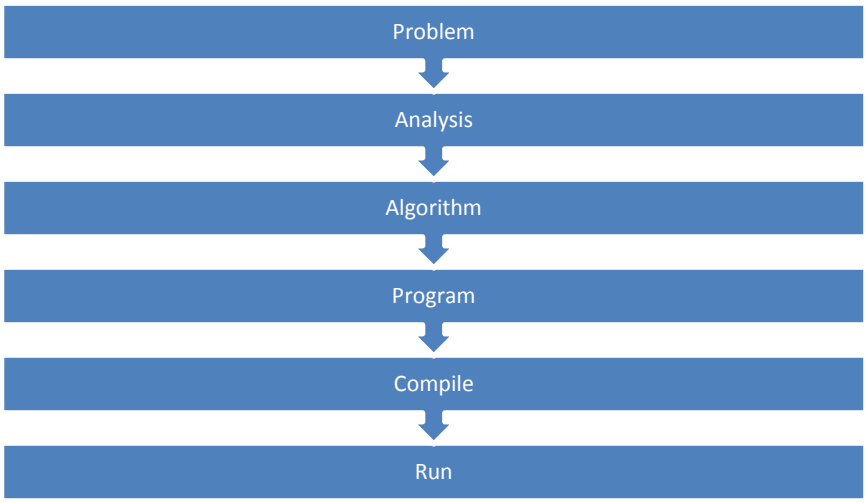
Programming: Introduction to programming and developing programs

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.

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.

ومع ذلك ، 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.

Programming consists of replacing the algorithms designed for solving automatic problems, by equivalent codes written in languages dedicated to programming, so that these algorithms can be interpreted by compile and executed automatically by machines.

Fortran is a language advanced programming software accompanied by a compiler called **G95**.

تتمثل البرمجة من استبدال أوامر الخوارزميات المصممة لحل مشكل ما ، برموز مكافئة مكتوبة بلغات البرمجة المناسبة ، بحيث يمكن ترجمتها عن طريق المترجم الى لغة الآلة وتنفيذها تلقائيا .

فمثلا الفورتران هو لغة عالية المستوى عندها برنامج مترجم الى لغة الالة يسمى **G95**

In this program, the symbols: (=) and (::) as well as the keywords: integer, program, end, and **implicit none** are words specific to the Fortran language. word **nb** designates a variable representing any memory space, to which is assigned a name

```
program addition
 implicit none
 integer :: nb
  nb = 2 + 4
5 end program addition
```

في هذا المثال الرمزان: (=) و (::) وكذلك الكلمات المفتاحية: **integer**،

program و **end program** هي كلمات خاصة بلغة فورتران.

تشير كلمة **nb** إلى متغير يمثل مساحة عمل في الذاكرة. تم تعيين اسم له،

دون أن يضطر المبرمج إلى تحديد مكان وجوده فعليًا

Fortran program struct

program noun
implicit none

- **declared part**
- **instruction part**

end program noun

save (name.f95) → **Compile** → **Run (name.exe)**

Programming Constructions:

1) Conditional Constructions:

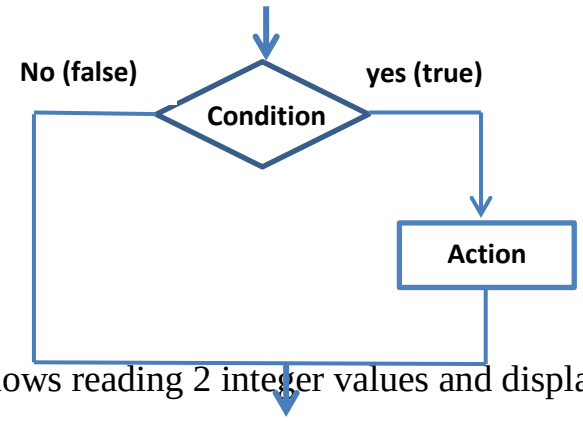
In fortran his construct has the following syntax:

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 Program 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

```
program Max
implicit none
integer :: x,y
print * ,'Enter two integer numbers:'
read *, x, y
 if(x>=y)then
  print *,' max value = ', x
 endif
 if(y>x)then
  print *,' max value = ', y
 endif
end program
```

2) Alternative Conditional Constructions:

Syntax:

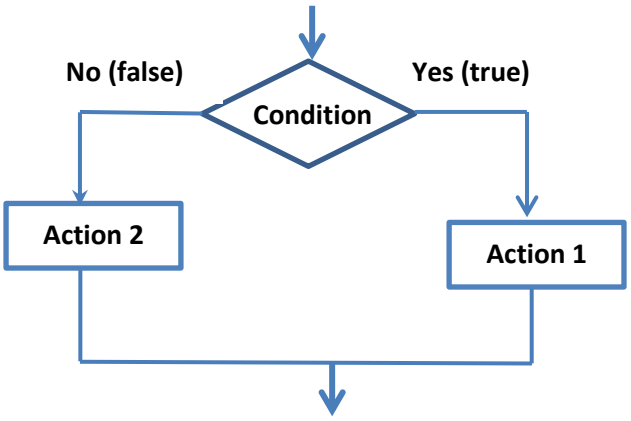
If** (condition) **then
Action 1;
Else
Action 2;
End if

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

```
program Max
implicit none
integer :: x,y
print * ,'Enter two integer numbers:'
read *, x, y
 if(x>=y)then
  print *,' max value = ', x
 else
  print *,' max value = ', y
 endif
end program
```

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

Organizational chart:



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 (**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

Note that **expi** can be one of the following 3 cases:
1. value: indicating the exact value of the variable tested;
2. value 1**,** value 2 **,** ...**,** value n: indicating a list of possible values for the variable tested;
3. [value 1**:** value n]: indicating a range of possible values for the variable tested;
if value **n** is not indicated, this means that the value of the variable is greater or equal to value1.

قيمة **exp i** يمكن أن يكون إحدى الحالات الثلاث التالية:
1. القيمة: تشير إلى القيمة الدقيقة للمتغير الذي تم اختباره؛
2. value 1, value 2, ... ,value n : تشير إلى قائمة القيم المحتملة للمتغير تم اختباره؛
3. [value1:valuen]: يشير إلى نطاق القيم المحتملة للمتغير الذي تم اختباره؛ إذا لم يتم الإشارة إلى value n، فهذا يعني أن قيمة المتغير أكبر أو يساوي القيمة 1.

Example : In this example, we want to program the following function f:

$$f(x)=\begin{cases}1, & x=1\\ 2, & x\in\{1,2\}\\ e^{\sin(x)} & x\geq 3\end{cases}$$

```
program Fx
implicit none
integer :: x
print*,"Enter an integer number: "
read*, x
select case (x)
case (0)
   print*,1
case (1,2)
    print*,2
case (3:)
    print*, exp(sin(real(x)))
case default
    print*, "unknown value"
end select
end program Fx
```

في هذا المثال، يتم ادخال قيمة x **ثم يتم تعريف حالة لكل قيمة (0) case تعني اذا كان** x=0 **اظهر النتيجة 1**
case (1,2) تعني اذا كان x=1 **أو** x=2 **اظهر النتيجة** 2 … **وإلا عرض الحالة الافتراضية**
case default **رسالة خطأ "** unknown value **" إذا كانت القيمة المدخلة ليست جزءًا من قيم التعريف للدالة.**