لغة البرمجة باسكال

الدكتورة سميرة عبد الله
منظمة الطاقة الذرية

1- المقدمـــــــــة :

يتردد كثيرا اسم هذه اللغه في اوساط مستخدمـــــي الحاسبات الالكترونيه وقد صدرت منشورات كثيرة بيـــــن مروج وناقد لهذه اللغه .

فما هي لغة باسكال ؟

باسكال هي لغة برمجة للاغراض العامة اى علمية وادارية . تم وضع الصيغ الاولى لهذه اللغه من قبل العالـم NIKLAUS WIRTH في " ETH , ZURICH " وقد قدمت على انهـــا اداة لتعليم البرمجه وعمل مترجمات لكفائتها ومرونتها .

خلال سنة 1971 ظهر مترجم (COMPILER) لهذه اللغه علـــى الحاسبه CDC-6600 وقد كتب بنفس اللغه ويبلغ حوالـــي 6000 سطر ووصف بانه سهل القراءة والتغيير ، وفي نهـــاية سنة 1973 اصدر العالمان C.HOARE و N.WIRTH تعريفـــات بديهيه لهذه اللغه وفي سنة 1974 صدر كتاب من قبـــل العالمين N.WIRTH & K.JENSEN يتضمن تقريرا عن هذه اللغه وقد اعتبر مصدرا الى " STANDARD PASCAL " .

وقد توالت الكتابات والتعديلات لهذه اللغه الى يومنا هذا.

سنتناول الفقرات التاليه شرح موجز عن البيانات والفعاليات وكيفية التعبير عنها بهذه اللغه ، علما بانى لم انطرق الى البرامج المساعده والدوال المعرّفه فى هذه اللغه STANDARD PROCEDURE & FUNCTION ولا الى العمليات الحسابيه والمنطقيه لان الهدف هو التعرف على الخطوط العريضه لهذه اللغه فقط .

2 – الهيكلة العامة لبرنامج مكتوب بلغة باسكال :

ان اى برنامج مكتوب بلغة من لغات الحاسبة الالكترونيه يحتوى على جزئين اساسيين هما وصف للفعاليات التي يجب تنفيذها وتتم بواسطة " STATEMENTS " ووصف للبيانات " DATA " اللازمة لتنفيذ هذه الفعاليات وتتم بواسطة "DATA DECLARATIONS & DEFINITIONS" لنأخذ برنامجا بسيطا مهمته قراءة رقمين حقيقيين وطبع مجموعهما , فعند استخدام لغة باسكال تكون الخطوات كالاتي :

```
PROGRAM ADD ( INPUT , OUTPUT ) ;
VAR FIRST , SECOND , SUM ; REAL ;
BEGIN ( * MAIN PROGRAM * )
READ ( FIRST , SECOND ) ;
SUM : = FIRST + SECOND ;
WRITE ( SUM )
END.
```

يحتوى البرنامج على المقدمه HEADING والجسم BLOCK

وينتهي بنقطة " . " (انظر الشكل رقم (1) . ان المقدمة تفيد لتسمية البرنامج ولتعداد الملفات التي يستخدمها البرنامج لاجل اخراج وادخال المعلومات من والى الحاسبه . اما الجسم فيحتوى على 6 اجزاء ، الخمسة الاولى منها اختيارية ، وهذه الاجزاء حسب تسلسلها بالبرنامج كما يلي:

1 . الاعلان عن العلامه ويتم بالكلمه LABEL

2 . تعريف الثابت ويتم بالكلمه CONST

3 . تعريف النوع ويتم بالكلمه TYPE

4 . الاعلان عن المتغير ويتم بالكلمه VAR

5 . الاعلان عن البرنامج المساعد او الدالة ويتم بالكلمتين "PROCEDURE" او " FUNCTION " على التوالي .

6 . قسم التعليمات STATEMENTS ، وهذه تنحصر بين الكلمتين BEGIN و END ، انظر الشكل (2) .

ان كل جملة تنحصر بين الرمزين " (** و " *) " تعتبر جملة توضيحية " COMMENTS " وليست تنفيذية .

3 . العلامه LABEL

وهي عبارة عن عدد صحيح موجب (بدون اشارة) لا يزيد

عن اربعة ارقام مثل 300 , 2103 . ان كل علامة تستخدم في البرنامج يجب ان تكون بالشكل التالي :

LABEL 3 , 20

وسوف يردنا استخدامها في بند التعليمات وخاصة تعليمة التفرع GOTO .

4 . تعريف الثابت CONSTANT

ان تعريف الثابت يعني اعطاء تسمية الى ثابت يراد استخدامه في البرنامج وهذا يعطي البرنامج صفة الوضوح وسهولة التوثيق والتغيير .

مثال :

```
CONST ALPHA = - 5;
BETA = ' * CONSTANT * ' ;
```

5 . البيانات وانواعها

5-1 تعريف نوع البيانات

بالامكان وصف نوع البيانات عند الاعلان عن المتغير

اما بشكل مباشر او بالاشارة الى تسمية تدل على ذلك النوع .

فمن الممكن تعريف نوع البيانات التي يستخدمها البرنامج بالشكل التالي :

```
TYPE BEET = ARRAY [1..8]  OF CHAR ;
```

2-5 انواع البيانات

ان البيانات تكون ضمن الانواع التاليه :

أ- النوع البسيط SIMPLE TYPE

1. النوع القياسي STANDARD TYPE

وهو على اربعة انواع.

1.1 INTEGER وتشمل جميع الاعداد الطبيعية

2.1 REAL وتشمل جميع الاعداد الحقيقة

3.1 CHAR وتشمل جميع الرموز المستعملة في الحاسبه وهي الارقام من 0 الى 9

والحروف من A الى Z اضافة الـــى

بعض الرموز الخاصه مثل + ,% ,$,

.... الخ .

4-1 BOOLEAN وهي عبارة عن العددين 1,0 او

القيم المنطقية FALSE,TRUE .

2 . الفترة INTERVAL TYPE

وتسمى ايضا SUBRANGE وهى مجموعة جزئيـــه مـن

نوع عددى من البيانات ويعرف بتحديد عنصريـــن

كحد اعلى وحد ادنى .

مثـــــال :

```
TYPE DAY = 1..31 ;
```

3 . العددى SCALAR TYPE

ويسمى ايضا ENUMERATED TYPE وهى يعرّف مجموعـة

مرتبة من العناصر بترقيم اسمائها.

مثـــل :

```
TYPE OPERATION = ( ADD , SUB , MULT , DIV );
```

حيث ان ADD تأخذ القيمه 0

و SUB=1 .... الخ

وان BOOLEAN معرفة من قبل اللغه على انهـــــا

TYPE BOOLEAN = (FALSE , TRUE) ;

ب – البيانات الهيكليه STRUCTURED TYPE

وهي مؤلفة من تراكيب محددة واضحه من حيث النوع والحجم وتتكون من الانواع التاليه :

1 – النسق ARRAY

TYPE TAB = ARRAY [1..100] OF INTEGER;

ان المتغير TAB عبارة عن نسق ذو 100 خليـــة او تركيب كل خلية تمثل عددا طبيعيا وان العبـــارة التاليه :

TAB [15] : = 20 ;

تعني التركيب رقم 15 من المتغير TAB يعـــادل 20

2 – الملف FILE

ان كلمة FILE في هذه اللغه تعني مجموعة من البيانات المتشابة والمتسلسلة , مثال :

```
TYPE F1 , F2 = FILE OF CHAR ;
```

ان كلا من المتغيرين F1 , F2 عبارة عن ملف اى سلسلة من الرموز ومن التعريف نجد ان سعة الملف او عدد مركباته غير محدد , وهذا ما يميز هذا النوع من البيانات عن النسق .

3 – المجموعه SET

مثال :

```
TYPE S1 = SET OF' A ' ... ' Z ' ;
```

ان مجموعة القيم التي يأخذها المتغير S1 هي كل المجموعات الجزئيه التي يمكن ان تتكون من الحروف من A الى Z ويضمنها المجموعه الخاليه فمن الممكن ان تكون :

```
S : = [ ] ;          مجموعة خالية
```

او

```
S : = [ ' S ' , ' E ' , ' T ' ]
```

4 - السجـــل RECORD

ويحتوى على عدد محدود من المركبات او الحقول FIELDS ليس بالضرورة ان تكون من نوع واحد كما هي في النسق ،

```
TYPE GAMA = PACKED ARRAY [1..10] OF CHAR ;

TYPE DATE = RECORD

                DAY :1..31 ;

              MONTH:1..12 ;

              YEAR:1900..2000

    END ;

PERSON = RECORD

          FNAME , INAME : GAMA ;

          SEX : ( MALE , FEMALE )

          BDAY : DATE

    END ;

VAR P : PERSON ;
```

نرى ان السجل PERSON مؤلف من ثلاثة حقول وان الحقل الثالث هو بدوره سجل مؤلف من ثلاثة حقول ايضا . بالامكان ان نجد العبارات التاليه في البرنامج

```
P. FNAME : = ' HASSAN ';
P. LNAME: = ' AHMAD   ';
P. SEX : = MALE ;
P. BDAY . DAY : = 23 ;
P. BDAY . MONTH : = 5 ;
P. BDAY . YEAR : = 1950 ;
```

ج - المؤشر POINTER

ان المتغير الساكن STATIC VARIABLE هو الذى يعلن عنه في البرنامج باسم معين ومن ثم يرجع له بواسطة ذلك الاسم، ونسميه ساكن لانه موجود ضمن ذاكرة الحاسبه طيلة فترة تنفيذ ذلك الجزء من البرنامج الذى اعلن فيه المتغير . اما المتغير الحركي DYNAMIC VARIABLE فهو لا يظهر

بوضوح في جزء الاعلان عن المتغيرات و لايمكن الرجوع اليه بواسطة تسمية معينة وانما بواسطة مؤشر يشير الى مكان معين في الذاكرة حيث يخزن هذا المتغير لنفرض ان P هو متغير من نوع المؤشر ومرتبط او يؤشر الى البيانات من نوع T . الاعلان التالي :

VAR P :↑T

او VAR P: @ T

يعني ان P هو مرجع الى متغير من نوع T وان P او P@ يمثل هذا المتغير .

NEW (P) تعني توليد وحجز مكان بالذاكرة لمتغير جديد من نوع T وخزن عنوانه في P .

DISPOSE (P) تعني الغاء المكان المحجوز بالذاكرة لعدم الحاجه اليه .

P=NIL تعني ان P لا يؤشر الى اى عنصر اى لا يخزن اى عنوان .

6 – الاعلان عن المتغير :

ان المتغيرات الداخله في اى برنامج مكتوب بهذه

اللغه يتم تعريفها للحاسبه بالصيغه التاليه :

VAR NAM1 NAM2 ..... : TYPE ;

حث NAM1 , NAM2 هي اسماء المتغيرات وان الغرض من كلمة TYPE هو وصف مجموعة القيم التي من الممكن ان تأخذها هذه المتغيرات , ففي اول برنامج كتبناه أعلن عن ثلاثة متغيرات (FIRST,SECOND,SUM) من نوع الاعداد الحقيقيه REAL .

7 – التعليمات والاوامر STATEMENTS

وهي عبارة عن فعاليات منطقية تنفذ من قبل الحاسبة ومن الممكن ترقيمها بـ LABEL اذا اريد الرجوع اليها باستخدام الامر GOTO .

الشكل رقم (3) يتضمن انواع التعليمات ومن الممكن تقسيمها الى نوعين وهما :

أ– التعليمات البسيطه SIMPLE STATEMENT

وهي تتكون من تعليمة واحدة فقط وتتضمن اربعة انواع من التعليمات وهي :

1- تعليمة تعيين ASSIGNMENT STATEMENT

وتعني ابدال قيمة المتغير او الدالة بقيمة جديدة متمثله بناتج العبارة الجبرية EXPRESSION

مثال :

```
alpha : = ( 5 + b ) / a ;
```

2- تعليمة برنامج مساعد PROCEDURE STATEMENT

وهي تفيد لتنفيذ البرنامج المساعد وذلك بذكر اسم البرنامج متبوعا بالقيم الحقيقية للمتغيرات ان وجدت والتي تأخذ قيم المتغيرات الصوريه التي تدرج عند تعريف البرنامج المساعد

3- تعليمة تفرع BRANCH STATEMENT

وهذه التعليمة تدل على ان الخطوة التالية للتنفيذ هي العبارة المرقمة ازائها والتي توجد في مكان ما من البرنامج ضمن ذلك 'BLOCK'

مثال

```
GOTO 50 ;
```

ب- التعليمات الهيكلية STRUCTURED STATEMENTS

وتتألف من عدد من التعليمات والتي تنفذ اما بشكل تتابعي كما في التعليمة المركبة او بتحقيق شرط معين كما في التعليمة الشرطية او تنفيذ بشكل تكراري كما في تعليمة التكرار وانواعها الثلاثة هي كالاتي :

1- التعليمة المركبة COMPOUND STATEMENT

وهذه مجموعة من التعليمات محددة بالكلمتين BEGIN و END وتنفذ تتابعيا .

مثال :

```
BEGIN
Z : = X ; X : = Y ; Y : = Z
END;
```

2- تعليمة تكرار ITERATION STATEMENT

وتتضمن ثلاثة انواع من التعليمات وهي :

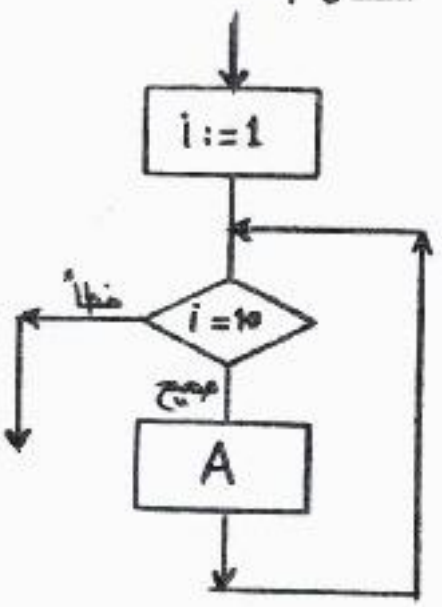
WHILE STATEMENT 1.2

WHILEE < EXP > DO < STATEMENT >

ان EXP تعني عبارة منطقية وان STATEMENT تعني تعليمة واحدة بسيطه او تعليمة مركبه .

مثال :

```
I : = 1 ;
WHILE  I<= 10 DO
A  BEGIN
     TAB(I):=0;
     I:=I+1
   END
```



هنا يتكرر تنفيذ الجزء A من البرنامج ما دامت العبارة المنطقية I<= 10 صحيحه .

REPEAT STATEMENT 2.2

REPEAT < Statement > UNTIL < exp >

مثـــال :

```
I:=1

A  REPEAT
      tab ( i ) : = 0 ;

      i : = i + 1
   UNTIL i>10 ;
```

يتكرر تنفيذ الجزء Aمن البرنامج ما دامـــــــــت العبارة المنطقيه غير صحيحه .

FOR Statement 3.2

```
FOR <V> : = <I> TO     <B> DO <Statement>
                DOWNTO
```

حيث ان V تعني control Variable وان I تعني القيمة الاولية لهذا المتغيير و B هي القيمة النهائية للمتغيير .

مثال :

```
FOR i = 1 10  DO

tab( i ) : = 0 ; ..... ( 1 )
```

يتكرر تنفيذ العبارة 1 حتى تصبح قيمة i اكبر من 10 اى تخرج عن الحد المذكور .

3- تعليمة شرطيه CONDITINAL STATEMENT

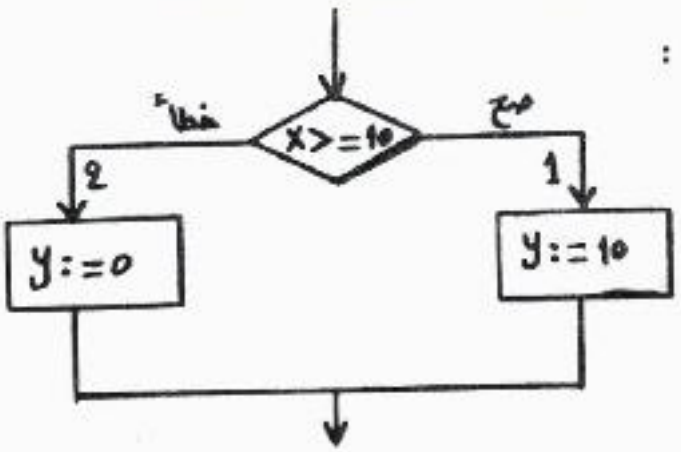
وتكون على نوعين :

IF STAT. 1-3

IF <exp> THEN <Statement>

او IF <exp> THEN <Statement> ELSE <Statement>

مثـــال :



IF X>= 10 THEN Y : = 10 ELSE Y: = 0

يكون تنفيذ العبارة رقم 1 مقرونا بتحقيق الشـــرط X> = 10 والا فستنفذ العبارة رقم 2 .

CASE <exp> OF

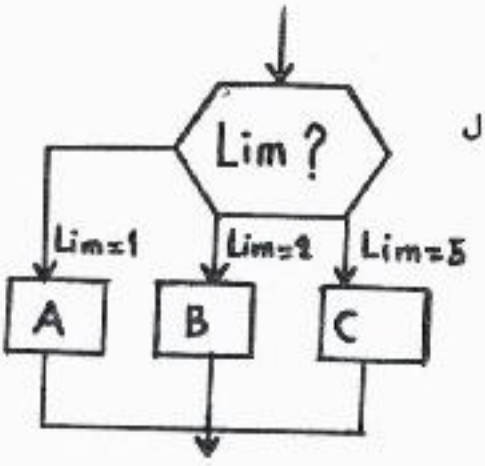
<label> : <stat.> ;

<label> : <stat.> ;

.
.
.

END

مثـــال




```
CASE LIM OF
1 : RESULT : = RESULT + VALUE ; ... ( A )
2 : RESULT : = RESULT * VALUE ; ... ( B )
5 : RESULT : = RESULT - VALUE ... ( C )
END ;
```

ان التعليمية التي ستنفذ هي التي ترقينها بعـــــادل القيمة الحقيقية للمتغير LIM فاذا كانـــــت LIM=10 مثـــلا فان تأثير التعليمه غير معرّف .

8 – البرامج المساعدة والدوال : PROCEDURES & FUNCTIONS

عندما يراد كتابة برنامج ما من المستحسن تقسيمه الى مقاطع منطقية كل مقطع يختص بعمل ما ثم كتابة برنامج لكل مقطع على شكل برنامج مساعد او دالة . ان هذه العملية توفر للبرنامج الوضوح وسهولة التغيير وتفادى تكرار كتابة المقاطع التي يحتاجها البرنامج الاصلي عدة مرات ، اضافة الى ذلك فان لغة باسكال توفر امكانية اشتراك اكثر من برنامج مساعد بجزء الذاكرة المخصص للمتغيرات المحلية LOCAL ، وهذا يؤدي الى اقتصاد في الذاكرة .

يعرف البرنامج المساعد في هذه اللغه بالصيغه التاليه :

```
PROCEDURE game( V 1 : INTEGER ; VAR V2 : REAL ) ;

{ .
  .   bloc
  .
```

حيث تفيد المقدمه لتسمية البرنامج المساعد لاستخدام هذه التسمية في تعليمة البرنامج المساعد عند الحاجه الى تشغيلية وان V1 ، V2 هي متغيرات صورية Formal Parameters وهذه تكون على ثلاثة انواع: Variable Parameter

وتكـــــون مسبوقـــــة بالكلمـــــة VAR كمـــــا فـــــي V2 PROCEDURE PARAMETER وتكون مسبوقة بالكلمـــه PROCEDURE او FUNCTION .

Value Parameter وهذه لا تسبقها اى كلمة كما في V1 امـــا BLOC فهو يحتوى على الاجزاء كما وضعت في الفقرة الثانيـة عند شرح الهيئة العامة للبرنامج اما الدالة فتعرف بنفس الطريقة مع ذكر نوعها .

```
FUNCTION theta ( VI INTEGER ) : INTEGER
{ :  bloc
```

9 – ادخال واخراج المعلومات INPUT , OUTPUT

ان اخراج وادخال المعلومات من والى الحاسبه يتـــم بواسطة البرامج المساعدة WRITE,READ على التوالـــــي ولغرض تنفيذ هذه العملية يجب توفر ملفين خاصيـن همـــا ملف الـ INPUT وملف الـ OUTPUT .

الغرض من الملف الاول هو لاستقبال وخزن المعلومات الوارده من المحيط الى الحاسبة ، والثاني لخزن النتائج التي تمت نتيجة لتشغيل البرامج الى المحيط الخارجي (كأن يكـــون مثقب بطاقات او قرص مغناطيسي ..... الخ) .

وهذان الملفان معرفان من قبل اللغه على انهما

VAR INPUT, OUTPUF : FILE OF CHAR ;

لنفرض ان f هو ملف وان V1 ، V2 هي متغيرات من نوع

(REAL او CHAR او INTEGER) فان :

READ (V1 , V2) يكافيء READ (INPUT , V1 , V2)

READ(f , V1 , V2) يكافيء READ (f , V1) ; READ

(f , V2)

ان البرنامج المساعد READLN مشابهه الى READ ويستعمل عندما يراد الانتقال الى بداية السطر الجديد بعد انتهاء القراءة . وكذلك

WRITE (V1 , V2) يكافيء WRITE (OUT PUT , V1 , V2)

WRITE (f , V1 , V2) يكافيء WRITE (f , V2) ; WRITE

وان البرنامج المساعد WRITELN (f) تعني وضع علامة نهاية السطر في الملف f .

الاستنتاجات والملاحظـــــــات :

أن المتخصصين في مجال البرمجة يأسفون كثيرا لعدم توفـــر مزايا معينه في هذه اللغة اعتادوا أن يجدوها في لغات آخـــرى ومنها :

1- تسلسل الرموز Concatenation of Strings .

2- اجراء ات العمليات الحسابية على متغيرات منطقية .

3- تحويل البيانات من نوع الى آخر اوتوماتيكيـــا AUTOMATIC TYPE CONVERSIONS

4- الاعلان الضمني عن المتغيرات Derault Declaration .

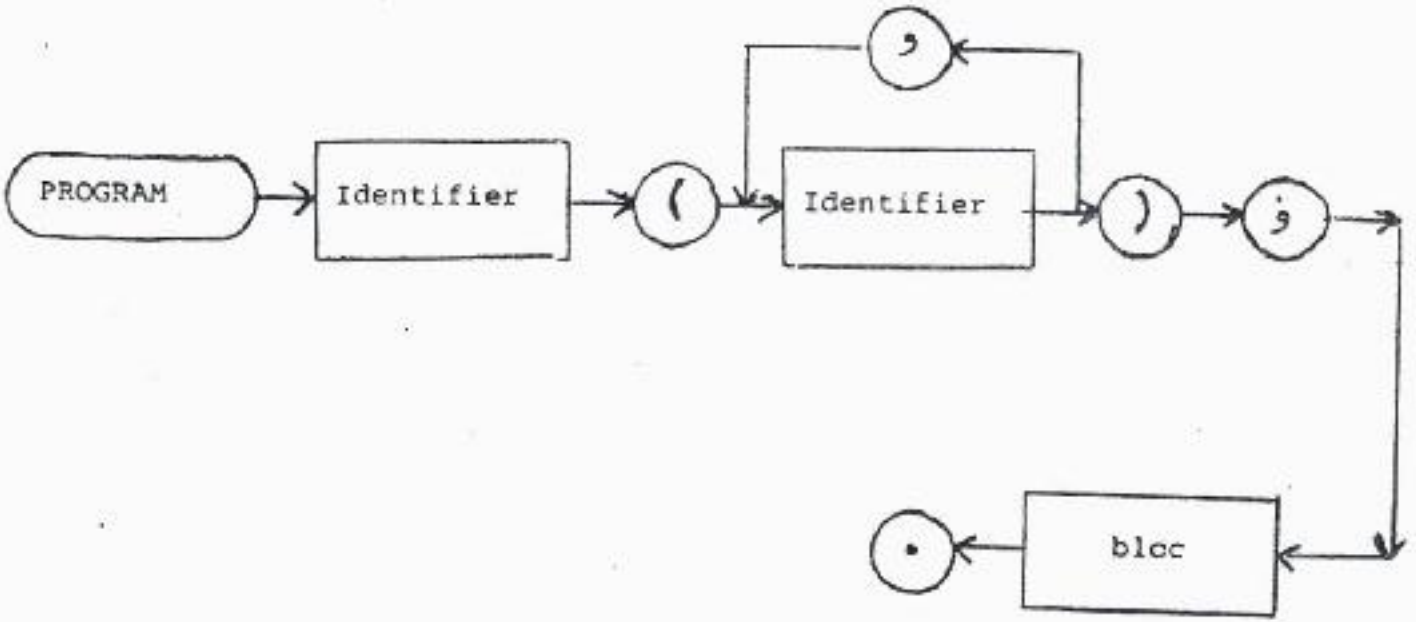
5 - النسق الديناميكي Dynamic Arrays .

" أن هذا النقص ليس سببه النسيان وانما هو حذف مقصود" كما عبر عن ذلك العالم WIRTH عندما سؤل من قبل المعنيين في احيان كثيرة يكون وجود مثل هذه التسهيلات في اللغة مؤديا الى عدم الوضوح وصعوبة تتبع الاخطاء والى عدم الوصول الى النموذج الجيد للبرمجة ، أضافة الى تعويد المبرمجين على الاتكاليـــة غيـــر المجدية ، فبالامكان الوصول الى سد النقـــص في هذه اللغة بأضافة برامج مساعدة .

كانت هناك اختيارات كثيرة مطروحة امام مصممي هذه اللغة لعمل

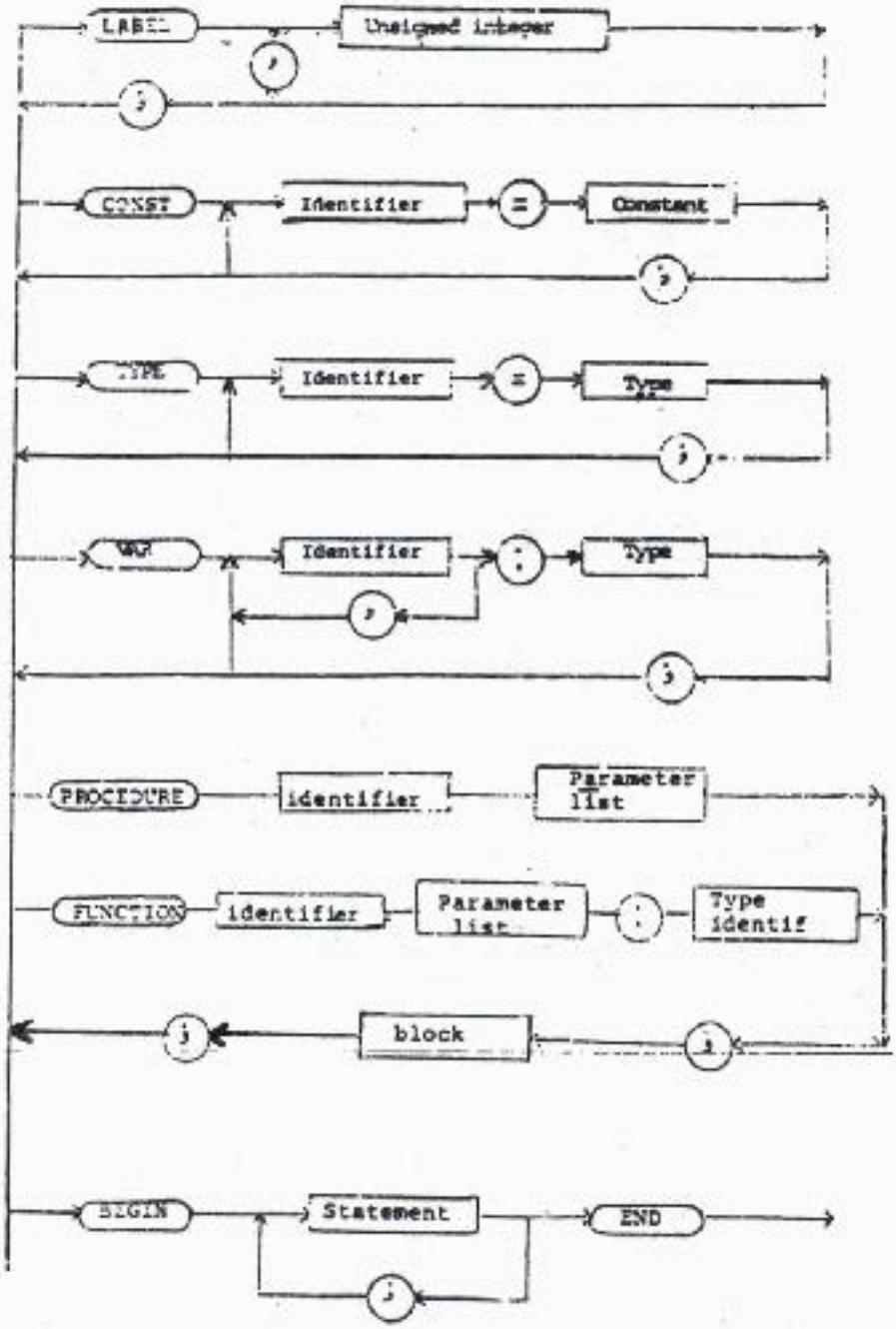
مترجم مركز نسبيا وفعال واقتصادي لكلا النوعين من المستعملين الاول الذي يكتب برنامج قصير باستعمال تراكيب قليلة من هذه اللغة والاخر الذي يكتب برنامج طويل ويستعمل كل مميزات اللغة ان هذه اللغة تستعمل في وقتنا الحاضر لاغراض مختلفة نظرا لسهولتها ووضوحها وكفائتها ، فقد اثبتت جدارتها لعمل المترجمات وذلك لكونها توفر البرمجة الهيكلية "Structured Programming" اضافة لكونها لغة معتمدة لتعليم البرمجة .

```
program Prime2 (Input, Output);
   (*************************************************************
   * Given a positive number N, this program is to locate the  *
   *     Nth prime number.                                     *
   * J is an odd integer, a candidate for a prime.             *
   * Prime is an array of the primes already found.            *
   * M is an index into the array of primes.                   *
   * PrimesFound is the number of primes already located.      *
   * HasNoFactor is a Boolean flag, used to record the out-    *
   *     come of testing J for primeness.                      *
   *************************************************************)
   const
      Maxindex = 1000;
   type
      Indextype = 1..Maxindex;
   var
      N, M, PrimesFound : Indextype;
      J : Integer;
      Prime : array [Indextype] of Integer;
      HasNoFactor : Boolean;
   begin
      (*******************************************************
      *  initialize the table of primes to contain 2        *
      *  and the first odd prime.                           *
      *******************************************************)
      Prime[1] := 2;  Prime[2] := 3;
      PrimesFound := 2;
      Read (N); (* if N is out of range, a Read error will occur *)
      J := Prime[PrimesFound] + 2;
      while PrimesFound < N do
         begin
            HasNoFactor := True;  M := 2;
            while HasNoFactor and (Sqr(Prime[M] <= J) do
               if J mod Prime[M] = 0 then
                  HasNoFactor := False
               else
                  M := succ(M);
            if HasNoFactor then
               begin (*** J is prime ***)
                  PrimesFound := PrimesFound + 1;
                  Prime[PrimesFound] := J;
               end;
            J := J + 2;
         end;
      Writeln ('PRIME[', N:3, '] =', Prime[N]);
   end.
```

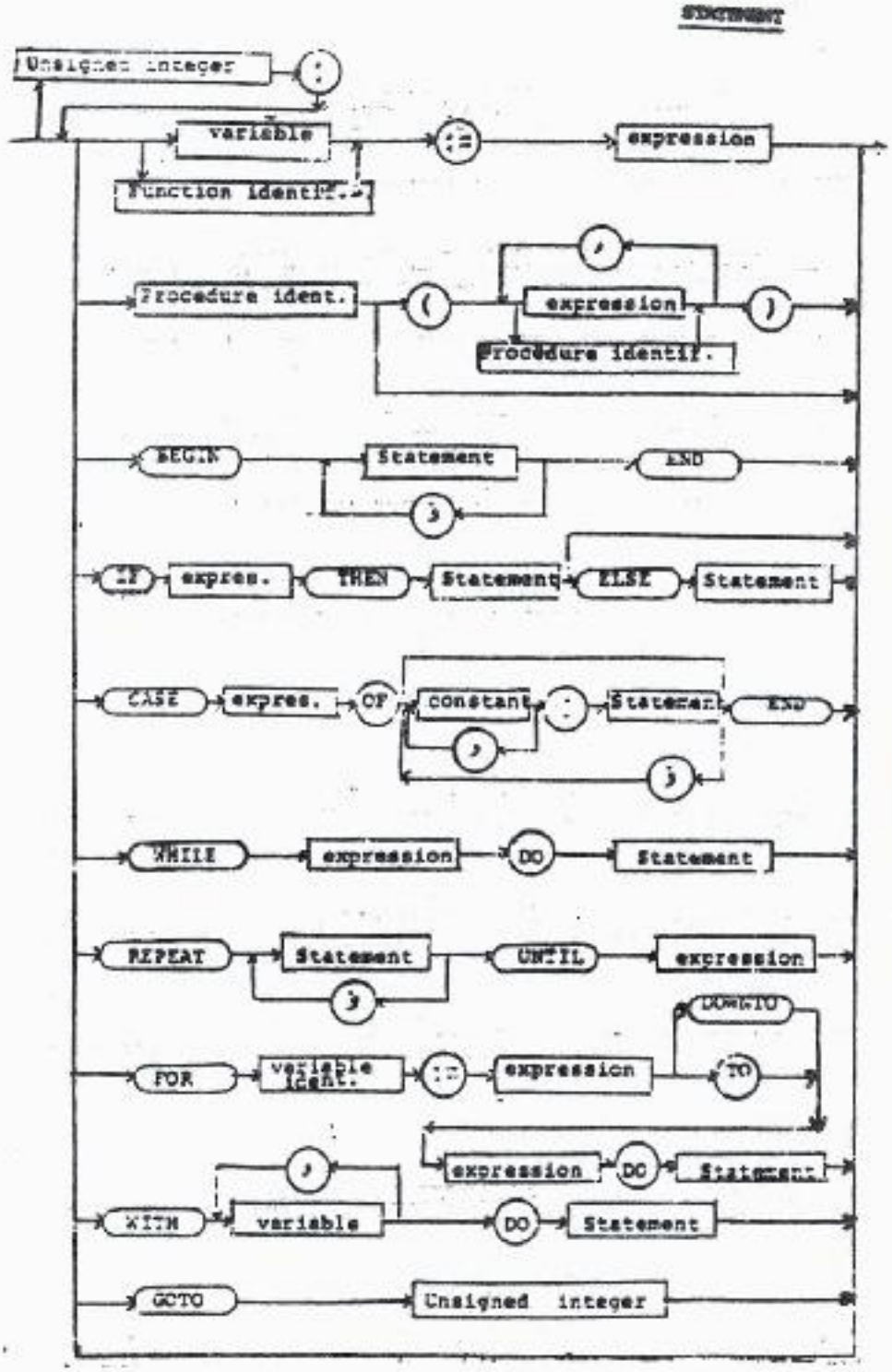


شكل رقم (I)

Block



شكل رقم (2) المجسم



شكل رقم (3) التعليمات

المصـــــــــادر

1. S. KUBBA These presentee A L Universite de paris - sud ,Centre D'Orsay.
   System de Simulaion du langage PASCAL sur mini-ordinateur
   HP-9830.
2. Kathleen Jensen-Niklaus Wirth
   PASCAL User Manual and Report(Second Edition)
   springer-Verlay .
3 J.M. CROZET - D. SERAIN
   LE Langage PASCAL.
   MASSON .
4 Alfred C. Hartmann
   Leture Note in Computer Science
   Springer - Verlay .
5. Kenneth, L. Bowles
   problem Solving using PASCAL
   Springer- Verlay .

6. Jim Walsh and John Elder

Introduction to PASCAL

C.A.R. Hoare Series Editor.

7. 01 Informatique No. 125 , October 1978

PASCAL, Un langage Tres evolue

(pages 59-63)

8. L'Ordinateur Individual No. 7-Mai 1979.

(pages 53-56).

9. Structured programming and problem solving With PASCAL

Richard B. Kieburtz

prentice- hall, INC .

10. 01 Informatique No. 125 , Octobre 1978

PASCL , Un Langage tre's evol'e

(pages 59-63)