Chapter 5 : Algorithmic

introduction

All program is a (very long) sequence of binary digits (either 0 or 1).

The first programs were written using this language (machine language).

These programs are therefore difficult to write and read as well as to maintain.

This is why another language intended for human beings was defined (it is assembly language) and the program responsible for translating into machine language is called “assembler”.

The goal of assembly language is to move away from machine details.

Unfortunately, the objective was not achieved.

It is for this reason that other languages closer to the human mind called “high-level languages” have been proposed (such as Pascal, C, Fortran, etc.).

The program responsible for translating into machine language is called "compiler" and the translation operation is called "compilation".

The output of a compiler is either the equivalent program in machine language or any existing errors in the source program text.

We can say that the compiler defines for the programmer as a virtual machine which understands the high level language associated with this compiler.

Notice : knowing a programming language is not enough to write programs.

Generally, we must go through a step called an algorithmic step to determine all the operations necessary to solve the given problem.

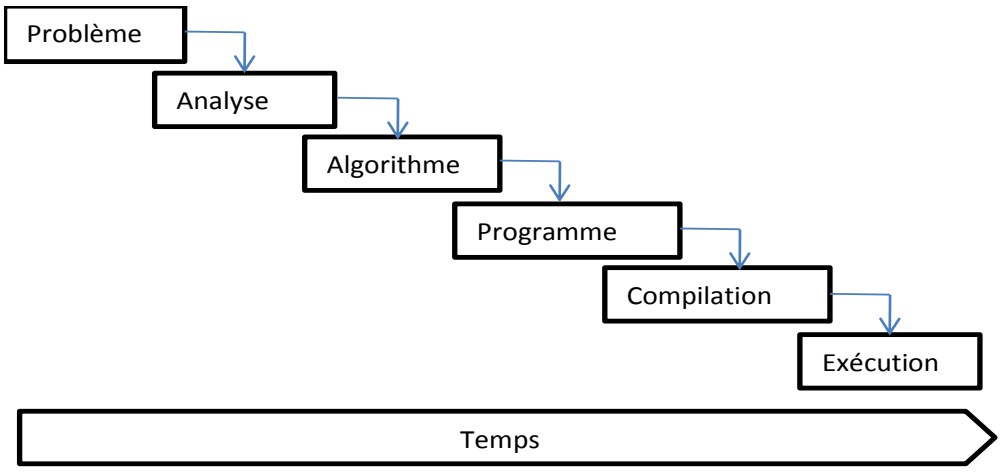
These operations must be coded using a machine-independent language; this language is called **algorithmic language** and the generated text is called **algorithm** .

An algorithm is divided into 2 parts:

- The header.
- The body.

Program development cycle:

The development cycle of a “computer program (or application)” can be summarized as follows :



البرامج عبارة عن تسلسل (طويل جدًا) من الأرقام الثنائية إما 0 أو 1 .

تمت كتابة البرامج الأولى باستخدام هذه اللغة (لغة الآلة).مما يصعب مهمة كتابة هذه البرامج أو قراءتها وكذلك صيانتها.

ولهذا تم تعريف لغة أخرى مخصصة للإنسان (وهي لغة التجميع) ونسمي البرنامج المسؤول عن الترجمة إلى لغة الآلة "المجمع. "

الهدف من لغة التجميع هو الابتعاد عن تفاصيل الآلة.

ولسوء الحظ، بقيت البرمجة بلغة التجميع صعبة و أقرب الى لغة الالة ، لم يتحقق الهدف.

ولهذا السبب تم اقتراح لغات أخرى أقرب إلى لغة العقل البشري تسمى "اللغات عالية المستوى" (مثل باسكال، وسي، وفورتران، وغيرها. يسمى البرنامج المسؤول عن الترجمة إلى لغة الآلة "المترجم" وتسمى عملية الترجمة "التجميع."

يكون مخرجات المترجم إما البرنامج المكافئ في لغة الآلة أو أي أخطاء موجودة في نص البرنامج المصدر.

يمكننا القول أن المترجم يُعرّف المبرمج بأنه جهاز افتراضي يفهم اللغة عالية المستوى المرتبطة بهذا المترجم.

ملاحظة : معرفة لغة البرمجة ليست كافية لكتابة البرامج .

يجب أن نمر بمرحلة إيجاد الفكرة أو التصميم المناسب لحل مشكل آليا هذه الخطوة تسمى الخوارزمية

وهي لغة مستقلة عن الآلة ، تسمى هذه اللغة لغة خوارزمية ويسمى النص الناتج خوارزمية.

تنقسم الخوارزمية إلى جزئين:

- رأس.
- الجسم.

An algorithm is a language designed to solve a problem in an automated manner intended for humans, not computers.

The goal of this language is to allow programmers to write the solution principle for a specific problem in the form of a sequence of instructions (commands) that will be executed by the machine after translating it into a programming language.

الخوارزمية هي لغة لتصميم حل مشكل بطريقة آلية موجهة للبشر وليست لأجهزة الكمبيوتر.

الهدف من هذه اللغة هو السماح للمبرمجين بكتابة مبدأ الحل لمشكلة معينة في شكل تسلسل من التعليمات (الأوامر) التي سيتم تنفيذها بواسطة الآلة بعد ترجمتها الى لغة برمجة.

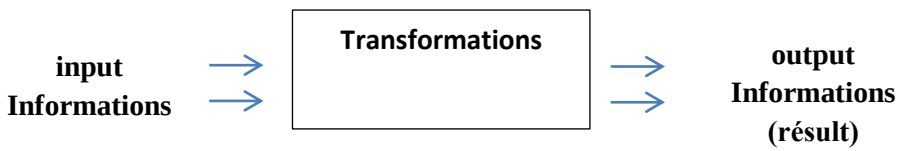
1. **instruction:** in algorithmic language and almost all programming languages there are only 3 types of instructions:

- **input instruction (read)**
- **assignment instruction (assignment)**
- **output instruction (write)**

1. التعليمات: في اللغة الخوارزمية وفي جميع لغات البرمجة تقريبًا يوجد 3 أنواع فقط من الأوامر :

- **تعليمات الإدخال (read)**
- **تعليمات الإسناد (←)**
- **تعليمات الإخراج (write)**

We can schematize any automatic processing of information as follows:



a) input instruction (Input)

In algorithmic language, any information input operation for processing is simply modeled by: **Read (information);**

More precisely, the input instruction is given syntactically by

Read (parameter list)

أ) تعليمات الإدخال (الإدخال)

في اللغة الخوارزمية، يتم تصميم أي عملية إدخال معلومات للمعالجة ببساطة من خلال: قراءة (معلومات)؛

بتعبير أدق، يتم إعطاء تعليمات الإدخال بشكل نحوي بواسطة الأمر Read (parameter list)

Example :

Read (x); parameters 1: x

Read (x,y,z); we have 3 parameters: x,y and z

RQ : type of each parameter is necessarily a **variable**.

b) assignment statement

Role: assign a value to a variable

Syntax: variable ⟵ "Expression" ;

Example :

x ⟵ 5 ;

x ⟵ 5x+1 ;

Semantics:

Evaluate the expression to the right of the assignment operator; then put the result of the evaluation in the variable located to the left of the assignment operator.

حساب العبارة الرياضية الموجودة على يمين أمر الإسناد ثم ضع النتيجة في المتغير الموجود على يسار أمر الاسناد

Exercise: Let there be 2 variables x and y which contain the values x_0 and y_0 respectively

Write a sequence of instructions that allows you to permute the 2 values.

تمرين: ليكن هناك متغيرين x و y يحتويان على القيمتين x0 و y0 على التوالي

اكتب سلسلة من التعليمات التي تسمح لك بتبديل القيمتين.

Answer

Let A be a variable of type x or y:

A ⟵ x ;

x ⟵ y;

y ⟵ A ;

c) output instruction (Write)

For reasons of simplicity, the Output Operations are written algorithmically as: **Write** ("parameter lists");

Example: Write ("hello");
Write ("the content of x is ", x);

RQ :

Any variable used must be declared before the body of the algorithm.

2) Declaration:

Really, we must declare in the declaration part, any object to be used in the body of the algorithm.

a) Variable declaration

Variables are declared using the **Var keyword**

Any variable must be declared with its type

Example: Var x, y: integer;

RQ: the type of a variable allows you to know the possible values of this variable as well as the operations applicable to this variable.

Language Data type → Predefined types (simple type) known by the

→ Compound type (programmer defined)

Simple types: we have the following types:
int (int in c), real (float), character (char) and boolean (int 1 or 0)

2) التصريحات :

يجب أن نعلن في جزء التصريحات عن أي كائن سيتم استخدامه في نص الخوارزمية.

أ) التصريح بمتغير

يتم الإعلان عن المتغيرات باستخدام الكلمة المفتاحية **Var**

يجب الإعلان عن أي متغير مع نوعه

مثال: **var x, y: integer**;

نوع المتغير يتيح لك معرفة القيم المحتملة لهذا المتغير وكذلك العمليات المطبقة على هذا المتغير.

الأنواع البسيطة: لدينا الأنواع التالية: الصحيح **(integer)** ، الحقيقي (**float**)، الحرف (**char**)، والمنطقي (1 أو 0)

Algorithmic Constructions:

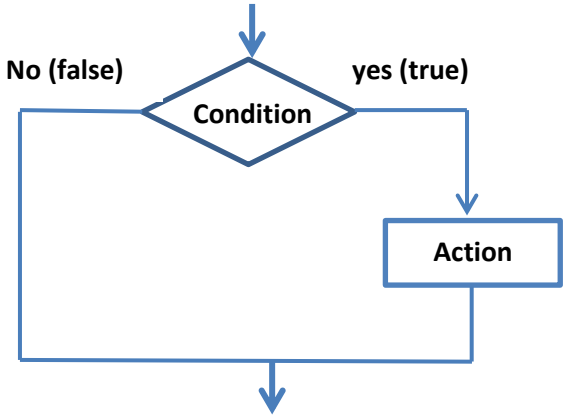
1) Conditional Constructions:

This construct has the following syntax:

Set of instructions 1 / start of program */*
If** (condition) **then
Instruction Set 2
end if
Set of instructions 3 / continuation of the program */*

if the condition is true, then we execute the *Instruction Set 2*

This means that instruction set 2 will be executed only if the condition is met.



Example :

Write an algorithm that allows reading 2 integer values and displays the max value?

```
Algorithm Max
  Var x,y: integer;
Begin
    Read x ,y ;
  If (x ≥ y) then
      write (x);
  End if
  If (x < y) then
      write (y);
  End if
END
```

2) Alternative Conditional Constructions:

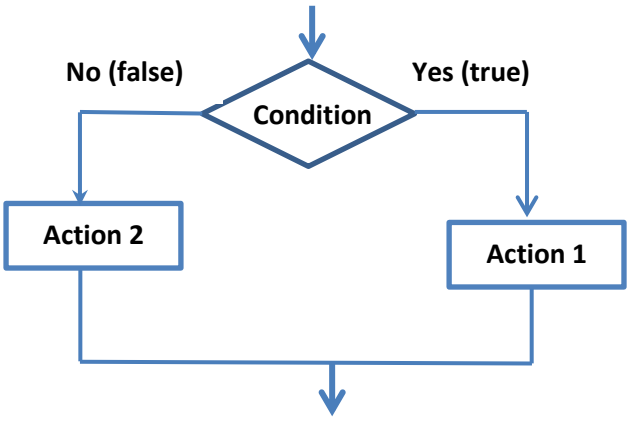
Syntax:

```
If (condition) then
    Action 1;
 Else
    Action 2;
End if
```

```
Algorithm Max
  Var x,y : integer;
Begin
    Read (x ,y) ;
  If (x ≥ y) then
      write (x );
  else
      write (y );
  End if
END
```

If (condition) is true then we execute **action 1** otherwise (if condition is false) we execute **action 2.**

Organizational chart:



Example

write an algorithm that allows you to solve a first degree equation (ax+b =0)

أكتب خوارزمية تسمح لك بحل معادلة من الدرجة الأولى **(a**x+**b** =0)

```
Algorithm Eq 1
 Var a, b: real;
Begin
    Read (a ,b) ;
  If (a = 0) then
    If (b ≠ 0) then
      write ( " impossible") ;
    else
     write (“ infinite solution”);
    End if
  else
     x ← -b/a;
     to write (x );
  End if
END
```

3) Selection constructs (choice):

In the case where the number of tests on the value of a variable is quite high (usually more than three) and in order to avoid writing several if-then else blocks, it is possible to use the **select-case** statement. Here is its general form:

في الحالة التي يكون فيها عدد الاختبارات على قيمة المتغير مرتفعًا جدًا (عادة أكثر من ثلاثة) ومن أجل تجنب كتابة عدة كتل **if-then else** ، من الممكن استخدام أمر اختيار الحالة ، وهنا شكلها العام:

Select case (**variable_name**)
case (exp1) :
// block of instructions to execute if the variable has the value of exp 1
case (exp 2) :
// block of instructions to execute if the variable has the value of exp 2
....
case (exp n) :
// block of instructions to execute if the variable has the value of exp n
case default
// block of instructions to execute if the variable has a value different from values exp1,..., exp n .
end select