الفصل الثاني

مقدمة في الخوارزميات

الهدف

✓جعل "الآلة "تعمل لصالحنا

Getting the machine to do work for us

المشكلة

- كيف نشرح لها ما يجب عليها فعله؟
- كيف نعلّمها؟
- كيف نتأكد أنها تؤدي المهمة بالشكل المطلوب؟
- بل وأفضل منا؟

الأهداف

- ✓حلّ المسائل بطريقة "الآلة“ Solve problems like a machine
- ✓معرفة كيفية شرح منطقك وتفكيرك
- ✓معرفة كيفية صياغة التفكير بشكل رسمي
- ✓تصميم (و كتابة) الخوارزميات

```
Algorithm Sort(arr)
  for i from 0 to n-1
    for j from 0 to n-i-1
      if arr[j] > arr[j+1]
        swap(arr[j], arr[j+1])
```

الخوارزميات

تعريف الخوارزمية

الخوارزمية هي ***تسلسل من التعليمات*** التي عند تنفيذها ، بشكل صحيح، تؤدي إلى ***النتيجة المطلوبة***

أمثلة على الخوارزميات:

- إرشاد سائح ضائع إلى وجهته
- كتابة وصفة طبخ
- صرف المشروبات تلقائياً من جهاز بيع
- تشغيل فيديو على يوتيوب

تعريف الخوارزمية- البيئة Environment

بيئة الخوارزميات هي مجموعة الأشياء أو العناصر اللازمة لإنجاز عملٍ تصفه خوارزمية.

نميّز بين:

- عناصر **الإدخال** :التي تُزوَّد بها الخوارزمية
- عناصر **الإخراج:** التي تنتجها الخوارزمية
- العناصر **الداخلية:** التي تُستعمل في **المعالجة** داخل الخوارزمية

➤ يمكن أيضًا تسمية بيئة الخوارزمية بـ **الإعدادات**(settings)

تعريف الخوارزمية – الاجراءات Actions

الإجراءات:

- هذه هي سلسلة التعليمات أو خطوات الخوارزمية.
- هي حدث ذو مدة محدودة يؤثر على البيئة.

- يُرجى ملاحظة أن:
  - تغيير ترتيب الإجراءات يمكن أن يغير النتيجة.
  - يمكن أن يظهر نفس الإجراء عدة مرات في نفس الخوارزمية.
- **الإجراء الأولي (Primitive Action):**

هو إجراء يتم تنفيذه دون أي معلومات أو خطوات إضافية.

خصائص الخوارزمية

الخوارزمية

التناهي

يجب أن تتوقف الخوارزمية بعد عدد منتهٍ من الخطوات

الشمولية

يجب أن تأخذ الخوارزمية بعين الاعتبار جميع الحالات الممكنة

التكرار

تحتوي الخوارزمية عادةً على تكرارات لبعض الإجراءات

الوضوح

يجب أن تكون كل إجراء في الخوارزمية محدداً بدون غموض

الاستقلالية

يجب أن تكون مستقلة عن لغات البرمجة والحواسيب

الكفاءة

يجب أن تعمل الخوارزمية بأقل وقت وأقل موارد ممكنة

تاريخ الخوارزميات

القرن الثامن عشر قبل الميلاد

البابليون :وضعوا خوارزميات شاملة لحسابات التجارة والضرائب

القرن الثالث قبل الميلاد

إقليدس :قدّم في كتابه "The Elements"الخوارزمية الشهيرة لإيجاد القاسم المشترك الأكبر

القرن التاسع الميلادي

الخوارزمي :أول من أضفى الطابع الرسمي على مفهوم الخوارزمية في كتابه "الجبر والمقابلة"

القرن الثاني عشر الميلادي

Adelard De Bath أدخل المصطلح اللاتيني (algorismus) نسبةً إلى اسم الخوارزمي

من المسألة إلى الحل

من المسألة إلى الحل

١	٢	٣	٤	٥
تعريف المسألة	تحليل المسألة	كتابة الخوارزمية	تنفيذ البرنامج	الاختبار والصيانة

المرحلة ١ -تعريف المسألة

تحديد المعلومات المتاحة وتعريف شكل النتائج المطلوبة
- ما المعطيات (المدخلات)؟
- ما النتائج المطلوبة (المخرجات) ؟

المرحلة ٢ -التحليل

إيجاد الطريقة للانتقال من البيانات إلى النتائج
- كيف نحل المسألة؟ • ما شكل الحل؟

المرحلة ٣ -كتابة الخوارزمية

كتابة حل واضح وغير غامض بالصياغة الخوارزمية
- أي هياكل تحكم تُستخدم؟ • أي متغيرات؟

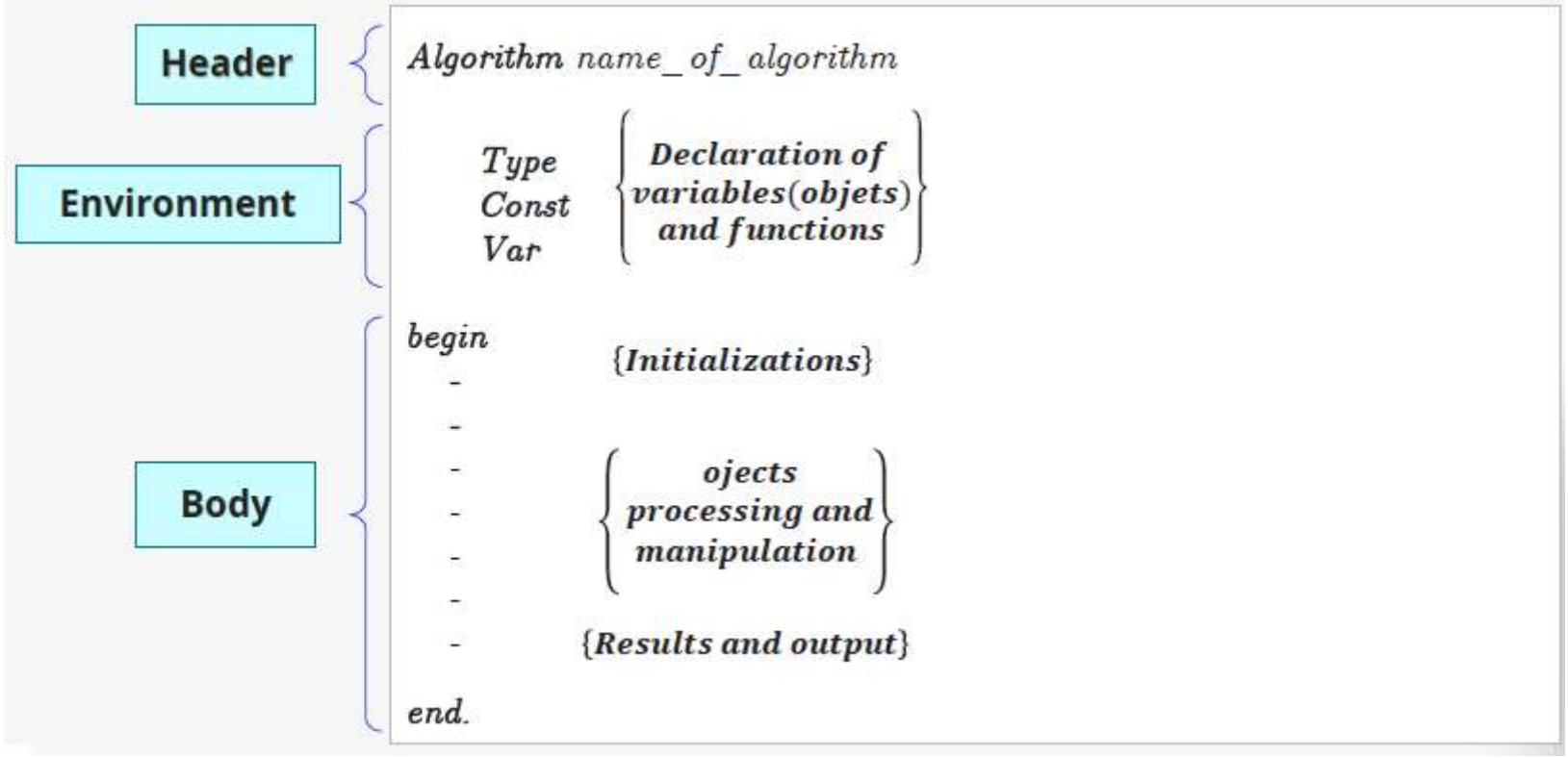
المرحلة ٤ -التنفيذ

اختيار لغة برمجة وترجمة الخوارزمية إليها
- أي لغة؟ • أي بيئة تطوير؟

المرحلة ٥ -الاختبار

تشغيل البرنامج والتحقق من صحته • أخطاء نحوية؟ • أخطاء منطقية؟ • نتائج صحيحة؟

بنية الخوارزمية


Header
Environment
Body
Algorithm name_of_algorithm
Type
Const
Var
Declaration of variables(objets) and functions
begin
{Initializations}
ojects processing and manipulation
{Results and output}
end.

التعليقات

```
//      a Single line comment
/*      Comment on
multiple lines */
```

- تعطي وصفاً بشرياً في كود الآلة
- تبسيط صيانة الكود وبالتالي تسريع كشف الأخطاء

الكائنات :الثوابت، المتغيرات والأنواع

الكائنات –Objects :تعريف

الكائن هو أي عنصر يُستخدم في الخوارزمية ليخزن بيانات أو يمثل قيمة يمكن التعامل معها أثناء تنفيذ الخوارزمية.

- يجب وصف أو إعلان جميع الكائنات المكونة للخوارزمية في البيئة (أو في جزء التصريحات).
- يتميز كل كائن بما يلي:
  - **الاسم** :(Name) معرف فريد، وهو سلسلة من الحروف والأرقام تسمح بتعيينه وتمييزه.
  - **النوع** :(Type) يحدد طبيعة المجموعة التي يأخذ منها قيمه.
  - **القيمة** :(Value) تشير إلى الحجم أو القيمة التي يتخذها الكائن في لحظة معينة.

الكائنات : المعرفات Identifiers

المعرف أو الاسم هو سلسلة من الحروف والأرقام يكون أول حرف فيها حرفًا أبجديًا.

يمكن أن يكون المعرف اسم خوارزمية، أو اسم متغير أو ثابت، أو اسم دالة.

قواعد المعرفات :(Rules for Identifiers)

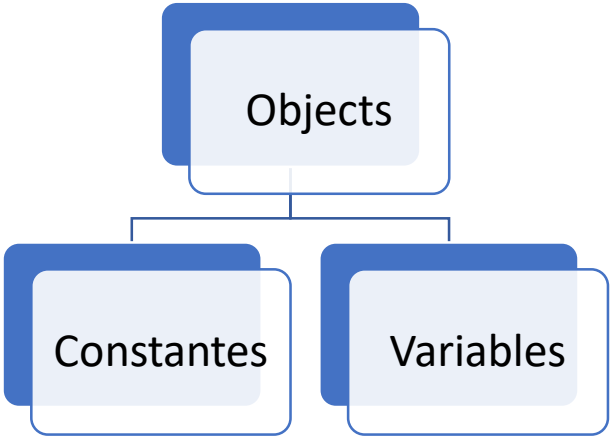
1. يجب أن يبدأ المعرف بحرف أبجدي.
2. يمكن أن يحتوي على حروف وأرقام و الرمز _ فقط.
3. لا يمكن استخدام المسافات داخل المعرف.
4. لا يمكن استخدام علامات الترقيم أو الرموز الخاصة.
5. يجب أن يكون المعرف فريدًا داخل البيئة أو الخوارزمية.

الكائنات : المعرفات Identifiers

أمثلة:

- product
- i
- j
- T1
- L_21
- surface
- student_name
- 5students
- release date
- x+y
- T 1
- algorithm

الكائنات :الفئات (Categories)



تستخدم الكائنات لتخزين البيانات التي تتعامل معها الخوارزمية.

هناك فئتان من الكائنات:

- **الثابت :(Constant)** هو كائن تكون قيمته غير قابلة للتغيير.
- **المتغير :(Variable)** هو كائن يمكن أن تتغير قيمته أثناء تنفيذ الخوارزمية.

الكائنات : الثوابت Constants

كائن لا تتغير قيمته أثناء تنفيذ الخوارزمية

كيفية التصريح بثابت:

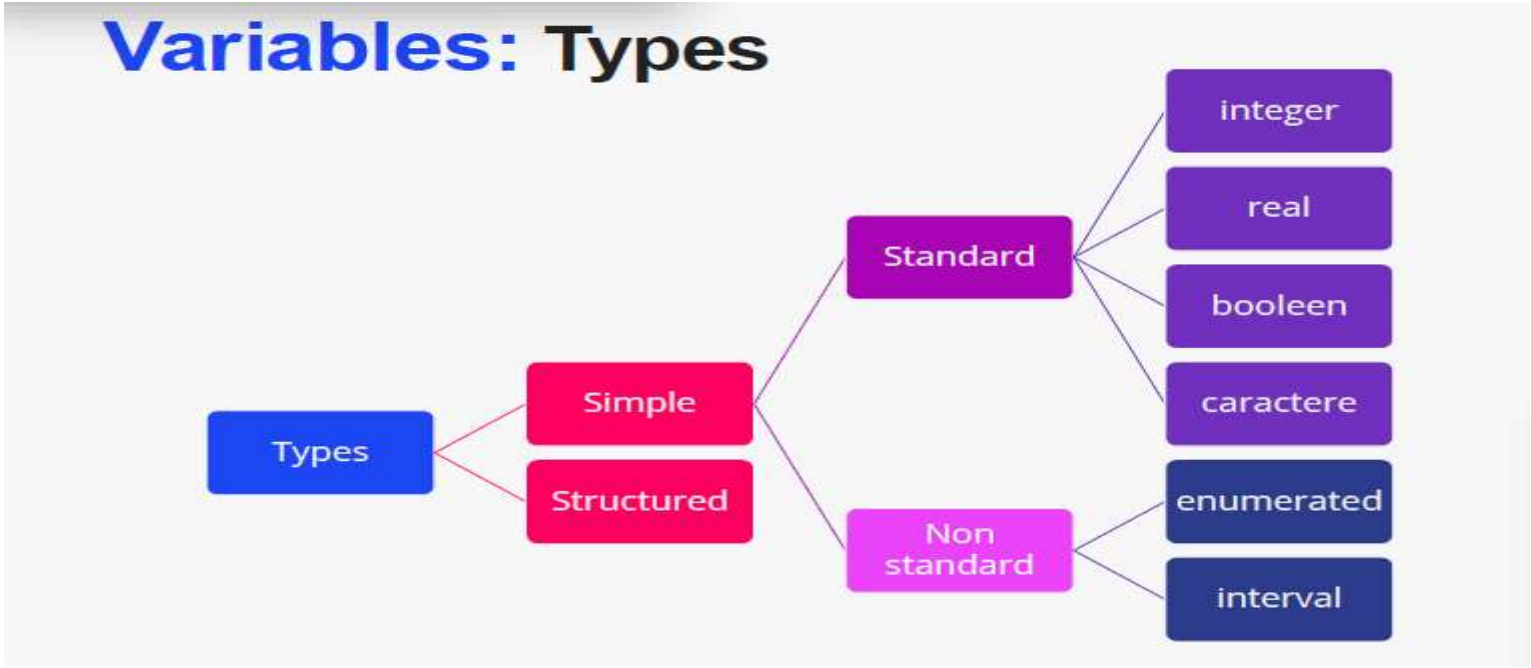
```
Const   nom_constante = Valeur
```

أمثلة:

```
Algorithm  exemple_const;
   Const     pi = 3.14
             cent = 100
             Lettre = 'M'
             Titre = 'Résultat :'
begin
   ...
   ...
end.
```

الكائنات : المتغيرات Variables

كائن تتغير قيمته أثناء تنفيذ الخوارزمية



الكائنات : المتغيرات

الأنواع العددية (Numerical Types)

- **عدد صحيح (Integer)** هو مجموعة الأعداد الصحيحة النسبية.
- **عدد حقيقي (Real)** هو مجموعة الأعداد التي تحتوي على جزء عشري.

الأنواع الأبجدية الرقمية (Alphanumeric Types)

- **المحرف (Character / CHAR)** يمثل حرفًا واحدًا.
  - يُكتب بين علامتي اقتباس مفردتين.
  - يشمل الحروف الأبجدية، الأرقام، علامات الترقيم، الرموز الخاصة، والمسافة... إلخ.
- **السلسلة النصية (String)** هي مجموعة من المحارف.

النوع المنطقي:(Boolean) هو مجموعة القيم المنطقية: **(True, False**صحيح أو خطأ).

الكائنات : المتغيرات

كيفية التصريح بمتغير:

```
Var      nom_variable : Type
```

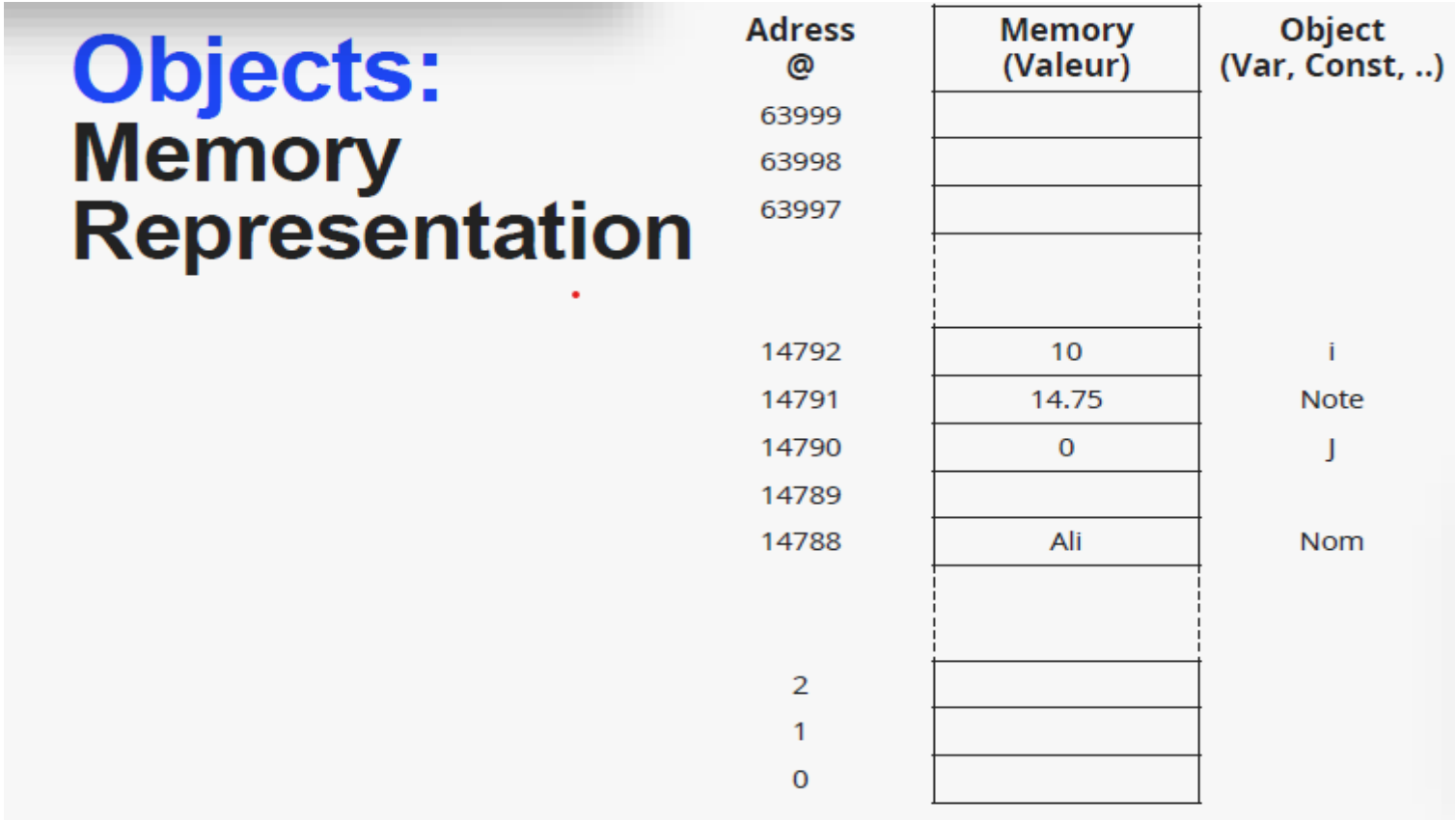
أمثلة:

```
Algorithme  exemple_vars;
   Var      C : character;
            N, i, j, jour, mois: integer
            x, y, racine : real
            trouver : boolean
begin
   ...
   ...
end.
```

الكائنات –Objects :التمثيل في الذاكرة الخوارزمية

- أي كائن يتعامل معه الخوارزمية (أو البرنامج) يتم تخزينه في الذاكرة المركزية.(RAM)
- تتكون الذاكرة المركزية من سلسلة من الصناديق المتجاورة تُسمى **خانات الذاكرة (Memory Boxes)** أو **الخلايا.(Cells)**
- يتميز كل صندوق ذاكرة بما يلي:
- **عنوان :(Address)** معرف فريد يشير إلى الخانة.
- **مساحة لتخزين قيم الكائن.(Space to store object values)**

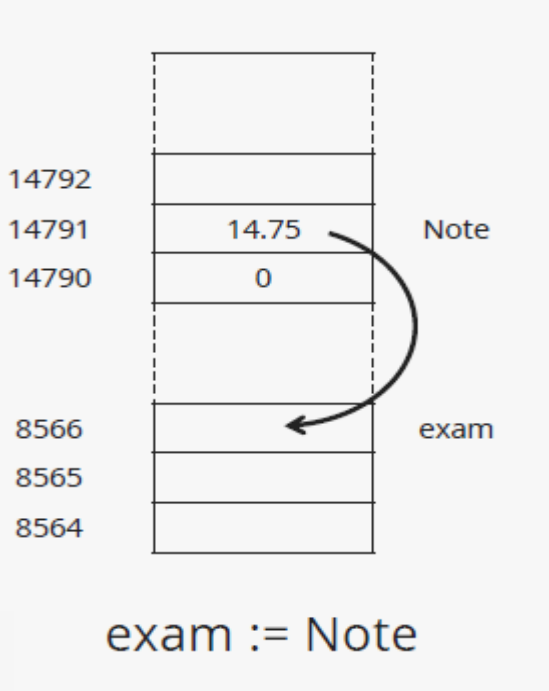
Objects: Memory Representation



الأوامر الأساسية

الاسناد -assignment

يتمثل دوره في إسناد (إعطاء) قيمة لمتغير.



يمكن أن تكون هذه القيمة:

- **ثابتًا.(Constant)**
- **قيمة متغير أو ثابت آخر.**
- **تعبيرًا (Expression)** مثل عملية حسابية أو منطقية

الشكل النظامي:

```
variable := expression
```

الاسناد -assignment

يتمثل دوره في إسناد (إعطاء) قيمة لمتغير.

الدور المزدوج لعملية الإسناد (Dual roles of assignment):

- **حساب وتقييم التعبير الموجود على يمين رمز الإسناد.**
- **إسناد وتخزين النتيجة في المتغير الموجود على يسار رمز الإسناد.**

```
Algorithm  exemple_ affect;
   Var        n, m, l: integer
begin
   ...
   n := 10
   m :=  n
   l := n*2 + m*3
   ...
end.
```

التعبيرات (Expressions)

التعبير هو مجموعة من القيم (المعاملات) المرتبطة بواسطة **مؤثرات (Operators)** وتكون مكافئة لقيمة واحدة.

المؤثر (Operator) هو رمز يربط بين قيمتين لإنتاج نتيجة.

expression = operand ***operator*** *operand*

التعبيرات (Expressions)

Operators	Operands	
+ - * / DIV MOD	Integer, Real	حسابية (Arithmetic)
+ (Conatenation)	Characters, Strings	أبجدية رقمية (Alphanumeric)
NOT AND OR	Boolean	منطقية (Logic)
< <= > >= = <>	Integer, Real, Character, String	علائقية (Relational)

Input/Output

القراءة (Reading)تسمح بقراءة قيم من لوحة المفاتيح

```
read (P1, P2, ..., Pn)
```

```
Algorithm  exemple_ affect;
   Var        n, m, l: integer
begin
   read (n, m)
    read (l)
   ...
end.
```

الكتابة (Writing)تسمح بعرض نتائج الخوارزمية على الشاشة

```
Write (E1, E2, ..., En)
```

يمكن أن تكون **E1**، **E2**، **...**، **En**متغيرات أو سلاسل نصية أو تعابير.

```
Algorithm  exemple_ affect;
   Var        n, m, l: integer
begin
   read   (n, m)
   write ('la somme =', n+m)
   ...
end.
```

مثال

حساب مساحة ومحيط مستطيل

المطلوب:

اكتب خوارزمية تقوم بـ:

1. طلب إدخال:
   1. طول المستطيل
   2. عرض المستطيل
2. حساب:
   1. **المساحة** = الطول × العرض
   2. **المحيط** = 2 × (الطول + العرض)
3. عرض النتيجتين

```
Algorithm RectangleCalculations;

Var
   length, width, area, perimeter : real;

Begin
   Write ("Enter the length:“);
   Read (length);
   Write ("Enter the width:“);
   Read (width);
   area := length * width;
   perimeter := 2 * (length + width);
   Write ("Area = ", area);
   Write ("Perimeter = ", perimeter);
End
```

Complex Instuctions

Block / Statements Block

الحاجة إلى بنية خوارزمية

بعض المشكلات معقّدة جدًا بحيث لا يمكن حلّها باستخدام تعليمات تسلسلية بسيطة فقط.

لذلك نحتاج إلى تنظيم التعليمات داخل **كتل** تسمح بتجميع مجموعة من العمليات المرتبطة لتنفيذها بطريقة منظّمة.

Blocs هي مجموعة منسجمة تتكوّن من تعليمة أولية واحدة أو أكثر

✓تنتهي بكلمة **نهاية الكتلة (End)**

التعليمة الشرطية/ Conditional Statement الخوارزمية

التعليمة الشرطية / الاختيار (Conditional / Selection Statement)

✓إذا **تمّ التحقق** من الشرط (أي إذا كان الشرط صحيحًا) يتم تنفيذ **Bloc 1**

✓ إذا لم **يتم التحقق** من الشرط (أي إذا كان الشرط غير صحيح) ننتقل إلى متابعة التنفيذ بالتسلسل بعد **Bloc1**

✓ الشرط هو عبارة عن **تعبير منطقي**(Logical Expression)

```
if (condition) then
    Block 1
  endif
...
```

مثال : حل معادلة من الدرجة الأولى

اكتب خوارزمية لحل المعادلة من الدرجة الأولى:

$ax + b = 0$ (نفترض أن $a \neq 0$)

التحليل:

باستخدام مفاهيم الرياضيات، فإن حل المعادلة من الدرجة الأولى هو:

$$x = -\frac{b}{a}$$

```
Algorithm equation_1er;
Var a, b, x: real;
begin
  read(a, b);
  if  (a <> 0)  then
     x := -b/a;
    write('la solution x = ',x);
endif
end.
```

التعليمة الاختيارية (Alternative Statement) الخوارزمية

```
if (condition) then
        Block statement 1
else
        Block statement 2
endif
...
```

- ✓ إذا تمّ التحقق من الشرط (أي إذا كان الشرط صحيحًا) يتم تنفيذ **Bloc 1**
- ✓ إذا لم يتم التحقق من الشرط (أي إذا كان الشرط غير صحيح) يتم تنفيذ **Bloc 2**

التعليمة الاختيارية (Alternative Statement) الخوارزمية

مثال : سالب ام موجب

اكتب خوارزمية تطلب من المستخدم ادخال عدد ما(لا يساوي الصفر)، تحديد اذا كان العدد سالب ام موجب، ثم اظهار كلمة "موجب" أو "سالب" حسب الحالة

التحليل:

- Read(): طلب ادخال عدد
- A<0 : اذا كان سالب
- A>0: اذا كان موجب

```
Algorithm Pos_Neg;
Var A : integer;
begin
  write("Give a number A :");
  read(A);
  if  (A > 0)  then
     write(A,"is positive");
  else
     write(A,""is negative");
  endif
end
```

التعليمة الشرطية المتداخلة(Nested CS)

التعليمة الشرطية المتداخلة (Nested Conditional Statement)

- ✓ هي تعليمة شرطية تحتوي داخلها تعليمة شرطية أخرى.
- ✓ تُستعمل عندما يعتمد القرار على أكثر من شرط واحد.

```
if (condition1) then
        Block statement 1
else
        if (condition2) then
            Block statement 2
        else {
            Block statement 3
        endif
        ...
endif
...
```

التعليمة الشرطية المتداخلة(Nested CS) الخوارزمية

مثال : سالب ام موجب او منعدم

اكتب خوارزمية تطلب من المستخدم ادخال عدد ما، تحديد اذا كان العدد سالب ام موجب، ثم اظهار كلمة "موجب" أو "سالب" أو معدوم حسب الحالة

التحليل:

- طلب ادخال عدد <-Read()
- اذا كان A<0 : سالب
- اذا كان A>0: موجب
- اذا كانA=0 : منعدم

```
Algorithm Pos_Neg_Null;
Var A : integer;
begin
 write("Donner un nombre A :");
 read(A);
 if  (A > 0)  then
   write(A, "est strictement positif");
 else
  if  (A < 0) then
    write(A, " est strictement négatif");
  else
    write(A, " est nul");
  endif
 endif
end.
```

التعليمة الشرطية السُلَّمية (Ladder Conditional Statement)

✓ هي سلسلة من الشروط المتتالية تُفحَص واحدًا بعد الآخر.

✓ عند تحقق أحد الشروط يتم تنفيذ الكتلة المرتبطة به، ويتم تجاهل باقي الشروط.

✓ تُستعمل عندما يكون لدينا عدة حالات ممكنة لنفس المشكلة (أكثر من شرطين).

✓ تمثّل عادةً بصيغة:

إذا ... وإلا إذا ... وإلا إذا ... وإلا.

```
if (condition1) then
        Block statement 1
else if (condition2)
        Block statement 2
else if (condition3)
        Block statement 3
else if (condition4)
        Block statement 4
else
        Block statement 5
endif
...
```

مثال : سالب ام موجب او منعدم

اكتب خوارزمية تطلب من المستخدم ادخال عدد ما، تحديد اذا كان العدد سالب ام موجب، ثم اظهار كلمة "موجب" أو "سالب" حسب الحالة

التحليل:

- طلب ادخال عدد <-Read()
- اذا كان A<0 : سالب
- اذا كان A>0: موجب
- اذا كانA=0 : منعدم

```
Algorithm Pos_Neg;
Var A : integer;
begin
  write('Donner un nombre A :');
  read(A);
  if  (A > 0)  then
     write(A,' est strictement positif');
  else if  (A < 0)
      write(A,' est strictement négatif');
   else
      write(A,' est nul');
  endif
end.
```

القيمة المطلقة

اكتب خوارزمية تقوم بحساب القيمة المطلقة لعدد ما

التحليل:

- طلب ادخال عدد <-Read()
- اذا كان A<0 : |A| = -A
- اذا كان A>= 0 |A| = A

```
Algorithm Val_Abs;
Var A : integer;
begin
  write('Donner un nombre A :');
  read(A);
  if  (A >= 0)  then
     write('la valeur absolut est',A);
  else
      write('La valeur absolut est', -A);
 endif
end.
```

التعليمات التكرارية
Repetetive Structers: Loops

التعليمات التكرارية Loops

توجد نوعين من البُنى التكرارية (الحلقات):

1. حلقة **FOR**

2. حلقة **WHILE**

✓ هذه البُنى لها نفس القدرة، لكن وفقًا للعرف يتم اختيار نوع الحلقة حسب طبيعة المشكلة المراد حلها.

✓ التكرار (Iteration) هو دورة كاملة للحلقة وتشمل:

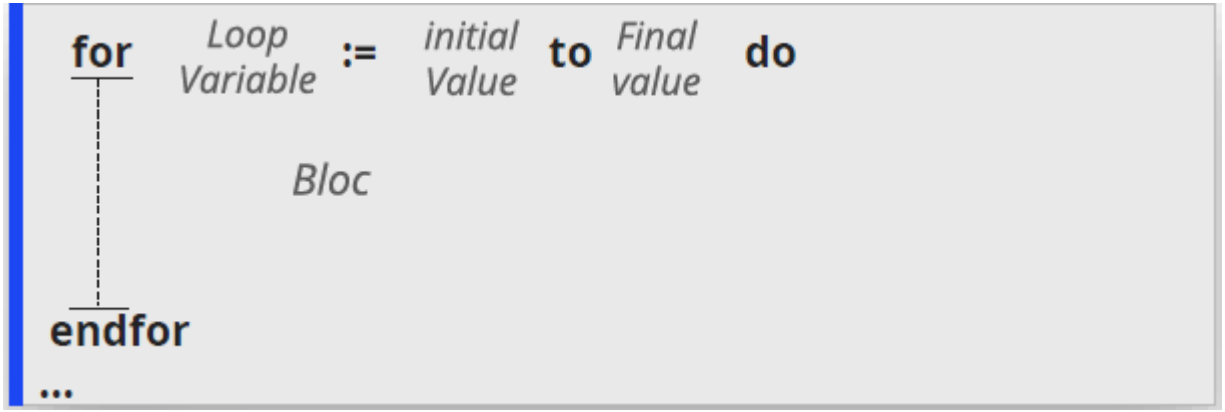
- تنفيذ تعليمات الحلقة،
- اختبار الشرط،
- وتعديل القيم اللازمة لاستمرار الحلقة أو إيقافها.

✓ **الحلقة اللانهائية (Infinite loop)** هي حلقة لا يتغير فيها الشرط أبدًا، مما يؤدي إلى عدد لا نهائي من التكرارات.

التعليمة التكرارية For

حلقة FOR هي بنية تكرارية تسمح بتكرار **نفس المعالجة** لمجموعة من القيم بين حد أدنى وحد أقصى.

✓ تُستخدم حلقة FOR عندما يكون **عدد مرات التكرار** معروفًا مسبقًا.



تنفيذ حلقة FOR

المرحلة 1: تهيئة متغير الحلقة بالقيمة الابتدائية و القيمة النهائية

المرحلة 2: تقييم الشرط والتحقق مما إذا كانت قيمة متغير الحلقة ضمن المجال المحدد أم لا.

- إذا كانت داخل المجال ننتقل إلى **المرحلة الثالثة**.
- إذا لم تكن داخل المجال ننتقل إلى **المرحلة الخامسة**.

```
For i := 1 to 5 do
        instructions
Endfor
```

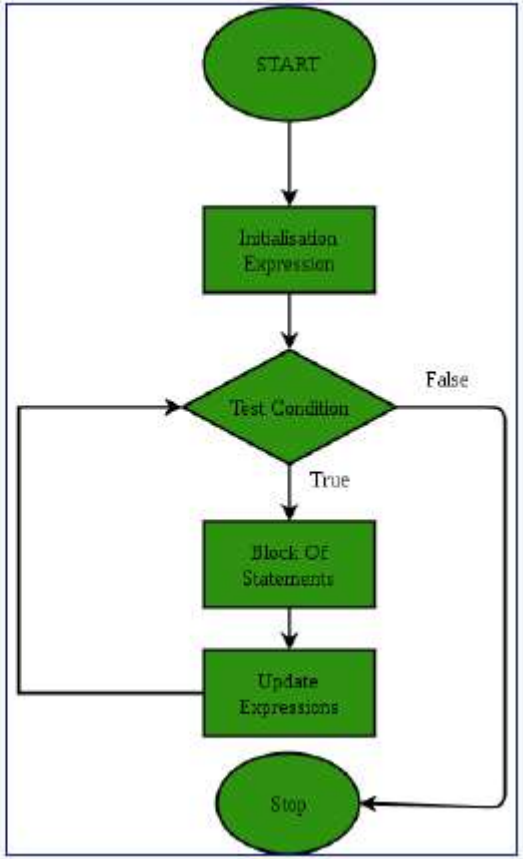
المرحلة 3: تنفيذ مجموعة التعليمات داخل الحلقة.

المرحلة 4: تحديث (زيادة) قيمة متغير الحلقة **تلقائيًا** حسب قيمة الخطوة(Step)

(القيمة الافتراضية: Step = 1) ثم العودة إلى المرحلة الثانية.

المرحلة 5: الخروج من الحلقة ومواصلة تنفيذ البرنامج ابتداءً من أول تعليمة بعد **EndFor**

for loop Flow Diagram



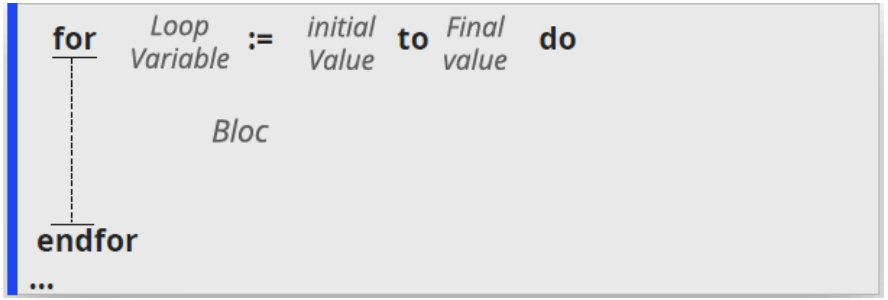
مثال 1

اكتب خوارزمية اكتب خوارزمية تقوم بعرض جدول الضرب لعدد صحيح بين 1 و 10.

```
Algorithm Multiples;
Var N, i, r : integer;
Begin
  write('Enter the number N:');
  read(N);

For i:= 1 to 10 do
      r:= N*i;
      write(N, "x", i, "=", r);
Endfor

end.
```



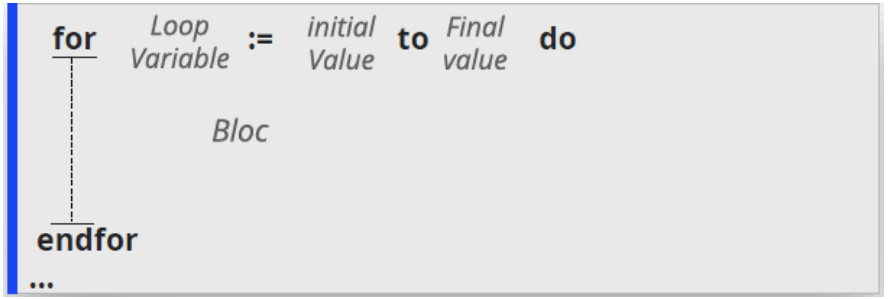
تنفيذ الخوارزمية

مثال 2

اكتب خوارزمية اكتب خوارزمية تقوم بحساب a^b علما ان a و bعددان صحيحان موجبان

```
Algorithm Power;
Var a,b, i,p : integer;
Begin
 write('Enter the numbers a and b:');
 read(a,b);
 p:=1;
 For i:= 1 to b do
     p:= p*a;
 Endfor
 write(a,"power",b,"is", p);

end.
```



تنفيذ الخوارزمية

التعليمة التكرارية While

✓تقوم حلقة WHILE بتنفيذ التعليمات داخل الحلقة طالما أن شرط التنفيذ محقق.

✓ يتوقف تنفيذ الحلقة بمجرد أن يصبح الشرط غير محقق.

```
loop variable initialization
while   execution test/condition   do
                Statements Block
    update loop variable
endwhile
```

تنفيذ حلقة WHILE

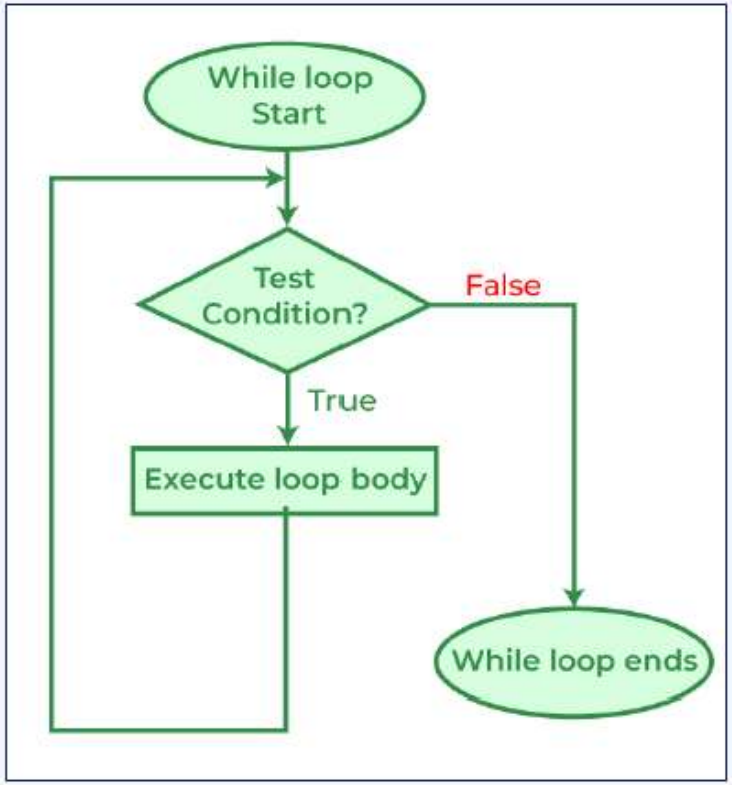
الخطوة 1: تقييم واختبار شرط التنفيذ.

- إذا كان الشرط محققًا ننتقل إلى **الخطوة الثانية**.
- إذا لم يكن محققًا ننتقل إلى **الخطوة الثالثة**.

الخطوة 2: تنفيذ مجموعة التعليمات داخل الحلقة.

الخطوة 3: الخروج من الحلقة ومواصلة تنفيذ البرنامج ابتداءً من أول تعليمة بعد **EndWhile.**

while loop Flow Diagram

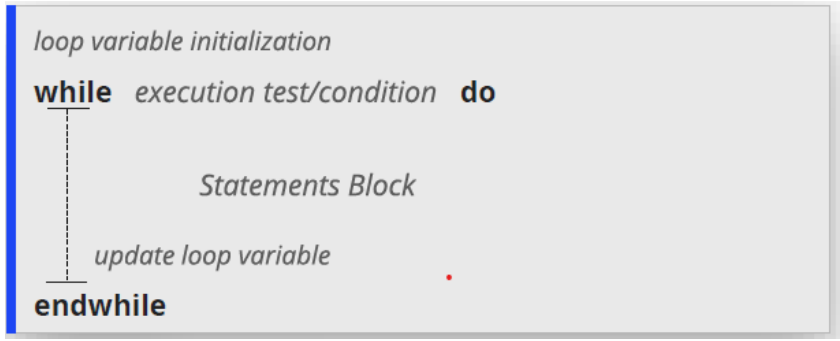


مثال 1

اكتب خوارزمية اكتب خوارزمية تقوم بعرض جدول الضرب لعدد صحيح بين 1 و 10.

```
Algorithm Multiples;
Var N, i, r : integer;
Begin
  write('Enter the number N:');
  read(N);
 i:=1;
 While i<= 10 do
      r:= N*i;
      write(N, "x", i, "=", r);
      i:=i+1;
Endwhile

end.
```

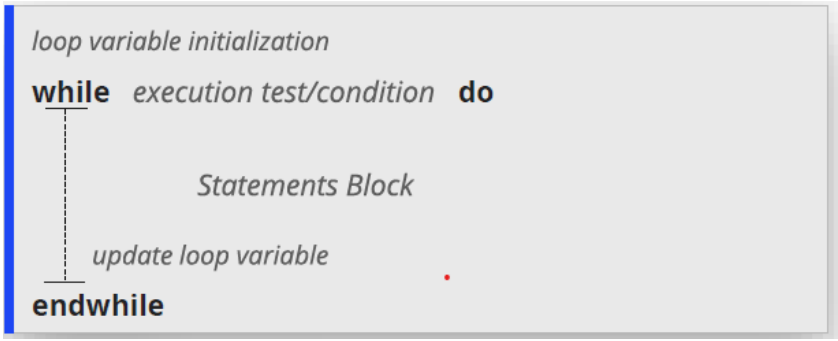


تنفيذ الخوارزمية

مثال 2

اكتب خوارزمية اكتب خوارزمية تقوم بجمع الأعداد التي يدخلها المستخدم، و تتوقف عندما يدخل المستخدم العدد 0

```
Algorithm SumUntilZero;
Var N, S : integer;
Begin
  S := 0;
  read(N);
  While N <> 0 do
    S := S + N;
    read(N);
  EndWhile
  write('Sum =', S);
End.
```



تنفيذ الخوارزمية