Ministry of Higher Education and Scientific Research
Djilali BOUNAAMA University - Khemis Miliana(UDBKM)
Faculty of Science and Technology
Department of Mathematics and Computer Science

Chapter 1

Introduction :
2. Introduction to Algorithms

MI-L1-UEF121 : Algorithms and Data Structures I

Noureddine AZZOUZA

n.azzouza@univ-dbkm.dz

Course Topics

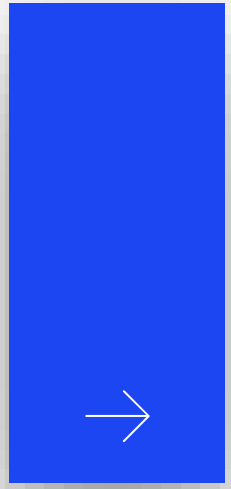
1. Algorithms
    - 1.1 Definitions
    - 1.2 Examples
    - 1.3 History
2. Programming
    - 2.1 Definitions
    - 2.2 Languages & tools
3. Solving a problem in computer science

Goal

- ✓ Getting the “**machine**” to do work for us.

Problem

- ✓ explain to the "**machine**" how it should do it.
  - ➢ *how to tell it?*
  - ➢ *How to teach it?*
  - ➢ *How do we make sure it does this job as well as we do?*
  - ➢ *Even, Better than us?*

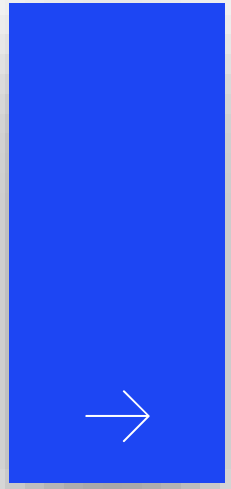


Objectives

- ✓ ***solve*** problems “like” a **machine**
- ✓ know how to ***explain*** your reasoning
- ✓ know how to ***formalize*** your reasoning
- ✓ ***design*** (and write) **algorithms**

Concepts covered

- ✓ **Basic Concepts**
  - ➢ "***basic***" algorithms for elementary problems
- ✓ **Learning a language**
  - ➢ Algorithmic ***formalism***,
  - ➢ ***programming*** languages: C, Pascal
- ✓ **Data Structures**
  - ➢ from the ***simplest*** to the most ***complex***
- ✓ **Complex problem solving**
  - ➢ ***clever*** and ***efficient*** algorithms



Algorithms

Definition

- ✓ An **ALGORITHM** is a ***sequence of instructions*** which, once ***executed correctly***, leads to a ***given result*** (desired).

- ✓ **Examples : Algorithms**

  - ➢ show a way to a lost tourist;
  - ➢ write a cooking recipe;
  - ➢ Dispense drinks automatically;
  - ➢ Play a video on YouTube

Example : Pancakes recipe

Preparing the pancakes

Algorithm

1.Put the flour in a bowl

2.Form a well

3.Add the whole eggs, sugar, oil and butter

4.Mix gently with a whisk, adding the milk.

5.Heat a pan and add a few drops of oil.

6.Cook the pancakes over low heat

The actions

The processor

The person making the recipe

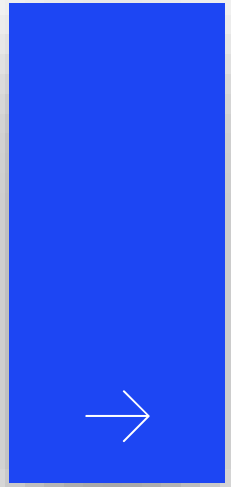
Definition : Processor

- ✓ An **algorithm** is always executed by a ***PROCESSOR***.
- ✓ An **algorithm** must therefore only contain instructions understandable by a ***PROCESSOR.***
- ✓ **Examples : Algorithms (Processor)**
  - ➢ show a way to a lost tourist (***a person***);
  - ➢ write a cooking recipe (***a person***);
  - ➢ Dispense drinks automatically (***a machine - dispenser***);
  - ➢ Play a video on YouTube (***a program***)

Definition : Environment

✓ ***Environment*** :

- It is the set of **objects** or **elements** required to carry out a work described by an algorithm,
- We distinguish:
  - ***input objects***: provided to Algorithm.
  - ***output objects***: produced by Algorithm.
  - ***internal objects***: internal manipulation of Algorithm
- The environment of an algorithm can also be called: the **settings** (parameterization)



Definition : Action

✓ ***Actions :***

- These are the "**sequence of instructions**" or **steps** of Algorithm
- It is an ***event*** of finite duration which ***modifies*** the environment
- Please note that:
  - Changing the order of actions can transform the result.
  - The same action can appear several times in the same algorithm
- A **primitive action** is an action executed (by a processor) without any ***additional information***.

Definition : Algorithm

An **algorithm** is a **sequence** (sequel) of primitive **actions**, which once executed by a well-defined **processor**, will carry out a very **specific job** (requested)

Properties

1. ***General:*** an algorithm must always be designed in such a way as to consider *all eventualities* of a treatment (take into account *all possible cases*).
2. ***Finitude:*** An algorithm *must stop* after a finite time (*finite number* of primitive actions).
3. ***Definition:*** all actions of an algorithm must be *unambiguously defined*
4. ***Repetitiveness:*** generally, an algorithm contains *several iterations*, that is to say actions that are *repeated* several times.
5. ***Efficiency:*** Ideally, an algorithm should be designed in such a way that it runs in *minimal time* and consumes *minimal resources*.
6. ***Independence:*** an algorithm must be *independent* of programming languages and computers

Learn Algorithms

To master Algorithms, three (3) qualities are required :

1. ***be methodical:***

   ✓ Before writing the instructions for an algorithm, you must *analyze the problem* to be solved. You must then *define the inputs* and *outputs* of Algorithm.

2. ***have intuition:***

   ✓ No recipe allows you to know a priori which instructions will achieve the desired result. The *reflexes* of algorithmic reasoning become spontaneous with experience.

3. ***Be rigorous :***

   ✓ Each time you write a series of instructions, you must systematically *put yourself* mentally in the *place of the machine* that will execute them.

History of Algorithms

1. ***18th century BC AD.*** :

   ✓ the **Babylonians** defined exhaustive descriptions of algorithms for calculations concerning trade and taxes;

2. ***3rd century BC AD*** :

   ✓ **Euclide** introduced (in his work The Elements) the famous algorithm which makes it possible to find the greatest common divisor (PGCD) of two numbers;

History of Algorithms

1. ***9th century***:
   - ✓ **Al Khuwarizmi** was the first to formalize the notion of algorithm in his work Algebra and Balancing ;

2. ***12th century***:
   - ✓ **Adelard de Bath** introduced the Latin term algorismus (with reference to the name of Al Khuwarizmi);

Programming

Program

- ✓ A **program** is a ***sequence of instructions*** written in a ***programming language*** **translating** an algorithm
- ✓ Each of its instructions specifies the operation that the computer must execute.

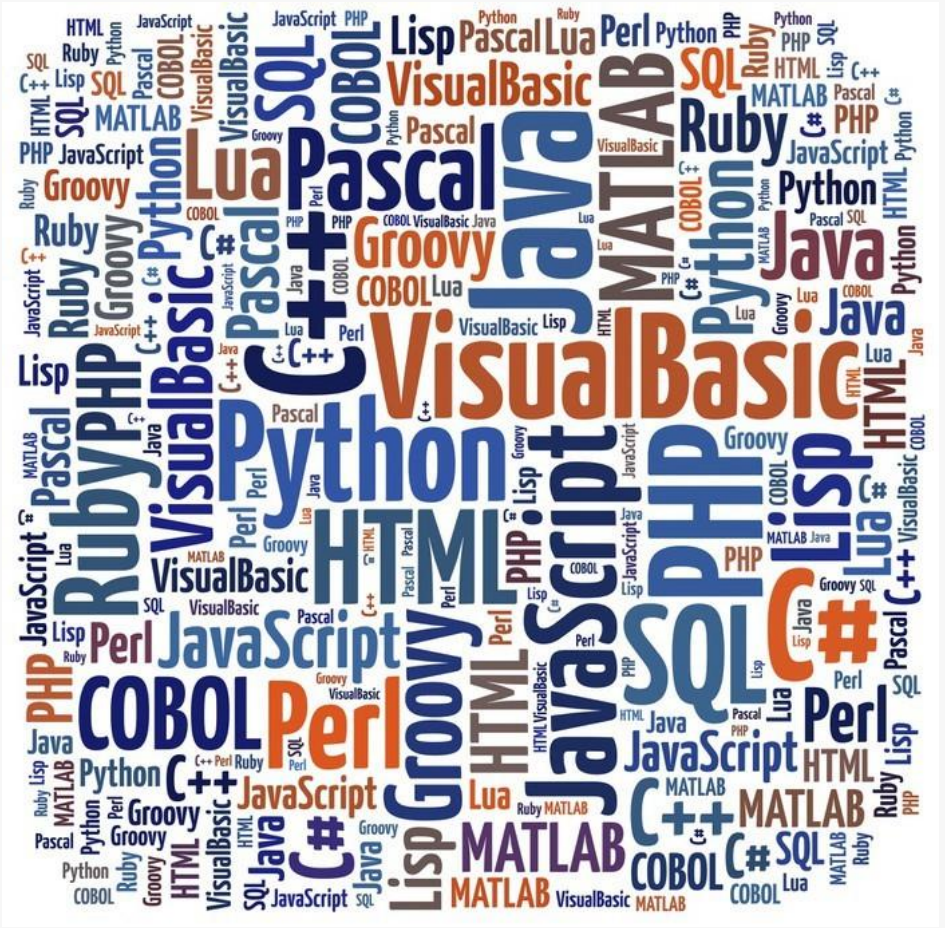
Algorithm vs. Program

Algorithm	Program
No machine required	Machine required
Written according to a formalism or an algorithmic description	Written in a programming language
Takes place	Runs or executed
Logical solution	Physical solution
result of the analysis	translation of an Algorithm

Programming language

- ✓ A **programming language** is a conventional notation intended for ***formulating*** (translating) algorithms and ***producing*** (developing) programs
- ✓ it is an **abstraction** of operations that can be carried out on a computer

- ✓ **Examples :**
  - ✓ Pascal, C, Python, Java, C++, C#, PHP, JavaScript



Programming

Programming language

C

C

```c
#include <stdio.h>

int main()
{
    printf("Hello World");
    return 0;
}
```

C#

C#

```csharp
using System;
namespace HelloWorldApp {

  class Geeks {

    static void Main(string[] args)
    {
        Console.WriteLine("Hello World");
        Console.ReadKey();
    }
  }
}
```

Java

Java

```java
class HelloWorld {

    public static void main(String args[])
    {
        System.out.println("Hello World");
    }
}
```

Python

Python

```python
print("Hello World")

```

Programming Tools

1. ***Editor***:

   ✓ A text or ***source code*** editor is software intended for creating and editing text files (program source files).

   Examples: NotePad++, Sublime Text, Atom, Brackets …

2. ***Compiler***:

   ✓ It is a program that transforms ***source code*** (written in a programming language) into ***object code*** to create a machine-***executable program***.

   ✓ **Examples:**

   ➢ ***C Langage***: GCC, Borland C,

   ➢ ***Pascal Langage***: Turbo Pascal, Free Pascal …

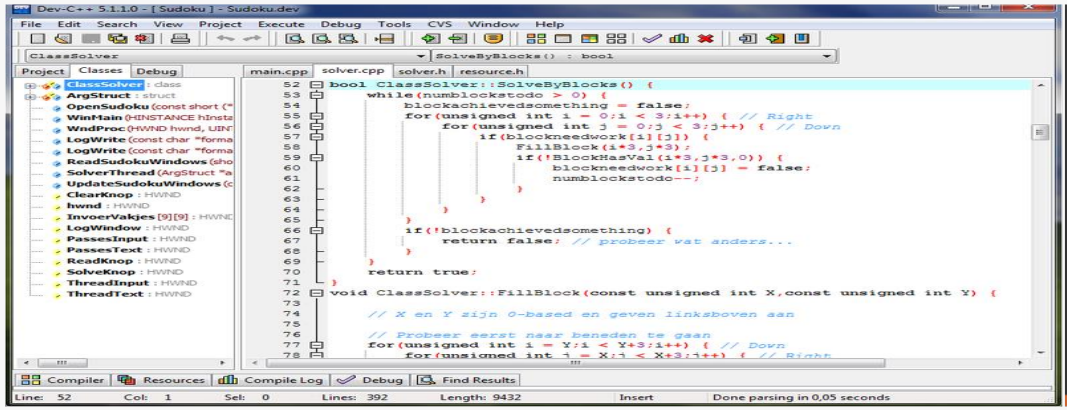
Programming Tools

3. ***IDE***:

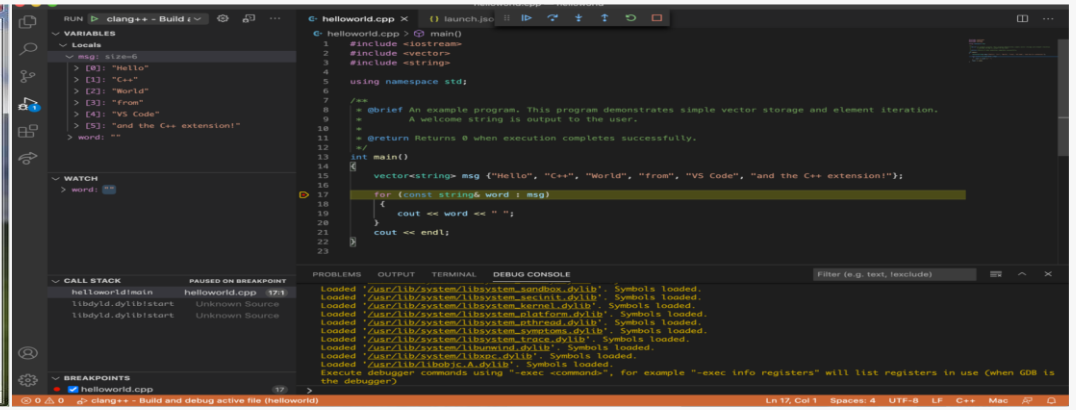
- ✓ The Integrated Development Environment (IDE) brings together a set of tools specific to program development. It can contain :
  - ➢ A text editor
  - ➢ A compiler
  - ➢ A debugger
  - ➢ A GUI creator ….etc
- ✓ **Examples:**
  - ➢ ***C/C++ Langage***: DevC++, Code::Blocks, Visual Studio Code, Eclipse + CDT …
  - ➢ ***Pascal Langage***: Lazarus, Free Pascal …
  - ➢ ***Python*** : PyCharm, Spyder, Visual Studio Code, Jupyter Notebook …

Programming Tools

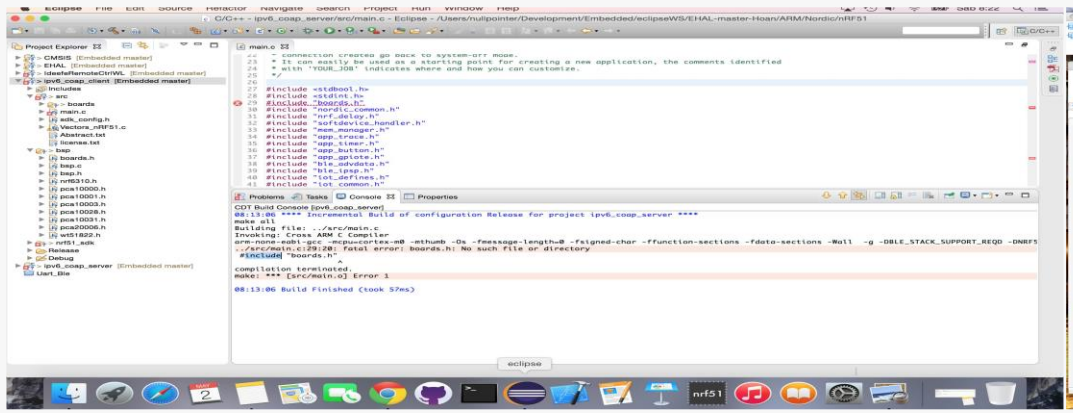
DevC++



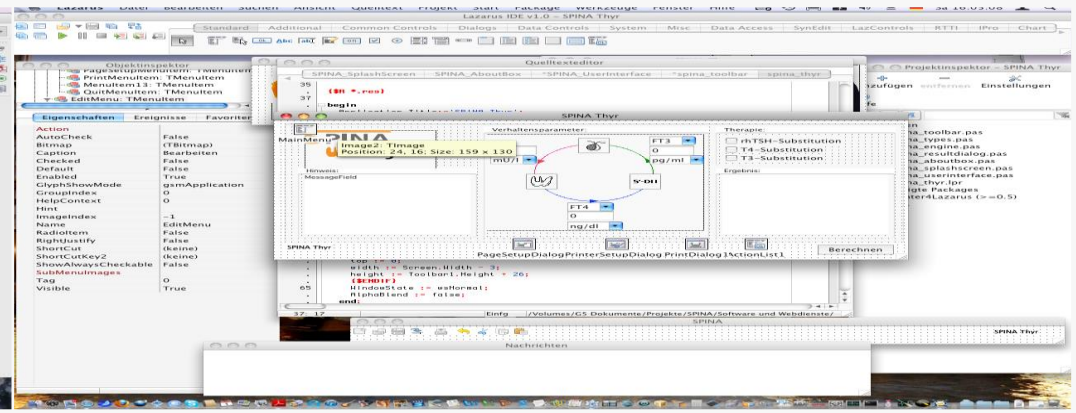
Visual Studio Code



Eclipse

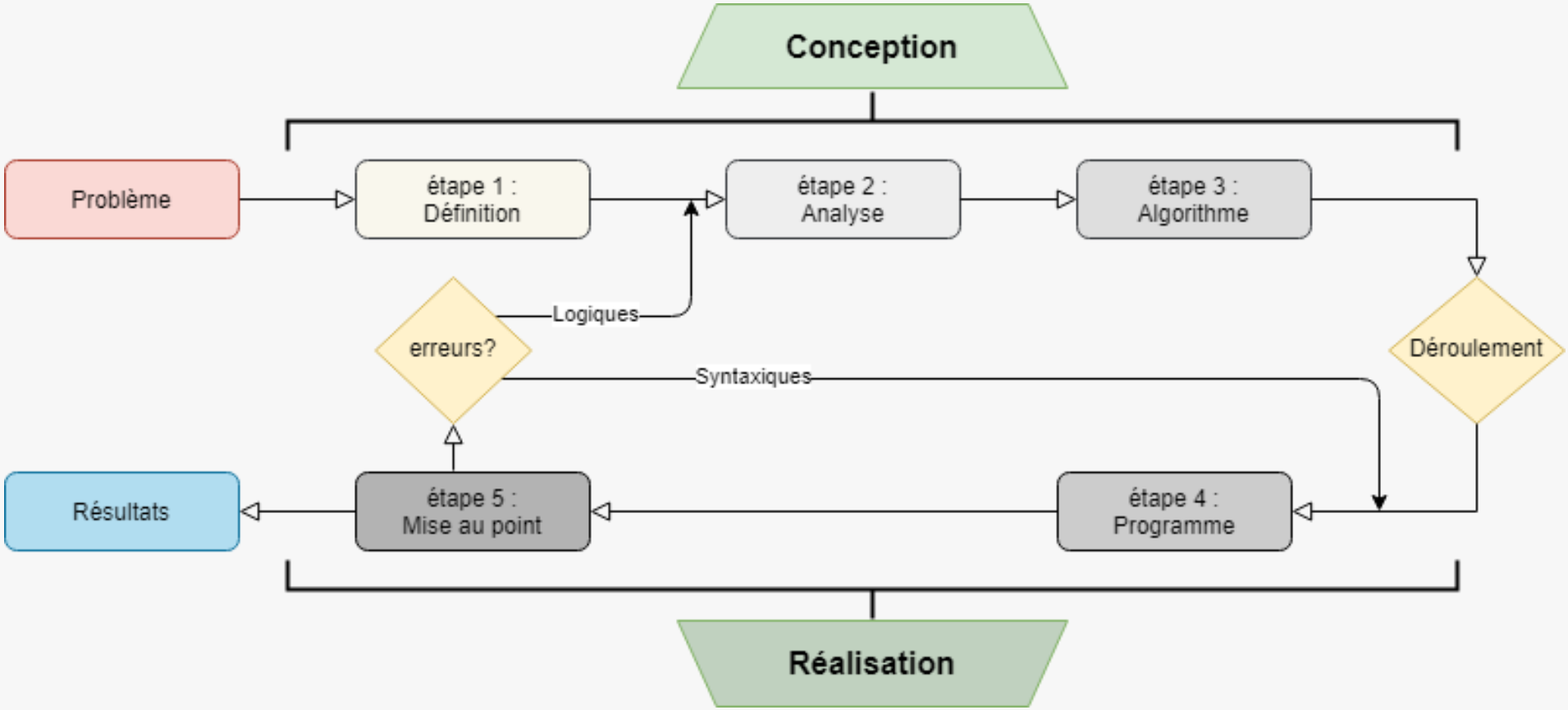


Lazarus



From Problem to Solution

Solving a problem in computer science



Step 1 : Problem definition

- It is about :
  1. ***Determine*** all ***available information***
  2. ***Define*** the shape of the ***desired results***

- In this step, we ask the following questions:
  1. ***What*** is ***given*** as ***input*** (***input*** or ***initial data***)?
  2. ***What outputs*** are ***requested*** (***output*** or ***results***)?

Step 2 : Problem Analysis

- It consists of :
  - ***a.*** ***Find*** the way to get from ***data*** to ***results***.
    - a. Natural language, scientific formula, diagram or drawing, …
  - ***b.*** ***Acquire*** the reflex to ***propose adequate solutions*** to a given problem
    - ***a. Solution by theory:*** mathematical, physical problem, etc.
    - ***b. Solution by synthesis:*** store management, sales management, etc.
    - ***c. Solution by experience:*** storage and search problem, …
- In this step, we ask the following questions :
  1. ***How*** to solve the ***problem*** ? How do we get to the ***results***?
  2. ***What*** is the solution to this ***problem***? What is the form of ***result?***

Step 3 : Writing an Algorithm

- You have to :
  1. ***Write*** a ***clear*** and ***unambiguous*** solution.
  2. ***Represent*** the solution by ***an algorithm*** written in "***Algorithmic Language***" or "***Algorithmic Formalism***"
  3. ***Run*** the Algorithm and check its operation (***general*** case, ***special*** cases)

- In this step, we ask the following questions :
  1. ***How*** to represent the solution in an algorithm?
  2. ***Which control structures*** to use?
  3. ***What variables*** and ***data structures*** to use?

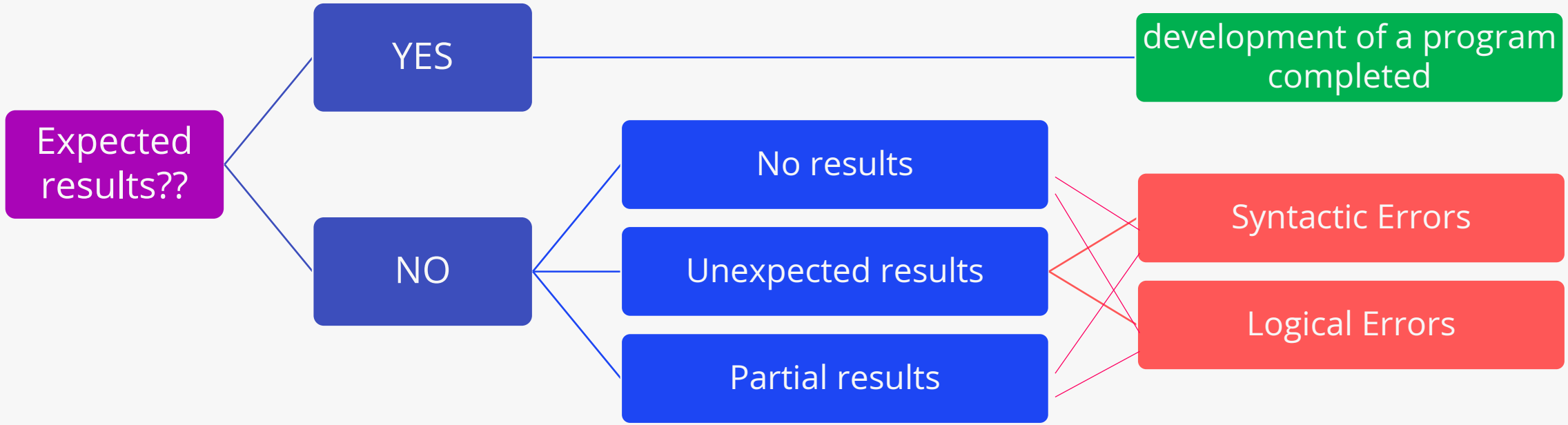
Step 4 : Program Implementation

- To concretely implement algorithm, we must :
  1. ***Choose*** a ***programming language***, and ***install*** its ***tools*** on a machine (PC)
  2. ***Translate*** Algorithm into this programming language

- In this step, we ask the following questions :
  1. ***Which programming language*** to use? (procedural, object-oriented, ...)
  2. ***Which IDE*** or ***code editor*** to use? (Paid, open source, free, ...)

Step 5 : Program maintenance (Test & Debug)

- When running the program on a machine :
  1. The machine ***checks*** the ***syntax*** (spelling) of the ***program***
  2. The machine ***translates*** and ***executes*** the meaning expressed by the ***program***



Example : Dividers of a Number

Problem: Return the list of dividers of a number

Problem : Return the list of dividers of a number

Step 1 – Definition : Let N be an integer; find a solution that allows you to display these dividers.

Step 2 – Analysis :

- We **divide** **N** by i=1, i=2, i=3, ….until i=N/2 (half of N)
  - Each time we **verify** the rest of the division of **N** by i
    - If it’s « 0 », we **display** the divider (i)

Example : Dividers of a Number

Step 3 – Algorithm

```
Algorithm Diviseurs;
Var N,i : entier ;

begin
    //les entrées
    read (N);

     //manipulation des données
    for i from 1 to N DIV 2 do
        begin
          if N MOD i = 0 then
                begin
                     //les sorties
                     write (i,' est un diviseur de', N);
                end
        end
end.
```

Exemple : Dividers of a Number

Step 4 – Program in « PASCAL » langage

```pascal
program diviseurs;
    uses Crt;
    var N, i : integer;

begin
    Readln(N);
    for i := 1 to N DIV 2 do
        begin
            if N mod i = 0 then
                begin
                    Writeln(i,' est un diviseur de ', N);
                end
        end
end.
```

Step 4 – Program in « C » langage

```c
#include <stdio.h>

int main()
{
    int N, i;
    scanf("%d", &N);

    for (i = 1; i < N / 2 ; i++)
    {
        if (N % i == 0)
        {
            printf("%d est un diviseur de %d \n", i, N);
        }

    }
    return 0;
}
```

Ministry of Higher Education and Scientific Research
Djilali BOUNAAMA University - Khemis Miliana(UDBKM)
Faculty of Science and Technology
Department of Mathematics and Computer Science

Chapter 1

Introduction :
2. Introduction to Algorithms

MI-L1-UEF121 : Algorithms and Data Structures I

Noureddine AZZOUZA

n.azzouza@univ-dbkm.dz