الوحدة 2: مقدمة في الخوارزميات

1. تعريف الخوارزم:

الخوارزم أو الخوارزمية (***Algorithme***) (***Algorithm***) هي مجموعة من الخطوات المنطقية والمتسلسلة والمحددة بدقة، والمرتّبة، والمنتهية تُستعمل لحل مشكلة معيّنة أو لإنجاز مهمة محددة، بحيث تؤدي هذه الخطوات دائمًا إلى نتيجة صحيحة عند تنفيذها بالشكل الصحيح.

أ. الخصائص الأساسية للخوارزم:

لكي نطلق على أي سلسلة خطوات اسم "خوارزمية"، يجب أن تتوفر فيها الشروط التالية:

- الوضوح :(*Precision*) كل خطوة يجب أن تكون محددة تماماً وغير قابلة للتأويل.
- المدخلات :(*Inputs*) البيانات التي تبدأ بها (مثل الأرقام التي تريد جمعها).
- المعالجة (*Processing*): العمليات الحسابية أو المنطقية التي تتم على البيانات.
- المخرجات :(*Outputs*) النتيجة النهائية (مثل حاصل الجمع).
- النهاية (*Finiteness*): يجب أن تتوقف الخوارزمية بعدد محدد من الخطوات ولا تستمر للأبد.

ب. أهميته:

تعتبر الخوارزميات الأساس لذي تبنى عليها جميع برامج الحاسوب والأنظمة الرقمية. فهي تمثل الحل المنطقي للمسألة التي يراد برمجتها. بعد كتابة الخوارزم والتأكد من صحته، يتم ترجمة هذه الخطوات إلى أي لغة من لغات البرمجة وكتابتها على الجهاز.

2. تصميم الخوارزم:

بعد فهم المسألة جيدا وتحديد مدخلاتها ومخرجاتها والعلاقة الموجودة بينها، نقوم تصميم الخوارزمية.

هناك طريقتين لتصميم الخوارزمية:

أ. باستعمال الكود (شبه كود) (Pseudocode):

وهي سلسلة تعليمات مكتوبة بلغة مفهومة من طرف الانسان، وتعتمد على نمط محدد في الكتابة وتسمى شبه كود (**Pseudocode**). وهي تعليمات متسلسلة (**Sequence**) يتم تنفيذها عادة الواحدة تلو الأخرى.

يتكون الخوارزم من ثلاث أقسام رئيسية:

- **الرأس :** يتكون من كلمة "**خوارزم**" (**Algorithm**) (**Algorithme**) متبوعة باسم الخوارزم.
- **منطقة التصريحات (Declarations) (Déclarations):** وهي الجزء الذي يتم فيه التعريف **بالمتغيرات** (**variables**) و**أنماط** المتغيرات (**types**) **والثوابت** (**constants**) المستعملة داخل الخوارزم. منطقة التصريحات مهمة جدا للجهاز من أجل تخصيص مساحة من الذاكرة لكل متغيرة أو لكل عدد ثابت، تختلف المساحة المخصصة على حسب نمط كل متغيرة.
- **جسم الخوارزم (Body) (Corps):** وهي المنطقة التي تحتوي على التعليمات اللازمة لحل المسألة.

1. التعليمات الأساسية:

- تعليمات البداية (**Begin**) والنهاية (**End**) : من أجل تحديد بداية ونهاية الخوارزمية أو تركيبة من التركيبات المستعملة.
- **تعليمة القراءة "اقرأ()" ، "()Read"، "()Lire": ** وتمثل عملية إدخال معلومة إلى البرنامج.

ملاحظة: إذا كتبنا التعليمة اقرأ(X) فإن X يجب أن تكون عبارة عن متغيرة قد تم الإعلان عنها مسبقا في قسم التصريحات. عندما يصادف الجهاز هذه التعليمة فإنه يتوقف عندها منتظرا من المستخدم إدخال قيمة من لوحة المفاتيح.

- **الكتابة "اكتب()" ، "()Write"، "()Print"، "()Ecrire":** : وهي تمثل عملية إخراج معلومة (إظهارها للمستعمل على الشاشة أو الطابعة) أو تسجيلها في ملف.

ملاحظة: إذا كتبنا التعليمة اكتب(X) فإن الجهاز يقوم بكتابة قيمة المتغيرة X على الشاشة، أما إذا كتبنا اكتب("X") فإن الجهاز يقوم بكتابة الحرف "X".

- **تعلمية الإسناد (Affectation):** ويستعمل فيها السهم المتجه إلى اليمين (→) بالنسبة للغة العربية والسهم المتجه إلى اليسار (←) بالنسبة للغات الأجنبية أو يستعمل رمز لمساواة (=). وتستعمل هذه التعليمة لوضع قيمة أو نتيجة عملية حسابية في متغيرة (للاحتفاظ بها). مثل : 5+ 3 ←A أي نحتفظ بنتيجة العملية 3+5 في المتغيرة A.
- **العمليات الحسابية البسيطة:** الجمع (+) والطرح (-) والضرب (*) والقسمة (/)، والقسمة الصحيحة (div) وباقي القسمة (mod) ..
- **العمليات المنطقية:** النفي (Not) والوصل (And) والفصل (Or).
- **تعليمات المقارنة:** $<$، $>$، $\leq$، $\geq$، $\neq$، $=$.

2. التركيبات:

- **تركيبة الاختيار (الشرط) (Selection):** وتستعمل للاختيار بين تنفيذ تعليمات من عدمه، أو للاختيار بين تنفيذ مجموعتين من التعليمات، حيث إذا تحقق الشرط يتم تنفيذ المجموعة الأولى وإن لم يتحقق يتم تنفيذ المجموعة الثانية:

إذا (شرط) إذن
(تعليمات)
نهاية اذا
	If (condition) **then**
	(*Instructions*)
	End if
		Si (condition) **Alors**
		(*Instructions*)
		Fin Si

إذا (شرط) إذن
(تعليمات1)
وإلا
(تعليمات2)
نهاية اذا
	If (condition) **then**
	(Instructions1)
	Else
	(Instructions2)
	End if
		Si (condition) **Alors**
		(Instructions1)
		Sinon
		(*Instructions2*)
		Fin si

إذا تحقق الشرط المذكور فإن الجهاز يتابع التنفيذ مباشرة في التعليمات المذكورة داخل التركيبة، أما إذا لم يتحقق فإن الجهاز يمر مباشرة إلى التعليمات التي تأتي بعد نهاية التركيبة بالنسبة للحالة الأولى وفي التعليمات الموجودة بعد "إلا" بالنسبة للحالة الثانية.

- **تركيبة الاختيارات المتعددة (حسب/حالة) (Swich/case) (Selon/cas): وتستعمل عند وجود العديد من الاختيارات الممكنة:**

حسب (المتغيرة) حالة 1: (تعليمات) حالة 2: (تعليمات) ... وإلا (تعليمات) **نهاية حسب**	**Switch** (variable) Case 1: (*instructions*) Case 2: (*instructions*) ... **Other** : (*instructions*) **End switch**	**Selon** (variable) Cas 1: (*instructions*) Cas 2: (*instructions*) ... Sinon: (*instructions*) **Fin selon**

ملاحظة: تركيبة "حسب" يمكن تحقيقها عن طريق استعمال عدة تركيبات من "إذا".

- **تركيبات التكرار(Iteration):** وتستعمل لتكرار تنفيذ سلسلة من التعليمات عدد معين من المرات أو إلى غاية تحقق شرط ما وتسمى أيضا بالحلقات (***loops***) (***boucles***)، وتتمثل في :

✓ **حلقة من أجل :** تستعمل لتكرار سلسة من العمليات عدد معين من المرات باستعمال متغيرة تتغير داخل مجال، القيمة التي تتغير بها هذه المتغيرة في كل مرة تسمى "الخطوة" (***Step***) (***Pas***)، في حالة عدم ذكر الخطوة فإن قيمتها تعتبر 1:

من أجل (i = 1 إلى 10 (خطوة = 2)) **افعل**
(تعليمات)
نهاية من أجل
	For i=1 :10 (step=2) **do**
	(Instructions)
	End For
		Pour i=1 :10 (pas=2) **faire**
		(Instructions)
		Fin Pour

✓ **حلقة كرر:** تستعمل لتكرار سلسة من العمليات حتى يتحقق الشرط (يتم تنفيذ التعليمات مرة واحدة على الأقل):

كرر (تعليمات) **حتى** (شرط)	**Do (Repeat)** (*Instructions*) **While (Until)** (Condition)	**Répéter** (*Instructions*) **Jusqu'à** (Condition)

✓ **حلقة طالما (مادام):** تستعمل لتكرار سلسة من العمليات حتى يتحقق الشرط (إذا كان الشرط غير محقق عند بداية الحلقة فإن التعليمات لا تنفذ نهائيا):

مادام (شرط) **افعل**
(تعليمات)
نهاية مادام
	While (Condition) **do**
	(*Instructions*)
	End While
		Tant que (Condition) ***faire***
		(*Instructions*)
		Fin tantque

ملاحظة:

- عند استعمال الحلقتين "كرر" أو "مادام" فيجب التأكد من أن الشرط سيتحقق في مرة من المرات، وإلا ستكون لدينا حلقة لا نهائية (***Infinite Loop***) (***Boucle Infinie***).
- عند استعمال الحلقة "كرر" فإنه يجب التأكد من أن الشرط محقق دائما في المرة الأولى.
- يمكن استعمال هذه الحلقات بالإضافة إلى تعليمة الشرط داخل بعضها البعض، لكن مع مراعاة أن آخر حلقة مفتوحة هي التي يجب أن تغلق قبل الآخرين.

الشكل العام للخوارزمية :

- نبدأ أولا باستعمال كلمة "**خوارزم**" (**Algorithm**) (**Algorithme**) ثم اسم الخوارزم (يمكن استعمال أي كلمة لكن يستحسن استعمال كلمات معبرة عن المعنى).
- ثم نتبعها **بالتصريحات** (**Declaration**) (**Déclarations**) والتي نذكر فيها المتغيرات والقيم الثابتة التي نستعملها لحل المسألة. كل متغيرة نتبعها بذكر نمطها (نوعها) (**Type**). إذا كان النمط مركبا فيجب ذكره قبل ذكر المتغيرة.

  الأنماط البسيطة هي : الأعداد الصحيحة (**integer**)، الأعداد الحقيقية (**real**)، القيم المنطقية (**boolean**)، الحروف (**char**)، سلسلة الحروف (**string**)، المؤشرات (**pointer**) وتسمى العناوين (**address**).

  الأنماط المركبة : مثل الجداول (**table**)، السجلات (**register**)، الأشجار (**tree**).
- ثم جسم الخوارزم والذي يبدأ بكلمة "**بداية**" (**Begin**) وينتهي بطلمة "**نهاية**" (**End**) وبينهما يتم كتابة التعليمات اللازمة لحل المسألة.

مثال:

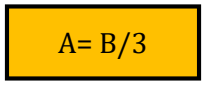
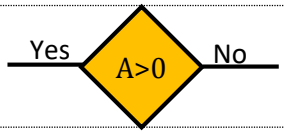
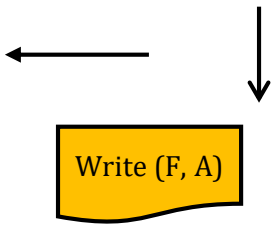
خوارزم جمع
A، B، S: أعداد صحيحة
بداية
اقرأ (A ، B)
S = A + B
اكتب (S)
نهاية

Algorithm Sum
A , B, S : integer
Begin
Read(A, B)
S=A + B
Print(S)
End

Algorithme Somme
A , B, S : entier
Début
Lire(A, B)
S=A + B
Ecrire(S)
Fin

ب. المخطط الانسيابي (Flowchart) (Organigramme):

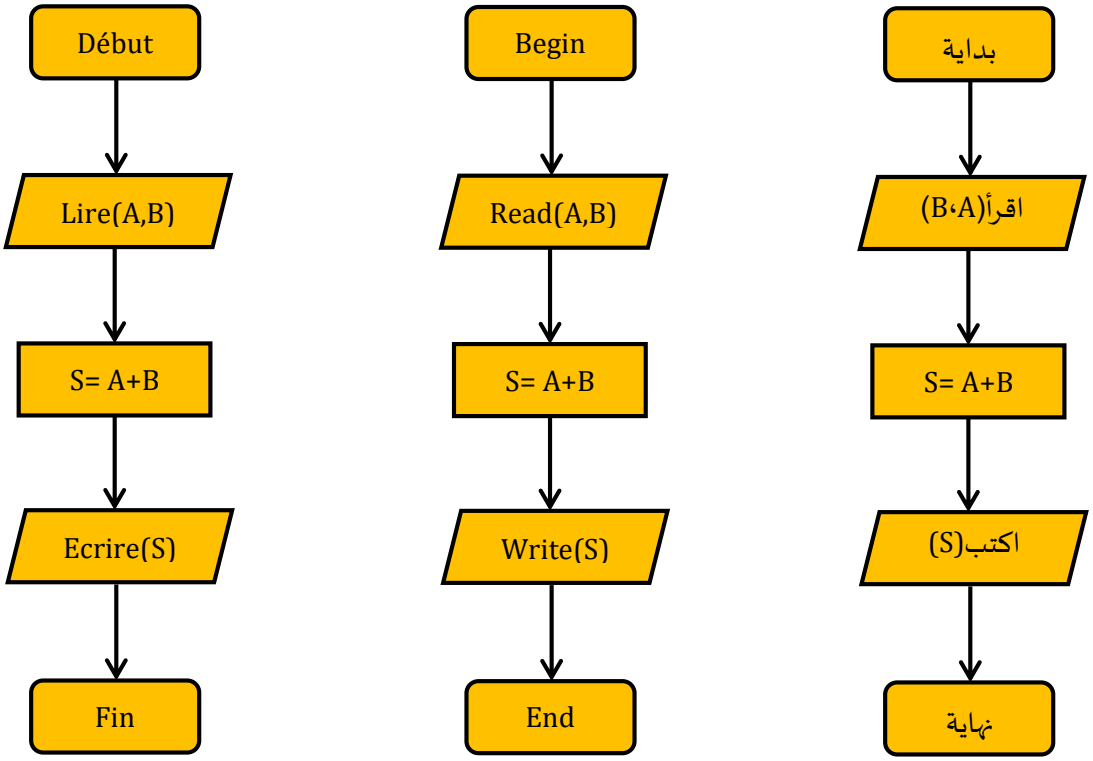
هو رسم بياني يُستخدم لتمثيل بنية وتدفق خوارزمية أو برنامج بشكل مرئي (تمثيل بصري). ويستخدم أشكالًا هندسية وروابط لتوضيح تدفق العمليات، والقرارات الشرطية، والحلقات، والإجراءات المطلوب تنفيذها. ويستعمل الأشكال التالية:

الشكل البيضاوي: للدلالة على البداية والنهاية.	
المستطيل: لتمثيل العمليات الحسابية.	
متوازي المستطيلات: للتعبير عن عمليات القراءة والكتابة.	
المعين: لتمثيل القرارات.	
الأسهم: للدلالة على الاتجاه.	

مستطيل منحي الخط السفلي: للدلالة على استعمال الملفات (للتخزين).

ملاحظة: المخطط الانسيابي مهم لفهم المبدأ العام لعمل الخوارزمية.

المثال السابق:



3. تحويل الخوارزم إلى برنامج:

بعد الانتهاء من كتابة الخوارزم والتأكد من صحته يتم ترجمته إلى لغة من لغات البرمجة مثل (Python، C، Java، ...)، حيث يتم ترجمة كل خطوة في الخوارزمية إلى تعليمات برمجية مكافئة مع احترام قواعد تلك اللغة (Syntax)، بعدها يجب اختبار البرنامج من أجل تصحيح الأخطاء.

3. أمثلة وتمارين:

مثال 1: اكتب خوارزم من أجل المقارنة بين عددين صحيحين A وB، بحيث يكتب على الشاشة إحدى العبارتين:

- العبارة « **A ≥ B** » إذا كان A ≥ B .
- العبارة « **B > A** » إذا كان A < B .

غير في هذا الخوارزم بحيث نفصل حالة المساواة عن الحالة الأولى.

مثال2: اكتب خوارزم يقوم بقراءة عددين صحيحين A وB ثم يقوم بتبديل القيمتين (Permutation) قيمة A يضعها في B وقيمة B يضعها في A، ومن ثم يظهر النتائج الجديدة.

تمرين 1: اكتب خوارزم يقوم بقراءة عددين صحيحين A وB، ثم يحسب حاصل القسمة الصحيحة لـ A على B وباقي قسمتها ويظهر النتائج المحصل عليها، إذا كان A<B يجب تبديل القيمتين A وB، (استعمل الرمز div للقسمة الصحيحة وmod لحساب باقي القسمة).

غير في هذا الخوارزم لمعالجة الحالة التي يكون فيها القاسم معدوما.

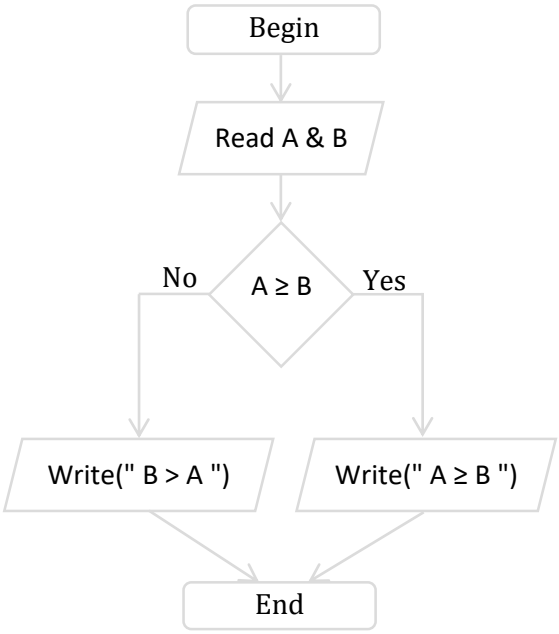
تمرين 2: ارسم مخطط انسيابي يمثل طريقة حل معادلة من الدرجة 2، ثم اكتب الخوارزمية بالتفصيل.

تمرين 3: اكتب خوارزم من أجل حساب معدل مادة الاعلام الآلي لطلبة فوجك، ثم مثله عن طريق مخطط انسيابي (الحل في الحصص التطبيقية).

حل المثال1:

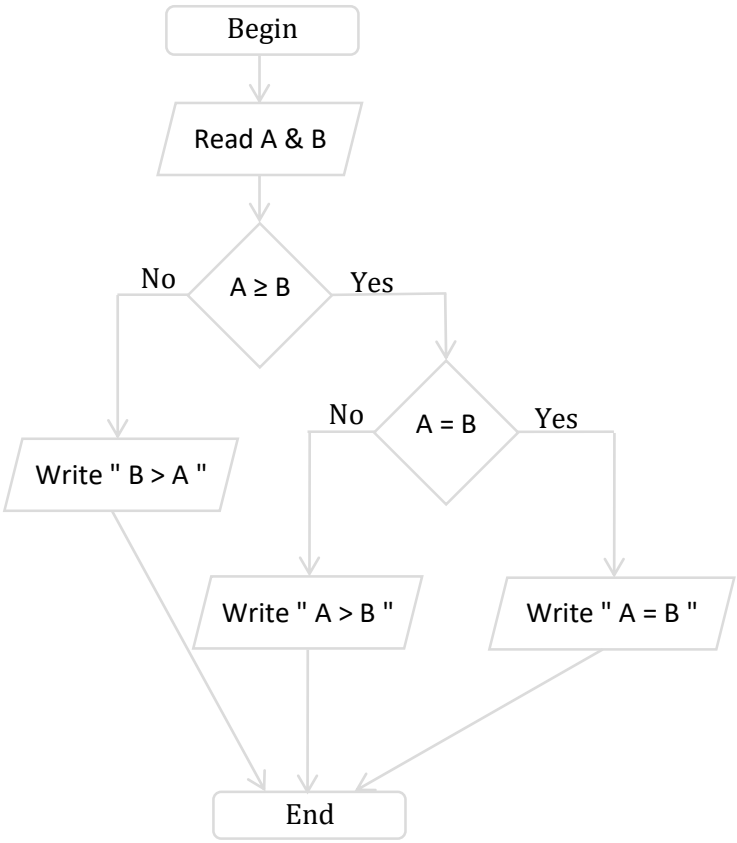
- المدخلات : العددين A و B.
- المخرجات: عبارة نصية.

```
Algorithm CompareNbr
Variables:
   A, B: integer
Begin
   Read(A); read(B)
   If A ≥ B then
      Write("A ≥ B")
   Else
      Write("B > A")
   Endif
end
```



عند إضافة حالة المساواة:

```
Algorithm CompareNbr
Variables:
   A, B: integer
Begin
   Read(A); read(B)
   If A ≥ B then
      If A=B then
         Write("A = B")
      else
         Write("A > B")
      endif
   Else
      Write("B > A")
   Endif
end
```



مثال 2: تبديل قيمتي متغيرتين فيما بينهما

- المدخلات : العددين A و B.
- المخرجات: العددين A و B.

خوارزم تبديل

المتغيرات:

A، B، C: أعداد صحيحة

بداية

اقرأ(A، B)

A → C

B → A

C → B

اكتب(A، B)

نهاية

```
Algorithm Permutation
Variables:
   A, B, C: integer
Début
   Lire(A,B)
   C ← A
   A ← B
   B ← C
   Ecrire(A,B)
Fin
```

```
Algorithm Permutation
Variables:
   A, B, C: integer
Begin
   Read(A,B)
   C ← A
   A ← B
   B ← C
   Write(A,B)
End
```

تمرين 1: حاصل القسمة وباقي القسمة الصحيحة لعددين:

- المدخلات : العددين A و B.
- المخرجات: العددين Q (Quotient حاصل القسمة) و R (Remainder باقي القسمة).

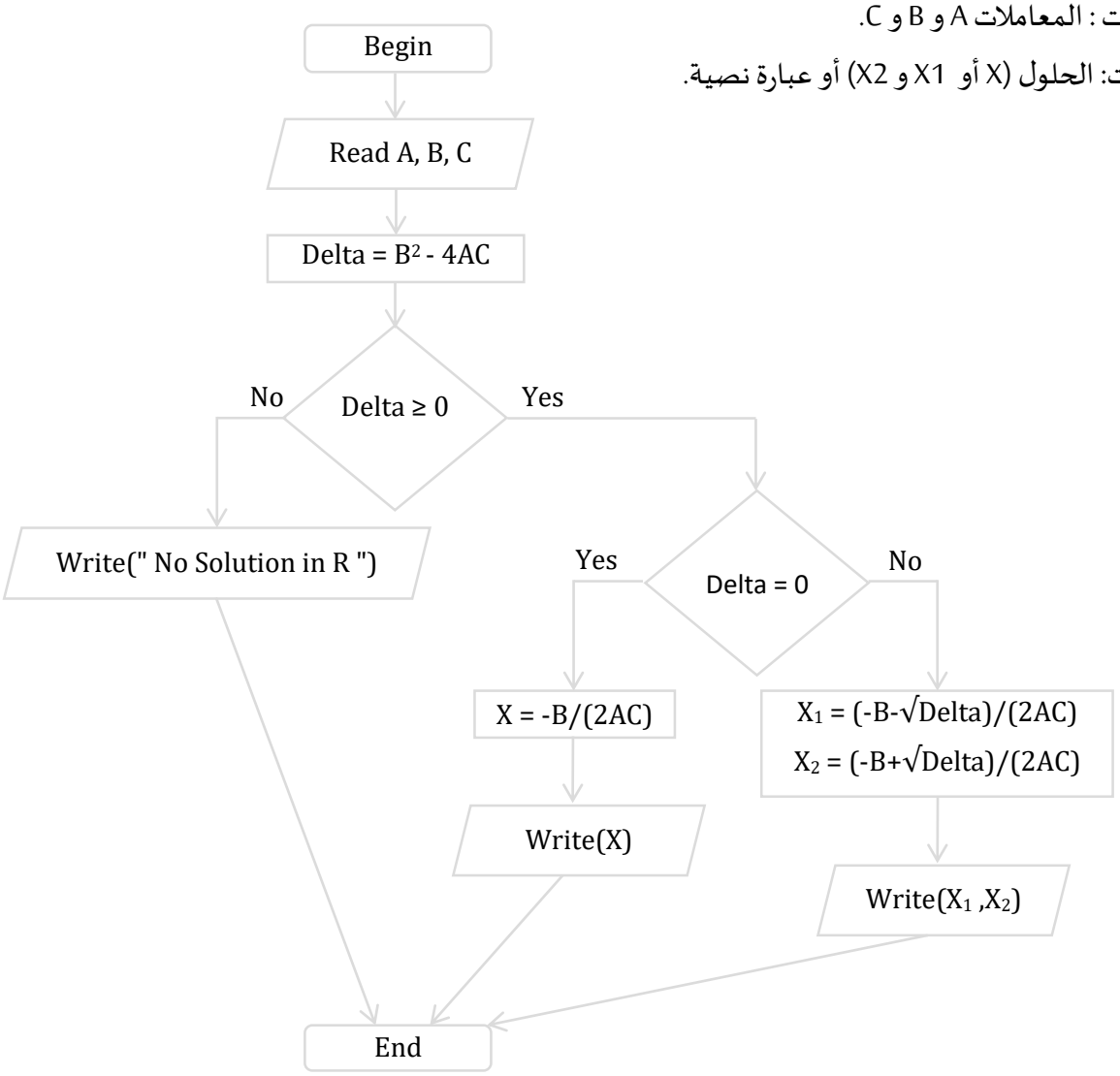
```
Algorithm Division
Variables:
   A, B, C, Q, R: integer
Begin
   Read(A,B)
   If A < B then
      C ← A
      A ← B
      B ← C
   End if
   Q ← A div B
   R ← A mod B
   Write("The quotient of division is :",Q)
   Write("The remainder of division is :",R)
End
```

معالجة حالة القسمة على 0:

بعد تركيبة تبديل العددين A و B، نضيف تركيبة جديدة "إذا" (if)بحيث إذا كان B غير معدوم نتم العملية بطريقة عادية أما إذا كان B معدوما نظهر رسالة (عبارة نصية) للمستعمل تبين بأنه لا يمكن القسمة على 0.

تمرين 2: مخطط انسيابي يمثل طريقة حل معادلة من الدرجة 2:

- المدخلات : المعاملات A و B و C.
- المخرجات: الحلول (X أو X1 و X2) أو عبارة نصية.



ملاحظة:

هذا المخطط لم يأخذ بعين الاعتبار حالة أن تكون المعاملات معدومة، وهي حالة جد مهمة.

4. هيكلة البيانات في الخوارزميات (Data structure):

هي طريقة تنظيم وتخزين البيانات في الحاسوب بحيث يمكن الوصول إليها وتعديلها واستخدامها بكفاءة. وهي الأساس الذي تعمل عليه الخوارزميات والبرامج.

تنقسم إلى :

أ. الأنواع البسيطة (Scalar Types):

وهي الأنواع التي تحتوي على قيمة واحدة فقط. وهي:

- **الأعداد الصحيحة (integer):** وهي كل الأعداد التي ليست لها جزء عشري (لا تحتوي على فاصلة)، وهي تشمل الأعداد الطبيعية مع الأعداد السالبة (النسبية). عند التصريح بأي متغيرة من هذا النوع فإن الجهاز يقوم بحجز مساحة من الذاكرة توافق حجم أكبر قيمة يمكن أن يحتويها هذا النوع. في معظم لغات البرمجة الحجم المخصص لهذا النوع هو 4 بايث (int32). هذا النوع يقبل كل العمليات الحسابية (+، -، *، /، القسمة الصحيحة وباقي القسمة الصحيحة)، بالإضافة إلى عمليات المقارنة (<، >، =، ≠، ..).

  ملاحظة : معظم لغات البرمجة تستعمل عدة أنواع مندرجة ضمن هذا النوع. مثلا لغة C تستعمل:

  short= int16، int=int32، وlong=int64.

- **الأعداد الحقيقية (real/float):** وهي الاعداد التي تحتوي على فاصلة (جزء صحيح وجزء عشري). المساحة المخصصة لتخزين هذا النوع تكون أكبر من المساحة المخصصة للأعداد الصحيحة. وهي تقبل كل العمليات المذكورة سابقا تقريبا (+، -، *، /) والمقارنات.

- **الأعداد المنطقية (Boolean):** وهي تقبل القيمتين "صحيح" أو "خطأ" (true/false). وتقبل العمليات المنطقية (And، Or، Not) والمقارنات.

- **الحروف (الرموز) (Character):** وهي تحتوي على حرف واحد أو رمز (...,'@','%','<','5','g','A'). للتفريق بين اسم متغيرة وقيمة من نوع حرف نستعمل الشولتين البسيطة ' ' (quotation marks) أو المضاعفة " " (Double quotation marks) للدلالة على هذه الأخيرة. العمليات التي تتقبلها هذه المتغيرات هي المقارنات والتحويل ما بين الرمز ورقمه.

  يخزن هذا النوع وفق نمطين: (ASCII table) وتتضمن 128 حرفا ورمزا تشمل الحروف الانجليزية مع الأرقام والرموز. والنمط (Unicode) وقد صمم لتمثيل كل حروف لغات العالم ورموز الايموجيه والرموز الرياضية، ويمكن أن يحتوي على أكثر من 1.1 مليون حرف (فعليا يستعمل لحد الآن 149186 رمز فقط).

ب. الأنواع المركبة البسيطة (Simple Compound Types):

وهي الأنواع التي تحتوي على عدة قيم ولكنها بسيطة من حيث التنظيم وليست هياكل معقدة وأهمها:

- **الجدول(Array / Tableau) :** هو مجموعة من القيم من نفس النوع مخزنة في أماكن متتالية في الذاكرة وهو غالبا ذو حجم ثابت. يمكن الوصول إلى عناصره بطريقة مباشرة بعن طريق المؤشرات (index). هذه العناصر قابلة للتعديل والترتيب والبحث والمقارنة.

  مثال1 : جدول ذو بعد واحد:

1	2	3	4	5	6	7	8
3	10	5	-15	3	50	100	20

ويتم التصريح به كما يلي:

T : جدول[1 : 100] من نوع صحيح ، T : Array[1 :100] of integer ، T : Tableau[1 :100] d'entier

بما أن الجدول ذو حجم ثابت فإنه يتوجب عند التصريح به إعطاءه حجما قياسيا، ثم استعمال عدد صحيح N للدلالة على عدد العناصر المراد استعمالها.

للدلالة على العنصر السادس نكتب T[6] وقيمته هي 50. عند كتابة 30←T[5] فإن العنصر الخامس تصبح قيمته 30 عوض القيمة السابقة 3.

مثال2 : مصفوفة (جدول ذو بعدين) (Matrix) (Matrice):

i \ j	1	2	3	4	5
1	10	5	-15	3	50
2	1	11	10	-20	17
3	5	13	36	12	5

M: جدول[3:1 ، 5:1] من نوع صحيح ، M : Array[1:3 ; 1:5] of integer.

عند كتابة M[i,j] فإن المؤشر i يدل على رقم السطر و المؤشر j يدل على رقم العمود.

عند كتابة 0←M[2,4] فإن العنصر الموجود في تقاطع السطر 2 والعمود 4 تصبح قيمته 0.

- **السلسلة النصية (string) (chaine de caractères):** هي جدول من الرموز (الحروف) (character) غير أن طولها قابل للتعديل أما عناصرها فهي غير قابلة للتعديل المباشر. العمليات التي يمكن القيام بها عليها هي: حساب الطول (عدد العناصر)، المقارنات، الدمج، استخراج جزء منها، المرور على عناصرها عن طريق مؤشر.

مثال:

التصريح : S1 و S2: سلسلة نصية ، S1, S2 : string ، S1,S2 : chaine de caractères.

الاستعمال : S1←"Algeria " ، S2←"my country" (يتم استعمال الشولتين للدلالة على أنها نص)

عند كتابة : S2 = S1 + "is " + S2 فإن S2 تصبح S2="Algeria is my country".

عند كتابة S1[3] نحصل على الحرف الثالث "g" .

الكتابة S1[7]←"e" خاطئة (لا يمكن تعديل العناصر مباشرة).

الكتابة : length(S1) تعبر على طول السلسلة = 7 (عدد العناصر).

قبل التعديل Length(S2) = 10 أما بعد التعديل فهي تعطي القيمة 21 (المسافات تعتبر حروفا).

- **القائمة (list) (liste):** هي قائمة من العناصر المرتبة ذات حجم ديناميكي (متغير) قابلة للتعديل المباشر وهي ليست بالضرورة متجاورة في الذاكرة.

مثال:

التصريح : Fruits = ("Apple", "Orange", "Grappes", "Banana", "Watermelon", "Strawberry")

الوصول: Fruits[3] = "Grappes"

التغيير: Fruits[6] ← "Pear"،

يمكن إضافة في وسط القائمة أو ي آخرها أو حذف عنصر عن طريق دوال خاصة.

- **السجل (record) (enregistrement)**: هو بنية تجمع عدة قيم من أنواع مختلفة تحت اسم واحد، تسمى حقولا (champs) (fields).

مثال:

التصريح : يتم التصريح بها عادة في الأنماط (Type)

```
Type  Student = record
        Name :string[20]
        Age : integer
        Mark : float
        end
```

مع المتغيرات نكتب S1 : Student

الاستعمال: S1.Name ← "Ali" ، S1.Age ← 20 ، S1.Mark ← 15

ت. الأنواع المعقدة (Complex Types):

وهي أنواع مركبة قد تحتوي على عدة أنواع من الأنواع المذكورة سابقا. وهي ديناميكية قابلة للتعديل. نذكر منها:

القوائم المرتبطة (Linked list) (Listes chainées).

البنى الشجرية (Tree) (Arbre).

الرسوم البيانية (Graph).

أمثلة متعددة:

مثال 1: خوارزم لحل معادلة من الدرجة 2 (مع معالجة كل الحالات الخاصة) (Edge cases):

```
Algorithm Equation2
Variables:
   A, B, C, X1, X2, Delta: integer
Begin
   Read(A,B,C)
   If A ≠ 0 then
      Delta←B*B-4*A*C
      If Delta <0 then
         Write("There is no solution in R")
      Else
         If Delta = 0 then
            X1← -B/(2*A)
            Write("Double Solution X =",X1)
         Else
            X1←(-B - sqrt(Delta))/(2*A)
            X2←(-B + sqrt(Delta))/(2*A)
            Write(Two solutions X1=",X1,"X2=",X2)
         End if
      End if
   Else
   If B≠0 then
      X←-C/B
      Write("1st degree equation; X =",X1)
   Else
      If C≠0 then
         Write("There is no solution in ℝ or ℂ")
      Else
         Write("All real and complex numbers are
         solutions")
      End if
   End if
End
```

ملاحظة:

- تستعمل sqrt للدلالة على الجذر التربيعي، كما يمكن استعمال رمز الجذر √.

مثال 2:

اكتب خوارزم لقراءة العلامات الي تحصل عليها طالب في عدة مواد، ثم حساب المعدل العام مع الأخذ بعين الاعتبار للمعاملات وأخيرا طباعة النتائج.

ارسم المخطط الانسيابي الموافق.

ملاحظات:

- Courses هي قائمة المواد المدروسة و Weights هي المعاملات المقابلة.
- Marks جدول من 6 عناصر لتسجيل العلامات المحصل عليها و A متغيرة لحساب المعدل.
- الأسطر التي تبدأ بـ // هي عبارة عن تعليقات تستعمل لشرح أجزاء الخوارزم للقارئ وليست لها أي تأثير على التنفيذ.

```
Algorithm StudentResult
Variables:
   Courses:["Maths", "Physics", "Chemistry", "Informatics", "Arabic", "English"]
   Weights:[2,2,1,2,3,1]
   Marks:array[1:6]of real
   A:real
   i:integer
Begin
   // Reading Marks
   For i=1:6 do
       Write("Enter the mark obtained in", Courses[i])
       Read(Marks[i])
   End for

   // Calculating the average
   A←0
   For i=1:6 do
       A← A + Marks[i]*Weights[i]
   End for
   A ← A/11

   // Printing result:
   Write("The final result is :")
   For i=1:6 do
       Write(Courses[i], " :",Marks[i])
   End for
   Write("The  average is :",A)
End
```

خوارزم نتائج

Courses : ["انجليزية" ,"عربية" ,"إعلام آلي" ,"كيمياء" ,"فيزياء" ,"رياضيات"]

Weights : [2,2,1,2,3,1]

Marks : جدول[1:6] حقيقي

A: عدد حقيقي

i: عدد صحيح

بداية

من أجل i = 1 : 6 افعل

اكتب("ادخل العلامة المحصل عليها في مادة : " ، Courses[i])

اقرأ(Marks[i])

نهاية من أجل

// حساب المعدل

A←0

من أجل i = 1 : 6 افعل

A←A+Marks[i]*Weights[i]

نهاية من أجل

A←A/11

// إظهار النتائج

اكتب(" النتائج المحصل عليها هي :")

من أجل i = 1 : 6 افعل

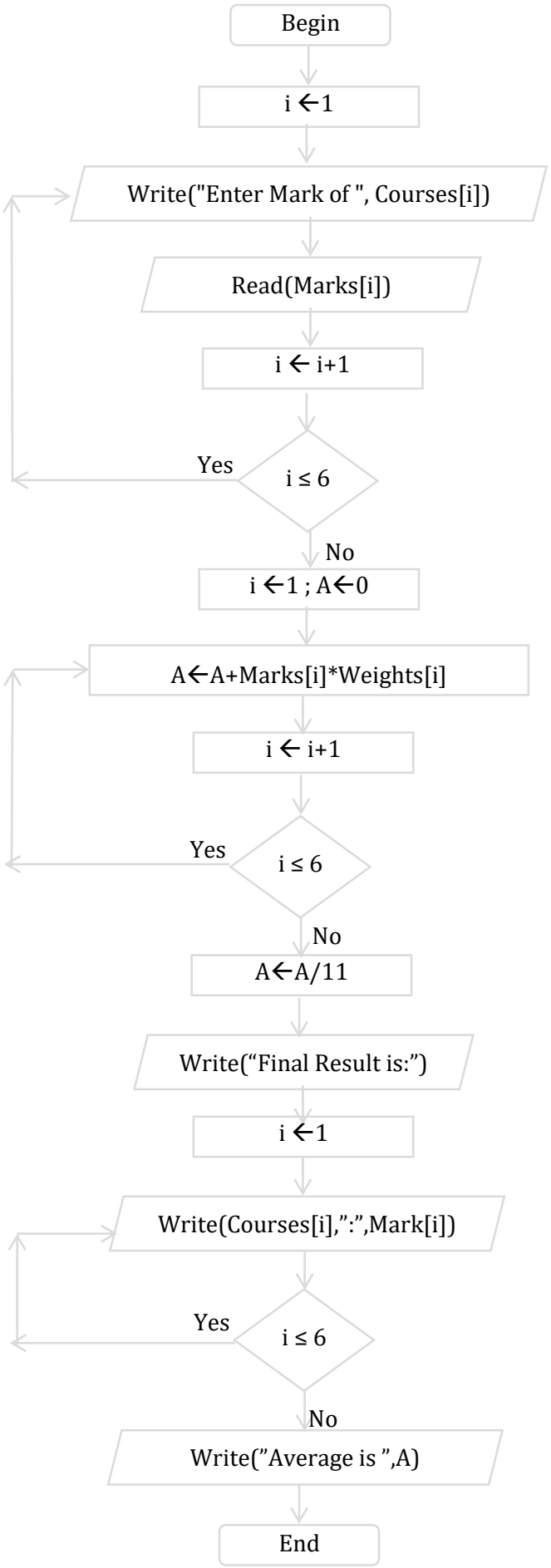
اكتب(Courses[i] ، ": "، Marks[i])

نهاية من أجل

اكتب(" المعدل المحصل عليه هو "، A)

نهاية

المخطط الانسيابي المو افق:



مثال 3:

اكتب خوارزم لقراءة كلمة في متغيرة S1 ثم ينسخها بالترتيب العكسي للحروف في متغيرة أخرى S2 ومن ثم يتحقق منها إن كانت متناظرة (palindrome) أم لا.

اكتب خوارزم للتأكد من أن كلمة متناظرة أم لا دون استعمال متغيرة أخرى.

```
Algorithm palidrome1
Variables:
   S1,S2: string
   i,N: integer
Begin
   Read(S1)
   N←Length(S1)
   S2←""
   For i=N:1 (step=-1) do
      S2←S2 + S1[N]
      N←N-1
   End for
   If S1=S2 then
      Write(S1, "is palindrome")
   Else
      Write(S1,"is not palindrome")
   End if
```

```
Algorithm palidrome2
Variables:
   S1: string
   i,N: integer
Begin
   Read(S1)
   N←Length(S1)
   While (i<N) and (S1[i]=S1[N]) do
      i←i+1
      N←N-1
   End while
   If i>N then
      Write(S1, "is palindrome")
   Else
      Write(S1,"is not palindrome")
   End if
End
```

مثال4:

اكتب خوارزم لقراءة وإظهار مصفوفة (جدول ذو بعدين) ذات M سطر و Nعمود من الأعداد الصحيحة، ثم حساب عدد العناصر المعدومة.

```
Algorithm Matrix
Variables:
   Mat: array[1:100,1:100]of integer
   M,N,I,j: integer
   Count: integer
Begin
   // Reading Matrix elements
   For i = 1 : M do
      For i = 1 : M do
         Read(Mat[i,j])
      End for
   End for
   // Desplaying Matrix elements
   For i = 1 : M do
      For i = 1 : M do
         Write(Mat[i,j])
      End for
   End for
   // Counting null elements
   Count←0
   For i = 1 : M do
      For i = 1 : M do
         If Mat[i,j]=0 then
            Count←Count+1
         End if
      End for
   End for
   Write("Number of zeros is :", Count)
End
```