Algorithms and Data Structures

Dr Cherif BENALI

University Centre Abdelhafid Boussouf-Mila

Mathematics and Computer Science Institute

Computer Science Department

EMAIL: *c.benali@centre-univ-mila.dz*

19/02/2025
4.2

Table of contents

Objectives

- **Prerequisites**
  Before starting this chapter, learners should:

  1. ✓ Logical Thinking : Understand basic logic operations (e.g., AND, OR, NOT) and be able to follow instructions step by step.
  2. + Mathematical Foundations : Be comfortable with simple math (addition, subtraction) and number systems like decimal and binary.
  3. Basic Programming Knowledge : Know basic computer usage and identify key components like the keyboard, screen, and mouse.

- **General objective of the course:**
  By the end of **this course**, the learner will be able to understand the basic concepts of general computing and design and interpret simple sequential algorithms, using fundamental algorithmic structures.
  Additionally, the learner will be able to:

  1. Understand basic algorithms and the basic concepts of general computing
  2. Design and interpret simple sequential algorithms using fundamental algorithmic structures
  3. Understand basic algorithms
  4. ✍ Write clean and readable code
  5. ⚙ Optimize simple programming solutions
  6. Improve logical thinking and problem-solving skills

- **Specific Objectives – Chapter 1: Introduction to General Computing**
  By the end of **this chapter (01)**, the learner will be able to:

  1. Define basic computer science concepts (computer science, computer, data, information)
  2. Distinguish between memory types and computer components
  3. Classify software according to type (system, application)
  4. Identify the different programming languages (machine, assembler, high-level)
  5. Describe how a computer system works according to Von Neumann architecture

- **Specific Objectives – Chapter 2 : Sequential Algorithms**
  By the end of this chapter, the student will be able to:

  1. Define an algorithm and understand its characteristics

2. Analyze a problem to identify the input data, expected results, and the solution method
3. Formulate a simple algorithm using algorithmic language
4. ÷ Apply precedence rules in arithmetic expressions
5. Design an algorithm using input/output, assignment, and comment statements

Introduction

This chapter lays the foundation for your journey into the world of computer science.

Whether you're new to the field or refreshing your knowledge, this chapter is your entry point into understanding how digital systems function. It introduces you to the essential building blocks that underpin all modern computing, preparing you to think logically, analyze systems, and appreciate the technology that shapes our world.

You will:

1. Learn essential definitions: Grasp the meanings of key terms such as informatics and computer, and learn to distinguish between conceptual ideas (like data and information) and physical components (like hardware).
2. Explore how information is represented: Discover how computers process and store information using number systems, including the familiar decimal system and the fundamental binary system used in digital logic.
3. Understand system components: Break down a computer into its hardware (e.g., CPU, memory, and input /output devices) and software (operating systems, applications), and understand how these work together.
4. Uncover data flow: Learn how data moves within the computer using elements such as buses and memory units, and see how software controls and coordinates the entire system.
5. Get introduced to programming languages: See how humans communicate with machines using different layers of language from low-level machine code, to assembly, up to high-level languages like Python or C++.
6. This chapter is more than just theory it equips you with the vocabulary, understanding, and context you'll need to confidently tackle more advanced computing topics in the chapters ahead.

By the end of this chapter, you will be able to:

1. Define key computing concepts such as informatics and computer
2. Explain how information is represented using the decimal and binary number systems
3. Identify hardware and software components of a computer system
4. Distinguish between different categories of programming languages (e.g., machine, assembly, high-level)

The following mind map allows you to have an overview of the course content in full:

Or you can access the **PDF** version from this **Google Drive link**:

: *https://drive.google.com/file/d/1JyLicFNLEpXqm2zahnS087PBXJECXih1/view?usp=sharing*

Pre-requisite Test (Entry-level) with orientation

I

Objectives

Pre-requisite Test (based only on what learners are expected to know before Chapter 1

Exercice : QUESTION 01

Q1. What does the term “information” refer to in everyday language?

- a) Only images
- b) Only numbers
- c) Any knowledge or fact that can be communicated
- d) Only sound

Exercice : QUESTION 02

Q2. What device is used to input text into a computer?

- a) Screen
- b) Mouse
- c) Keyboard
- d) Printer

Exercice : QUESTION 03

Q3. Que donne l'expression : 5 - 2 * 3 en respectant la priorité des opérateurs ?

- a) 9
- b) 1
- c) -1
- d) 6

Exercice : QUESTION 04

Q4. What is the common numbering system we use in daily life?

- a) Binary
- b) Decimal
- c) Hexadecimal
- d) Octal

Exercice : QUESTION 05

Q5. Which of the following is an example of data?

- a) A conclusion
- b) An organized report
- c) A raw number or word
- d) A compressed file

Exercice : QUESTION 06

Q6. What is required to operate a computer?

- a) Just electricity
- b) Only hardware
- c) Software and hardware
- d) A printer

Exercice : QUESTION 07

Q7. Why do we study algorithms?

- a) To play video games
- b) To write essays
- c) To create music
- d) To solve problems logically

Exercice : QUESTION 08

Which of the following devices is commonly used to display information from a computer?

- a) Monitor
- b) Keyboard
- c) USB stick

Exercice : QUESTION 09

Q8. What does a mouse help you do when using a computer?

- a) Navigate and select items on the screen
- b) Display images
- b) Control the operating system
- d) Install software

Exercice : QUESTION 10

Which term best describes a set of instructions given to a computer to perform a task?

- a) Processor
- b) Input
- c) File
- d) Program

I.1 Definition and Fundamentals

II

1. I.1.1 Definition of Informatics

Definition

The term **INFORMATIC** is a term resulting from the contraction of the two words

"**Information**" and "**automatic**".

We have two main concepts:

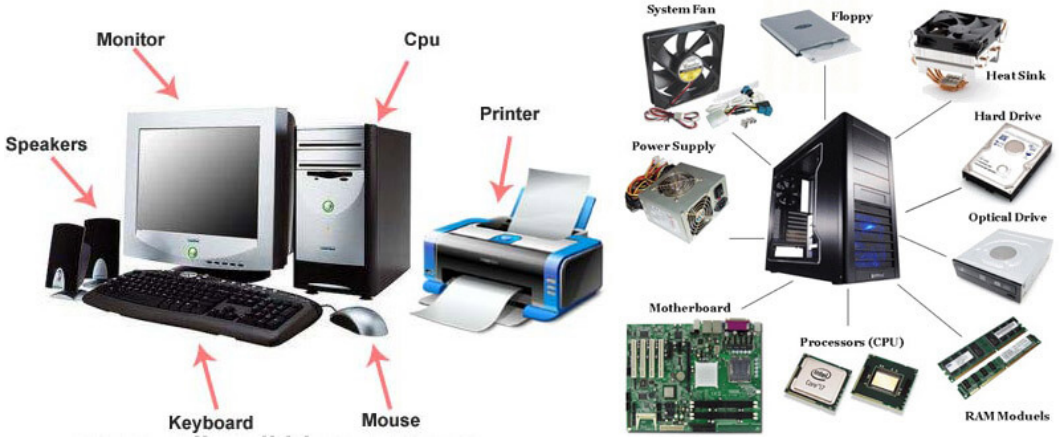
1. **Automatic processing:** It represents the sequence of operations (instructions) performed by software (program). The **Instruction: (or command)**is an order given by the user (or Program) to the computer.
   Example: in Microsoft Word changing the size of a text is an operation (It's automatic processing).
2. **Information:** means anything that can be processed by a machine (computer). In other words, the facts and knowledge deduced from the data.
   Data: Is the processed information.
   Example : text, number, image, sounds, video...

So, computer science is the science of automatic processing of information using automatic machines (computers).

2. I.1.2. Computer

Definition

Computer is an automatic (programmable) information processing machine. It can process various types of information (texts, drawings, images, sounds) but internally all this information is converted into digital form.



Computer Parts (B.ch)

I.2. Representation of information

1. All communications inside the computer are done with electrical signals.
2. These electrical signals have only two states:
3. So the computer manipulates information in binary form
    - **0:** off (no electrical signal).
    - **1:** on (presence of electrical signal).
4. A unit of information (**0 or 1**) is called a **bit (binary digit).**

Why binary digits?

1. a) Decimal system

Definition

It's a base 10 number system.

- **Origin:** ten fingers in the hands
- **Representation: 10** different symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Example

representation of a number (**580**):

5 hundreds, 8 tens, 0 ones

Math equivalent : 5 *100 + 8 *10 + 1 * 1

2. b) The Binary System

Definition

It is a base 2 number system.

- easier and more reliable to read an electrical signal
- representation: two states **0** (false) and **1** (true)

Example

- representation of number **6** : **110**
- Mathematical equivalent: 1* **4** + **1*** 2 + **0** * 1

Decimal to Binary Table

Decimal (Base 10)	Binary (Base 2)	Decimal (Base 10)	Binary (Base 2)
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

So the representation of a character = the combination of several bits. The most used combination is that corresponding to 8 bits witch call byte.

Exercice

IV

Exercice

Which of the following statements are TRUE about how a computer handles information?

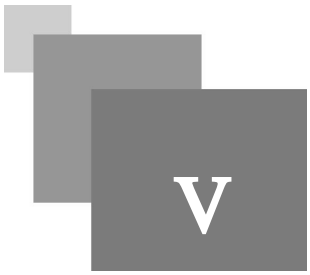
- ☐ a) A computer can directly process information in various forms like text and images.s
- ☐ b) Internally, all information is converted into a digital form.
- ☐ c) Computers primarily use analog signals for communication.
- ☐ d) A computer is a programmable machine that processes information.

Exercice

Which of the following statements are TRUE about electrical signals and binary form in computers?

- ☐ a) Computer communication relies on electrical signals.
- ☐ b) Electrical signals in computers have more than two states.
- ☐ c) The binary system uses "0" to represent "off" and "1" to represent "on."s
- ☐ d) A bit is a unit of information representing a decimal digit.

I.3. Computer system



This is the set of software and hardware means necessary to meet the computer needs of users. A computer system is made of two parts: Hardware and Software.

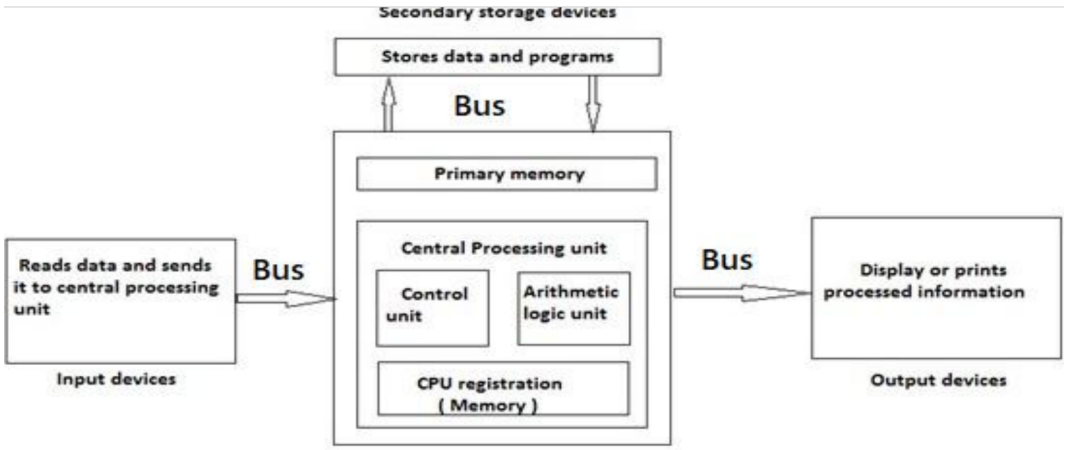
1. I.3.1. Hardware

1.1. Definition

Definition

It is all the hardware that makes up the computer and its peripherals.

All current computers are based on the following Von Neumann model (1903-1957):



Von Neuman Architecture (B.C)

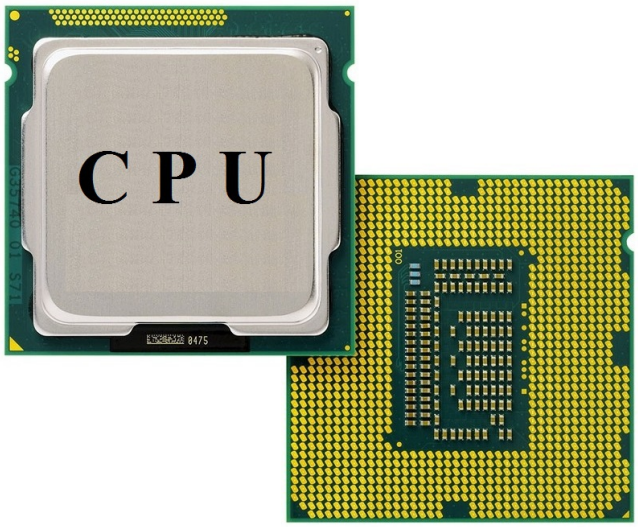
1.2. Central unity

The central unit consists of a processor and a primary memory:

1.2.1. The processor (central processing unit)

it is the “brain” of the computer. its role is to execute the programs stored in central memory by loading the instructions, decoding them and executing them one after the other. The processor mainly comprises a command and control unit (**CCU**) and an arithmetic logic

unit (**ALU**):



The processor (central processing unit) (by B.C)

a) Control Unit (CU)

Definition

It is the intelligent part of the microprocessor. It makes it possible to seek the instructions of a program located in the RAM memory, to interpret them and then to route the data to the ALU in order to process them.

b) Arithmetic Logic Unit (ALU)

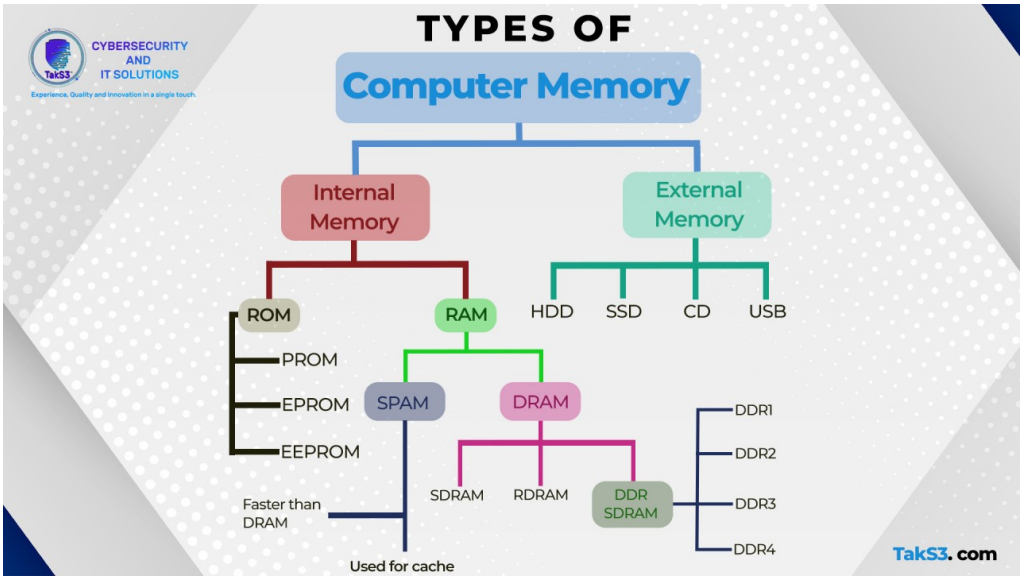
Definition

It is made up of a set of circuits (memory registers) responsible for performing arithmetic operations (addition, subtraction, multiplication, division) and logical operations.

1.2.2. Main memory

Primary Memory is a section of computer memory that the **CPU** can **access** directly. Primary Memory has a **faster** access time than **secondary memory** and is **faster/shorter** than cache memory in a memory hierarchy. Primary Memory, on average, has a storage capacity that is lower than secondary memory but higher than cache memory.

The primary memory is a basic component of the computer, without which any functioning becomes impossible. Its role is to store the data before and during their processing by the processor. There are essentially **02 types** of internal memories:

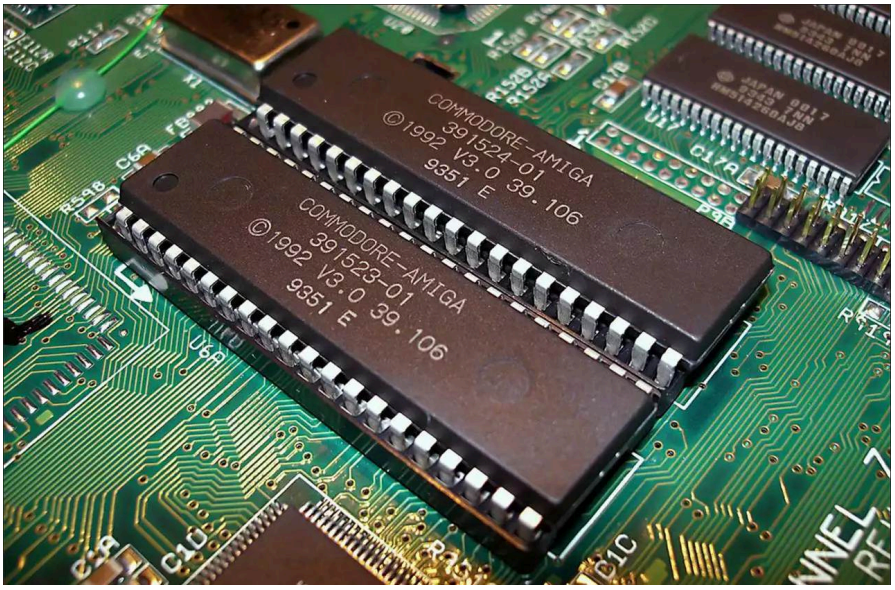


Types of memory (B.c)

a) ROM (Read Only Memory)

Definition

is a dead and non-volatile memory that can only be read. it contains information needed to start the computer.

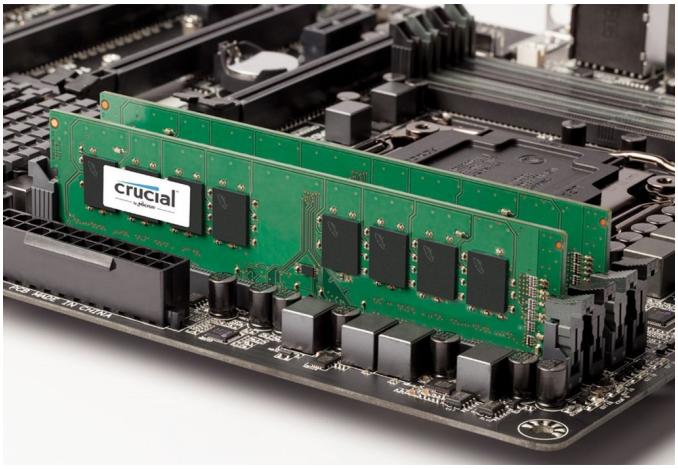


ROM (Read Only Memory) (by B.C)

b) RAM (Random Access Memory)

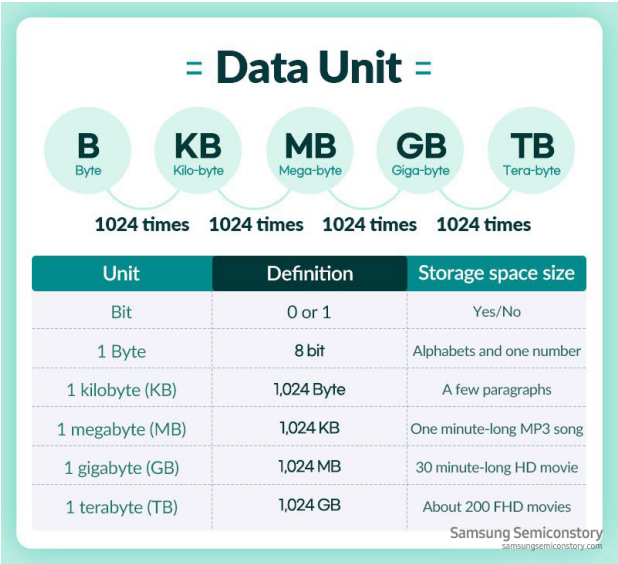
Definition

is a random access memory (in which the processor can read and write) and volatile (empty when the computer is turned off). it contains the programs and data being processed.



RAM (Random Access Memory) (By B.C)

The **capacity** of a memory can be measured in the number of **bytes** available such as:



Data unites(B.C)

1.2.3. Exercice

Exercice

Which of the following are functions of the CPU?

- ☐ a) To store data permanently.
- ☐ b) To seek the instructions of a program.
- ☐ c) To interpret instructions.
- ☐ d) To display information on the screen.

Exercice

Which of the following statements are TRUE about Primary Memory? (Select two)

- ☐ a) It has a slower access time than secondary memory.

- [ ] b) The CPU can access it directly.
- [ ] c) It is faster than cache memory.
- [ ] d) It has a larger storage capacity than secondary memory.

Exercice

The capacity of a memory is measured in:

- ○ a) Bits
- ○ b) Bytes
- ○ c) Hertz
- ○ d) Watts

Exercice

- ○ a) It is non-volatile.
- ○ b) It is read-only memory.
- ○ c) It is random access memory.
- ○ d) It stores information needed to start the computer.

Exercice

Which of the following statements is TRUE about ROM?

- ○ a) It is volatile memory.
- ○ b) It is random access memory.
- ○ c) It contains the programs and data being processed.
- ○ d) It is non-volatile memory that can only be read.

Exercice

Which of the following is considered Primary Memory?

- ○ a) Hard disk drives
- ○ b) Solid state drive
- ○ c) RAM
- ○ d) USB drive

1.3. Secondary or auxiliary memories:

Definition

Are storage media that are used to permanently store information (they keep the information even in the absence of electric current). Unlike main memory, secondary memories are slow. To do this, an existing program on the hard disk must be loaded into central memory to be executed.

Example

hard drive, floppy disk,The blank disk, and the CD-ROM... c)

1.4. The buses

Definition

A bus is a set of electrical lines allowing the transmission of signals (information)

between the various components of the computer

2. I.3.2. The software :(software):

The automatic processing of information or data by computer is based on tools called Software (or programs).

2.1. Computer program

Definition

A computer program is a list of commands telling a computer what it should do. It comes in the form of one or more sequences of instructions, which must be executed in a certain order by a processor.

Example

program that calculate an average, password checking program...

2.2. Software

Set of programs and data that cooperate with each other to provide a service to the user. Two types of Software are installed on a computer:

2.2.1. Basic software (operating system)

Definition

set of **programs** which manages the operation of the microcomputer with respect to its peripherals and which provides a "**bridge**" between the user and the physical machine

Example

MS-DOS, Windows, Mac-OS, Linux, etc.

2.2.2. Application software (application programs)

Definition

Programs that perform tasks that users expect from computers, these are programs developed usually by software companies (groups of engineers) or by users themselves (in the case of simple programs).

Example

- Office software: Word processing (Word), Spreadsheet (Excel)...
- Messaging and communication software via a network, the Internet.
- Programming software: Dev C++, Eclipse (for Java), Delphi, etc.

I.4. Computer languages

VI

In order to be able to communicate with a computer, developers designed several programming languages, without it, we will not be able to manipulate a computer or transmit instructions to it. There are several classifications of its languages, but the one we are interested in is the following:

1. I.4.1. Machine language

Definition

Machine language also called binary language, it is with this language that computers **work**. It consists of using two states (represented by the numbers **0 and 1**) to encode information (text, images, sound, etc.)

In machine language, the programmer must enter every command and **all data in binary form.**

Example

11001010 00010111 11110101 00101011 : (this is an instruction in machine language).

2. I.4.2. Assembly language

Definition

Assembly languages a low-level language close to machine language which can be directly interpreted by the computer's microprocessor while remaining readable by a human being.

The assembler was created to facilitate the work of programmers. It consists in representing the combinations of bits used in binary language by easy-to-remember symbols: each instruction expressed in machine language, the programmer coding his programs in assembly language, these are then transcribed by software called assembler in machine language, then executed by the computer.

Example

- **MOV AX, 5** (instruction1 in assembly language)
- **ADD DX, 1** (instruction2 in assembly language)

3. I.4.3. Evolved language (high level language)

Definition

We designate by evolved language all the languages being situated above the low level languages (machine language, assembler). The high level language is a language that accomplishes a lot for a minimum of code and programming effort, there is a whole package of them , we cite as an example: Pascal, Java, C, C++, C#, Visual Basic (or VB), Delphi, Python, Perl, PHP, JavaScript, VBscript, ASP ...etc.

These languages are called advanced because they hide the complexity of programming. Unlike assembly language which is very close to machine language, they offer layers that make the hardware abstract; to write Internet messaging software, you don't need to know the brand of the network card or the modem. Another thing the assembler does not use advanced structures such as loops (while, for) and conditions (if, switch), but it constitutes an integral part in the languages evolved. The syntax of advanced languages is very simplified, there are, for example, words in English (if, do while, switch, integer, string) so it is more accessible and understandable to people than the assembler itself witch is more accessible than machine language. In advanced languages, commands are entered using the keyboard, or from a program in memory. They are then intercepted by a "compiler" program, which translates them into machine language.

Example

- **X=Y+5;** (instruction in **C++** language)
- **X:=Y+5;** (instruction in **Pascal** language)

Exercice

VII

Exercice

Which of the following statements are TRUE about Machine Language?

- ☐ It is also called binary language.
- ☐ It is easy for humans to read and write.
- ☐ It uses two states (0 and 1) to encode information.
- ☐ Programmers use English-like words for commands.

Exercice

Which of the following are characteristics of Assembly Language?

- ☐ a) It is a low-level language.
- ☐ b) It uses binary code for instructions.
- ☐ c) It is close to machine language.
- ☐ d) It hides the complexity of hardware.

Exercice

Which of the following are characteristics of High-Level Languages?

- ☐ a) They hide the complexity of programming.
- ☐ b) They use advanced structures like loops and conditions.
- ☐ c) They are directly interpreted by the computer's microprocessor.
- ☐ d) They require the programmer to enter commands in binary form.

Exit Test (based on what learners are expected to master after the chapter 01)

VIII

Objectives

The objective of this assessment is to evaluate your understanding of the foundational concepts from Chapter 1. Based on your results, there are suggested resources available to help you recap and address any weak points, ensuring a stronger grasp of the material.

This 10-question assessment is designed to evaluate your understanding of the key concepts covered in Chapter 1. Your responses will help identify areas where further review or practice may be needed. Based on your performance, we have provided suggested resources to help you strengthen your knowledge and improve in specific areas.

Exercice : QUESTION 01

Q1. Quel est l'élément principal qui exécute les instructions dans un ordinateur ?

- a) Mémoire RAM
- b) Disque dur
- c) Processeur
- d) Carte graphique

Exercice : QUESTION 02

Q2. Que signifie le terme "bit" ?

- a) Une unité de mémoire équivalente à 8 octets
- b) Une opération mathématique

- c) Une unité binaire représentant 0 ou 1
- d) Une fonction logique

Exercice : QUESTION 03

Q3. Que donne l'expression : 5 - 2 * 3 en respectant la priorité des opérateurs ?

- a) 9
- b) 1
- c) -1
- d) 6

Exercice : QUESTION 04

What is the definition of "informatics" (informatique)?

- a) The study of human behavior
- b) The automatic processing of information using computers
- c) A type of programming language
- d) A communication system

Exercice : QUESTION 05

Which of the following is not a main component of a computer system?

- a) Hardware
- b) Software
- c) Network cable
- d) Central Processing Unit (CPU)

Exercice : QUESTION 06

What is the basic unit of binary information called?

- a) Byte
- b) Bit
- c) Block
- d) Code

Exercice : QUESTION 07

What is the base number system used by computers to process information?

- a) Base 10
- b) Base 8
- c) Base 16
- d) Base 2

Exercice : QUESTION 08

Which of the following is an example of secondary memory?

- a) RAM
- b) ROM
- c) Hard disk

Exercice : QUESTION 09

What is the role of the Control Unit (CU) in the processor?

- a) Interpret and manage instructions
- b) Store long-term data
- c) Execute arithmetic calculations
- d) Control the power supply

Exercice : QUESTION 10

Which of the following is not a main component of a computer system?

- a) Hardware
- b) Software
- c) Electricity
- d) Data

II.1 Definition of Algorithm

IX

Definition

The term "algorithm" comes from the name of the 9th-century Muslim Arab mathematician, Al?Khwarizmi (780-850), who was of Persian descent.

II.2. Historical Background

- The term "algorithm" is not limited to computer science, and the concept of an algorithm predates that of computer science.
- The earliest discovered algorithms describe methods of mathematical calculation.

Example

The **Euclidean algorithm**, which allows for the calculation of the **greatest** common divisor (**GCD**) of two integers a and b:

1. Divide a by b to obtain the remainder, **r.**
2. Replace a with **b.**
3. Replace b with **r.**
4. Continue as long as it is possible. Eventually, the **GCD** will be obtained.

Example:Introductory Example: Algorithm (Concrete Preparation).

1) Purchase the raw materials (Cement, crushed stone, sand) - Input

2) Pour two wheelbarrows of crushed stone onto one wheelbarrow of sand. (Step 1)

3) Pour one bag of cement onto the mixture and mix. (Step 2)

4) Pour 60 liters of water onto the mixture and mix. (Step 3)

5) This results in obtaining the concrete. – Output

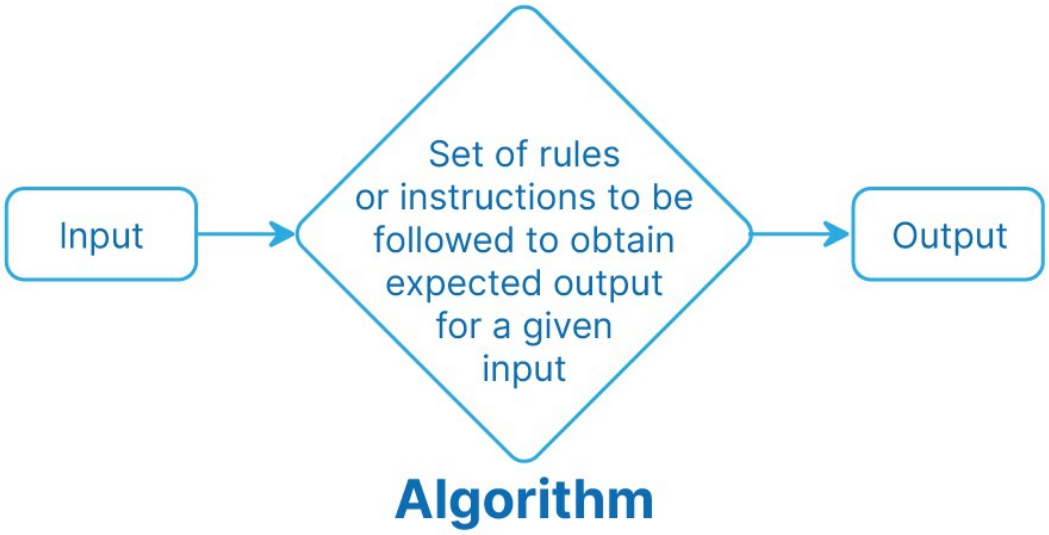
II.3 Fundamentals of Algorithms

XI

1. II.3.1 Definition of an Algorithm

Definition

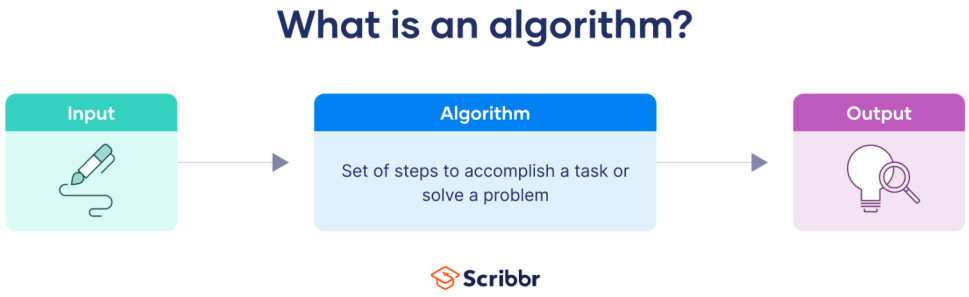
Basically, the word Algorithm means " A set of finite rules or instructions to be followed in calculations or other problem-solving operations, Therefore Algorithm refers to a sequence of finite steps to solve a particular problem



what is an algorithm?(b.ch)

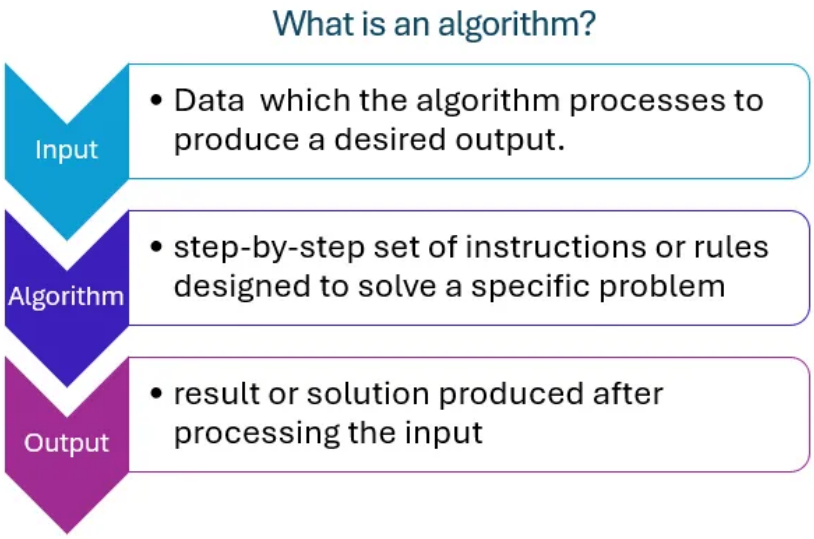
Definition

An algorithm is a finite and unambiguous sequence of operations or instructions that allows solving a problem.



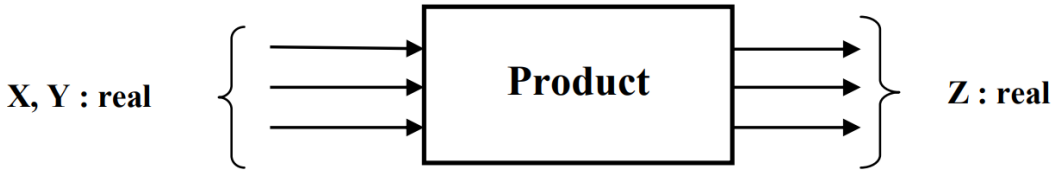
Definition

An algorithm is a method, which, in a finite amount of time, leads to a determined result from a given situation. The processing of information by a computer involves having this machine generate information, called results, from information called data.



Example

calculate the **product** of two real numbers **Z= X * Y** ?

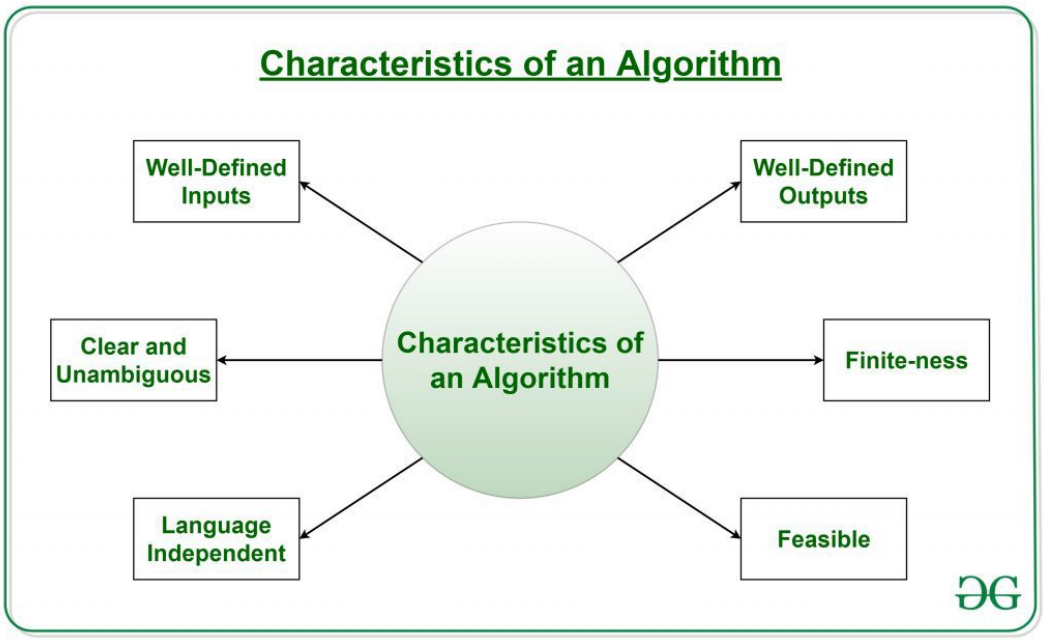


2. II.3.2 Characteristics of an Algorithm

An algorithm must have the following characteristics:

- **Clear and Unambiguous**: The algorithm should be unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
- **Well-Defined Inputs:** If an algorithm says to take inputs, it should be well-defined inputs. It may or may not take input.
- **Well-Defined Outputs**: The algorithm must clearly define what output will be yielded and it should be well-defined as well. It should produce at least 1 output.
- **Finiteness**: The algorithm must be finite, i.e. it should terminate after a finite time. Further, An algorithm must terminate after a finite number of steps in all test cases. Every instruction which contains a fundamental operator must be terminated within a finite amount of time. Infinite loops or recursive functions without base conditions do not possess finiteness.
- **Feasible:** The algorithm must be simple, generic, and practical, such that it can be executed with the available resources. It must not contain some future technology or anything.

- **Language Independent:** The Algorithm designed must be language-independent, i.e. it must be just plai instructions that can be implemented in any language, and yet the output will be the same, as expected.
- **Input:** An algorithm has zero or more inputs. Each that contains a fundamental operator must accept zerc or more inputs.
- **Output**: An algorithm produces at least one output. Every instruction that contains a fundamental operator must accept zero or more inputs.
- **Definiteness**: All instructions in an algorithm must be unambiguous, precise, and easy to interpret. By referring to any of the instructions in an algorithm one can clearly understand what is to be done. Every fundamental operator in instruction must be defined without any ambiguity.
- **Effectiveness**: An algorithm must be developed by using very basic, simple, and feasible operations so that one can trace it out by using just paper and pencil.



Complement:Properties of Algorithm

- It should terminate after a finite time.
- It should produce at least one output.
- It should take zero or more input.
- It should be deterministic means giving the same output for the same input case.
- Every step in the algorithm must be effective i.e. every step should do some work.

3. Exercice

Which of the following statements BEST describes the "Definiteness" characteristic of an algorithm?

- ☐ An algorithm must produce at least one output.
- ☐ Each step of the algorithm must be clear and lead to only one interpretation.
- ☐ An algorithm should be written in a programming language.
- ☐ All instructions in an algorithm must be unambiguous, precise, and easy to interpret.s

4. II.3.3 Algorithmic Language and Keywords

Definition: Algorithmic Language

This is the language used to describe and define algorithms. By using a set of keywords and structures that allow for a complete and clear description of all operations to be performed on data to obtain results.

Definition: Keywords

A keyword is an identifier that has a specific meaning (algorithm, start, end, if, then...).

Example

Algorithm Addition;

// variable declaration

num1, num2, sum: **integer**;

Begin

// Input

write('Enter the first number (num1): ');

read (num1);

write('Enter the second number (num2): ');

read (num2); //Performing Addition

Sum

num1 + num2;

// Output

write('The sum is: ', sum);

End.

5. II.3.4 Algorithmics and Programming

Definition: Algorithmics

Algorithmics is the science that studies algorithms.

Definition: Program

A computer program is a sequence of instructions written in a language understandable by a computer (programming language). In other words, a computer program is the translation of the algorithm into a programming language.

Definition: Programming Language

A set of commands and keywords necessary for writing a program so that it can be understood by the computer.

Example

Basic, Fortran, C, C++, Pascal, Delphi, Java...

II.4 Algorithm Structure and Execution

XII

1. II.4.1 General Structure of an Algorithms

An algorithm is composed of **three** parts: the **header**, the **declaration**, and the **body** of the algorithm.

ALGORITHM Algorithm-Name

Constants

Types

Variables

Functions

Procedures

BEGIN

Instruction 1;

Instruction 2;

...

Instruction n;

END

- The header: it simply allows for the identification (naming) of an algorithm.
- Declarations: this is a list of all objects (constants, variables, types, functions, and procedures) used and manipulated within the body of the algorithm.
- The body: The body contains the sequence of instructions to be executed (a set of operations to be performed on the data, i.e., variables)

Example

an algorithm that calculates the **product** of two integer numbers (First algorithm).

```
Algorithm
Multiply_Two_Numbers
```

header

```
Declare Integer a, b,
result
```

Declarations

```
Begin
    Print "Enter the first
number: "
    Read a

    Print "Enter the
second number: "
    Read b

    result ← a * b

    Print "The product is:
", result
End
```

body

2. II.4.2 Execution Trace of an Algorithms

- The execution of an algorithm (program) proceeds instruction by instruction in the order specifiedby the algorithm (program). Execution begins from the first instruction that follows the keyword **BEGIN** and ends at the last instruction just before the keyword **END**.
- - Initially, the values of variables are **unknown (?)**.
- - When a variable receives a value, that value is **stored** until another instruction **changes** it.

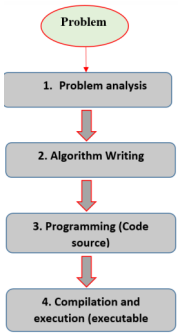
Step	a	b
Instruction1	10	?
Instruction2	10	15
Instruction3	3	15

```
Algorithm Trace
      a : integer;
      b : integer;
begin
     a←10 ;
     b← a+ 5 ;
     a← 3 ;
end.
```

II.5 Problem Solving with Algorithms

XIII

The creation of a program that can be executed by a computer requires following a process consisting of **four phases**. The role of the algorithm (Phase 2) is shown in the following figure:



1. II.5.1 Problem Analysis

Analysis involves:

- Extracting initial data (inputs),
- Defining the goal or goals of the problem (outputs or results),
- Deriving the method to follow to solve and achieve these objectives.

2. II.2.5.2 Writing Algorithms

Writing the set of corresponding algorithms.

3. II.5.3 Programming and Execution

Programming

In this step, the algorithm is translated into a programming language (C, C++, Pascal, Fortran, Java, etc.). This results in the source code of the program, which is stored in one or more files on the computer.

Compilation and Execution

- - In this step, the source code of the program is translated into machine language using special software called a compiler. This process generates the machine code of the program, which is then executed directly by the computer.
- - There are several software applications that include a compiler for the C language, such as Dev C++, Turbo C, Microsoft Visual C++ Express, and others.

Example:Problem

Create a program that **calculates** the electrical current "I" flowing in a closed

circuit powered by a source with a potential difference of "U=220 Volts" and containing a resistance

"R=12 Ohms."

Solution:

1) Problem Analysis:

- - Input Data: U=220 Volts, R=12 Ohms.
- - Results or Outputs (Objectives): Calculated current I.
- - Method: U=R*I, hence I=

2) Writing Algorithms: (This is where the algorithm for solving the problem would be written.)

Algorithm current-intensity-calculation

variables

U, R, I: integers;

Begin

U ← 220;

R ← 12;

I ← U / R;

Write (I);

End.

3) Programing : (see practical work)

Program in language *pascal*	**Program in langage *C***	**Program in langage *C++***
Program intensitycalculation ; Uses crt ; Var U,R,I : integer ; **Begin** U :=220 ; R :=12; I := U/R ; **End.**	#include <stdio.h> int U,R,I ; **void main()** **{** U = 220 ; R=12; I = U/R ; **}**	#include <iostream> using namespace std; int U,R,I ; **void main()** **{** U = 220 ; R=12; I = U/R ; **}**

4) Compiling the program : (voir Practical work)

Exercice

XIV

Exercice

The process of translating the source code of a program into machine language is called:

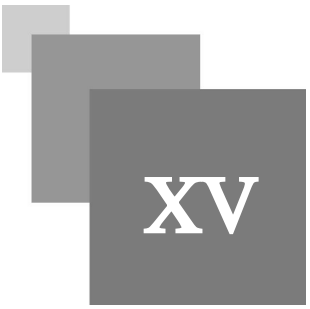
- a) Execution
- b) Programming
- c) Compilation
- d) Interpretation

Exercice

The step where an algorithm is translated into a programming language is called:

- a) Execution
- b) Programming
- c) Compilation
- d) Debugging

II.6 Writing Good Algorithms



- Clearly and unambiguously define the given problem.
- An algorithm must be well-structured.
- The result must be achieved in a finite number of steps, so there should be no infinite loops.
- Consider all possible data cases, ...
- The result must address the problem at hand. Please note that a set of test cases NEVER proves or guarantees that a program is correct. It can only demonstrate that it is incorrect.
- An algorithm must be commented. Comments are sentences that can be inserted at any level of the algorithm (program). They serve solely to provide explanations for the reader of the algorithm and are completely ignored by the executor. {comment}.
- If applicable, an algorithm should be modular, meaning it can be broken down into smaller parts.
- An algorithm should be efficient, considering criteria such as execution time and memory usage.

II.2.7 Variables and Data Types

XVI

1. II.7.1 Concept of a Variable

Definition

- The variables in an algorithm contain the information necessary for its execution.
- A variable in a program is a memory space identified by a name, intended to store a value that can be modified during processing (operations).
- Each variable is characterized by a name (identifier) and a type.

2. II.7.2 Naming and Declaration

The Identifier (Name)

To identify a variable, letters of the alphabet (a, b, c, A, ...) and digits (0, 1, 2, ...) can be used,

provided that the first character is an alphabet letter. The identifier can also contain an underscore "_".

Examples:

- X, y, name, price, X1 are **variable identifiers** (**names**).
- 1x is **not** an identifier since it starts with a **digit**.
- nom*, Met%un1 are **not** variable identifiers because they contain special characters.

The Type

The type corresponds to the kind of information you want to use:

- Integer for handling integers (1, 115, -7, ...),
- Real for handling real numbers (11.3, 15.7, -4.3, ...),
- Boolean for handling Boolean values (true or false),
- Character for handling alphabetic and numeric characters (a, b, 3, ...),
- String for handling strings of characters used to represent words or phrases (mila, omar, ...)

Note

- The identifier must be different from all keywords.
- For code readability, choose meaningful names: e.g., TotalSales2004, Price_Incl_Tax, Price_Excl_Tax, sum, ...

Variable Declaration

All variables must be declared before they are used.

Syntax: **Variable Name: Variable Type;**

Algorithm **exp_decl**

a: integer;

c: char;

Age: integer;

prénom: string;

x, y, z: real;

Begin

{Sequence of Instructions}

End.

Note:

Multiple variables of the **same type** can be declared on a single line by separating them with commas (for example: a**,** b**,** c : integer;)

3. II.7.3 Types of Variables

3.1. Integer Type (int)

The integer type is equipped with the following operators:

- Classical arithmetic operators: + (addition), - (subtraction), * (multiplication).
- Integer division, denoted as div, where n div p gives the integer part of the quotient of the integer divisior of n by p.
- Modulus, denoted as mod, where n mod p gives the remainder of the integer division of n by p.
- Classical comparison operators: <, >, =, ...

Examples:

- 7 div 2 = 3
- 7 mod 2 = 1
- sqrt(5) = 25
- abs(-17) = 17

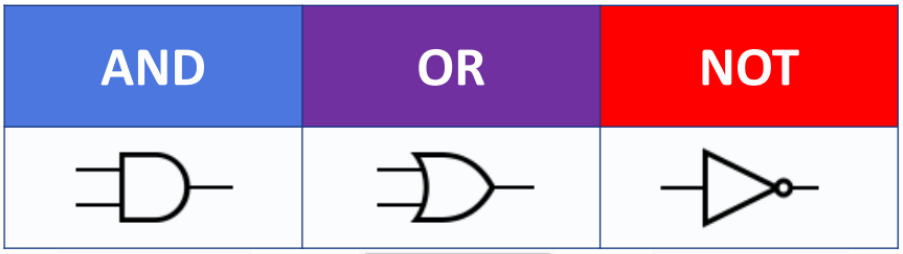
3.2. Real Type (float and double)

Valid operations on real numbers include:

- Classical arithmetic operations: + (addition), - (subtraction), * (multiplication), / (division).
- Classical comparison operators: <, >, =, ...
- The function that provides the square root (sqrt).

3.3. Boolean Type (bool in language C++)

This is a type with only two values: true or false. The logical operators (operations) on this type are **not, or, and** . These operators are defined by the following truth tables:



INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

INPUT		OUTPUT
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

INPUT	OUTPUT
A	NOT A
0	1
1	0

3.4. Character Type

This is a type consisting of alphabetic and numeric characters. A variable of this type can only contain a single character. Predefined operations on characters include:

- **Succ(c):** It provides the character that immediately follows the character c.
- **Pred(c)**: It provides the character that immediately precedes the character c

Example :

Succ('B') = 'C', Pred('z') = 'y'.

3.5. String Type

A string is a sequence of characters enclosed in double quotation marks. Example: "computer" , "course" , "mila" , "-1.6 "

Predefined operations on strings include:

- Comparisons: <, >, =, ... based on lexicographic order (ASCII).
- Concatenation represented by +: It provides the string obtained by concatenating two strings.

Note:

Strings are ordered based on lexicographic order.

Example :

'art' < 'course', 'course' < 'courses', 'courage' < 'courses'.

4. II.7.4 Constants

A constant is an algorithmic object that holds a **single value** throughout its execution.

Example:

Const int Max_value = 100; /*cpp; `Max_value` is a constant that is set to 100 and **cannot be changed** throughout the execution of the program */

Algorithm exp_cons

Const Pi ← 3,14 ;

Variable X : integer ; Y : real;

Begin

X ← 5 ; Y←X+Pi ;

Pi ← X+1 (false) /* Pi will always have the value 3.14.*/

end

Remarks:

- To declare a variable, you specify its name and type.
- To declare a constant, you must use the keyword CONST. You define its name and value.
- Always start with the declaration of constants before variables.

II.8 Expressions and Operators

XVII

1. II.8.1 What is an Expression?

Definition

An expression represents a combination of operands and operations performed on these operands.

The operands can be:

- Variables
- Values

The operators (operations) can be:

- Algebraic operators: +, -, *, /, div, mod.
- Logical operators: and, or, not.
- Relational operators: <, <=, >, >=, =, <>.

Every expression is associated with a type, which is the type of the value of that expression.

Example

"a + 8" is an expression representing an addition operation involving the operands "**a**" and "**8**".

Note:

An operator that operates on **two** operands is called a binary operator (e.g., +), while an operator that operates on a **single** operand is called unary (e.g., not).

2. II.8.2 Rules for Evaluating Expressions

The calculation of the value of an expression with more than one operator depends on the interpretation given to that expression.

Example

"5 - 4 div 2" is an ambiguous expression.

Parentheses can resolve this issue:

- (5 - 4) div 2 = 0.
- 5 - (4 div 2) = 3.

Note

In the absence of parentheses and to avoid ambiguity, evaluation rules have been established.

There is a descending order of precedence among operators as follows:

1) Unary operators: not.

2) Multiplicative operators: *, /, div, mod, and.

3) Additive operators: +, -, or.

4) Relational operators: <, <=, >, >=, <>.

Example

Therefore, the interpretation assigned to "5 - 4 div 2" is indeed "5 - (4 div 2) = 5 - 2 = 3".

If an expression contains operators of the same precedence, then the operators of the same precedence are left-associative.

Example

"5 - 3 - 2" is interpreted as "(5 - 3) - 2 = 2 - 2 = 0".

Note

In algorithmics, to avoid ambiguity, it is always advisable to use parentheses.

II.2.9 Algorithm Instructions

XVIII

An instruction represents one or more actions (operations) involving one or more variables. There are two types of elementary instructions:

- Assignment instruction
- Input/output instruction.

1. II.2.9.1 Assignment Instruction

Assignment is an instruction that stores the value of an expression in a variable.

Syntax of assignment: X ← exp; reads as: x receives exp.

Where:

- X: variable,
- ← : The assignment operator is the arrow
- exp: expression.

This replaces the initial value of x with the value of the expression.

Example

Algorithm exp_Aff

a, b : integer;

Begin

a ← 12 ;

b ←a+ 4 ;

End.

_____

Execution:

a=12 and b=16

2. II.2.9.2 Input/Output Instructions

Input Instruction

This instruction allows reading the value of a variable from the keyboard.

Syntax: **Read (variable);**

The execution of this instruction involves assigning a value to the variable by taking this value from the input device (keyboard).

Example

Suppose X is an **integer** variable, then:

Read(X); will assign an integer value typed from the keyboard to the **variable X.**

Output Instruction

This instruction allows displaying output on the screen.

There are two types of output:

- Displaying variable values: Syntax: Write (variable);
- Displaying text: Syntax: Write ('text');

Example

- Write (2 * x + 5): displays the value of the expression 2 * x + 5. If x is 10, then this instruction displays 25.
- Write ('The result is: '): displays the text "The result is: ".
- Write ('The result is: ', X): if x is 15, this instruction displays "The result is: 15."

3. II.2.9.3 Comments in Algorithms

To write the algorithm in a clear and logical manner, it is possible to add comments to algorithmic

actions (instructions).

By convention, a comment:

- Starts with a forward slash (/) followed by an asterisk * and ends with an asterisk * followed by a forward slash: /* multiple lines */
- Starts with a double slash for a comment that spans a single line: // Single line A comment is not an instruction. It has no effect on the execution of the algorithm.

Example

Write an algorithm that calculates the sum of two integers?

Algorithm exp_comt

a, b, som : integer ;

Begin

Lire (a) ;

Lire (b) ;

/* "This instruction allows for reading the values of variables a and b from the keyboard." ***/**

Som ← a+b ; **//** calculates the sum

write (som) ; **//** displays the result on the screen.

end

References

Exercice

XIX

Exercice

Which of the following statements are TRUE about instructions in algorithms? (Select two)

- ☐ a) An instruction represents one or more actions.
- ☐ b) Instructions do not involve variables
- ☐ c) Assignment and input/output are types of elementary instructions.
- ☐ d) Instructions are only used for mathematical calculation

Exercice

Which of the following statements are TRUE about assignment and output instructions?

- ☐ a) The assignment instruction stores the value of an expression in a variable.
- ☐ b) The assignment operator is typically represented by a plus sign (+).
- ☐ c) Output instructions are used to display information on the screen.
- ☐ d) The 'Read' instruction is used to display output.

Exercice

Which of the following statements are TRUE regarding operator precedence in algorithms?

- ☐ a) Parentheses are not necessary to avoid ambiguity in expressions.
- ☐ b) Unary operators have the lowest precedence.
- ☐ c) Multiplicative operators have higher precedence than additive operators.
- ☐ d) Parentheses can be used to override the default precedence rules.

Exercice

Which of the following statements are TRUE about variables in algorithms?

- ☐ a) A variable is a memory space identified by a name.
- ☐ b) A variable's value cannot be modified during processing.
- ☐ c) Each variable is characterized by a name (identifier) and a type.

- ☐ d) Variable identifiers can start with a digit.

Exercice

Which of the following statements are TRUE regarding operator precedence in algorithms?

- ☐ a) Parentheses are not necessary to avoid ambiguity in expressions.
- ☐ b) Unary operators have the lowest precedence.
- ☐ c) Multiplicative operators have higher precedence than additive operators.s
- ☐ d) Parentheses can be used to override the default precedence rules.

References

XX

1. References fo chapter 01

- **[1]** GeeksforGeeks, "What is Computer Science?," GeeksforGeeks. Available: https://www.geeksforgeek org/what-is-computer-science/, accessed Feb. 21, 2025.1.2 History of Computers
- **[2]** Computer History Museum, “Timeline of Computer History – Computers,” Computer History Museum. [Online]. Available: https://www.computerhistory.org/timeline/computers/. [Accessed: Feb. 14, 2025].
- **[3]** Carnegie Mellon University, “The World of the Internet - Handouts,” Gelfand Center for Service Learning and Outreach. [Online]. Available: https://www.cmu.edu/gelfand/lgc-educational-media /digital-education-modules/dem-documents/new-the-world-of-the-internet-handouts.pdf. [Accessed: Mar. 3, 2025].
- **[4]** MIT OpenCourseWare, "Introduction to Computer Architecture," MIT OCW. Available: https://ocw. mit.edu/computer-architecture/, accessed Feb. 21, 2025.1.5 Programming Languages
- **[5]** Coursera, "Introduction to Programming Languages," Coursera. Available: https://www.coursera.org /learn/programming-languages/, accessed Feb. 21, 2025.

2. References fo chapter 02

- **[1]** GeeksforGeeks, "Introduction to Algorithms," GeeksforGeeks. https://www.geeksforgeeks.org /introduction-to-algorithms/ (accessed Feb. 21, 2025).
- **[2]** EnjoyAlgorithms, "History of Algorithms," EnjoyAlgorithms. https://www.enjoyalgorithms.com/blog /history-of-algorithms/ (accessed Feb. 21, 2025).
- **[3]** The Hub, "Algorithms Shape Modern Computing. But Where Did They Come From?," The Hub. https://www.ttuhub.net/2024/01/algorithms-shape-modern-computing-but-where-did-they-come-from/ (accessed Feb. 21, 2025).
- **[4]** Pseudo-code - Glossaire MDN _ définitions des termes du Web _ MDN . https://developer.mozilla.or /fr/docs/Glossary/Pseudocode (accessed Feb. 21, 2025).
- **[5]** GeeksforGeeks, "Compiler Design - Introduction to Compilers," GeeksforGeeks. https://www. geeksforgeeks.org/compiler-design-introduction-to-compilers/ (accessed Feb. 21, 2025).
- **[6]** GeeksforGeeks, "Variables and Its Types in C and C++," GeeksforGeeks. https://www.geeksforgeeks org/variables-in-c/ (accessed Feb. 21, 2025).
  https://www.geeksforgeeks.org/cpp-variables/
- **[7]** GeeksforGeeks, "Data Types in C," GeeksforGeeks. https://www.geeksforgeeks.org/data-types-in-c/ (accessed Feb. 21, 2025).
- **[8]** What are C Expressions? https://www.scaler.com/topics/c-expression/ (accessed Feb. 21, 2025).
- **[9]** Comments in C. https://www.codechef.com/learn/course/c/LBCL01/problems/CCOMENT (accessed Feb. 21, 2025).

- **[10]** Introduction to Programming: Variables and Objects . https://www.purdue.edu/hla/sites/varalalab/wp content/uploads/sites/20/2018/02/Lecture_7.pdf(accessed Feb. 21, 2025).

Exit Test (based on what learners are expected to master after the chapter 02)

XXI

Exercice

01) What is an algorithm?

- A) A programming language
- B) A set of finite steps to solve a problem
- C) A database system

Exercice

02) Which of the following is NOT a characteristic of an algorithm?

- A) It must be finite
- B) It must have well-defined inputs and outputs
- C) It must be ambiguous

Exercice

03) What is the first step in executing an algorithm?

- A) Writing source code
- B) Identifying inputs and outputs
- C) Writing RESULTS

Exercice

04) What is the purpose of a variable in an algorithm?

- A) To store values

- B) To execute instructions
- C) To define functions

Exercice

06) What does a compiler do?

- A) Converts source code into machine code
- B) Runs the program
- C) Identifies variables

Exercice

07) What is the correct order of precedence in mathematical expressions?

- A) Addition before multiplication
- B) Parentheses, multiplication, addition
- C) Multiplication before parentheses

Exercice

08) Why are comments used in algorithms?s

- A) To explain the code
- B) To change variable values
- C) To execute instructions

Exercice

09) What makes an algorithm efficient?

- A) Minimal execution time
- B) More lines of code
- C) Complex logic

Exercice

10) Which of the following is the first step in the problem-solving process?

- a) Writing the program
- b) Analyzing the problem

- c) Debugging the solution