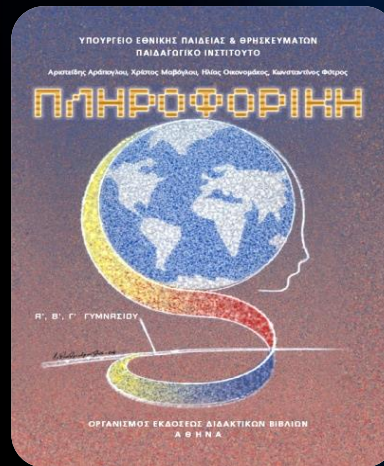


<http://www.zioulas.gr>



THE ALGORITHM CONCEPT

CHAPTER 1



EVANGELOS C. ZIOULAS (IT TEACHER)

PROBLEM

It is a situation that needs to be addressed, a problematic state that needs to be solved.

However, the solution of a problem can't be known or obvious but needs some actions in order to be confronted and resolved.

- School problems are usually **calculative** problems which demand a set of:
 - **Logical thoughts** (comparisons)
 - **Mathematical operations** (additions, subtractions etc.)

- All people confront daily problems either in their **personal** or in their **social** environment.
 - the calculation of the shortest route to our school
 - the planning and management of a school trip
 - the arrangement of our books in a library
 - the solving of a linear or a quadratic equation
 - the calculation of the perimeter or an area of a shape

- Apart from daily common problems, there are also a number of **more complex problems** which have arisen during the last years e.g.
 - the pollution of the atmosphere
 - the ozone layer depletion
 - the energy problem
 - the exploration of space
 - the malnutrition, the unemployment, the economic crisis e.g.

THE ELEMENTS OF A PROBLEM

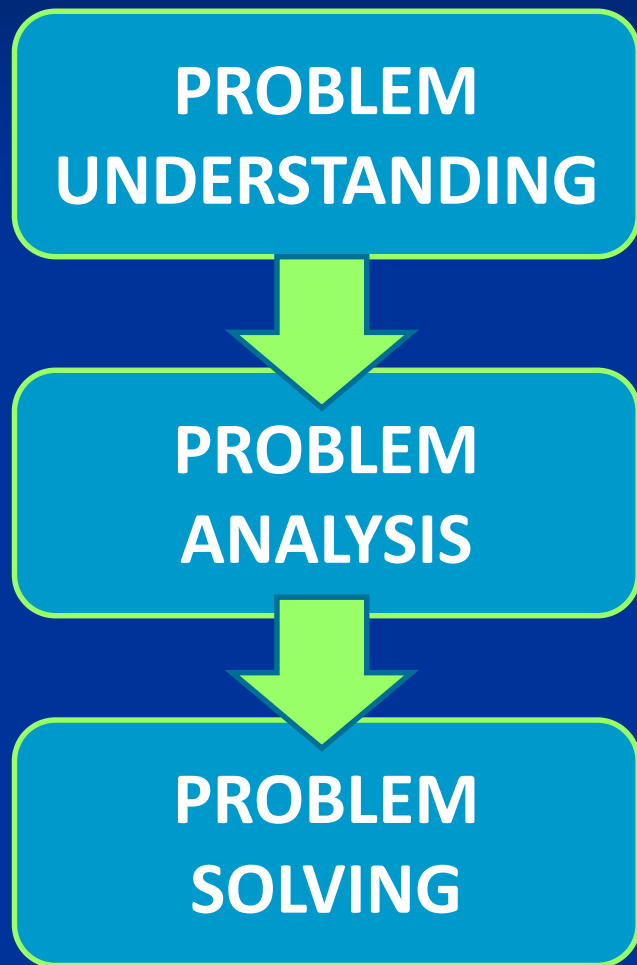


They are all the **standard elements** of a problem that are known from the beginning. They are indisputable and can be used by the solver as the basis for solving the problem.

It is the **procedure** a solver should follow in order to collect data, to understand their meaning as well as to combine and process them in an appropriate manner to **meet the objectives** of the problem, which is the conclusion of the final information.

It is anything the solver tries to find in a problem in order to get out of the problematic state. It is a matter of continuous exploration and research. Information is **the result of data processing**.

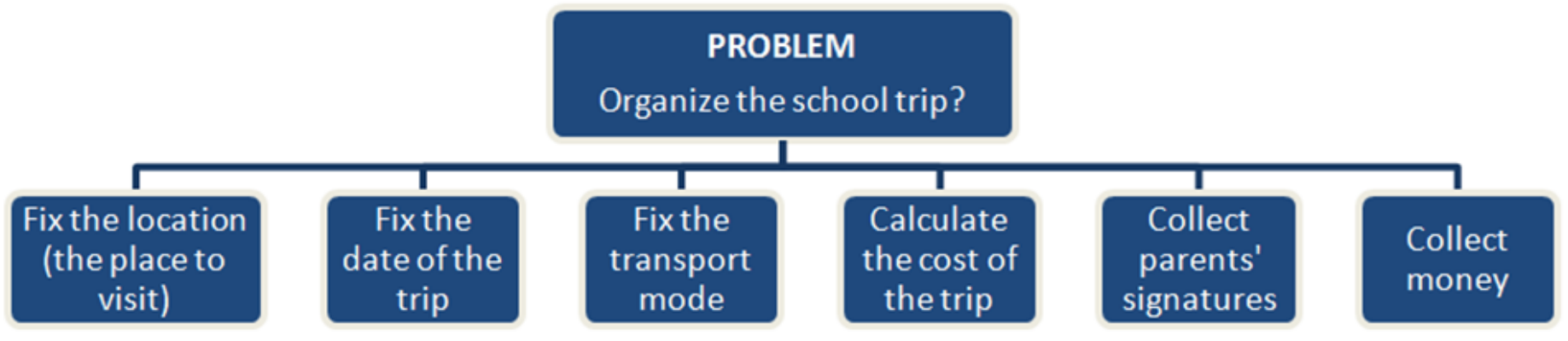
STAGES OF A PROBLEM



- During the first stage, we need to **clarify the data and the goals** (the information we need to get) of the problem, as well as to understand the specifications of the environment in which this problem is going to be solved.
- Due to its difficulty and complexity the initial problem cannot be addressed directly by the solver. So, in order to be confronted it needs to be **analyzed in individual sub-problems** which will have a simpler solution.
- It is the **implementation of the problem** solution in the form of a set of simple and clear **commands** (algorithm).

EXAMPLE

PROBLEM ANALYSIS



Analysis of the problem “Organizing our school trip” in simpler individual sub-problems.

PROBLEM CATEGORIES

Solving Ability

SOLVABLE

Problems that can be solved and their solution is already known and formulated *e.g. the calculation of the area of a shape*

OPEN

Problems whose solution has not yet been found but also we haven't proved that they are unsolved *e.g. is there life on other planets, is there life after death?*

UNSOLVED

Problems that have been proved not to have any chance to be solved *e.g. the squaring of the circle, the prevention of death etc.*

Structuring Level

STRUCTURED

Problems whose solution derives from one individual automated procedure (a series of known steps) that is already formulated *e.g. how can we find the roots of a linear or a quadratic equation?*

SEMI-STRUCTURED

Problems that can be solved within a range of possible automated procedures leaving to the human factor the ability to choose *e.g. the selection of transportation mode for a trip*

UNSTRUCTURED

Problems whose solution is based on human intuition and doesn't follow any automated procedure *e.g. how can we organize our party?*

Solving Type

DECISION

Problems that often can be answered with a single "Yes" or "No". They are closed type problems that leave solver the ability to decide what of the available choices is correct *e.g. is number 15 odd or even? is number 7 a primary number?*

COMPUTATIONAL

Problems whose solution derives from a number of calculations and at the end the final answer of the problem is a number *e.g. a mathematical operation*

OPTIMIZATION

Problems for which we are trying to find the optimum result, the result that satisfies the data of the problem in the best possible way *e.g. the finding of the fastest (or slowest) route for a specific location.*

ALGORITHM

Algorithm is a finite set of individual instructions (logical steps or commands) that are formulated with accuracy and definiteness having as major objective the solution of a problem.

- This term derives from the name of Persian Mathematician Mohammed ibn-Musa al-Khuwarizmi (750 – 850 A.D.) who dealt with the number processing (**algorismus**).
- In 17th century, this word was combined with the Greek word «**αριθμός**» and thus transformed in its current form.

- An algorithm is a term that is not only associated with **scientific** problems but also with any problem derived from daily **personal** or **social** life e.g.
 - The steps of a cooking recipe
 - The steps for a mathematical operation
 - The steps for finding the shortest route to a particular place

Algorithm Example

Create a meal

1. Gather materials
2. Prepare all pans
3. Cook the meal
4. Prepare the salad
5. Prepare the table
6. Lunch your meal
7. Clean the table
8. Wash the dishes
9. Arrange the cuisine

When we design an algorithm, we should be very careful to put its set of **instructions in a logical order**, so that we can build a correct solution for the problem.

- The steps that constitute the algorithm are called **commands** or **instructions** and should always be placed in a logical order.
- The one who executes the commands of an algorithm is called an **executioner** and might be a human being or a computer machine.
- When an algorithm is expressed in a set of commands that are understandable to computer then it is called a **program**.

THE GOAT, THE WOLF & THE CABBAGE

Play the game

A sailor needs to bring a wolf, a goat, and a cabbage across the river. The boat is tiny and can only carry one passenger at a time. If he leaves the wolf and the goat alone together, the wolf will eat the goat. If he leaves the goat and the cabbage alone together, the goat will eat the cabbage. How can he bring all three safely across the river?

Data:

1 goat, 1 wolf, 1 cabbage, 1 available seat in the boat,
2 banks in the river

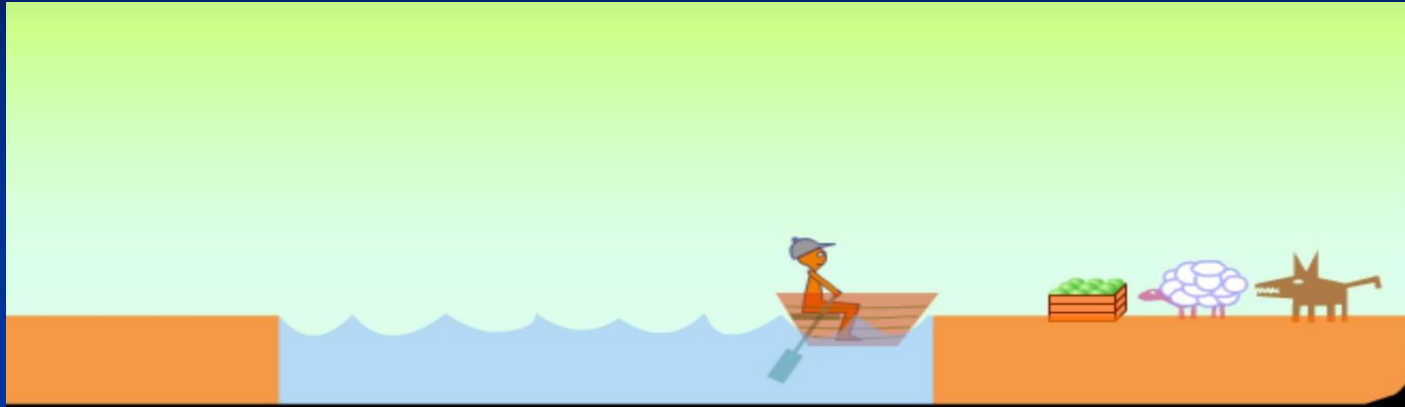
**Context of the
problem:**

The wolf and the goat shouldn't stay alone together.
The goat and the cabbage shouldn't stay alone together.

Goal:

The goat, the wolf and the cabbage should pass to the opposite bank

PROBLEM SOLVING



1. Put the goat in the boat.
2. Move the boat to the second bank.
3. Leave the goat to the second bank.
4. Turn back the boat to the first bank.
5. Put the cabbage in the boat.
6. Move the boat to the second bank.
7. Leave the cabbage to the second bank.
8. Put the goat in the boat.

9. Turn back the boat to the first bank
10. Leave the goat to the first bank.
11. Put the wolf in the boat.
12. Move the boat to the second bank.
13. Leave the wolf to the second bank.
14. Turn back the boat to the first bank.
15. Put the goat in the boat.
16. Move the boat to the second bank.
17. Leave the goat to the second bank.

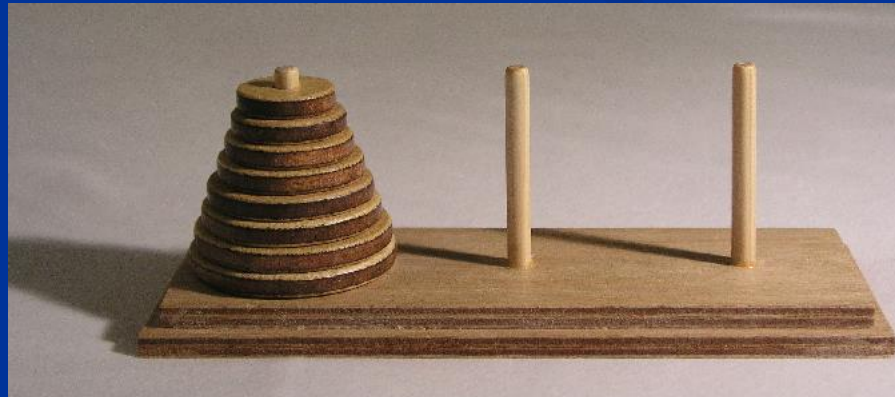


Play the game

THE TOWERS OF HANOI

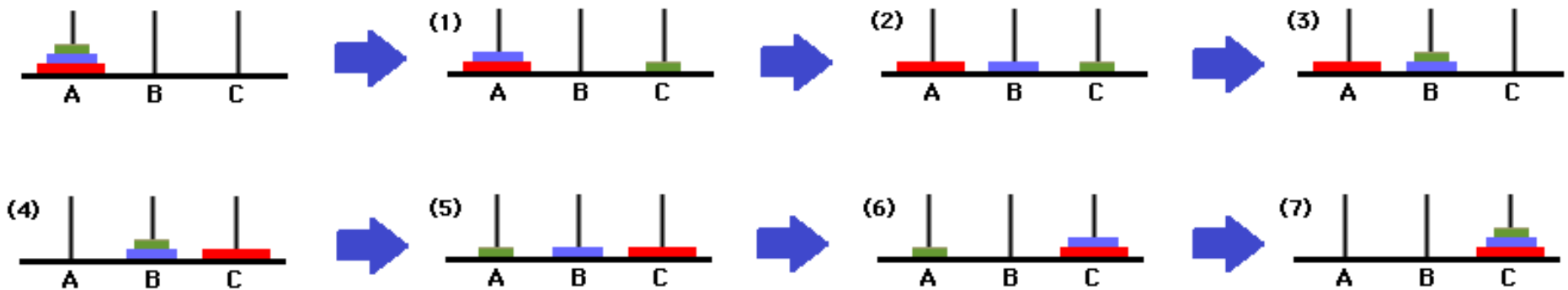
We have three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack to another rod, obeying the following rules:

- A) Only one disk must be moved at a time.
- B) To move a disk we take the upper disk from one of the rods and slide it onto another rod, on top of the other disks that may already be present on that rod.
- C) No disk may be placed on top of a smaller disk.



PROBLEM SOLVING

Algorithmic solution for 3 disks



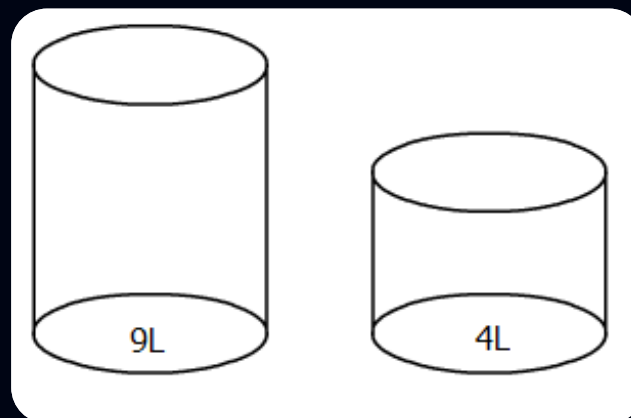
1. Move the small disk to the 3rd rod
2. Move the medium disk to the 2nd rod
3. Move the small disk to the 2nd rod
4. Move the big disk to the 3rd rod
5. Move the small disk to the 1st rod
6. Move the medium disk to the 3rd rod
7. Move the small disk to the 3rd rod

ALGORITHM ATTRIBUTES

CLEAR	Every step leaves no doubt about its correct execution
EFFECTIVE	It gives its results in a finite time period
FEASIBLE	It is executable by an individual executioner
FAST	It uses the least possible number of steps to be completed
ECONOMIC	It requires fewer resources as possible
GENERAL	It is able to solve a set of related problems

EXAMPLE 1

- Chapter 1 – Activity 3 (p. 201)
- List the steps of the algorithm that resolves the following problem of measuring water:
- *You have 2 containers of water. The first holds 9 liters and the second 4 liters. Can you measure out exactly 6 liters of water using only these two containers and unlimited amount of water?*



SOLUTION

Command	BIG	SMALL
Fill the small can	0	4
Move 4L to the big can	4	0
Fill the small can	4	4
Move 4L to the big can	8	0
Fill the small can	8	4
Move 1L to the big can	9	3
Empty the big can	0	3
Move 3L to the big can	3	0
Fill the small can	3	4

Command	BIG	SMALL
Move 4L to the big can	7	0
Fill the small can	7	4
Move 2L to the big can	9	2
Empty the big can	0	2
Move 2L to the big can	2	0
Fill the small can	2	4
Move 4L to the big can	6	0

EXAMPLE 2

- Chapter.1 – Activity 3 (p.201)
- List the steps of the algorithm that resolves the following problem of measuring time:
- Instead of a watch, we have two hourglasses of 7 minutes and 4 minutes respectively.
- How can we measure exactly 9 minutes?



SOLUTION

Command	Hourglass 7'	Hourglass 4'	Total time
Fill both hourglasses simultaneously	4 min	4 min	4 min
Turn the small hourglass upside down		0 min	
Fill both hourglasses simultaneously	7 min	3 min	7 min
Turn the big hourglass upside down	0 min		
Fill both hourglasses simultaneously	1 min	4 min	8 min
Turn the big hourglass upside down	0 min		
Empty the big hourglass	1 min		9 min

PROGRAM

It is an algorithm that is formulated in such a way that its **set of instructions** is fully **understood by the computer** machine so they can be executed instantly.

- The rule that derives from the previous definition is that while **a program is always a type of an algorithm**, the opposite doesn't happen necessarily.
- Any algorithm to be conceivable and executable by a computer, needs to respect a set of spelling and syntax specifications.

```

PROGRAM primecheck;
  { This is Turbo Pascal. }

VAR n,i,max: INTEGER;
    continue: BOOLEAN;
    status: STRING[13];

BEGIN
  REPEAT
    write('Enter a whole number (type 0 to quit): ');
    readln(n);
    status := ' is prime';
    IF n>2 THEN continue := TRUE
      ELSE continue := FALSE;
    i := 1;
    max := TRUNC(SQRT(n));
    { max is the largest divisor that must be
      checked to see if n is prime}
    WHILE continue DO
      BEGIN
        i := i + 1;
        IF (n MOD i)=0 THEN
          BEGIN
            status := ' is not prime';
            continue := FALSE
          END;
        continue := (i < max);
      END; {end of WHILE loop}
    write(n);
    writeln(status);
  UNTIL n=0
END.

```

Example of a
program
(Turbo Pascal)



Checking a number if it is
prime or not

PROGRAMMING LANGUAGE

It is an **artificial language** that is a powerful tool of the user in order to communicate with the computer and express his algorithms in an intelligible manner.

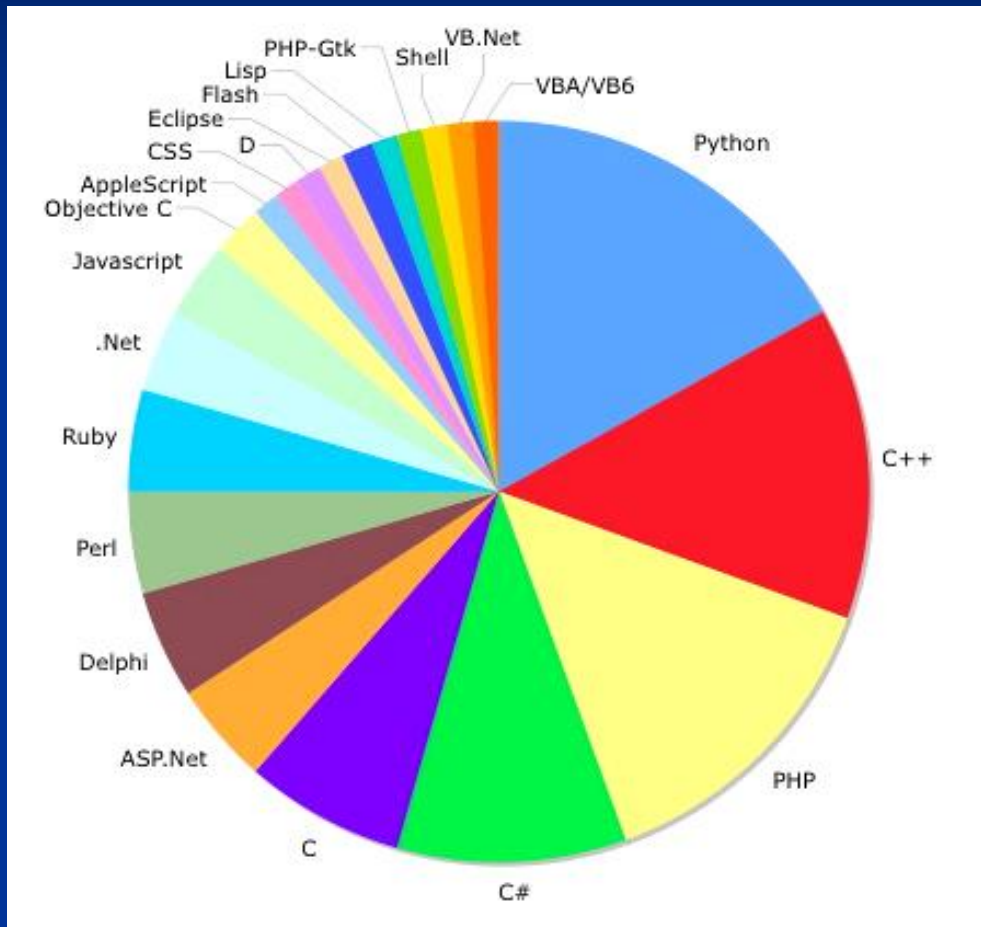
- As any natural language, a programming language has its own **syntactic** and **conceptual rules** to form its instructions.

- Each programming language has its own **syntax**, **vocabulary** and **alphabet**.
- Today there are some hundreds of programming languages, that some of them might be useful to an individual programmer e.g.

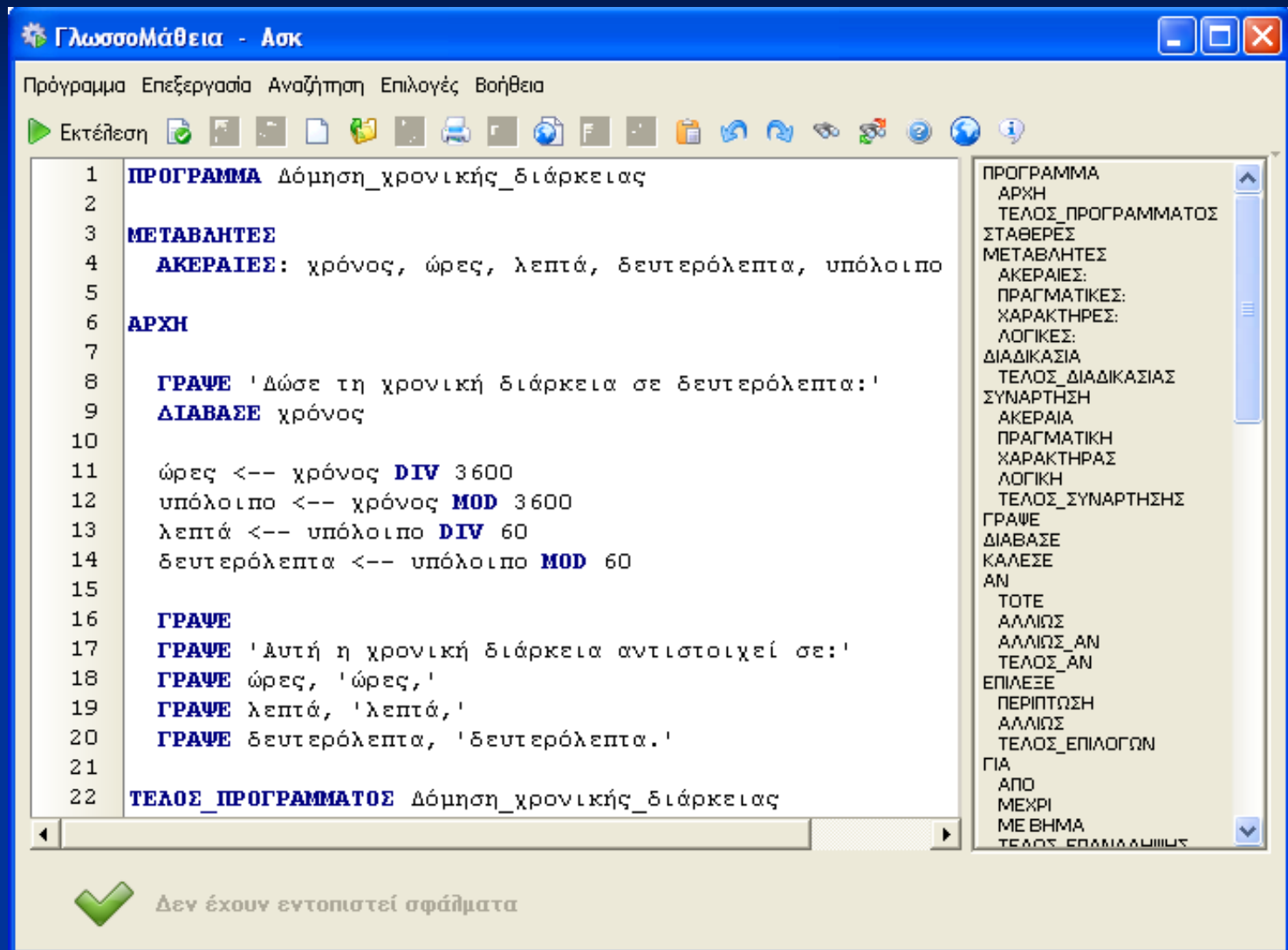
Pascal, Basic, Fortran, Cobol, Logo, C, C++, Java, Python,
Perl, Lisp, Visual Basic, C#, SQL, Ada, Prolog

- Each programming language is made to implement specific range of applications.

PROGRAMMING LANGUAGES



Example of a program in Glossomatheia programming environment



The screenshot shows the Glossomatheia programming environment. The main window title is "ΓλωσσοΜάθεια - Ασκ". The menu bar includes "Πρόγραμμα", "Επεξεργασία", "Αναζήτηση", "Επιλογές", and "Βοήθεια". The toolbar contains icons for execution, file operations, and help. The main editor displays a Pascal program for calculating the breakdown of a duration into hours, minutes, and seconds. The symbol table on the right lists the program's components, including variables, constants, and procedures.

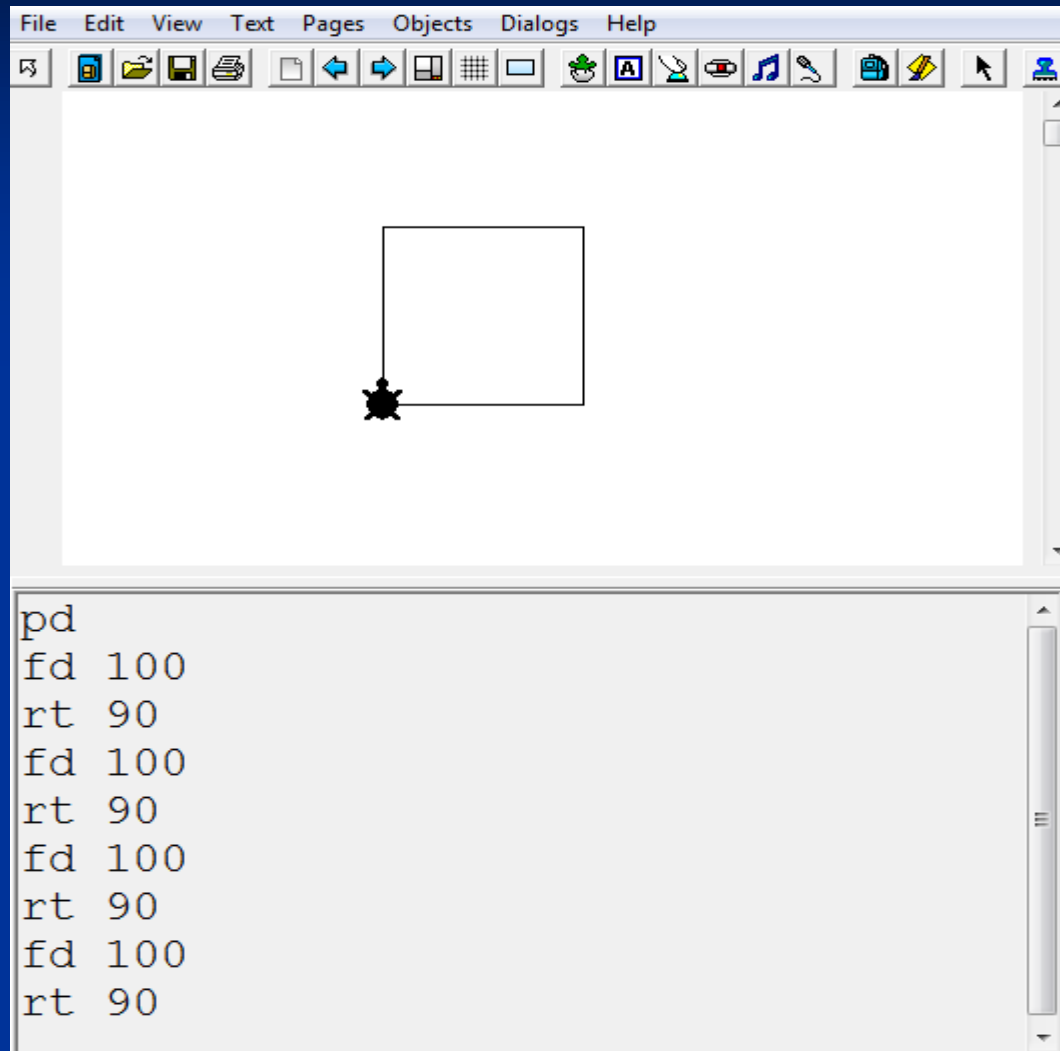
```
1  ΠΡΟΓΡΑΜΜΑ Δόμηση_χρονικής_διάρκειας
2
3  ΜΕΤΑΒΛΗΤΕΣ
4  ΑΚΕΡΑΙΕΣ: χρόνος, ώρες, λεπτά, δευτερόλεπτα, υπόλοιπο
5
6  ΑΡΧΗ
7
8  ΓΡΑΨΕ 'Δώσε τη χρονική διάρκεια σε δευτερόλεπτα:'
9  ΔΙΑΒΑΣΕ χρόνος
10
11  ώρες <-- χρόνος DIV 3600
12  υπόλοιπο <-- χρόνος MOD 3600
13  λεπτά <-- υπόλοιπο DIV 60
14  δευτερόλεπτα <-- υπόλοιπο MOD 60
15
16  ΓΡΑΨΕ
17  ΓΡΑΨΕ 'Αυτή η χρονική διάρκεια αντιστοιχεί σε:'
18  ΓΡΑΨΕ ώρες, 'ώρες,'
19  ΓΡΑΨΕ λεπτά, 'λεπτά,'
20  ΓΡΑΨΕ δευτερόλεπτα, 'δευτερόλεπτα.'
21
22  ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Δόμηση_χρονικής_διάρκειας
```

Symbol Table:

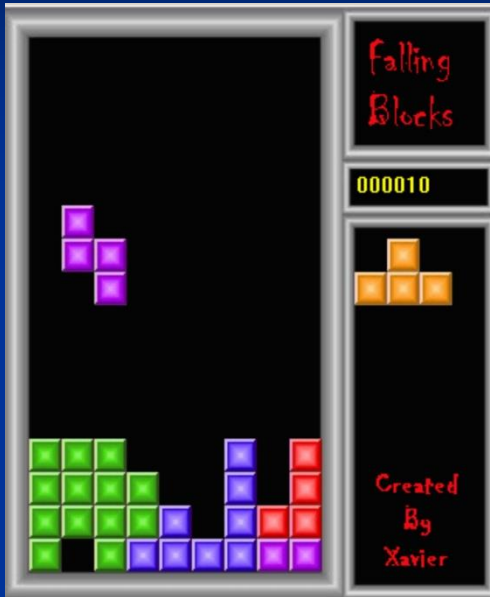
- ΠΡΟΓΡΑΜΜΑ
- ΑΡΧΗ
- ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
- ΣΤΑΘΕΡΕΣ
- ΜΕΤΑΒΛΗΤΕΣ
- ΑΚΕΡΑΙΕΣ:
- ΠΡΑΓΜΑΤΙΚΕΣ:
- ΧΑΡΑΚΤΗΡΕΣ:
- ΛΟΓΙΚΕΣ:
- ΔΙΑΔΙΚΑΣΙΑ
- ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
- ΣΥΝΑΡΤΗΣΗ
- ΑΚΕΡΑΙΑ
- ΠΡΑΓΜΑΤΙΚΗ
- ΧΑΡΑΚΤΗΡΑΣ
- ΛΟΓΙΚΗ
- ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
- ΓΡΑΨΕ
- ΔΙΑΒΑΣΕ
- ΚΑΛΕΣΕ
- ΑΝ
- ΤΟΤΕ
- ΑΛΛΙΩΣ
- ΑΛΛΙΩΣ_ΑΝ
- ΤΕΛΟΣ_ΑΝ
- ΕΠΙΛΕΞΕ
- ΠΕΡΙΠΤΩΣΗ
- ΑΛΛΙΩΣ
- ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
- ΓΙΑ
- ΑΠΟ
- ΜΕΧΡΙ
- ΜΕ ΒΗΜΑ
- ΤΕΛΟΣ_ΕΡΩΤΗΜΑΤΟΣ

Δεν έχουν εντοπιστεί σφάλματα

Example of a program in **MicroWorlds Pro** programming environment



PROGRAMMING GAMES



```
void DisplayBlock(SBlock Block)
{
    if (Block.nY < 1) return;
    RECT rcBlock = g_rcBlock;
    rcBlock.left = Block.nColor * BLOCK_DIAMETER;
    rcBlock.right = Block.nColor * BLOCK_DIAMETER + BLOCK_DIAMETER;

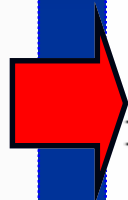
    g_pDisplay->Blit( (DWORD)Block.nX * BLOCK_DIAMETER - 2 ,
                     (DWORD)Block.nY * BLOCK_DIAMETER ,
                     g_pSecondarySurface, &rcBlock );
}
```

The famous game **TETRIS** is based in a program that includes a wide set of instructions. A small subset of commands written in C programming language is depicted in the above image.

THE EVOLUTION OF LANGUAGES

```
10101000 00001010
10001100 00000001
00111100
01010001 00000001
01000011 00000001
11000000 11111010
10001100 00000010
11111111
```

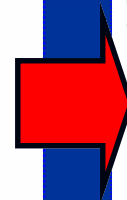
Machine Language



LOOP

```
INDEX=$01
SUM=$02
LDA #10
STA INDEX
CLA
ADD INDEX
DEC INDEX
BNE LOOP
STA SUM
BRK
```

Symbolic Language
(Assembly)



```
sum = 0
FOR index=1 TO 10
    sum=sum+index
NEXT index
END
```

High-Level Language



MACHINE LANGUAGE

- It is the **mother language** of computer (the only language it understands).
- All the instructions are written in binary form in **bits** (series of 0 and 1).
- It is a very hard type of programming with high complexity.
- A program in binary form is executed directly by the computer because it is fully understandable without the need of any translation.

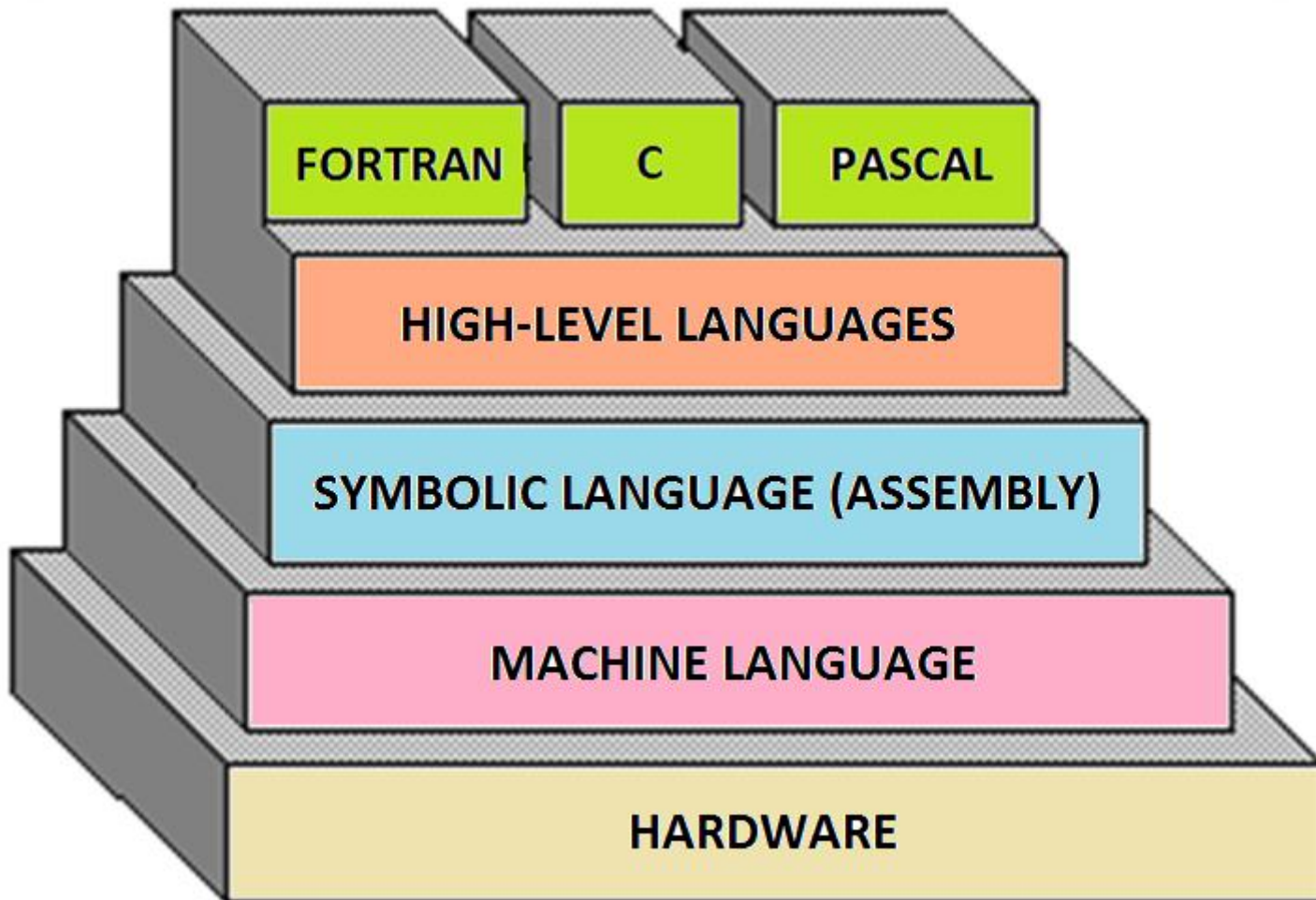
SYMBOLIC LANGUAGE

- It is a more comprehensible kind of language which uses **symbolic names** instead of numbers to express its commands.
- Assembly is also known as **low level language** because it is not independent from the computer architecture.
- The symbolic program is not executable directly by the computer, so it **needs to be translated** to machine language first.
- The program which makes the necessary translation is called an **assembler**.

HIGH-LEVEL LANGUAGE

- They resemble to human languages so they seem to be more familiar to programmers.
- They use a **short vocabulary** of English words as a set of commands, a simple syntax and many **mathematical symbols**.
- Each instruction of a high level language corresponds to a set of binary instructions in machine language.
- To be translated in machine language, high-level languages have two types of translation programs: **compilers** and **interpreters**.
- Unlike Assembly, they are portable so they are **independent** from computer architecture.

LANGUAGE CATEGORIES



ATTRIBUTES OF A PROGRAMMING LANGUAGE

ALPHABET	It is the set of characters - symbols which are used by the language to express its commands (e.g. letters, numbers, punctuation, mathematical symbols)
VOCABULARY	It is the set of words (instructions) which are recognizable by the language as accepted commands and they have a fixed meaning.
SYNTAX	It is the set of rules which determine the spelling and syntax restrictions of a language. It constitutes the grammar of a programming language.

PROGRAMMING ENVIRONMENT

- It is a software suite which allows programmers to develop their applications and programs.
- The environment should provide programmer 2 basic programming tools:

- EDITOR

It is the program we need to edit and correct the source code of the program (it has a similar usage to a word processor).

- TRANSLATOR

It converts program instructions from high level language (**source file** in form of English commands) to machine language (**object file** in form of bits) searching simultaneously for possible syntax errors.

TRANSLATOR CATEGORIES

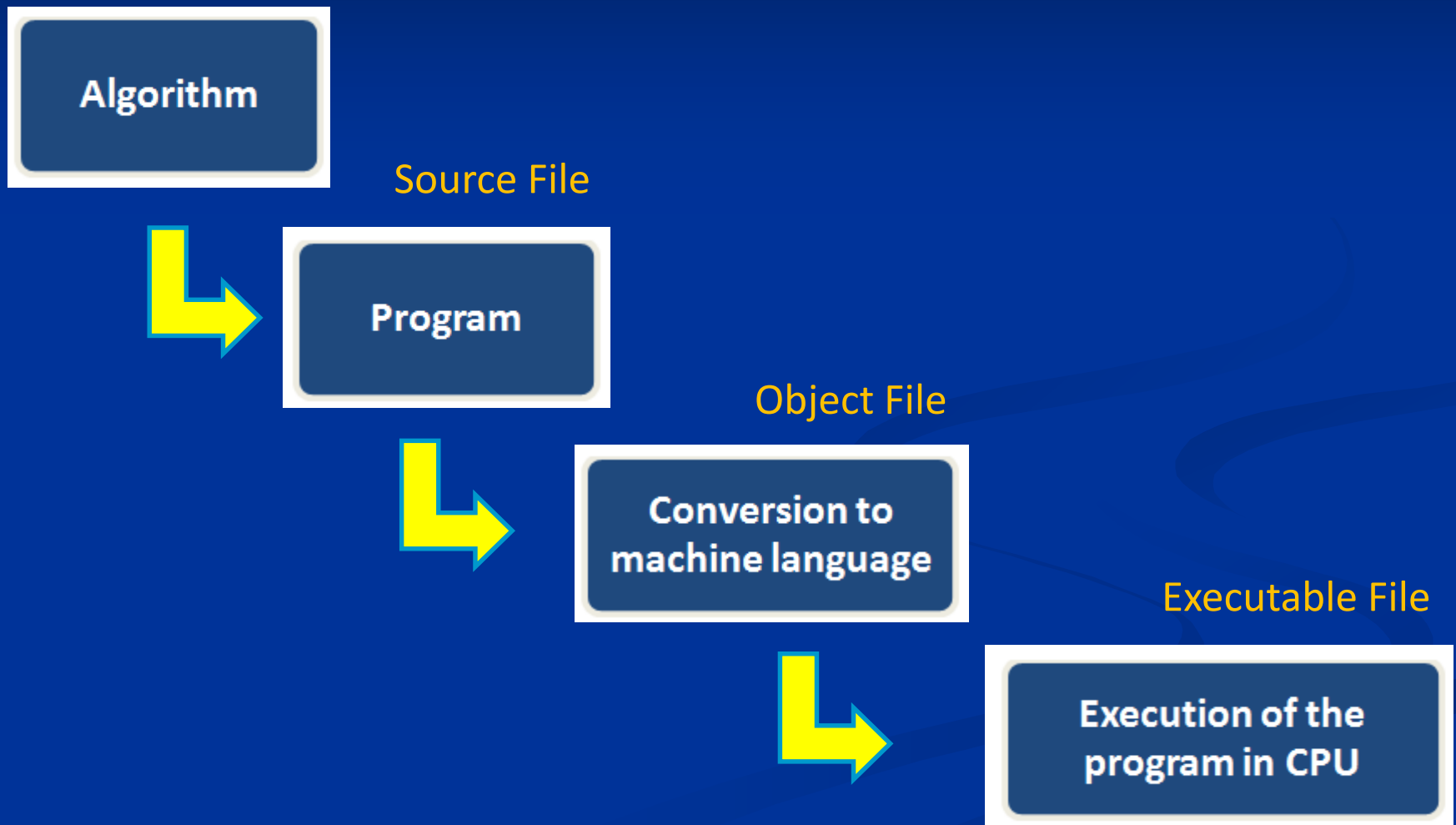
COMPILER

It checks **the entire program** for syntax errors and in case it doesn't find any, it converts the program into machine language in order to be executed.

INTERPRETER

It checks the program **line by line** (each one command individually) and in case it is syntactically correct, it converts it to machine language and executes it.

STAGES OF A PROGRAM EXECUTION



ERROR TYPES

- There are two basic categories of error that are made in programming very often:

- **SYNTAX ERRORS**

- They are errors caused by a **violation of grammatical rules** of the programming language (e.g. a violation of syntax or spelling limit the execution of a program) and **can be detected** by the compiler during translation.

- **RUNTIME ERRORS**

- They are errors caused by an **incorrect algorithm** made by the programmer in order to implement the program and they appear mainly during execution. They **cannot be detected** by any compiler during translation.