الفصل 2 : الخوارزميّة المتسلسلة البسيطة

1. تمهيد

عندما ابتُكِر الحاسوب في أربعينيات القرن الماضي، كان يعمل بواسطة الصمّامات الإلكترونية، وكانت برمجته (كتابة الأوامر) تتمّ بإدخال سلسلة من الأعداد تتكون من الصّفر (0) والواحد (1). وكان ذلك صعبا على المبرمجين. ولكن بابتكار الترانزستور صَغُر حجم الحاسوب كثيرا، وزادت إمكانيّاته. وتمّ ابتكار لغات أسهل للاستخدام تشبه الى حدّ كبير لغة الانسان. وتُسمّى هذه اللغات باللغات عالية المستوى. حيث يمكن ترجمة أيّ برنامج مكتوب في هذه اللغات إلى لغة الحاسوب (0،1) باستعمال برنامج التجميع (compilateur) بطريقة سريعة واوتوماتيكية.

في هذا الفصل سوف نتطرّق إلى مفهوم اللغة، بنية الخوارزميّة، ومفهوم المتغيّرات والثوابت، كما سنرى بعض الأنواع البسيطة للتعليمات. مثل: الإسناد، وتعليمتي الإدخال والإخراج.

2. مفهوم اللغة ولغة الخوارزميّة

2.1. اللّغة Language

هي طريقة الاتّصال والتفاهم بين الأشخاص، وفي حالة الحاسوب، هي الطريقة التي يفهم بها الحاسوب أوامر الإنسان. وتتكون اللغة من حروف، رموز، ،مفردات، قواعد نحويّة ومعاني.

- **الأبجدية** L'alphabet: وهي مجموعة الحروف والأرقام والرموز.
- **المفردات** vocabulaire: وهي مجموعة الرموز والكلمات، سواء كانت كلمات **محجوزة** او كلمات **يعرّفها** المبرمج.
- **القواعد النحوية** la syntaxe: عبارة عن قوانين أو ضوابط التي تتحكّم في تجميع وتمَوضع الرموز والمفردات.
- **الدلالة او المعنى** la sémantique: تُحدِّد معنى كلّ تعليمة من التعليمات التي يمكن بناؤها في اللغة ، ولا سيما، ما سينتج عنها عند التنفيذ.

2.2. لغة البرمجة Programming Language

تُوفِّر لنا لغة البرمجة إطارا لتطوير الخوارزميّات، وإنتاج البرامج التي يمكن الحاسوب تطبيقها. وبصفة أخصّ، تسمح لنا بوصف هياكل البيانات التي سيتمّ التعامل معها في الحاسوب، والعمليات التي سيتمّ تنفيذها. وهي لغة متوسطة بين لغة الإنسان ولغة الالة. حيث يمكن للإنسان فهمها، كما يمكن للحاسوب ترجمتها الى لغة 0 و 1 التي يفهمها.

هناك العديد من لغات للبرمجة، لكلّ منها مزاياها وقواعدها الخاصّة، التي تجعلها مناسبة بدرجات متفاوتة لنوع معيّن من البرمجيات، وعلى المبرمج أن يكون مُلمّا ببعض منها، وأن يعرف ما هي اللغة المناسبة لكلّ نوع من التطبيقات.

2.2.1. أنواع لغات البرمجة

يمكن تقسيمها إلى قسمين

- Interprété: إذا كان البرنامج لا يُترجَم كُليّا إلى لغة الآلة، وإنّما يُترجَم ويُنفّذ تعليمة بتعليمة. مثل: Matlab ولغات الويب.
- compilé: إذا كان البرنامج يتمّ ترجمته كُليّا الى لغة الآلة قبل أن يتمّ تنفيذه. مثل: C.

أمثلة

- لغة C (سي) و++C: وهي التي ستُستعمل في هذا الدرس وتُعتبر اللغة الأم للعديد من اللغات الأخرى.
- لغة باسكال pascal: قريبة من الخوارزم وتُستعمل في الأوساط البيداغوجية.
- لغة دالفي dephi ولغة الوين داف: جيّدة من أجل تطوير برامج خاصّة بالتسيير.
- لغة جافا: جيّدة لتطبيقات الشبكة والهاتف المحمول
- لغة #C (سي شاربْ) وفيجوال بيسك: جيّدة لتطوير برامج خاصّة ببيئة ويندوز.
- لغة Objective C (Xcode): خاصّة بتطوير برامج منتجات Apple (ماك، أي باد وايفون).
- لغة php: خاصّة بتطوير مواقع الواب
- لغة Matlab وبايثون Python: خاصّة بتحليل البيانات، وموجّهة للمهندسين.

2.2.2. *المترجِم او المُجمِّع Compiler*

هو برنامج كمبيوتر يقوم بتحويل شفرة المصدر(code source) المكتوبة بلغة برمجة معينة إلى شفرة مستهدفة يمكن تنفيذها مباشرة بواسطة جهاز كمبيوتر.

هناك العديد من لغات البرمجة، لا يمكن للمبرمج ان يلِمّ بها جميعا. وحتى يتسنّى للمبرمجين العمل كفريق واحد، وتبادل الحلول فيما بينهم، لابدّ لهم من صياغة هاته الحلول في لغة الخوارزميّة. وتعتبر لغة الخوارزمية اللغة المشتركة بين جميع المبرمجين.

2.2.3. *بيئة التطوير المتكاملة The Integrated Development Environment*

لكتابة برنامج يمكن استعمال أيّ محرّر نصوص، مثل المفكرة. لكن هذه الطريقة تجعل عملية التطوير صعبة للغاية. لذلك توجد مجموعة من البرامج تُوفّر جميع الأدوات الضروريّة لعمليّة التطوير تدعى بـ IDE. حيث يُوفّر IDE جميع الأدوات التي نحتاجها لتصميم التطبيقات، وتطويرها، واختبارها، وتصحيحها، ونشرها، ممّا يُتيح سهولة وسرعة في التطوير.

يتضمّن IDE جميع الأدوات اللاّزمة لبدء تصميم التطبيقات. مثل:

- محرّر الكود Code Editor: لكتابة وتحرير شفرة البرنامج. حيث يقوم بالتنسيق التلقائيّ، ممّا يُسهّل عمليّة القراءة.
- إدارة المشروع Project Manager: لإدارة الملفات التي تُشكّل مشروعًا واحدًا.
- مصحّح الأخطاء debugger: لإيجاد وإصلاح الأخطاء في التعليمات البرمجيّة.
- اختصارات لتجميع وتنفيذ البرنامج.
- أدوات أخرى ...

مثال: Dev-C++ ، Embarcadero، visual studio...

2.3. العناصر النحوية الأساسيّة

2.3.1. *الكلمات المحجوزة Mots clés*

هي كلمات لها معنى مسبّق بالنسبة للغة البرمجة، ولا يمكن استعمالها من طرف المبرمج لإنشاء عناصر جديدة خاصّة به. لكل لغة كلماتها الخاصّة، مثل كلمة algorithme وdébut وfin في الخوارزم، وكلمة if، while في لغة C.

2.3.2. *القيم valeurs*

وهي الأعداد، والرموز(توضع دوما بين علامتي اقتباس أُحادية ' ')، وسلاسل الرموز (توضع دوما بين علامتي اقتباس ثنائيّة " ")، وvrai وfaux. انظر الأنواع.

مثال: 7، 5، -2، 3.12، -2.6e-7 ، 'خ'، 'a'، '5'، '!'، "قيمة"، "azerty"، "A"، vrai، faux

2.3.3. المُعرّف *L'identifiant*

هو الاسم الذي يعطيه المبرمج لأيّ عنصر من عناصر الخوارزميّة التي يريد انشاءها. مثل: اسم الخوارزميّة، اسم متغيّر، اسم نوع، اسم ثابت، اسم دالة... وله قواعد وشروط في لغة C والتي سنتبنّاها في الخوارزميّة.

قواعد تسميّة المُعرّفات:

- يمكن لاسم المُعرّف أن يحتوي فقط على رموز حرفية وعددية منA إلى Z، من a إلى z ومن0 إلى 9، كما يمكن استعمال الرمز « _ » شرْطة سُفليّة (tiret du huit)
- يجب أن يكون كلمة واحدة، أي لا يمكن للاسم أن يحتوي على فراغ (مسافة) " ".
- يجب أن تبدأ التسميّة بحرف او « _»، ولا تبدأ برقم.
- يجب ألاّ يكون كلمة محجوزة.
- يجب أن يكون المُعرّف وحيدا، فلا يمكن تعريف أكثر من عنصر بنفس الاسم.
- يستحسن استعمال أسماء ذات دلالة، مثلا نستعمل Largeur بدلا من x.

أمثلة

معرّفات صحيحة

x, pi, Mat_info, estVide ,n5, _debut, _0a (يستحسن تجنبها)

معرّفات خاطئة

رموز غير مقبولة α, ج, π, é ,élève

تبدأ برقم 3a

وجود الفراغ Mat info

كلمات محجوزة debut, fin

2.3.4. العمليّات *Operators*

- العمليّات الحسابيّة Arithmetic Operators

العملية	C	ملاحظة	مثال
+ -	+ -	للإشارة	-a 7- 3+
+ -	+ -	المعاملين صحيحين النتيجة عدد صحيح، أحدهما حقيقي النتيجة عدد حقيقي.	5+3.0 حقيقي
*	*	للجداء مثل الجمع والطرح.	5*3 صحيح
/	/	للقسمة الحقيقية. النتيجة دوما عدد حقيقي	5/3 او 5.0/3 en C
mod	%	لحساب باقي القسمة. المعاملين صحيحين النتيجة دوما عدد صحيح.	5 mod 3 او 5%3 en C
div	/	لحساب حاصل القسمة (القسمة الصحيحة). مثل باقي القسمة.	5 div 3 او 5/3 en C
^		لحساب الاس وفي لغة C نستعمل الدالة pow() النتيجة عدد حقيقي	5^2 او pow(5,2) en C
√		لحساب الجذر وفي لغة C نستعمل الدالة sqrt() النتيجة عدد حقيقي	√5 او sqrt(5) en C

- عمليّات المقارنة Relational Operators

العملية	C	ملاحظة
>, >=, <, <=		نفسها في لغة C
=	==	في C "=="مرتين = وتُقرأ تساوي، أما "=" تُستعمل للاستناد وتُقرأ يأخذ.
≠	=!	وتُقرأ لا يساوي

نتيجة المقارنة دوما تكون من نوع منطقيّ Boolean أي vrai أوfaux

- العمليّات المنطقيّة Boolean Operators

العملية	C	ملاحظة
non للنفي	!	صحيح في حالة المعامل خاطئ، وخاطئ في حالة المعامل صحيح.
et للوصل "و"	&&	صحيح vrai في حالة المعاملين صحيحين، والا خاطئ.
ou للفصل "او"	||	خاطئ faux في حالة المعاملين خاطئين، والا صحيح.
xou او الشرطية		صحيح في حالة أحدها صحيح والأخر خاطئ، وخاطئ في حالة التساوي.
= التكافؤ	==	صحيح في حالة متساويين

- العمليّات على السلاسل String Operators

+ تُستعمل لدمج سليلتين مثل:

"bonne"+"Chance" تنتج "bonneChance" دون إضافة الفراغ. "bonne" + " " + "Chance" تنتج "bonne Chance"

- الأولويّة

عند حساب أيّ عبارة نتّبع الأولويّة الملخصّة في الجدول التالي، وفي حالة تساوي الأولوية، تكون الأولوية للعملية التي على اليسار.

الأولوية	العملية
0	()
1	non، + و- الإشارة
2	* / div mod
3	+ -
4	>, >=, <, <=
5	≠, =
6	et
7	ou

تُستعمل الاقواس من أجل تغير الأولويّة (وفي بعض الأحيان التوضيح)

مثال:

2.3.5. العبارة *Expression*

هي عبارة عن بُنية من القيم ومُعرّفات، مربوطة بعمليات، وعند حسابها نتحصّل على قيمة واحدة. ويتمّ انشاؤها باستخدام القيم والمتغيرات والاقواس والعمليات.

مثال: لنفرض ان a=2 وb=3 وok=vrai

العبارة	النتيجة	العبارة	النتيجة	العبارة	النتيجة
5	5	a+3	5	ok	vrai
a	2	"In"+"fo"	Info	a*(b-7)>8 et ok	faux

2.3.6. التعليمة *Instruction*

وهي جملة او خطوة من الحلّ، أي الأمر الذي سيُنفّذ. وتدعى أيضا بالبيان statement.

2.3.7. كتلة تعليمات *Bloc d'instructions*

هي مجموعة تعليمات تبتدئ بكلمة Début وتنتهي بـ Fin، أو تبدأ بكلمة محجوزة تحدد البداية مثل si وتنتهي بكلمة fin + كلمة البداية مثل finSi. وتبدأ بـ { وتنتهي بـ } في C.

مثال

خوارزم	C

Début
  Instruction 1
  …
  Instruction n
Fin
	{
	  Instruction 1 ;
	  …
	  Instruction n ;
	}

2.3.8. التعليقات *Les commentaires*

هي نصوص يتمّ تجاهلها أثناء ترجمة البرنامج، وليست جزءا من الحلّ (الخوارزمية). يتمّ إضافتها إلى البرامج لترك الشروحات وتسهيل الفهم. يتمّ تحديد التعليقات في لغة C باستعمال "//" لإضافة تعليق يتكون من سطر واحد. حيث يبدأ بـ // وينتهي بالرجوع الى السطر. كما يمكن إضافة تعليقات تبدأ بـ « /* » وتنتهي بـ « */ »، والتي يمكنها أن تمتد لعدّة أسطر.

مثال:

```
تعليق من سطر واحد //
تعليق من */
عدة أسطر*/
```

2.4. التعبير عن الخوارزميّة

يمكن التعبير عن خوارزميّة عبر الكتابة باللغة الطبيعيّة. مثل: العربية او الفرنسية. إلّا أنّ اللغة الطبيعية ينتابها الغموض وغير دقيقة. لذلك نقوم بكتابة الخوارزميّة باستعمال أشباه الكود، الهياكل التنظيميّة ولغات البرمجة.

1- شبه الكود (Pseudocode): وصف الخوارزميّة بلغات البشر كالعربيّة أو الفرنسيّة أو الإنجليزيّة بطريقة مشابهة للغات البرمجة. البعض يستخدم الكثير من التفاصيل (لتصبح قريبة من لغات البرمجة)، والبعض الآخر يستخدم القليل (أي أقرب للغة البشر)... فلا قاعدة معيّنة لكتابة هذا النوع من الشفرات.

2- الهيكل التنظيميّ: هو تمثيل مصوّر للخوارزميّة يوضّح خطوات حل المشكلة من البداية إلى النهاية، مع إخفاء التفاصيل لإعطاء الصورة العامّة للحلّ. ويتم فيها استعمال أسهم واشكال هندسية متّفق عليها لتمثيل الخطوات.

3- الشفرة البرمجية code: حيث يتمّ كتابة الخوارزميّة بلغة البرمجة مباشرة مثل لغة C، وفي هذه الحالة يمكن للحاسوب ترجمتها إلى اللغة الثنائية، ليتمّ تنفيذها مباشرة من طرف المعالج.

3. أجزاء الخوارزميّة

تتكوّن الخوارزميّة من مجموعة من البيانات ومجموعة من التعليمات على هذه البيانات. وتشبه الخوارزميّة إلى حدّ كبير في بنيتها وصفة الطبخ. ففي العادة، تتكوّن الوصفة من: عنوان الوصفة، ثم تأتي المكونات والمقادير، وفي الأخير، طريقة التحضير.

وتأخذ الشكل التالي:

```
Algorithme الاسم
   التصريح بالبيانات التي نحتاجها
Début
   التعليمات
Fin
```

وتتكوّن الخوارزميّة من ثلاث أجزاء أساسية هي:

- العنوان en-tête: ويتكوّن من كلمة **Algorithme**، متبوعة بالاسم الذي يشرح المسالة المراد حلها. ويجب أن يكون معرّفا صحيحا.
- التصريحات Déclarations: ويتمّ فيها حجز أماكن في الذاكرة للبيانات (الثوابت والمتغيرات)، التي سيتمّ استعمالها كمعطيات ونتائج.
- التعليمات Instructions : وهي مجموعة الخطوات أو الأوامر التي ستنفّذ عند تشغيل الخوارزميّة. وتبتدئ بكلمة Début، وتنتهي بكلمة Fin. ولها 5 أنواع اساسية:

1. تعليمة الإسناد.
2. تعليمة القراءة (لإدخال المعطيات).
3. تعليمة الكتابة (لإظهار النتائج).
4. التعليمة الشرطية.
5. التعليمة التكرارية.

4. البيانات: الثوابت والمتغيّرات

4.1. الثابت Constante

هو قيمة (أعداد أو رموز) لها اسم، ولا يمكن تغييرها أثناء تنفيذ البرنامج.

التصريح عن الثوابت :

Const Identificateur=valeur

Const أو Constante: هما كلمتان محجوزتان، تسمحان بالتصريح عن الثوابت.

identificateur: هو اسم المعرّف الذي يُطلق على الثابت.

valeur : هي القيمة التي تُعطى للثابت.

أمثلة:

```
Const
     Pl= 3.1415926
     DEP  = "قسم الاعلام الالي"
```

فوائد الثوابت

- تقليص الكود حيث يمكن تعويض عبارة كبيرة بكلمة صغيرة. مثل: P بدلاً من 3.1415926.
- تجنّب الاخطاء من خلال توفير اسم ذي معنى. مثلا: PI بدلاً من 3.1415926.
- تبسيط صيانة الكود، بحيث يتمّ تغيير القيمة في مكان واحد فقط.

4.2. المتغيّر variable

هو مكان في الذاكرة يستخدم لحفظ البيانات. له اسم، نوع وقيمة، (له عنوان في السداسي الثاني).

- **اسم**: عبارة عن معرّف يستعمله المبرمج للرجوع إلى القيمة المخزنة والتعامل مع المتغيّر. مثل: poid.
- **نوع**: كل شيء في الحاسوب عبارة عن 0 و1. فالنوع يحدّد كيفيّة ترجمتها، كما يحدّد الحجم اللاّزم حجزه على الذاكرة. أي عدد البيتات (bits)، والعمليّات المسموح بها. مثال: int (32 bits).

- **قيمة**: هي محتوى البيتات التي يتكوّن منها، أي قيمتها. وفي العادة، هي الشيء الذي يتغيّر أثناء تنفيذ البرنامج. مثل: 1101 والذي يمثل العدد 13، أو العدد -5 إذا اعتبرنا 1 الذي على اليسار هو الإشارة "-".

التصريح عن المتغيّرات :

```
Var Identificateur: Type
```

Var أو Variable: هما كلمتان محجوزتان، تسمحان بالتصريح عن المتغيّرات.

Identificateur: هو اسم المعرّف الذي يُطلق على المتغيّر.

Type: هو نوع المتغيّر.

يمكن استعمال "," للتصريح بعدّة متغيّرات من نفس النوع.

أمثلة:

```
Var
     age : entier
     sexe : caractères
     x, y, z: réel
```

ملاحظة: اصطلاحا، كتابة الثوابت تكون بحروف كبيرة MAJUSCULES، والمتغيّرات بحروف صغيرة minuscules

5. أنواع البيانات Types de donnés

النوع : هو **المجال** الذي تنتمي إليه البيانات. أي أعداد، نصوص، صور، صوت او فيديو. حيث يحدّد النوع كيفيّة **ترجمة** البيتات (bits) المكونة للمتغيّر، كما يحدّد **الحجم** اللاّزم حجزه على الذاكرة، أي عدد البيتات، **والعمليات** المسموح بها. عندما تقوم بتعريف متغيّر ، يجب عليك تحديد نوعه. ويوجد 5 أنواع أساسية في الخوارزم هي:

1. الأعداد الصحيحة Integer Entier مثل: -5، 0، 1، 13
2. الأعداد الحقيقية Real Réel مثل: -8.17، 1.0، 3.14
3. منطقي Boolean Booléen: يتضمّن إحدى القيمتين: صحيح أو خطأ faux vrai.
4. الرموز Character Caractère: ويشمل جميع الرموز الموجودة على لوحة المفاتيح. مثل: الأرقام، الحروف بجميع اللغات، والرموز المطبوعة (المرئية)، والغير مطبوعة. وتكون دوما بين علامتي اقتباس أحاديّة (فاصلة علويّة). مثل: 'a'، 'M'، '1'، '+'، ',' ، 'س' .
5. سلسلة الرموز Chaines de caractères، String: هي مجموعة من الرموز، يكون طولها 0 او أكثر، وتكون دوما بين علامتي اقتباس ثنائيّة. مثل: "informatique"، "Bonne chance\n"، "1"، "3.14".

ملاحظات:

- نستعمل "." بدل "," للتعبير عن الأعداد العشرية.
- 1 ليست نفسها .1، وليست نفسها '1'، وليست نفسها "1". الأول عدد صحيح، والثاني عدد حقيقي، والثالث رمز، والأخير سلسلة رموز.
- 'a' ليست نفسها "a". الأولى رمز، والثانية سلسلة طولها 1.
- الحروف الصغيرة ليست نفسها الكبيرة. أي 'a' ليست نفسها 'A'.
- هناك رموز (أزرار) لا تُطبع. مثل: الفراغ ' ' ، الرجوع للسطر '\n'.

- يتمّ استخدام الخطّ المائل العكسيّ، (\) backslash، للتمثيل المرئيّ لبعض الرموز الغير مرئية أو الخاصّة. مثل: سطر جديد '\n'، والجدولة أي فراغ كبير '\t'. ولطباعة " نستعمل "\ ولطباعة \ نستخدم \\.
- هناك سلسلة رموز فارغة. أي لا تحوي أي رمز، طولها 0، ونرمز لها بـ "".

6. العمليّات الأساسيّة

6.1. الإسناد Affectation

وهي العمليّة التي تسمح لنا بتخزين قيمة داخل متغيّر.

variable←exp

variable هو اسم لمتغيّر

exp هي عبارة، (معرّفات، قيم وعمليّات انظر 2.3.4)، يتمّ حسابها للحصول على قيمة واحدة، يتمّ وضعها داخل المتغيّر variable.

← تُقرأ: يأخذ، بالفرنسيّة reçoit، وبالإنجليزيّة get. حيث يشير السهم دوما إلى المتغيّر variable.

لا يمكن للمتغيّر أن يحمل إلّا قيمة واحدة في كل لحظة من تنفيذ البرنامج. وعند تنفيذ العمليّة لا يتغيّر إلّا المتغيّر الذي على اليسار. حيث يفقد القيمة القديمة، ويأخذ القيمة الجديدة. ولكي تتمّ عمليّة الاسناد بشكل صحيح، يجب أن تكون قيمة العبارة التي على اليمين، والمتغير الذي على اليسار، من نفس النوع، أو على الأقل، من نوع متوافق.

مثال

a←5	a تأخذ 5
b←a*2	b تأخذ 10
a←0	a تأخذ 0
b←b-1	b تأخذ 9
c←'b'	c تأخذ حرف b
d←b>a	d تأخذ vrai
s←"name"	s تأخذ كلمة name

قبل استعمال أيّ متغيّر يجب التصريح به، واسناد قيمة ابتدائيّة له. وللحصول على قيمة أيّ متغيّر او ثابت، يكفي كتابة اسمه.

6.2. تعليمات الإدخال والإخراج I/O

للتعامل مع المستعمل، يمتلك المبرمج تعليمتين هما: ()Lire و()Ecrire.

6.2.1. الإدخال Lire()

تُعتبر()Lire دالّة جاهزة في الخوارزميّات. حيث تقوم بإدخال قيمة من عند المستعمل، عن طريق لوحة المفاتيح، وإسنادها إلى المتغيّر الذي بين القوسين. وتُستعمل دائماً لإدخال المعطيات.

الصيغة

```
Lire(variable)
```

variable هو اسم لمتغيّر، ولا يمكن استعمال lire إلّا مع المتغيّرات.

عند تنفيذ البرنامج، وعند مصادفة تعليمة الادخال ()lire، يتوقف التنفيذ مؤقتا حتى يقوم المستخدم بإدخال البيانات. وتنتهي عملية الادخال عن طريق الضغط على مفتاح الإدخال entré. ليُواصل البرنامج تنفيذه.

يمكن إدخال عدّة متغيّرات مرّة واحدة ، مفصولة بفاصلة « , ». وفي هذه الحالة، يقوم المستخدم بإدخال قيمة المتغيّر الأول، ثم يضغط على مفتاح الفراغ espace، ليُدخل بعدها قيمة المتغيّر الثاني، ولا يضغط على مفتاح الإدخال entré، إلّا بعد إدخال قيمة المتغيّر الأخير.

مثال:

`Lire(nom)`	يدخل مجموعة من الحروف مثلا « محمد » ثم يضغط على مفتاح الإدخال entré
`Lire(a, b)`	يدخل عددا مثلا « 15 » ثم يضغط على espace ثم يدخل العدد الثاني مثلا « 20 » ثم يضغط على مفتاح entré

6.2.2. الإخراج Ecrire()

تُعتبر Ecrire() دالّة جاهزة في الخوارزميّات. حيث تقوم بعرض أي شيء نكتبه بداخلها على الشاشة. وتُستعمل دائماً لطباعة النتيجة.

الصيغة

```
Ecrire(exp)
Ecrire("message")
```

`exp` هي عبارة، يتمّ حسابها للحصول على قيمة واحدة، ليتمّ إظهارها على الشاشة.

`"message"` هي أيّ نصّ تريد اظهاره كما هو على الشاشة. حيث لا تتمّ عمليّة حسابه. ويكون بأيّ لغة، أو مجموعة حروف كانت. ويجب أن يكون بين علامتي اقتباس ثنائيّة، والتي لا يتمّ اظهارها على الشاشة.

يمكن اظهار عدّة قيم ونصوص مرّة واحدة، مفصولة بفاصلة « , ».

مثال

`Ecrire(nom)`	تظهر قيمة المتغير `nom` على الشاشة مثلا « محمد »
`a←5` `Ecrire(a+3)`	تظهر على الشاشة 8 دون ان تتغير قيمة a.
`Ecrire("مربع",a,"هو",a*a)`	تظهر على الشاشة مربع 5 هو 25
`Ecrire("b=",a)`	تظهر على الشاشة b=5

ملاحظات

- دوما تأتي قبل العملية Lire العملية Ecrire، وذلك كي تشرح للمستخدم ما هو الشيء المطلوب منه إدخاله.
- يقوم المستعمل بالكتابة على لوحة المفاتيح، بينما يقوم البرنامج (الحاسوب) بالقراءة Lire() من لوحة المفاتيح، ويقوم البرنامج بالكتابة Ecrire() على الشاشة، بينما يقوم المستعمل بالقراءة من الشاشة.
- يمكن تعميم استخدام Lire() للإدخال من جميع وحدات الادخال، وEcrire() للإخراج من جميع وحدات الإخراج.

7. بناء خوارزميّة بسيطة

بعد أنْ رأينا أنّ الخوارزميّة تتكوّن من 3 أجزاء، هي: العنوان، التصريحات والتعليمات، وتعلّمنا كيفيّة التصريح بالثوابت والمتغيّرات، وتعلّمنا 3 أنواع من التعليمات، وهي: الإسناد، القراءة، والكتابة، الآن، يمكننا كتابة بعض الخوارزميّات البسيطة. ولمعرفة المتغيّرات نطرح السؤال: "ما هي المعطيات اللاّزمة، وما هي النتيجة المنتظرة؟". أمّا جزء التعليمات، ففي العادة، يتضمّن ثلاث مراحل أساسية هي:

- المرحلة الأولى "المدخلات" : يتمّ فيها إدخال المعطيات اللاّزمة للّتنفيذ. باستعمال التعليمة Lire().
- المرحلة الثانية "المعالجة" : وتحوي مجموعة من التعليمات اللاّزمة لحل المسألة. باستعمال تعليمة الاسناد.

– المرحلة الثالثة " المخرجات": تُعْرَض فيها النتائج. باستعمال التعليمة ()Ecrire.

مثال:

اكتب الخوارزميّة التي تحسب مساحة دائرة.

```
Algorithme surf_cercle
Const
  P=3.14
Var
  r, s :entier //r est le rayon et s est la surface
Début
  Ecrire("ادخل نصف القطر")
  Lire(r)
  s←p*r*r
  Ecrire("مساحة الدائرة هي:" , s)
Fin
```

اكتب الخوارزميّة التي تحسب معدّل مادّة ASD1.

```
Algorithme moy_ASD1
Var
  cont, td, tp, moy :réel
Début
  Ecrire("ادخل نقطة الامتحان ونقطة الاعمال الموجهة ونقطة الاعمال التطبيقية")
  Lire(cont, td, tp)
  moy←(cont*3+td+tp)/5
  Ecrire("المعدل هو:" , moy)
Fin
```

7.1. تنفيذ الخوارزمية

يهدف تنفيذ الخوارزميّة إلى معرفة قيمة كل متغيّر بعد كل تعليمة. حيث يبدأ التنفيذ من كلمة début الى كلمة fin. ففي البداية، تكون قيم المتغيّرات غير محددة (ليست فارغة)، ثم بعد كل عملية إسناد أو قراءة، تتغيّر قيمة المتغيّر.

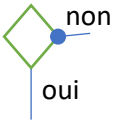
مثال

Algorithme miroir	**التنفيذ**		
Var			
a, b, c :entier	a	b	c
Début			
a←357	357	?	?
c←0	357	?	0
b←a mod 10	357	7	0
c←c*10+b	357	7	7
a←a div 10	35	7	7
b←a mod 10	35	5	7
c←c*10+b	35	5	75
a←a div 10	3	5	75
b←a mod 10	3	3	75
c←c*10+b	3	3	753
Fin			

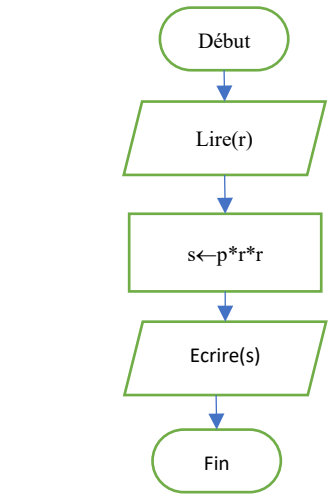
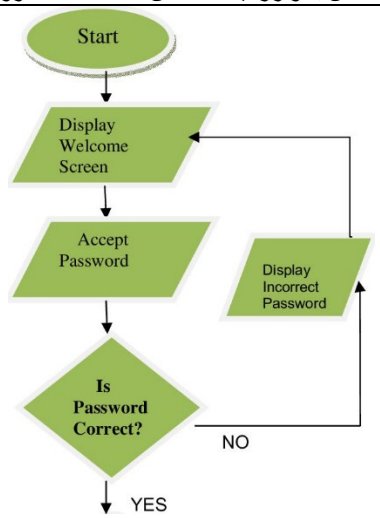
8. تمثيل الخوارزميّة بواسطة الهيكل التنظيميّ L'algorigramme

الهيكل التنظيميّ هو تمثيل مرئيّ للخوارزميّة قبل برمجتها. حيث يوضح لنا تسلسل العمليّات، ويُعطي لنا الصورة العامّة للأجزاء المكوّنة للخوارزميّة. ويتمتّع الهيكل التنظيمي بالعديد من المزايا: فهو يجعل من الممكن تصوّر الأفكار بشكل أفضل. وبما أنه مفهوم من قبل الجميع، فهو يتيح العمل في فريق بسهولة أكبر.

وتُستعمل في الهيكل التنظيميّ مجموعة من الأشكال أهمها:

الرمز	الاستعمال
	Début, fin, interruption رمز للبداية والنهاية والانقطاع.
	Entrée – Sortie رمز الإدخال والإخراج.
	Symboles De Traitement الرمز العام "المعالجة". مثل الإسناد.
	Symboles Logiques: Choix avec condition رمز منطقي، يستعمل في اختبار عبارة، واختيار طريق من بين عدة طرق.

مثال:

الهيكل التنظيمي الخاص بالخوارزميّة التي تحسب مساحة دائرة	الهيكل التنظيمي الخاص بخوارزميّة التأكد من صحة كلمة المرور.
	

9. الترجمة إلى لغة C

لغة C هي لغة أمريّة، مترجمة كليّا، ومن المستوى العالي. وهي واحدة من أكثر لغات برمجة استخدامًا في العالم. وتعتبر اللغة الأم للعديد من لغات البرمجة. في هذا الدرس سنستعمل Dev-C++ كبيئة التطوير المتكاملة IDE. ونُنَوِّهُ هنا إلى أنّ C حسّاس لحالة الأحرف، فهو يميّز بين الأحرف الكبيرة MAJUSCULES، والصغيرة minuscules. فـ main ليست نفسها Main أو MAIN أو mAin. لذلك ننصح بالكتابة دوما بحروف صغيرة. ويجب أن تنتهي كل التعليمات البسيطة (تصريح، إسناد ،ادخال وإخراج، إرجاع) بفاصلة منقوطة ";".

9.1 المعالج المسبّق Le préprocesseur

قبل أن يتمّ تجميع (compilation) البرنامج فعليًا ، تتمّ معالجة ملفات مصدر الشفرة (source) بواسطة معالج مسبّق، والذي يحلّ بعض التوجيهات المعطاة له. مثل: تضمين ملفات أخرى (مكتبات)، تغيير كلمات بعبارات أخرى (ماكرو). ويبدأ التوجيه المعطى للمعالج المسبّق دائمًا بـ #.

9.1.1 #include

يُخبر التوجيه #include المعالج بإدراج محتوى ملف آخر في كود البرنامج الحالي.

```
#include <اسم_الملف>
```

في العادة، تكون هذه الملفات عبارة عن مكتبات من الدوال المُعرّفة والجاهزة للاستعمال.

مثال

- لاستخدام الدوال الخاصّة بالإدخال والإخراج I/O (printf وscanf)، نستخدم المكتبة stdio.h.
- لاستخدام الدوال الرياضيّة (sin، cos، exp، pow، sqrt، ...)، نستخدم المكتبة math.h.
- لاستخدام الدوال الخاصّة بالسلاسل (strlen، ...)، نستخدم المكتبة string.h.

```
#include <stdio.h>
#include <math.h>
```

9.1.2 الماكرو macro

يتمّ تعريف الماكرو، في أبسط أشكاله، على النحو التالي:

```
#define macro <le texte de remplacement>
```

```
#define العبارة_المستبدلة اسم_ماكرو
```

مثال

```
#define N 10
```

يقوم المعالج باستبدال جميع تكرارات الكلمة N بـ 10.

9.2. الأنواع Les types

9.2.1. الأنواع المعرّفة مسبقًا Prédéfinis

الجداول التالية تلخص الأنواع الأساسية في الخوارزم وما يقابلها في C

- الأعداد الصحيحة في الخوارزم من -∞ الى +∞

النوع	الحجم بالبَايْتْ octet	الحجم بالبِتْ bit	المجال
char	1	8	-2^7 ، $+2^7-1$
short	2	16	-2^{15} ، $+2^{15}-1$
long	4	32	-2^{31} ، $+2^{31}-1$
int	4	32	-2^{31} ، $+2^{31}-1$

- الأعداد الطبيعية في الخوارزم من 0 الى +∞. بما أنّ الأعداد الصحيحة تحتوي الأعداد الطبيعية، ففي العادة، نستعمل الأعداد الصحيحة للتعبير عنها. كما يمكن إضافة unsigned أمام النوع في C للتعبير عن الأعداد الطبيعية فقط.

النوع	الحجم بالبَايْتْ octet	الحجم بالبِتْ bit	المجال
unsigned char	1	8	0 ، $+2^8-1$
unsigned short	2	16	0 ، $+2^{16}-1$
unsigned long	4	32	0 ، $+2^{32}-1$
unsigned int	4	32	0 ، $+2^{32}-1$

- الأعداد الحقيقية

النوع	الحجم بالبَايْتْ octet	عدد الأرقام بعد الفاصلة	المجال
float	4	6	
double	8	8	
long double	10	8	

- النوع المنطقيّ boolean: لا يوجد نوع منطقي في لغة C وإنّما يتمّ استعمال int. حيث vrai هي العدد 1 وfaux هي 0. ويتم ترجمة أيّ عدد يختلف عن 0 على أنّه vrai.
- النوع رموز (حروف) هو char.
- النوع سلسلة الرموز: للتعبير عن سلسلة الرموز في C، نستعمل الجداول (الفصل 5) char[] أو المؤشرات (السداسي الثاني) char*.

9.2.2. ملاحظات

- النوع int هو النوع النمطي (generic) للأعداد الصحيحة.
- النوع char يُستعمل للأعداد الصحيحة والرموز. حيث أنّ كلّ رمز يقابله عدد.
- في C++ يوجد النوع bool للنوع المنطقيّ وstring لسلسلة الرموز.
- في هذا الدرس نستعمل char للرموز، int للأعداد الصحيحة والمنطقية وfloat للأعداد الحقيقية.

9.2.3. التحويل بين الانواع *La conversion de type*

- ضمنيّ implicite: ويتمّ من طرف المجمّع أوتوماتيكيّا. ويكون من النوع الصغير إلى النوع الكبير دون ضياع المعلومة. مثل: من char إلى int، أو من int إلى float، أو من float إلى double. فعند تحويل 5 إلى float تصبح 5.0.
- الصريح (cast) explicite: عندما يؤدّي التحويل إلى احتمال ضياع المعلومة، يجب التصريح بأنّ المبرمج هو من يقوم بالعمليّة، وكلّ ما عليه فعله، هو تحديد نوع الوجهة بين قوسين أمام العبارة لتحويلها. مثلا : (int)3.1416 لتصبح 3.

9.2.4. تعريف أنواع جديدة

لإنشاء نوع جديد، أو تغيير اسم نوع معيّن، نستعمل الكلمة المحجوزة **typedef**. وتأخذ الصيغة التالية:

```
typedef اسم_النوع_القديم  الاسم_الجديد;
```

مثال : للتصريح بنوع جديد اسمه Banane والذي أصله int، نستعمل:

```
typedef int Banane;
```

9.3. التصريح بالمتغيّرات والثوابت

9.3.1. التصريح بالمتغيّرات

الصيغة

```
Type Identificateur;
```
اسم_المتغير النوع ;

مثال

```
int age;
char sexe;
float x, y, z;
Banane b;
```

9.3.2. التصريح بالثوابت

الصيغة

```
const Type Identificateur=valeur;
Type const Identificateur=valeur;
```
const النوع اسم_الثابت = قيمة ;

مثال

```
int const N = 10;
const float Pl = 3.1415926;
const char[] DEP  = "قسم الاعلام الالي";
```

ملاحظة: يمكن استعمال الماكرو للتصريح بالثوابت. مثل:

```
#define DEP  "قسم الاعلام الالي"
```

9.4. الإسناد

نستعمل = بدل ← وتُقرأ تأخذ وليس تساوي.

الصيغة

```
Identificateur = expression;
```

مثال

a=5;	a تأخذ 5
b=a*2;	b تأخذ 10
a=0;	a تأخذ 0
b=b-1;	b تأخذ 9
c='b';	c تأخذ حرف b
d=b>a;	d تأخذ 1
s="name";	s تأخذ كلمة name

التصريح مع الإسناد في نفس الوقت:

```
float x, y=3, z;//y est initialisé avec 3
```

في حالة ورود الإسناد عدّة مرّات، فإن الأولية تكون على اليمين. مثل:

```
b=3 ;
a=b=5+3 ;
```

b تأخذ 8، ثم a تأخذ قيمة b، أي 8.

اختصارات الإسناد في C.

العبارة	ملاحظة	مثال
v+=exp ; ⇔ v=v+(exp) ;		x=2 ; x*=5+3 ; ⇔ x=x*(5+3) ; ≠ x=x*5+3 ; x تصبح 16
v-=exp ; ⇔ v=v-(exp) ;	الاقواس مهمة	
v*=exp ; ⇔ v=v*(exp) ;		
v/=exp ; ⇔ v=v/(exp) ;		
v%=exp ; ⇔ v=v%(exp) ;		
v++ ; ⇒ v=v+1 ;	في حالة ورودها في عبارة أخرى يتم الحساب بالقيمة الحالية للمتغير وبعد الانتهاء يتم إضافة 1 او إنقاصه للمتغير حسب العملية	x=2 ; y=3+x++ ;⇔ y=3+x ;x=x+1 ; أي ان y تصبح 5 و x 3
v-- ; ⇒ v=v-1 ;		
++v; ⇒ v=v+1 ;	في حالة ورودها في عبارة أخرى يتم إضافة 1 او إنقاصه حسب العملية قبل حساب العبارة والتي تتم بالقيمة الجديدة للمتغير.	x=2 ; y=3+ ++x ;⇔ x=x+1 ; y=3+x ; أي ان y تصبح 6 و x 3
--v ; ⇒ v=v-1 ;		

ملاحظة

- v++ ليست نفسها v+1، وإنّما v=v+1.
- v++، ++v، v=v+1، v+=1 كلّها متكافئة، إذا أتت كتعليمة مستقلة.
- يكون الفرق بين v++، ++v حين ورودها في عبارة أخرى، هل الإضافة قبلية ام بعدية.

التعليمة الفارغة (instruction vide): هي تعليمة لا تفعل شيئا، وهي التعليمة فاصلة منقوطة « ; » . وتلحق بها تعليمات مثل i+1، حيث يتم حساب النتيجة وتجاهلها، دون أيّ تغيير في حالة البرنامج.

9.5. الإدخال والإخراج

9.5.1. printf (print formatted)

تُستخدم الدالّة printf، المُعرّفة في المكتبة stdio.h، لكتابة البيانات المنسّقة على وحدة الإخراج القياسيّة ، افتراضيًا الشاشة.

الصيغة

```
printf (format, expression_1,… , expression_n);
printf (نص منسق, عبارة);
```

- `format` عبارة عن نصّ إو سلسلة رموز، يتمّ إظهارها على الشاشة كما هي. إلّا الرمز % الذي يمثِّل تنسيق `expression`، و\ الذي يمثل رمز الهروب.
- `expression` هي عبارة، يتمّ حسابها للحصول على قيمة واحدة. ليتمّ إظهارها على الشاشة بالتنسيق الموجود في `format`.

يأخذ التنسيق **format** الشكل التالي:

```
% [flags] [width] [.prec] type_char
```

حيث يبدأ دوما بـ %

- flags
  - \- : جعل المحاذاة تكون على اليسار. أي الإزاحة تكون نحو اليسار.
  - \+ : اظهار إشارة العدد.
  - فراغ : اظهار فراغ بدل + في حالة العدد موجب.
- width يمثّل أقل عدد خانات على الشاشة لإخراج قيمة معيّنة، حيث إذا كان هذا العدد أكبر من الحجم المطلوب، يتم ملأ الفارق بفراغات او أصفار، حسب هذا العدد إذا كان يبدأ بـ 0 أم لا.
- prec. عدد الأرقام بعد الفاصلة في حالة float.
- type_char حرف يُمثّل نوع القيمة المراد إخراجها. يوضّح الجدول التالي التنسيقات الأكثر استخدامًا.

التنسيق	الاستعمال
%d	لإدخال او اخراج عدد في النظام العشري (10) **d**écimal
%o	لإدخال او اخراج عدد في النظام الثماني (8) **o**ctal
%x	لإدخال او اخراج عدد في النظام السادس عشر (16) he**x**adécimal
%u	لإدخال او اخراج عدد طبيعي **u**nsigned
%i	لإدخال او اخراج عدد صحيح **int** مثل %d
%f	لإدخال او اخراج عدد حقيقي **f**loat
%c	لإدخال او اخراج رمز **c**hat
%s	لإدخال او اخراج سلسلة **s**tring (char*)
%e	لإدخال او اخراج عدد حقيقي بالكتابة العلمية (مثلا 3.5e-7) اي 3.5 * 10^-7

- بعض الرموز خاصة، لذلك علينا استخدام ما يسمّى بتقنية الهروب لاستخدامها. في C ، حرف الهروب (escape) هو الرمز (\)، الخط المائل العكسي backslash، ونستخدمه لإضافة سطر جديد ’\n‘ والجدولة أي فراغ كبير ’\t‘. ولطباعة " نستعمل "\ ولطباعة \ نستخدم \\، ولطباعة % نستخدم %%.

مثال:

تظهر السلام عليكم	`printf("السلام عليكم")`
a=(13)10 a=(15)8 a=(D)16	`int a=13;` `printf("a=(%d)10\ta=(%o)8\ta=(%X)16\n",a ,a ,a);`
a=66 a=0.000000 a=B a=b لان ترميز 66 ليس بالضرورة 66 في float و66 إذا رأيناها على أنّها حرف فهي تمثّل الحرف B بينما 66+32=98 تمثّل ترميز الحرف b صغيرة.	`a=66;` `printf("a=%i\ta=%f\ta=%c\ta=%c\n",a ,a ,a ,a+32);`
3.141593 3.1416 003.14	`float pi=3.1415926;` `printf("%f\t%.4f\t%06.2f" ,pi ,pi ,pi);`

(scan formatted) scanf .9.5.2

تُستخدم الدالّة scanf، المُعرّفة في المكتبة stdio.h، لقراءة البيانات المنسّقة على وحدة الإدخال القياسيّة ، افتراضيًا لوحة المفاتيح. حيث تقوم الدالّة بنسخ القيمة التي أدخلها المستعمل إلى عنوان المتغير (المكان في الذاكرة). لذلك نستعمل & قبل اسم المتغيّر.

الصيغة

```
scanf(format, &variable_1, … , &variable_n);
```

`format` سلسلة رموز، تمثّل التنسيق الخاصّ بالقراءة.

`variable` يمثّل اسم المتغيّر. ويكون مسبوقا بـ &، إلّا إذا كان من نوع سلسلة (نوع مؤشر).

يأخذ التنسيق **format** الشكل التالي:

```
% [width] type_char
```

- width: عدد يتحكّم في الحد الأقصى لعدد الأحرف المراد قراءتها من حقل الإدخال الحالي.
- type_char: حرف يُمثّل نوع القيمة المراد إدخالها. نفس الرموز في الجدول الخاص بـ printf.

مثال

`scanf("%s", nom);`	لا يتمّ استعمال & لقراءة سلسلة.
`scanf("%d%f",&a, &b);`	يدخل عددا ثمّ يضغط على espace ثمّ يدخل العدد الثاني ثم يضغط على مفتاح entré.

ملاحظة: من الأفضل ادخال قيمة واحدة بكل تعليمة scanf.

مشكلة scanf مع الرموز والسلاسل.

عند قراءة حرف باستعمال scanf("%c"…)، فإنّ المستعمل يدخل الحرف الأول، ثمّ يضغط على زر الادخال enter والذي بدوره ينتج الرمز '\n'، الذي لا يُحذف من الذاكرة، وعندما يصادف البرنامج التعليمة scanf("%c"…) للمرة الثانية، فإنّه لا ينتظر ما سيدخله المستعمل، وإنّما يسند الرمز \n للمتغيّر الثاني. ولتفادي هذا المشكل نستعمل في scanf الثانية فراغ ' ' بعد % هكذا: scanf(" %c"…). أو نستعمل الدالّة getch المعرّفة في المكتبة string.h.

مثال سنحاول ادخال 3 حروف a، b، c إلى المتغيّرات c1، c2 و c3. نستخدم:

```
scanf("%c", &c1) ;
scanf("%c", &c2) ;
scanf("%c", &c3) ;
```

يضغط المستخدم على a، ثمّ enter. فيقوم البرنامج بإسناد a إلى c1 و \n الى c2، وينتظر المستعمل لإدخال الحرف الثاني الذي سيسنده لـ c3. ولتجنب هذا المشكل نستخدم:

```
scanf("%c", &c1) ;
scanf(" %c", &c2) ;
scanf(" %c", &c3) ;
```

أمّا مشكلة scanf مع السلاسل تظهر عندما نحاول ادخال سلسلة تحوي فراغات في متغير واحد. مثلا:

```
scanf("%s%s", v1,v2) ;
```

عندما ندخل الكلمتين math info، ونضغط على enter، فإنّ البرنامج يسند الكلمة الأولى لـv1 والكلمة الثانية لـv2. ولكن، عندما نريد إدخال الكلمتين (اسم مركب مثلا) في متغيّر واحد نستخدم:

```
scanf("%s", v1) ;
```

وعندما ندخل الكلمتين math info، ونضغط على enter، فإنّ البرنامج يسند الكلمة الأولى لـv1، وتضيع الكلمة الثانية. ولتفادي هذا المشكل، نستعمل الدالّة gets المعرّفة داخل المكتبة string.h.

```
#include <sting.h>
gets(v1);
```

9.6. هيكل البرنامج في C

```
1.  #include <stdio.h>
2.    التصريح بالثوابت والأنواع والمتغيرات
3.  int main()
```

```
4.  {
5.    التصريح بالثوابت والمتغيرات
6.    التعليمات
7.    return 0;
8.  }
```

الشرح:

1. لإدراج المكتبة stdio.h والتي تحتوي على printf، scanf.
2. مكان التصريحات العامّة.
3. main() : يجب أن يحتوي كل برنامج على دالّة بدء تشغيل تسمى main والتي تشير إلى نقطة دخول البرنامج.
4. بداية جسم الدالة main ويقابلها في الخوارزم début
5. مكان التصريحات المحليّة.
6. التعليمات.
7. الارجاع: يجب أن تُرجع الدالّة main عددًا صحيحًا (int). حيث أنّها تُرجع 0 لنظام التشغيل، لتقول أنّ كلّ شيء سار على ما يرام.
8. نهاية الدالة main والبرنامج، ويقابلها في الخوارزم fin.

ملاحظة

- يمكن التصريح إمّا في مكان التصاريح العامّة او المحليّة.
- تنسيق الكتابة في كود الشيفرة والمحاذاة والهوامش والفراغات والرجوع للسطر التي بعد الرموز الخاصّة مثل :; ,+= ([{}]) الخ ليس لها معنى في البرنامج. إذ يمكن كتابة جل البرنامج في سطر واحد، والذي يفصل بين التعليمات هي ; وليس الرجوع الى السطر.
- يجب احترام تنسيق الكتابة والمحاذاة والهوامش والفراغات والرجوع للسطر من اجل مقروئيّة البرنامج.

مثال يوضح عمليّة الترجمة من الخوارزميّة الى لغة C

الخوارزمية	C	ملاحظة
Algorithme surf_cercle		اسم الخوارزمية يصبح اسم الملف surf_cercle.c
Const P=3.14	**Const** float P=3.14;	يمكن استخدام #define P 3.14
Var r, s :entier	int r, s;	لا توجد كلمة var في C
//r le rayon et s surface	//r le rayon et s surface	ليست جزءا من البرنامج وانما للشرح فقط.
Début	int main() {	يمكن التصريح بالمتغيرات بعد {
Ecrire("ادخل نصف القطر")	printf("ادخل نصف القطر\n");	\n للرجوع الى السطر
Lire(r)	scanf("%d", &r);	لا تنس & قبل المتغير
s←p*r*r	s=p*r*r;	كل تعليمة تنتهي بـ ;
Ecrire("مساحة الدائرة هي:" , s)	printf("مساحة الدائرة هي:%d" , s);	يجب إعطاء الصيغة
Fin	}	

مثال 2

اكتب البرنامج الذي يحسب معدّل مادّة ASD1.

```
#include <stdio.h>
int main()
{
  float cont, td, tp, moy ;
  printf("ادخل نقطة الامتحان\n") ;
  scanf("%f", &cont) ;
```

```
  printf("ادخل نقطة الاعمال الموجهة\n") ;
  scanf("%f", &td) ;
  printf("ادخل نقطة الاعمال التطبيقية\n") ;
  scanf("%f", &tp) ;
  moy = (cont * 3 + td + tp) / 5 ;
  printf("المعدل هو:%.2f" , moy) ;
  return 0;
}
```