Unit 1

Introduction

1. Introduction

A computer, basically can only perform 3 types of operations:

1. Sum, subtract, multiply or divide two numbers
2. Compare two values and know if they are equal, or the first one is bigger or smaller than the second one.
3. Store and retrieve information.

Through a program, very complex tasks can be accomplished by combining those 3 operations.

The tasks performed by a program can be carried out by a person. The advantages of using a computer are:

1. The high velocity of the calculations.
2. Total reliability of the calculations. As long as the program has been well designed.
3. Big capacity to store information.

2. Algorithm and program.

Algorithm: is a step-by-step procedure to solve a problem in a finite amount of time.

The term comes from the name of a Persian mathematician: Abu Abdallah Mohammad Ibn Musa Al Khwaritzmi (IX century).

One essential characteristic of an algorithm is that it **always finishes**.

Procedure: is a sequence of instructions that can be performed in a mechanical way.

How to start the engine of a car:

1.- Plug the key
2.- Put the gear in neutral position.
3.- Press the gas pedal.
4.- turn the key to the start position (or the start button).
5.- If the engine starts before 6 seconds, leave the key in ignition position.
6.- If the engine doesn't start in 6 seconds, wait 10 seconds and repeat stetp 3 to 6 (**but no more than 5 times**)
7.- If the car doesn't start, call the garage.

15 coins game.
This two-players game starts placing 15 coins on the table.
Player one picks from 1 to 3 coins. Next, player 2 picks from 1 to 3 coins from the remaining coins on the table. This process is repeated until there are no coins left on the table. The player which is forced to take the last coin looses.

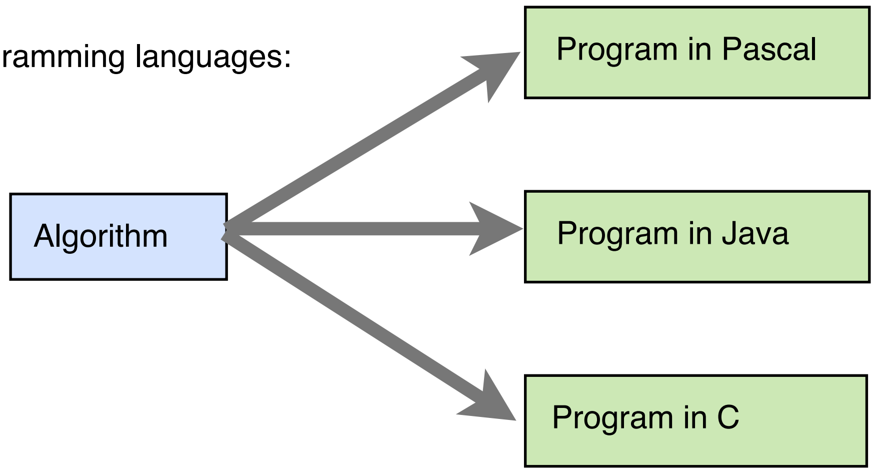
The problem consists on finding a winning strategy that allows player 1 to force player 2 to take the las coin.

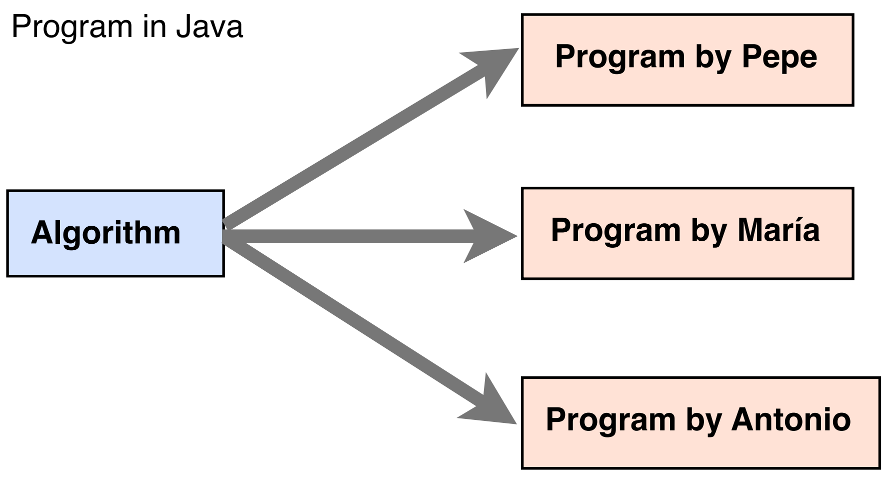
1.- Player 1 takes 2 coins
2.- Player 2 takes x coins (x>=1 and x<=3)
3.- Player 1 takes 4-x coins
4.- Repeat steps 2 and 3 until there is only 1 coin left.

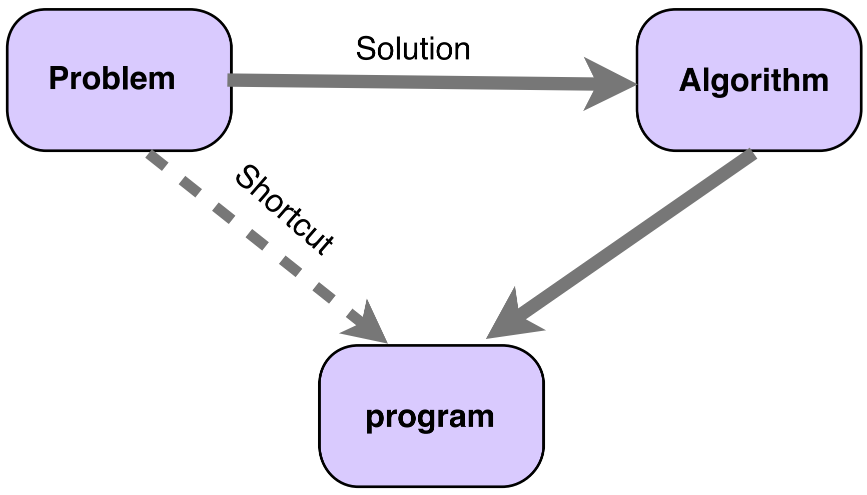
Programming language: is a set of rules, symbols and special words used to build a program.

Program: Is a sequence of instructions that indicates the actions that have to be run by a computer.

Programming languages:
….



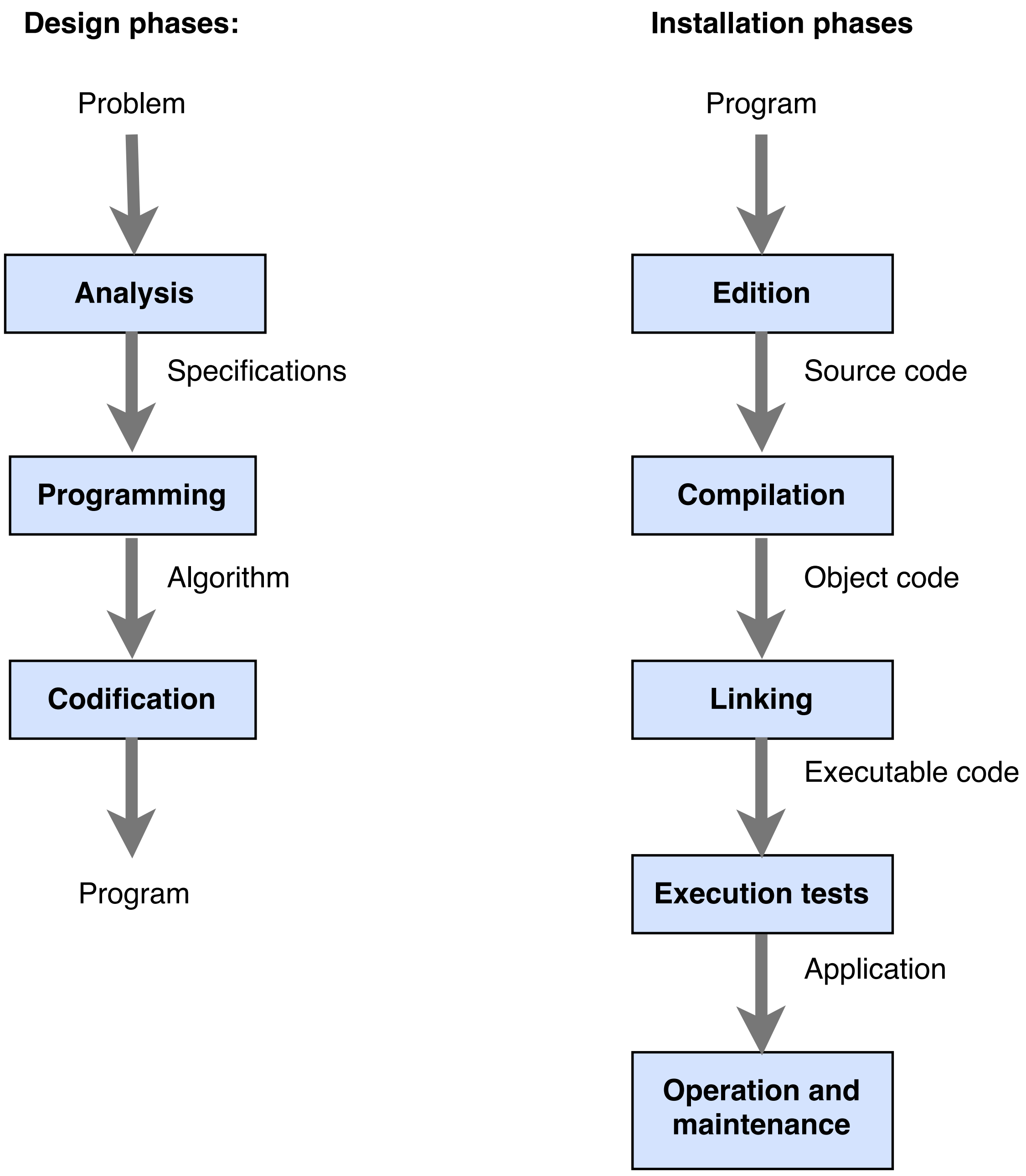




3. Application’s life cycle

An **application** is composed of one or several interrelated programs to carry out a certain task in an automatic way using a computer.

Application life cycle: Is the process followed from the Identification of the problem or task until we have a solution installed and working on the users’ computers, as long as the Application is useful for them.



3.1. Design of the program

3.1.1. Analysis phase

Consists of the study and detailed description of the problem, including:

- Analysis of the input data
- Analysis of the output data
- Relations between input and output
- Breakdown of the problem in modules.

The result of this phase is known as the **specifications** of the problem, and is a set of documents implementing all these aspects.

3.1.2. Programming phase

Is the design of the solution to the problem using an **algorithm**. To represent it there are different notations like flowcharts, pseudocode, etc.

3.1.1. Codification phase

In this phase the algorithm is translated into a specific programming language, like Java. Generally it¡s a mechanic task.

The result of this phase is the **program**.

3.2. Installation

In this section are included the installation an setup of the program in the computer.

3.2.1. Edition

Writing of the **source code** on the computer, saving it on a permanent support. This task is accomplished by a program called **editor**.

3.2.2. Compilation

Is the translation of the program in source code to machine language. The result is the **object code**. This is done by programs called **compilers and interpreters**, which besides check the correctness of the syntax of the program.

3.2.3. Linking

It is usually necessary to join together some pieces of system routines or another modules (in object code as well) to our object code. This is done by the **linker**. The result is the **executable** file.

3.2.4. Execution tests

Consists in running the program several times with different test data to ensure it works as expected.

3.2.5. Operation and maintenance

Operation consists in the continuos and usual use of the application by the users immersed in a productive environment while it is useful.

Along with the operating of the application, maintenance tasks are performed to check that is working well and to adapt it to any new circumstance that implies its modification.

4. Errors.

Depending on the moment or phase they are detected, errors can be classified in several categories.

Compilation errors:

The compilation errors (or also known as syntactic errors) are caused by breaking a syntactic rule of the programming language, such as a misspelled reserved word, an incomplete sentence etc. This errors are the easiest to fix, because they are detected by the compiler, which will indicate where the error is and what kind of error has been detected via a text message.

Running errors:

They are usually due to not allowed operations, such as dividing a number by zero, to reading a non-numeric value when a numeric value was expected, exceeding a range of allowed values, etc. They are detected because an halt of the program is produced while the program was running. We say that the system has “aborted” the program. They are more difficult to detect than the syntcatic errors because they can happen or not depending on the used input data.

Logic errors:

Correspond to obtaining wrong results by the program. The only way to detect them is to design a wide range of test data and comparing the resulting data obtained by the program with the right results calculated by other means. They are the most difficult to fix, not just because of the difficulty to detect them but because they are caused by the design of the program.

Specification errors:

They are the worse kind of error and most costly to fix. They are caused by wrong specifications due to a bad communication by the programmer and who explains the problem (usually a client). They are usually detected when the applications has already been installed and it can imply to have to make a big amount of the work again.

5. The quality of the programs

To solve a particular problem you can design different algorithms or programs. The election of the most adequate one must be based on series of quality requirements that have a great importance to assess the cost of design and maintenance.

The desired qualities for a program are:

Legibility:

It must be clear and easy, in such a way that it is easy to read and understand.

Portability:

Its design must allow the codification in different programming languages and its installation in different systems.

Modifiability:

It must facilitate its maintenance. It must be easy to modify and make the necessary updates to adapt it to a new situation.

Efficiency:

It must make the most of the computer resources, minimizing the used memory and running time or process time, as long as it isn't at the expense of the above requirements.

Programming methodology is the set of methods and techniques that help to develop programs that fulfills all the quality requirements exposed above.

6. Documentation of programs

In order to facilitate the operation and maintenance of a program it is basic to generate a wide documentation, clear and precise. It must include the specifications of the problem obtained in the analysis of the problem and all the details about how to get the maximum performance or efficiency.

There are two types of documentation depending on where they are internal and external.

Internal documentation:

It's basically formed by the source code of the programs, and it's main goal is to make it easy to read and understanding of them. Part of this documentation are:

Comments:

They are sentences that are inserted in any place of the source code of the program and that are ignored by the compilers or interpreters. They are not translated into object code and thus they don't consume extra memory. You must include as many comments as necessary to clarify the meaning of not obvious code lines.

Autocommented code:

The reserved words used in programming languages constitute by themselves part of the documentation because they correspond to terms (in English) which express it's purpose.

But the Internal documentation can be improved by:

- Creation of adequate and meaningful identifiers for variables, constants, functions, etc...
- Declaration of constants for fixed values: Ex.: IVA = 0,21
- Indentation, pagination, and insertion of empty lines to mark blocks, loops, etc...

External documentation:

Is the set of documents included with the programs but without being part of them.

- Specifications from the analysis.
- Description and diagrams of the design of the programs
- User's manual
- Maintenance manual