني :
Convert Inchs To Cm : حول من بوصات إلى سنتمترات .

وقد تم دمج هذه العبارة في كلمة واحدة ؛ لأن اسم البرنامج يتكون من كلمة واحدة فقط بدون فراغات ، ولكن يسمح بوجود أحرف كبيرة في داخل الاسم لغرض التوضيح .

كما نلاحظ أن البرنامج يحتوي على متغيرين كسريين REAL هما Inchs و Cents كما هو واضح في التحديد VAR .

أما الجزء الثالث فيحتوي على ثلاث جمل ، كل جملة تنتهي بفاصلة منقوطة (:) كما هو الحال في جميع جمل باسكال ، أما END (بدون القاطعة) فتعني نهاية الجزء التنفيذي ، والقاطعة التي تليها تحدد نهاية البرنامج .

3-2 المتغيرات والثوابت وأنواعها Variables & Constants

عندما نحدد رمزا لمتغير فذلك يعني أننا قد حجزنا موقعا في الذاكرة الرئيسية لتخزين قيمة هذا المتغير، ونشير لهذا الموقع بالاسم المحدد (وتعتمد سعة هذا المتغير على نوعه) وتغير القيمة التي يحتوي عليها هذا الموقع من وقت إلى آخر أثناء تنفيذ البرنامج . أما الثابت فلا تتغير قيمته .

ويوجد في لغة باسكال أساسا أربعة أنواع من المتغيرات والثوابت ، وهي النوع الكسري والصحيح والرمزي والمنطقي ، ولكن بالإمكان أيضا تعريف أنواع أخرى قد يحتاجها المبرمج في برنامجه . وتعرض الآن لكل نوع على حدة :

1- النوع الصحيح INTEGER

من الناحية النظرية فإن المتغير الصحيح ينتمي إلى الفئة

$$\{ \dots , -2 , -1 , 0 , 1 , 2 , 3 \dots \}$$

أي إن هذه الفئة يوجد بها عدد لا نهائي من العناصر، ولكن من الناحية العملية ، لا بد أن نضع حدا أقصى لهذه الفئة ، ويسمى في لغة باسكال MaxInt ، أي أن الفئة تبدأ من - MaxInt وتنتهي عند MaxInt . وقد تختلف قيمة الثابت MaxInt من صيغة إلى أخرى ، فنجد في الأنظمة التي تخصص 16 بتا bit (خانة ثنائية) للمتغير الصحيح أن :

$$\text{MaxInt} = 2^{15} - 1 = 32767$$

وإذا تجاوز متغير صحيح هذا الرقم يحدث ما يسمى بخطأ الفيض overflow ويرتب على ذلك نتائج حساسية غير صحيحة .

ولكن ماذا لو كان لدينا أعداد تفوق الحد 32767 ؟ في هذه الحالة توفر لنا صيغة تربو باسكال النوع LongInt ، الذي يخصص حيزا أكبر للمتغير الصحيح ، أي 32 خانة ثنائية بدلا من 16 خانة ، وبالتالي علينا بتحديد هذا النوع في جملة VAR بدلا من النوع INTEGER إذا كنا نتوقع أن يكون العدد كبيرا جدا بحيث لا يتجاوز 2147483647.

2- النوع الكسري REAL

المتغير الكسري هو الذي ينتمي إلى فئة الأعداد الحقيقية ، ويمكن تمثيله

بطريقة النقطة العائمة Floating Point :

$$v = a * 10^b$$

حيث a عدد كسري و b عدد صحيح .

وكمثال على هذه الطريقة في تمثيل الثوابت الكسرية ، نجد أن

$$1230000.0 = 0.123 * 10^7$$

$$0.0000456 = 0.456 * 10^{-4}$$

وفي لغة باسكال نستخدم التمثيل aEb للدلالة على :

$$aEb = a * 10^b$$

فمثلا :

$$0.246E2 = 0.246 * 10^2$$

$$0.369E-3 = 0.369 * 10^{-3}$$

ولكن ليس من الضروري استخدام هذا التمثيل في التعبير عن قيمة كسرية،

فإذا استخدمنا الثابت 12.5 أو 0.125E2 فإن المعني واحد .

3- النوع الرمزي CHAR

يشمل هذا النوع كل الرموز المستخدمة في فئة الرموز (Character Set)

بما في ذلك الحروف الأبجدية الصغيرة :

a, b, c, ..., x, y, z

والحروف الكبيرة Capital letters :

A, B, C, ..., X, Y, Z

والأرقام digits :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

والإشارات :

+, -, *, /, &, ...

الموجودة بلوحة المفاتيح (الملامس) .

عند تعيين ثابت رمزي لا بد من حصره بين إشارات الاقتباس ، مثل :

'a', '*', 'c', ...

4- النوع المنطقي BOOLEAN

المتغير المنطقي هو المتغير الذي تكون قيمته إما صوابا TRUE أو خطأ FALSE . فإذا كان المتغير compare من النوع المنطقي فإن الجملة :

compare := (5 > 4);

تكافئ :

compare := TRUE;

وبالمثل فإن :

compare := (4 > 5);

تكافئ :

compare := FALSE;

لاحظ هنا أن (5 > 4) مثال لعبارة منطقية . وبالمثل فإن :

$$6 = 8$$

تعتبر عبارة منطقية قيمتها FALSE و العبارة (10 أكبر من أو تساوي 5):

$$10 >= 5$$

تعتبر عبارة صحيحة TRUE . أما العبارة :

$$3 < > 4$$

فتعني أن 3 لا تساوي 4 وهي بالطبع صحيحة TRUE .

2-4 العبارات الحسابية

Arithmetic Expressions

العبارة الحسابية هي مجموعة من المتغيرات أو الثوابت أو مزيج منها

تفصل بينهما مؤثرات operators حسابية ، ونرمز لها بالرموز التالية :

* مؤثر الضرب .

/ مؤثر قسمة الأعداد الكسرية .

+ مؤثر الجمع .

- مؤثر الطرح .

DIV مؤثر قسمة الأعداد الصحيحة .

MOD مؤثر لإيجاد باقي قسمة الأعداد الصحيحة .

وتستخدم الأقواس () لتقسيم العبارة الواحدة إلى عبارات جزئية . لاحظ
عدم وجود رمز لعملية الأس (القوى) في لغة باسكال ، و لا بد من برجة هذه
العملية عند الحاجة إليها.
و كأمثلة على العبارات الحسابية ما يلي :

$$a + b * c$$

$$2 * x + 5$$

$$1.5 - (PAY / 2.5 + 4.0)$$

$$(2.456 + rate - 23.8) * 4.0$$

$$65 \text{ DIV } 4 * 2$$

وعند إيجاد قيمة عبارة حسابية ، فإن القواعد التالية يجب مراعاتها:

- 1- يتم التقييم من اليسار إلى اليمين حسب الأسبقية .
- 2- تعطي الأسبقية لعمليتي الضرب و القسمة على عمليتي الجمع و الطرح .
- 3- تستخدم الأقواس لإعطاء الأسبقية للعبارة المحصورة بين القوسين .

مثال (1-4-2) :

يبين الرمز (=) فيما يلي العبارات المتكافئة :

$$2 + 5 * 3 = 2 + 15 = 17 \quad (1)$$

$$2.7 + 4.0/2.0 * 3.1 = 2.7 + 2.0 * 3.1 \quad (ب)$$

$$= 8.9$$

$$3 + 4 * 2.0/4.0 = 3 + 8.0/4.0 \quad ()$$

$$= 5.0$$

$$(5 + 6) * 3 = 11 * 3 = 33 \quad (د)$$

$$(-5 + 6) * (3 - 5) = (1) * (-2) = -2 \quad ()$$

ويستخدم المؤثر DIV لقسمة عددين صحيحين ، فمثلا :

$$9 \text{ DIV } 2 = 4$$

أي إن الجزء الكسري (الباقي) قد تم إهماله .

هذه العملية خاصة بالأعداد الصحيحة و لا يجوز كتابة عبارة مثل :

$$\rightarrow \text{خطأ} \quad 9.0 \text{ DIV } 2.0$$

وللحصول على بقية القسمة نستخدم المؤثر MOD ، فمثلا :

$$9 \text{ MOD } 2 = 1$$

$$8 \text{ MOD } 4 = 0$$

أي أن بقية قسمة 9 على 2 هو 1 ، وبقيّة قسمة 8 على 4 هو الصفر . ويمكن

دمج DIV و MOD في عبارة واحدة مثل :

$$(-34) \text{ DIV } (29 \text{ MOD } 6)$$

وهي تكافئ :

$$(-34) \text{ DIV } 5 = -6$$

ويجوز أيضا تقسيم الأعداد السالبة ، فمثلا :

$$\begin{aligned} 11 \text{ DIV } (-4) &= -2 \\ 11 \text{ MOD } (-4) &= 3 \\ (-16) \text{ MOD } 3 &= -1 \\ (-23) \text{ MOD } 5 &= -3 \end{aligned}$$

$$\begin{aligned} 11 \text{ MOD } (-4) &= 3 \\ 11 \text{ DIV } (-4) &= -2 \\ (-16) \text{ DIV } 3 &= -5 \\ (-23) \text{ DIV } 5 &= -4 \end{aligned}$$

ويجب الانتباه إلى عدم التقسيم على صفر فذلك يعتبر خطأ ينهي تنفيذ البرنامج .

2-5 تحديد المتغيرات Declaration of Variables

من الضروري في لغة باسكال تحديد أسماء المتغيرات المستخدمة في البرنامج وأنواعها. ويكون ذلك في بداية البرنامج في جملة VAR . فمثلا الجملة :

VAR x, y; REAL;

تحدد أن x و y متغيران كسريان . والجملة :

VAR gr: CHAR;

تبين أن المتغير gr من النوع الرمزي ، والجملة :

VAR correct: BOOLEAN;

توضح أن المتغير correct من النوع المنطقي (البولي) . ونلاحظ في هذه الأمثلة كتابة أسماء المتغيرات بالحروف الصغيرة ، ولكن ذلك ليس ضروريا ، فلغة باسكال لا تميز بين الحروف الصغيرة small والحروف الكبيرة CAPITAL ، أي أن المتغيرات التالية ترمز للموقع نفسه في الذاكرة :

Numofitems numofitems NUMOFITEMS
numOFitems

والغرض الوحيد من استخدام الحروف الصغيرة في أسماء المتغيرات هو توضيح أن هذه الأسماء من اختيارنا ، بخلاف الكلمات (REAL ، VAR ، CHA ، ...) التي كتبت بحروف كبيرة لتوضيح أنها كلمات خاصة بلغة باسكال . وعند اختيار اسم للمتغير variable identifier يجب مراعاة ما يلي :

• يمكن أن يتكون الاسم من أي عدد من الحروف و الأرقام على أن يبدأ الاسم بحرف من اليسار.

• الحروف الكبيرة (A, B, C, ...) والحروف الصغيرة (a, b, c, ...) تعتبر متكافئة ، أي أن المتغير RATE مثلا هو نفسه rate . ونستفيد من هذه الخاصية في توضيح أسماء المتغيرات باستخدام خليط من الحروف الصغيرة والكبيرة مثل :

StudentName , YourAddress , SumOfSquares , ...

• عدم استخدام الكلمات المحجوزة كأسماء ، مثل VAR و BEGIN و END وغيرها من الكلمات المحجوزة والمبينة بعض منها في جدول (2-4-1) .
جدول (2-4-1) : بعض الكلمات المحجوزة في تربو باسكال .

Turbo Pascal Reserved Words

ABSOLUTE	INTERRUPT
AND	LABEL
ARRAY	MOD
BEGIN	NIL
CASE	NOT
CONST	OF
DIV	OR
DO	PACKED
DOWNTO	PROCEDURE
ELSE	TO
END	TYPE
FILE	UNIT
FOR	UNTIL
FORWARD	USES
FUNCTION	VAR
GOTO	WHILE
IF	WITH
IN	XOR

• من إحدى خصائص البرمجة الجيدة اختيار أسماء مناسبة للمتغيرات المستخدمة في البرنامج بحيث تدل هذه الأسماء على مسمياتها ، فمثلا

اختيار الاسم age للدلالة على العمر ، أو price للدلالة على السعر ، أو NumberOfStudent للدلالة على عدد الطلبة إلى غير ذلك . . . هو أفضل من استخدام رموز مختصرة غير واضحة المعني .

- تسمح بعض صيغ باسكال باستخدام " _ " كأحد رموز المتغير مثل Age_group أو name_of_wife .

مثال (1-4-2)

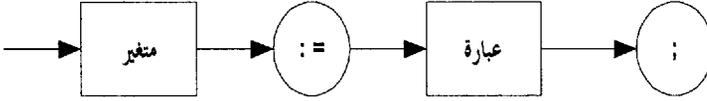
الأسماء التالية جائزة في لغة باسكال لتعريف المتغيرات :

a	temp
MidPoint	Y12
BookList	MOHAMMED
AccountNumber	ItIsFound
XgreaterThanY	too_hot

أما الأسماء التالية فلا تجوز للأسباب المبينة :

لا يجوز استخدام كلمة محجوزة.	UNIT
لا يجوز البدء برقم.	3X5
لا يجوز أن يحتوي على الرمز * .	*FIFTY
لا يجوز احتواء إشارة حسابية	EX - 5

تتبع هذه الجملة الشكل اللغوي التالي :



حيث يتم تعيين قيمة العبارة الواقعة على يمين المؤثر =: للمتغير الواقع على يسار هذا المؤثر. ومن الضروري هنا أن تكون العبارة و المتغير من النوع نفسه إلا في حالات خاصة .
فمثلا الجملة :

$$y = 2.54 * x;$$

تعتبر جملة تعيين للعبارة الحسابية ($2.54 * x$) للمتغير y حيث x و y متغيران من نوع REAL .
والجملة :

Condition: = TRUE;

تعين القيمة TRUE للمتغير المنطقي Condition على أن يتم تحديد نوع هذا المتغير مسبقا في جملة VAR . وإذا كان المتغير ch من النوع CHAR وكتبنا جملة على النحو:

$ch = 5;$

فإن ذلك يعتبر خطأ عدم تناظر النوع Type Mismatch ؛ إذ إن 5 تعتبر ثابتا عدديا وليست رمزا . ولتصحيح هذا الخطأ نكتب :

$ch = '5';$

والاستثناء لقاعدة تناظر نوع العبارة والمتغير يحدث في الجملة التي تحتوي على عبارة عددية صحيحة ومتغير كسري ، مثل :

$$x := 12 \text{ DIV } 3;$$

في هذه الحالة ، تأخذ x القيمة 4 حتى ولو كان متغيرا كسريا وليس صحيحا . أما الجملة التي تحتوي على متغير صحيح في الطرف الأيسر وعبارة كسرية في الطرف الأيمن ، مثل :

$$I := 2.5;$$

فهي غير جائزة إذا كان I متغيرا صحيحا لعدم تناظر الطرفين .

2-7 الإدخال و الإخراج Input & output

تعرضنا لمعنى إدخال البيانات وإخراجها عندما درسنا المصطلحين (اقرأ) و (اكتب) في الحل الخوارزمي . والآن عند ترجمة الخوارزمية إلى برنامج بلغة باسكال ، نستخدم جملة READ أو READLN بدلا من (اقرأ) وجملة WRITE أو WRITELN بدلا من (اكتب) ، وسنوضح الفرق بينهما فيما بعد .
نبدأ أولا بعملية قراءة البيانات .

الصورة العامة لقراءة عدد n من البيانات من لوحة المفاتيح هي :

$$\text{READ} (v1, v2, v3, \dots, vn);$$

حيث ترمز $v1, v2, \dots, vn$ لأي n متغير. يتم تعيين البيانات على التوالي للمتغيرات المسرودة في الجملة . لاحظ أن البيانات يجب أن تطابق في نوعها المتغيرات المناظرة لها . وأن هذه البيانات يجب فصلها بفاغ واحد على الأقل ، أو أن تكون كل واحدة في سطر منفصل .

فإذا كان المتغير TXM من النوع الكسري ، و NUM من النوع الصحيح

واستخدمنا الجملة :

$$\text{READ} (\text{TXM}, \text{NUM})$$

مع إدخال البيانات التالية عند تنفيذ البرنامج :

$$12.34 \quad 987$$

فإن القيمة 12.34 تتعين للمتغير TXM و القيمة 987 تتعين للمتغير NUM .

إلا أن تربو باسكال لا يتقبل الجملة .

READ (TXM, NUM, SIGN)

حيث SIGN من النوع CHAR نظرا لعدم القدرة على تمييز البيانات العددية من الرمزية في سطر واحد ؛ لأن الفراغ لا يعتبر فاصلا بل هو رمز كأى رمز آخر ، ولذلك نلجأ إلى وضع قيمة SIGN في سطر جديد ، ونستخدم للقراءة جملتين مثل:

READLN (TXM, NUM);

READ (SIGN);

حيث READLN لها تأثير READ نفسه ، ولكن مع التحول إلى سطر جديد بعد القراءة .

لاحظ أن الفرق بين READLN و READ أن الأولى تحمل بقية القيم في السطر نفسه وتتحول إلى سطر جديد، ولكن READ لا تحملها. لذلك يجوز قراءة قيمتين في سطر واحد بجملتين من نوع READ ، ولكن لا يجوز قراءتهما بجملتين من نوع READLN إلا إذا وضعناهما في سطرين متتاليين .

لعرض مخرجات البرنامج على الشاشة نستخدم جملة WRITE ، مثل :

WRITE (NUM, TXM, SIGN);

أو نستخدم WRITELN على النحو:

WRITELN (NUM, TAX, SIGN);

في الحالة الأولى (أي باستخدام WRITE) يبقى المؤشر في نفس السطر بعد كتابة المطلوب ، وفي الحالة الثانية (أي باستخدام WRITELN) يتحول إلى بداية سطر جديد بعد الانتهاء من الكتابة . لاحظ عدم وجود مشكلة في اختلاف أنواع المتغيرات المطبوعة باستخدام WRITE أو WRITELN . فمثلا إذا كان المتغير ch من النوع CHAR بحيث :

ch := ' ';

WRITELN (NUM, ch , TXM, ch , SIGN);

فهي تقوم بطباعة القيم الثلاثة مع وجود فراغ واحد بين القيم المطبوعة .
ونلاحظ أيضا أن القيمة المطبوعة من متغير كسري تكون على الصورة aEb
(أي a مضروب في 10 أس b) .

لتفادي هذه الصورة ، وطباعة القيمة على الصورة المعتادة ، نستخدم
الجملة :

```
WRITELN (X: m: n);
```

أو

```
WRITE (X: m: n);
```

حيث m عدد الخانات المخصصة للعدد كله (بما في ذلك الجزء الكسري
والصحيح والفاصلة والإشارة) . و n عدد خانات الجزء الكسري .

فمثلا الجملة :

```
WRITELN (TAX: 6: 2);
```

ستطبع قيمة TAX على الصورة

```
xxx.xx
```

والجملة :

```
WRITELN ('GRADE = ', grade: 4: 1);
```

ستطبع على الصورة

```
GRADE = xx.x
```

لاحظ أن الرموز الواقعة بين علامتي الاقتباس المفردة (وتسمى نضيد
STRING) تعتبر مقدارا ثابتا وتتم طباعتها كما هي .

ولتترك سطرا فارغا بين السطور نكتب WRITELN بدون متغيرات . ويمكننا
تكرار ذلك بعدد السطور الفارغة المطلوبة .

2-8 مثال لبرنامج كامل

نقوم الآن بعرض مثال لبرنامج يبين لنا الأجزاء الثلاثة لبرنامج بلغة باسكال . يقوم هذا البرنامج (المبين بالشكل 1-7-2) بحساب مساحة المستطيل بمعلومية طوله وعرضه . ونستخدم فيه المتغيرات الكسرية التالية :

الطول = length

العرض = width

و area = المساحة .

ويتبين من جملة VAR أن هذه المتغيرات من النوع الكسري REAL. كما نلاحظ أن اسم البرنامج هو rectangle (أي مستطيل) . ويتكون الجزء التنفيذي من ست جمل ، هي :

1- الجملة الأولى :

```
WRITE ('enter length →');
```

وهي تقوم بطباعة النضيد:

```
enter length
```

أي أدخل الطول . والسهم هنا لغرض تحديد الموقع الذي يتم فيه كتابة البيانات المطلوبة .

2- الجملة الثانية :

```
READLN (length);
```

وهي تقوم بقراءة قيمة الطول .

3- الجملة الثالثة :

```
WRITE ('enter width→');
```

وتقوم بطباعة النضيد :

```
'enter width→'
```

4- الجملة الرابعة :

```
READLN ( width);
```

وهي تقوم بقراءة قيمة العرض width .

5-جملة التعيين :

area: = length * width

وهي تقوم بضرب الطول في العرض وتعيين الناتج للمتغير area.

6-الجملة :

WRITELN ('area =', area);

وهي تطبع المساحة على الصورة الأسية :

area = aEb

تنفيذ البرنامج :

عند تنفيذ البرنامج (باستعمال الأمر run في منظومة تربو باسكال) تظهر

على الشاشة العبارة :

enter length→ -

حيث يبين المؤشر (-) موضع إدخال البيانات المطلوبة ، فمثلا إذا كان

الطول 23 أدخلنا العدد 23 في الموقع المحدد .

بعد الضغط على الزر enter تظهر العبارة :

enter width→-

فإذا كان العرض المطلوب هو مثلا 45 ، نقوم بطباعة هذا العدد والضغط على

الزر enter لتحصل على الناتج :

area = 1.0350000000E + 03

وهي طبعا تعني :

area = 1035

شكل (1-7-2) : برنامج لحساب مساحة المستطيل .

```
PROGRAM rectangle;
VAR length : REAL;
    width : REAL;
    area : REAL;
BEGIN
    WRITE('enter length-->');
    READLN(length);
    WRITE('enter width-->');
    READLN(width);
    area := length*width;
```

```
WRITE ('area=' ,area:10:3) ;  
READLN ;
```

END .

نلاحظ عن هذا البرنامج ما يلي :

- 1- وجود الفاصلة المنقوطة في نهاية كل جملة ، و أيضا كفاصل بين المتغيرات في جملة VAR بعد تحديد نوعها .
 - 2- ليس من الضروري أن تبدأ الجملة عند سطر جديد ، ولكن يمكن وضع أكثر من جملة في سطر واحد ؛ لأن الفاصلة المنقوطة هي التي تفصل بين الجملة والتي تليها.
 - 3- ليس من الضروري إزاحة الجمل التي تقع بين BEGIN و END عددا من الفراغات إلى اليمين كما هو موجود في هذا البرنامج ، ولكن ذلك مفيد جدا في توضيح بداية الجملة المركبة ونهايتها . لاحظ أن كل كلمة BEGIN يجب أن تقابلها كلمة END ، وبواسطة ترك هذه الفراغات نستطيع تمييز بداية ونهاية جملة داخل جملة أخرى ، وذلك بوضع كلمة END في العمود نفسه الذي تقع فيه كلمة BEGIN المناظرة لها . تسمى هذه العملية بإدخال الأسطر Indentation وهي خاصية هامة في برامج لغة باسكال أو ما يسمى بصورة عامة بالبرمجة الهيكلية structured programming .
- وأخيرا يجب ملاحظة أنه هناك بعض الحالات التي لا تستوجب وجود الفاصلة المنقوطة في نهاية الجملة. مثال ذلك الجملة الأخيرة في البرنامج الواقعة قبل النهاية END .

2-9 توثيق البرنامج

لقد سبق وأن أشرنا إلى أهمية كتابة البرنامج بطريقة واضحة ومفهومة لمن يستخدمه ، بل أحيانا لا يستطيع كاتب البرنامج أن يفهم برنامجه - إذا لم يكن مكتوبا بأسلوب واضح - بعد مرور فترة طويلة من الزمن . وحتى يكون

البرنامج واضحا يجب أن تكون الأسماء فيه تدل على مسمياتها ، فاستخدام المتغير SALARY للدلالة على المرتب أفضل من استخدام المتغير S رغم أن كلاهما صحيح ، واستخدام المتغير StudentGrade أكثر وضوحا من استخدام SG مثلا .

وهناك وسيلة في لغة باسكال تمكننا من توضيح البرنامج والغرض من كل جزء أو جملة أو متغير ، وهي استخدام الأقواس { } حيث إن الكلمات التي تقع داخل هذه الأقواس في البرنامج سوف لن تعتبر جزءا تنفيذيا منه بل هي توضيح لقارئ البرنامج . كما يمكن استخدام الأقواس (* *) الرمز للغرض نفسه ، مثل :

```
{ main program }
```

أو

```
(* main program *)
```

وكمثال ، نرجع للبرنامج السابق الذي يقوم بحساب المساحة ، ونضيف إليه التوضيح :

```
{ a program to compute the area of a rectangle }
```

لاحظ أن هذه الجملة التوضيحية يمكن أن تكتب باللغة العربية على النحو التالي:

```
{ برنامج لحساب مساحة مستطيل }
```

وذلك بشرط توفر أنظمة تعريب (مثل النافذة NAFITHA) .

أخيرا نلاحظ أنه يمكن توثيق وتوضيح البرنامج في أي موقع نشاء من

البرنامج ، دون أن يؤثر ذلك عليه .

10-2 تمارين

1- أوجد الأخطاء اللغوية في البرنامج الموجود بالشكل (1-9-2).

شكل (1-9-2) : برنامج به أخطاء .

```
program tax
var income,tax, real;
begin
readln(Income);
TAX = .15*income;
Writeln('tax=';tax);
End;
```

2- اكتب برنامجا لحساب مربع عدد يتم إدخاله عن طريق لوحة المفاتيح عند التنفيذ . وبين في هذا البرنامج :

- جملة تعيين
- جملة إدخال
- جملة إخراج

4- بين فيما يلي الثابت الكسري من الثابت الصحيح :

2.0E+5
1.0
-2
3.5
12345

5- ضع الأعداد التالية على صورة كسر عشري :

3.1672E-3
-2.135E+5
1.56E+3

5- أي من الأسماء التالية جازز كأسماء متغيرات في لغة باسكال؟

var	ABCD	e5	2D
a*b	Book_title	HeIsOld	MyName

6- أي من العبارات الحسابية تعتبر صحيحة في لغة باسكال ؟

- $T + U * V$
- $5 + X + S$
- $6(A + B)$
- $312.45 + 15.0/6.1$
- $((15.0 + E5)/GH)/X$

7- أوجد قيمة العبارات الحسابية التالية :

- a) $3 + 4 * 6 + 2$
- b) $12.0 + 16.2/2.0$
- c) $3.1 + 9.0/3.0 * 4.0$
- d) $(1.0 + 5.0) * 3.2$
- e) $(4 * (14 + 5) + 3) * 2$
- f) $(17MOD5) + (14DIV5)$

8- اكتب برنامجا لحساب مساحة مثلث من الصيغة :

$$2 \setminus \text{المساحة} = \text{القاعدة} * \text{الارتفاع}$$

بحيث يكون الناتج على الصورة التالية

$$\text{AREA} = \text{xxxx.xx}$$

حيث كلمة AREA تعني المساحة .

ويجب أن يكون البرنامج واضحا في عملية الإدخال بحيث يتم إدخال القاعدة ،

عندما تظهر على الشاشة العبارة :

ENTER BASE

ويتم إدخال الارتفاع عندما تظهر العبارة :

ENTER HIGHT

9- ما معني الكلمات التالية :

Syntax Diagram

Identifier Word

Variable

Real number

Assignment Statement

Input / Output

Character

True / False

Boolean

Arithmetic Expression

10- إذا كان I متغيرا صحيحا و R متغيرا كسريا و B متغيرا منطقيا و C متغيرا

رمزيا ، بين أي من الجمل التالية غير جائزة في لغة باسكال ولماذا ؟

- | | | | |
|----|---|----|---------------|
| a) | I | := | 2345 |
| b) | I | := | 4/5 |
| c) | C | := | I |
| d) | C | := | 'C' |
| e) | B | := | True |
| f) | B | := | '*' |
| g) | R | := | 45.6 |
| h) | R | := | '45.6' |
| I) | R | := | 10 DIV (-3) |
| j) | I | := | (-15) MOD 4.0 |

11- اكتب برنامجا لحساب عدد الساعات والدقائق والثواني في عدد من

الثواني ، بحيث يكون الناتج على النحو التالي :

عدد الساعات = xxx

عدد الدقائق = xx

عدد الثواني = xx