جامعة حماة
كلية العلوم في مصياف / السنة الأولى

المادة: لغات البرمجة

المحاضرة الخامسة: الإجراءات والدوال في لغة باسكال

Procedures an Functions in Pascal

العام الدراسي ٢٠١٩- ٢٠٢٠

الإجراءات والدوال في لغة باسكال

- يفضّل تقسيم البرنامج إلى عدة كتل برمجية (برامج جزئية) يكون لكل كتلة وظيفة محددة، فمبدأ التجزئة هدفه التحكم والسيطرة، حيث أن قاعدة (فرق تسد) مفيدة جداً في البرمجة للأسباب التالية :

١. سهولة فحص كل كتلة على حده، لسهولة معرفة موقع الخطأ في البرنامج بدقة من قبل المبرمج .

٢. سهولة تعديل الكود عند الحاجة .

٣. لمنع التكرار، فإذا اضطررنا إلى تكرار مجموعة من التعليمات التي تقوم بمهمة محددة عدة مرات فمن الأفضل وضعها في برنامج جزئي ونكتفي بذكر اسم هذا البرنامج الجزئي عند الحاجة إليه .

٤. سهولة فهم الكود (النص البرمجي) عند قراءته مبسّطاً ومقسّماً .

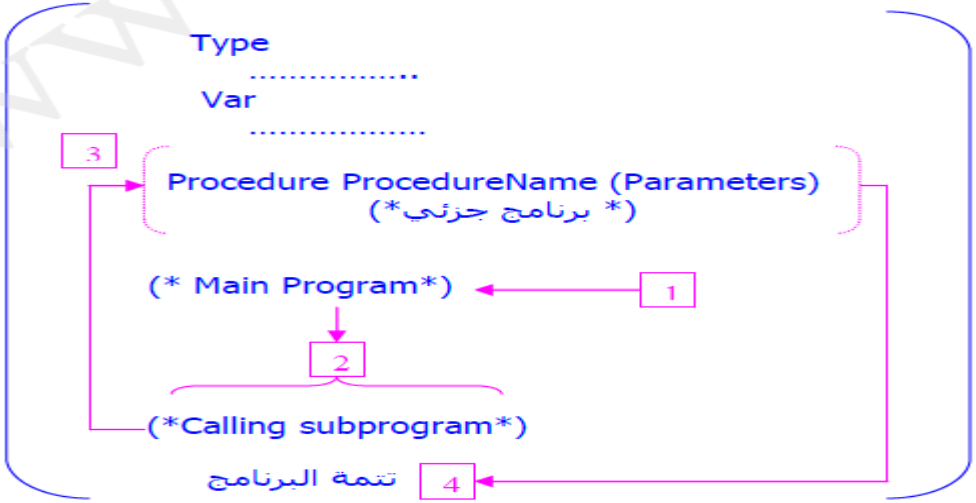
- تسمى هذه الكتل البرمجية بالبرامج الجزئية وتتوضع هذه البرامج Sub Programs في الجسم الرئيسي للبرنامج بعد قسم التعريفات **Var**، و هي على نوعين :

١. الإجراء **Procedure**

٢. التابع **Function** .

الإجراءات Procedures

- **الإجراء Procedure:** هو برنامج جزئي يحتوي على مجموعة من الأوامر والتعليمات البرمجية التي تقوم بتنفيذ مهمة محددة، يتم تعريفه باسم في بداية البرنامج الرئيسي، ويتم استدعاءه عن طريق كتابة اسم الإجراء متبوعاً بفاصلة منقوطة أثناء كتابة تعليمات البرنامج الرئيسي .
- **استدعاء الإجراء:** عملية يتم من خلالها انتقال تنفيذ البرنامج من العبارة المعالجة قبل الاستدعاء إلى تعليمات الإجراء المستدعى، فعندما نستدعي الإجراء تنفذ مجموعة التعليمات المحتواة في هذا الإجراء ، وبعد الانتهاء من تنفيذ آخر تعليمة من تعليمات الإجراء ينتقل التحكم إلى النقطة التي تأتي مباشرة بعد أمر استدعاء الإجراء في البرنامج الرئيسي . و المخطط التالي يوضح كيف يتم ذلك :



الإجراءات Procedures

- البنية العامة للإجراء تشبه إلى حد كبير بنية البرنامج الرئيسي في لغة باسكال، ويتكون الإجراء من الأجزاء التالية :

أولاً : رأس الإجراء Procedure Head : يبدأ الإجراء بالكلمة المحجوزة Procedure ويتبعها اسم الإجراء Procedure_Name ويجب أن ينتهي رأس الإجراء بفاصلة منقوطة لنفصله عن باقي أجزاء الإجراء، يتبع الاسم بلائحة من الوسطاء Parameters. و بذلك يكون الشكل العام لترويسة الإجراء:

Procedure name (**parameter1 :** type ; **parameter2 :** type ;.....) **;**

ثانياً : قسم التصريح Declaration Section: يملك كل إجراء قسم تصريح خاص به var، يمكن أن نصرح فيه عن ثوابت أو متحولات أو أنماط مختلفة ، أو أن تعرف فيه إجراءات وتوابع أخرى أيضاً. و طريقة التصريح عن هذه الأنواع تخضع لنفس الطريقة المستخدمة في البرنامج الرئيسي مع بعض الفوارق البسيطة في إمكانية الوصول إلى التعريفات والتصريحات حيث لا يمكن استخدام هذه المتحولات إلا ضمن التابع نفسه لأنها متحولات محلية.

ثالثاً: جسم الإجراء Procedure Body : يحتوي جسم الإجراء على مجموعة من التعليمات والأوامر البرمجية (إدخال، إخراج، عمليات حسابية و منطقية) التي تقوم بتنفيذ المهام المطلوبة منه (الأوامر المتبوعة بفاصلة منقوطة) . يبدأ جسم الإجراء بـ Begin وينتهي بـ End; الموكلة إليه .

قواعد عامة عن الإجراءات Procedures

- يتم الإعلان عن الإجراء خارج الجسم الرئيسي للبرنامج باستخدام الكلمة procedure .
- يجب إعطاء اسم مميز لكل إجراء procedure_ name .
- الإجراءات لها بداية begin ونهاية end; خاصة بها.
- الإجراءات يجب أن تكون معرّفة دائماً قبل المكان الذي يتم استدعائها منه، أي يجب أن نقوم بالإعلان عن الإجراء ثم استدعائه ولا يجوز استدعاء الإجراء قبل الإعلان عنه .
- **مثال**: طريقة الإعلان و الاستدعاء لإجراء اسمه Hello يطبع الكلمة "Hello" على الشاشة .

```
Program Proced1;
procedure Proc1;
        begin
        Writeln('Hello');
        end;
Begin
 Proc1;
End.
```

القالب العام لتعريف الإجراء ضمن برنامج باسكال ←

لاستخدام الإجراء يجب أن نقوم باستدعائه من خلال اسمه المميز في جسم البرنامج (مكان كتابة أوامر البرنامج)

الإجراءات Procedures

- في المثال التالي سوف يتم استدعاء إجراء داخل إجراء آخر .

```
Program PROCEDURES;

Procedure Proc1;
        begin
        Writeln('Hello');
        end;

Procedure Proc2;
        begin
         Proc1;
         Writeln('Haysam');
        end;
Begin
 Proc2;
End.
```

مثال: طباعة عبارة Hello

بواسطة إجراء ضمن إجراء ⬅

ناتج الطباعة:

Hello

Haysam

الإجراءات Procedures

الوسطاء (Parameters): يفضل أن نحدد للإجراء المتحولات اللازمة لعمله عند استدعائه، ويتم ذلك في بداية الإجراء حيث يملك رأس الإجراء لائحة اختيارية من الوسطاء مع نوع كل منها وتضم وسيط واحد أو أكثر كما يمكن الاستغناء عنها كلها بحسب الوظيفة المطلوبة من الإجراء .

- السؤال الذي يطرح نفسه، لماذا لا نستخدم المتحولات العامة في الإجراء بدلاً من الوسطاء ..؟؟؟
- الجواب: قد نستخدم نفس الإجراء أكثر من مرة على أكثر من متحول، وعندما نستعمل المتحولات العامة بدلاً من الوسطاء سوف نضطر إلى كتابة إجراء جديد مماثل للسابق و لكن بأسماء متحولات مختلفة ... و عندئذٍ لن تتحقق الفائدة المرجوة من استخدام الإجراءات.

```
Program Proced1;
Procedure Print( s: String; i: Integer);
    begin
    Write(s);
    Writeln(i);
    end;
Begin
Print ('Hello',3);
Print (‘Thank you’,5);
End.
```

ناتج الطباعة:

Hello3

Thank you 5

الإجراءات Procedures

- **المتغيرات العامة و المحلية :** المتغيرات التي نستخدمها في بداية البرنامج بعد تعليمة Var عادة تعتبر متغيرات عامة، أي يمكننا استخدامها في أي وقت وفي أي مكان في البرنامج . بينما المتغيرات المحلية يمكن استخدامها فقط داخل الإجراء و المتغيرات المحلية لا تأخذ مكانها من الذاكرة إذا لم يبدأ الإجراء بالتنفيذ . يتم الإعلان عن المتغيرات المحلية تحت إعلان اسم الإجراء مباشرة.

```
Program Procedures;
Procedure Print ( s: String );
    Var
    i: Integer;
    begin
    for i := 1 to 3 do
    Writeln(s);
    end;
Begin
        Print ( ' Hello ' );
End.
```

مثال: طباعة عبارة Hello ثلاث مرات بواسطة الإجراء

ناتج الطباعة:

Hello

Hello

Hello

الإجراءات Procedures

ملاحظة : إذا قمنا بتعريف متحول عام وآخر محلي بنفس الاسم فالإجرائية سوف تتعامل مع المتحول المحلي ولا يمكنها أن تجري أي تغيير على المتحول العام

```
Program test ;
Var
X : integer ;      (Global متحول عام )
Procedure change ;
Var
X : integer ;    (Local متحول محلي)
Begin
X := 100 ;      // (* x local *)
End ;
Begin
X := 50 ;       // (* x global *)
Change ;
Writeln ( ‘ x = ', x );        // (* x global *)
End.
```

إن خرج هذا البرنامج : X = 50

لأن التغييرات ضمن الإجرائية طرأت على

المتحول المحلي x . و ليس على المتحول العام x

الدوال (التوابع) Functions

- الاختلاف بين التوابع Functions والإجراءات Procedures بسيط جداً من ناحية القواعد الإملائية، لكن الأهم هو أن نميز بينهما من ناحية المفهوم، لكي نعرف متى نستخدم التابع ومتى نستخدم الإجرائية .
- لقد صممت الإجرائية للقيام بعمل ما فقط، مثل قراءة مصفوفة أو طباعتها أو التبديل بين متحولين .
- أما التابع فقد صمم لحساب قيمة ما أيضاً، مثل حساب قيمة الجذر التربيعي لرقم، أو إيجاد مربع رقم ما ، الخ ... أي أننا نتوقع من التابع أن يعيد قيمة ما عند انتهاء تنفيذه حتى لو كان ذو مهمة معينة مثل قراءة مصفوفة إلا أننا قد نستخدمه في هذه الحالة ليعيد عدد العناصر التي تمت قراءتها بنجاح مثلاً .
- تعريف التوابع في لغة باسكال : يبدأ رأس التابع بالكلمة المحجوزة Function يليه اسم التابع ثم لائحة من الوسطاء الاختيارية محصورة بين قوسين، وننهي الترويسة بالنمط الذي سيرده التابع و **يجب أن يكون نمطاً بسيطاً .**
- **الشكل العام لترويسة التابع :**

Function name **(parameter1:** type **; parameter2 :** type **;..):**function_type **;**

ما تبقى من باقي أجزاء التابع فإنها مماثلة لمقابلاتها في الإجرائية، إلا أن جسم التابع يجب أن يحوي على تعليمة إسناد قيمة ما لاسم التابع، و لكن بشرط أن تكون هذه القيمة من نفس نوع المعطيات الذي يعيده التابع .

الدوال (التوابع) Functions

- التوابع تشبه الإجراءات باستثناء أنها تعيد قيمة .
- عند تعريف التوابع تستخدم الكلمة Function بدلاً من الكلمة Procedure.
- لتعريف نوع البيانات للقيمة المعادة يجب أن تستخدم النقطتين : وبعدها نوع البيانات وقبل النقطتين اسم الدالة .

القالب العام لتعريف التوابع ضمن برنامج باسكال ⬅

```
Program Functions;
Function Name ( i , j :Integer) : Integer ;
    begin
    …..
    end;
Begin
Function_name ;
End.
```

الدوال (التوابع) Functions

استدعاء التابع: يمكن أن نعامل التابع عند الاستدعاء كما نعامل الثوابت تقريباً ، فمن الممكن إسناد قيمة التابع لمتحول ما ومن الممكن استخدام قيمته في تعبير ما مباشرة كما من الممكن تمرير قيمته لنستخدمها في إجراء أو تابع آخر، ولكن من غير الممكن إسناد قيمة ما للتابع مثلاً ومن غير الممكن قراءة التابع .

- أمثلة على الاستدعاء الصحيح :

```
y := sqr ( x ) ;
Write ( sqr ( x ) ) ;
x := 1 + sqr ( y ) ;
```

- أمثلة على الاستدعاء الخاطئ :

```
Sqr ( x ) := y ;
Read ( sqr ( x ) ) ;
Sqr ( x ) := 1 + y ;
```

- بالطبع إذا كنا نفكر بطريقة منطقية فلن نقع في أخطاء استدعاء التابع؛ لأن جميع الكتابات الخاطئة السابقة لاستدعاء الإجراء تعطي التابع قيمة ما و**التابع يرد قيمة ولا يأخذ قيمة .**

الدوال (التوابع) Functions

- عند إسناد قيمة التابع لمتغير سيجعل المتغير يساوي القيمة المعادة إليه من الدالة.
- إذا استخدمنا التابع في أمر مثل Writeln فسوف يتم طباعة القيمة المعادة من التابع . لإسناد قيمة العائد أنشئ اسم التابع تساوي القيمة التي تريدها أن تكون هي العائد.

```
Program Functions;

Var
Answer: Integer;

Function Add ( i , j : Integer ) : Integer ;
        begin
        Add := i + j;
        end;

Begin
Answer  :=  Add ( 1 , 2 ) ;    // or // Writeln( Add(1,2));
Writeln ( Answer );
end.
```

مثال: جمع عددين صحيحين بواسطة التوابع ←

- يمكنك الخروج من الإجراء أو الدالة في أي وقت باستخدام الأمر Exit .

الدوال Functions

مثال على التابع : تابع يقوم بإرجاع القيمة الكبرى بين قيمتين تم تمريرهما له :

```
Function Max (a , b : integer) : integer;
Begin
If ( a > b ) then
Max := a ;
Else
Max := b ;
End;
```

ملاحظة :

١. نطلق اسم روتين على البرنامج الجزئي سواء كان تابع أو إجرائية .

٢. بما أن الروتين هو عبارة عن برنامج جزئي وينطبق عليه صفات البرنامج الرئيسي فيمكن استدعاء روتينات أخرى بداخله شريطة أن تكون معرفة قبله أو مصرح عنها قبله كما يمكن للروتين نفسه أن يستدعي نفسه بداخله (العودية)، وأكثر من ذلك ... يمكن أن نعرف روتينات أخرى ضمنه كما نعرفها في البرنامج الرئيسي عادة (الروتينات المتداخلة)، و لكنها تخضع لعدة شروط الصلاحية و النفاذ.

التعليمة Exit في الإجراءات والدوال

- يمكننا الخروج من الإجراء أو الدالة في أي وقت باستخدام الأمر Exit .

```
Program Proced5 ;

    Procedure GetName ;
    Var
    Name : String ;
    begin
    Writeln ( ' What is your name? ' ) ;
    Readln ( Name ) ;
    if  ( Name = '  ' ) then
    Exit ;
    Writeln ( ' Your name is ' , Name ) ;
    end ;

    Begin
    GetName ;
    End.
```

مثال: طباعة الاسم بواسطة إجراء

العمليات والدوال والإجراءات القياسية
Standard Operators, Function, Procedures

١- العمليات على الأعداد الصحيحة:

بالإضافة إلى العمليات الأساسية (الجمع +، الطرح - ، الضرب *، القسمة الصحيحة div، باقي القسمة الصحيحة mod) التي يتم تنفيذها على الأعداد العشرية والست عشرية (يسبقها إشارة $)، يمكن تنفيذ مجموعة من العمليات على مستوى الخانات الثنائية للعدد الصحيح، وهي:

- لدينا العددين الصحيحين x = 38 و y = 42 من النوع byte ، سنقوم بتنفيذ العمليات المذكورة أدناه على العددين بعد أن نقوم بتحويلهما إلى عددين ثنائيين.

x = 00100110 (38) , y = 00101010 (42)

- **العملية not** : تعطي المتمم الثنائي لجميع خانات العدد الصحيح

not (x) = not (00100110) = 11011001 (217)

- **العملية and** : تقارن منطقياً بين الخانات الثنائية المتقابلة في عددين مختلفين و تعطي عدداً ثنائياً فيه الواحدات المتقابلة.

x and y = 00100110 and 00101010 = 00100010 (34)

- **العملية or** : تقارن منطقياً بين الخانات الثنائية المتقابلة في عددين مختلفين و تعطي عدداً ثنائياً فيه كل الواحدات المتقابلة أو غير المتقابلة في العددين.

x or y = 00100110 or 00101010 = 00101110 (46)

١- العمليات على الأعداد الصحيحة

- **العملية xor** : تنجز عملية الفرق التناظري بين الخانات الثنائية المتقابلة في عددين مختلفين و تعطي عدداً ثنائياً تتوافق قيمه مع الجدول التالي: X xor Y

X	Y	X xor Y
0	0	0
0	1	1
1	0	1
1	1	0

x = 00100110 (38)

y = 00101010 (42)

x xor y = 00100110 xor 00101010 = 00001100 (12)

- **العملية shl** : تنجز عملية إزاحة الخانات الثنائية إلى اليسار وفق العدد المحدد على يمين التعليمة، واضعة الأصفار مكان الفراغات الحاصلة على اليمين .

x shl 2 = 00100110 shl 2 = 10011000 (152)

- **العملية shr** : تنجز عملية إزاحة الخانات الثنائية إلى اليمين وفق العدد المحدد على يمين التعليمة، واضعة الأصفار مكان الفراغات الحاصلة على اليسار.

x shr 1 = 00100110 shr 1 = 00010011 (19)

٢- الدوال العددية الحسابية

- **الدالة العددية { +$N } ** تجعل قيم الدوال الحقيقية real من النوع الموّسع extended بعد التعريف عنها في بداية البرنامج قبل Begin.
- **الدالة العددية { -$N } ** تجعل قيم الدوال حقيقية real بعد التعريف عنها في بداية البرنامج قبل Begin.
- **الدالة الحسابية ABS** : تعطي القيمة المطلقة للعدد الصحيح أو الحقيقي و تكون النتيجة من نفس النوع.

function **ABS** (x: real / integer) : real / integer ;

- **الدالة ARCTAN** : تعطي الزاوية التي ظلها x مقدرة بالراديان.

function **ARCTAN** (x: real / integer) : real ;

- **الدالة COS** : تعطي جيب تمام الزاوية x مقدرة بالراديان.

function **COS** (x: real / integer) : real ;

- **الدالة SIN** : تعطي جيب الزاوية x مقدرة بالراديان. function **SIN** (x: real / integer): real ;
- **الدالة SQR :** تعطي مربع القيمة x. function **SQR** (x: real / integer): real / integer;
- **الدالة SQRT :** تعطي الجذر التربيعي للعدد غير السالب x .

function **SQRT** (x: real / integer): real / integer ;

٢- الدوال العددية الحسابية

- **الدالة LN** : تعطي اللوغاريتم الطبيعي للعدد x . function **LN** (x: real / integer): real ;

- **الدالة INT :** تعطي الجزء الصحيح من العدد الحقيقي x . function **INT** (x: real): real ;

- **الدالة FRAC**: تعطي الجزء الكسري من العدد الحقيقي. function **FRAC** (x: real): real ;

- **الدالة EXP** : تعطي المقدار e^x . function **EXP** (x: real / integer): real ;

- **الدالة UPCASE**: تعطي المحرف اللاتيني الكبير المقابل للمحرف اللاتيني الصغير في المتحول x .

function **UPCASE** (x: char): char ;

- **الدالة الحسابية RANDOM** : تعطي أعداداً حقيقية في المجال (0,1) بشكل عشوائي .

function **RANDOM** (x: real):real ;

الإجراء RANDOMIZE : يعطي بداية عشوائية لموّلد الأعداد العشوائية الحقيقية في المجال (0,1)
بحيث لا تعاد القيم نفسها عند كل تنفيذ للبرنامج . Procedure **RANDOMIZE** ;

٣- دوال التحويل بين الأنواع القياسية

- **الدالة round** : تدوير العدد الحقيقي إلى أقرب عدد صحيح

function **round** (x: real): integer;

round (1.4) = 1 round (1.8) = 2

round (1.5) = 2 round (-1.5) = -1

- **الدالة trunk** : اقتطاع الجزء الكسري من العدد الحقيقي والإبقاء على الجزء الصحيح

function **trunk** (x: real): integer; trunk (1.6) = 1

-**الدالة chr:** المحرف المقابل لعدد موجود في جدول ASCII ضمن المجال (255.. 0).

function **char** (x: byte): char ;

-**الدالة ord** : العدد المقابل للمحرف الموجود في جدول ASCII، وهي معاكسة لـ chr.

function **ord** (x: char): byte ;

٤- دوال الأنواع المرتبة المنتهية

- **الدالة PRED** : تعطي هذه الدالة القيمة السابقة لقيمة المتحول المرتب X.

function **PRED** (x: ordinal type): ordinal type;

PRED (2) = 1 PRED (' j ') = ' i ' PRED (FALSE) = TRUE

- **الدالة Succ** : تعطي هذه الدالة القيمة اللاحقة لقيمة المتحول المرتب X.

function **Succ** (x: ordinal type): ordinal type;

Succ (2) = 3 Succ (' i ') = ' j ' Succ (FALSE) = TRUE

- **الدالة Odd** : تعطي قيمة بوليانية تقوم بتحديد قيمة المتحول X " فردي أو زوجي "

function **Odd** (x: integer): Boolean ;

Odd (x) = true " المتحول فردي "

Odd (x) = false " المتحول زوجي "

٤- دوال الأنواع المرتبة المنتهية

- **الإجراء Dec** : يعطي المتحول x قيمة جديدة مساوية لقيمة سابقة بمقدار n.

Procedure Dec (var x : ordinal type ; [n]) ;

Ord (x) = ord (x) – n x = 5 ; y = ' O '

Dec (x , 2); Dec (y , 4); x = 3 ; y = ' K '

- **الإجراء INC** : يعطي المتحول x قيمة جديدة مساوية لقيمة لاحقة بمقدار n.

Procedure INC (var x : ordinal type ; [n]) ;

Ord (x) = ord (x) + n x = 5 ; y = ' K '

Dec (x , 3); Dec (y , 2); x = 8 ; y = ' M '

٥- إجراءات التحكم بمسار البرنامج

- **الإجراء Exit** : يؤدي إلى الخروج من البرنامج الفرعي إلى البرنامج الرئيسي، أو لتوقيف البرنامج الرئيسي والخروج منه، وله الصيغة التالية:

Procedure **Exit** ;

- **الإجراء Halt** : يؤدي إلى قطع تنفيذ البرنامج والخروج منه إلى الوسط ، بغض النظر عن مكان استخدامه في البرنامج الرئيسي أو الفرعي، وله الصيغة التالية:

Procedure **Halt** ;

- **الإجراء Runerror** : يؤدي إلى قطع تنفيذ البرنامج وإظهار رسالة الأخطاء رقم n من جدول أخطاء التنفيذ المعروفة في لغة باسكال، وله الصيغة التالية:

Procedure **Runerror** (n : word) ;