

# Algorithmic & Data Structure - 1

1

**M. BERRABAH Sid Ahmed**

berrabah.sidahmed@gmail.com

<https://sites.google.com/view/1inf-engineer-algo>

Department of computer science  
Faculty of Sciences  
University ABBT of Tlemcen

# Algorithmic & Data Structure - 1

2

## □ The objectives of the subject:

- The main objective of this course is to learn how to analyze a problem, describe it in algorithmic terms, choose the appropriate data structures, and master the inherent difficulties of programming.
- At the end of this course, the student must be able to: model and analyze a problem, program a solution, and validate it.

# Algorithmic & Data Structure - 1

3

## □ Content

- Chap - 1 : Introduction - Algorithmic & Programming
- Chap - 2 : Variables – Data Types – Operators
- Chap - 3 : Conditioned statement
- Chap - 4 : Loops
- Chap - 5 : Functions and Procedures
- Chap - 6 : Arrays
- Chap - 7 : Sorting Algorithms

**First Year Engineer in computer science**

# **CHAPTER 1 : Algorithms & Programming**

# Definitions

5

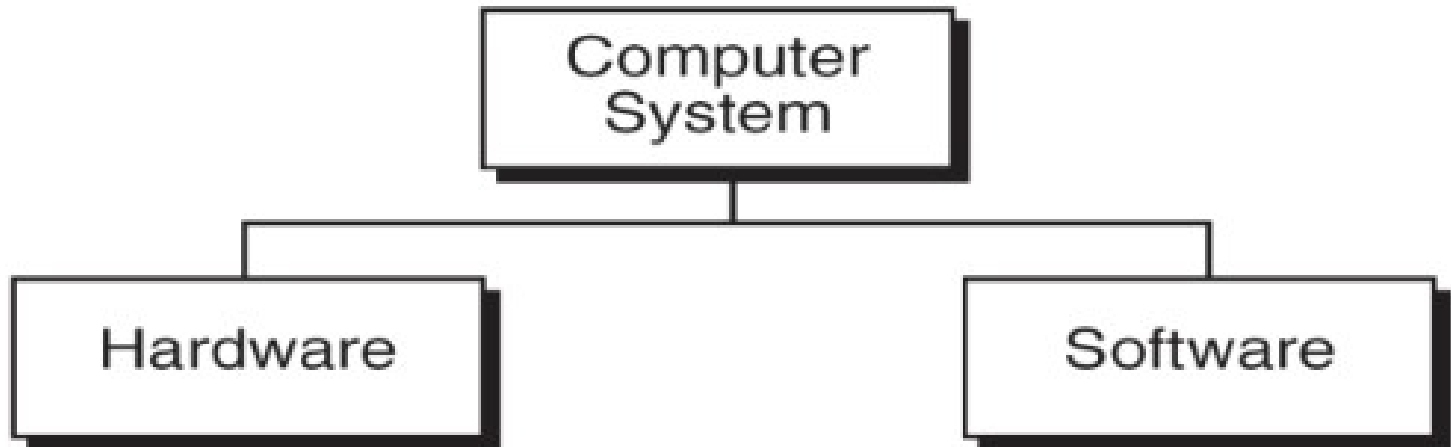
- **Computer Science** deals with automatic processing of data using a computer

# Computer System

6

A Computer is an electronic device which performs operations such as accepts data (as input), stores the data, manipulates or processes the data and produces the results (as output).

A computer system consists of hardware and software.



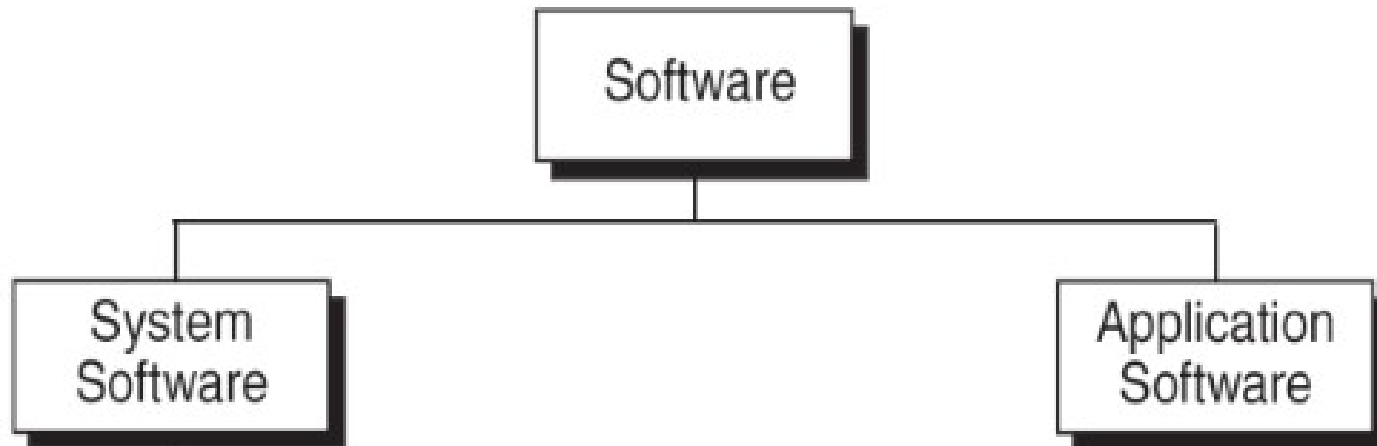
# Computer System

7

**Computer hardware** is the collection of physical elements that comprise a computer system.

**Computer software** is a collection of computer programs that provides the instructions for a computer what to do and how to do it.

Basically computer software is of two main types:



# Software

8

**System Software:** System software is responsible for managing a variety of independent hardware components, so that they can work together. Its purpose is to unburden the application software programmer from the often complex details of the particular computer being used, including such accessories as communications devices, printers, device readers, displays and keyboards, and also to partition the computer's resources such as memory and processor time in a safe and stable manner.

**Application Software:** Application software is developed to aid in any task that benefits from computation. It is a broad category, and encompasses Software of many kinds such as image editing, office toolbox, ...



# Solving a computational problem

9

The steps to solve a computational problem are:

## 1. Statement of Problem

Define clearly what inputs are available, what outputs are required and what is needed for creating workable solution.

## 2. Analysis and design

Identify the step-by-step method of solving a problem (algorithm).

## 3. Programming

Convert the developed algorithm into actual programs in a given programming language like C.

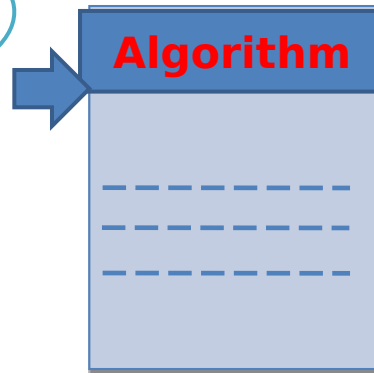
# Solving a computational problem

10

Step-by-step  
method of solving  
a problem



Language translating the imagined method  
in a way understandable to everyone :  
**Algorithm**



Language translating the algorithm  
in a way understandable for the  
computer. : **Program**

# Algorithm

11

- An **algorithm** is a procedure through which we obtain the solution of a problem. In other words, an algorithm is a step-by-step method of solving a problem.
- The domain that studies algorithms is called "**Algorithmic**"

# Algorithm

12

The name "**Algorithmic**" is derived from the name of the mathematician Muḥammad ibn Mūsā al-Khwārizmī, born in the 780s and died around 850.



# Properties of an Algorithm

13

1. **Finiteness**: An algorithm terminates after a finite numbers of steps.
2. **Definiteness**: Each step in algorithm is unambiguous. This means that the action specified by the step cannot be interpreted (explain the meaning of) in multiple ways & can be performed without any confusion.
3. **Input**: An algorithm accepts zero or more inputs
4. **Output**: An algorithm should produce at least one output.
5. **Effectiveness**: It consists of basic instructions that are realizable. This means that the instructions can be performed by using the given inputs in a finite amount of time.

# Representation of Algorithm

14

Two ways can be used to represent an algorithm:

**1. Pseudo-code**

**2. Flowchart**

# Representation of Algorithm

15

**1. Pseudo-code :** An algorithm can be written in human language, like sentences and using mathematical formulas.

# Representation of Algorithm

16

**1. Pseudo-code :** An algorithm can be written in human language, like sentences and using mathematical formulas.

**Example :** Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.



# Representation of Algorithm

17

**1. Pseudo-code :** An algorithm can be written in human language, like sentences and using mathematical formulas.

**Example :** Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

**Solution :**

# Representation of Algorithm

18

**1. Pseudo-code :** An algorithm can be written in human language, like sentences and using mathematical formulas.

**Example :** Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

**Solution :**

**1.** Input 4 marks: M1,M2,M3,M4

# Representation of Algorithm

19

**1. Pseudo-code :** An algorithm can be written in human language, like sentences and using mathematical formulas.

**Example :** Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

**Solution :**

1. Input 4 marks: M1,M2,M3,M4
2. Calculate  $\text{GRADE} \leftarrow (M1+M2+M3+M4)/4$

# Representation of Algorithm

20

**1. Pseudo-code :** An algorithm can be written in human language, like sentences and using mathematical formulas.

**Example :** Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

**Solution :**

1. Input 4 marks: M1,M2,M3,M4
2. Calculate  $\text{GRADE} \leftarrow (M1+M2+M3+M4)/4$
3. if (GRADE < 10) then Print "FAIL"  
    else Print "PASS"

# Representation of Algorithm

21

**2. Flowchart:** a flowchart is a graphical representation of an algorithm.

# Representation of Algorithm


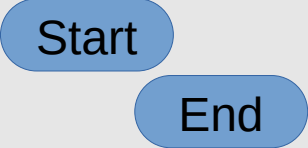
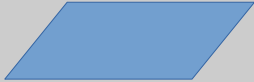
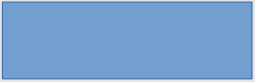


22

**2. Flowchart:** a flowchart is a graphical representation of an algorithm.

As different symbols are used to specify the type of operation performed, it is easier to understand an algorithm represented using a flowchart.

# Flowchart symbols

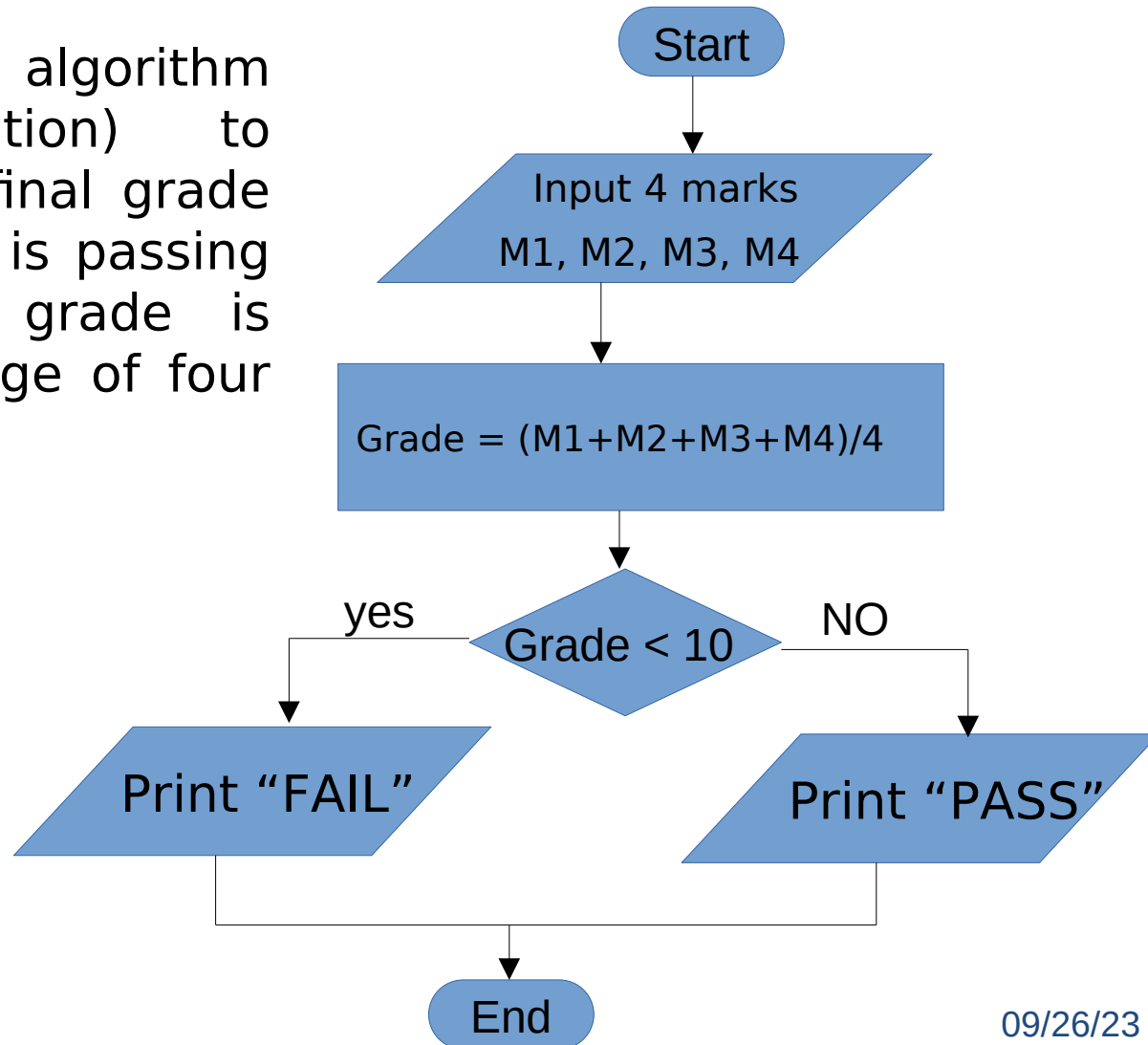
23

Symbol	representation	description
Arrows (flow-lines)		To show the flow of control through the program.
Terminal Symbols		Indicate the start and the end of a flowchart
Input / Output		Parallelogram shaped symbol is used for both input (Read) and Output (Write)
Process Symbol		Rectangle shaped symbol indicates a specific processing (calculation or initialisation)
Decision symbol		The diamond shaped symbol is used for decision making. It has two exits: one is labeled YES and other labeled NO.
Connectors		Whenever a complex flowchart is more than one page, in such a situation, the connector symbols are used to connect the flowchart.

# Flowchart example

24

**Example :** Write an algorithm (flowchart representation) to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

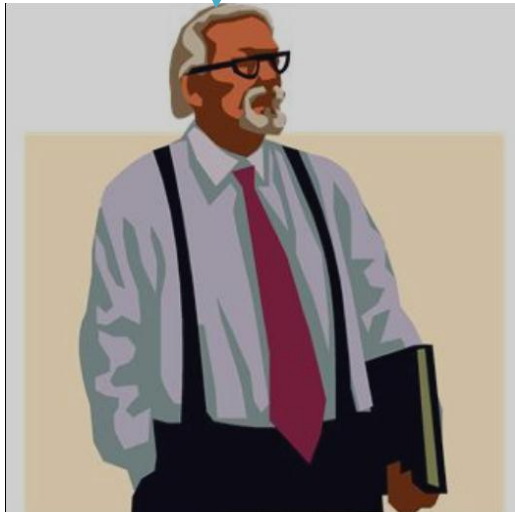




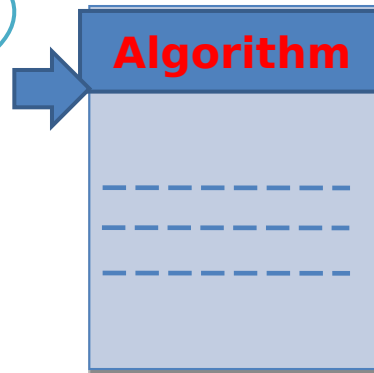
# Solving a computational problem

25

Step-by-step  
method of solving  
a problem



Language translating the imagined method  
in a way understandable to everyone :  
**Algorithm**



Language translating the algorithm  
in a way understandable for the  
computer. : **Program**

# PROGRAMMING

26

## **Definition :**

- A program is a sequential set of basic instructions translating an algorithm into programming language.
- Each of these instructions specifies the operation to be executed by the computer.

# PROGRAMMING LANGUAGE

27

## Definition :

- **The programming language** is the intermediary between humans and the machine. It allows writing operations that the computer must perform in a language close to the machine but understandable by humans.
- **A programming language** is a language comprising a set of characters, symbols, and words governed by rules that allow them to be assembled and used to provide instructions to a computer.
- There are several **programming languages**, most of them being reserved for specialized domains. Examples include Fortran, C, C++, Java, Pascal, Python, and more.

# PROGRAMMING LANGUAGE

28

- **QUESTION** : Since the computer only understands the binary language (machine language), how will it execute a program written in a high-level programming language?

# PROGRAMMING LANGUAGE

29

- **QUESTION** : Since the computer only understands the binary language (machine language), how will it execute a program written in a high-level programming language
- **RESPONSE** : Indeed, while the computer can only comprehend the binary language consisting of 0s and 1s, it needs an intermediary to translate a program written in a high-level programming language (human-readable language) into machine-understandable language.

# PROGRAMMING LANGUAGE

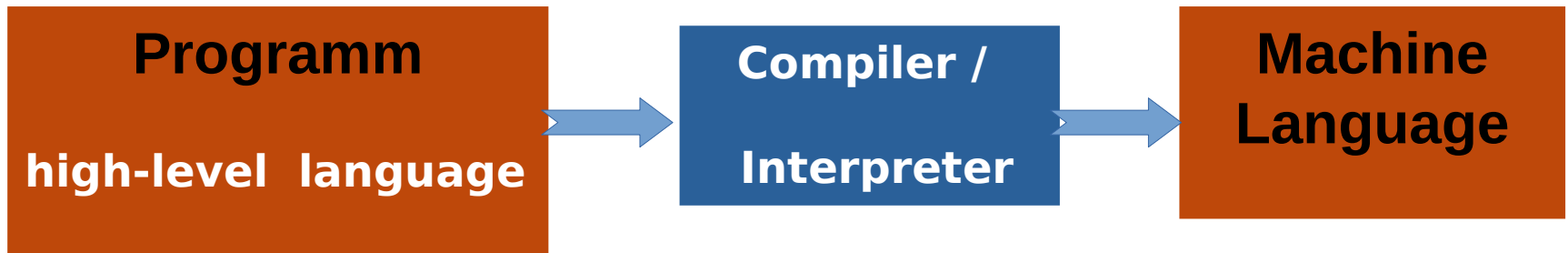
30

- **QUESTION** : Since the computer only understands the binary language (machine language), how will it execute a program written in a high-level programming language?
- **RESPONSE** : Indeed, while the computer can only comprehend the binary language consisting of 0s and 1s, it needs an intermediary to translate a program written in a high-level programming language (human-readable language) into machine-understandable language.
- There are two main approaches to achieve this: Compiler or Interpreter.

# PROGRAMMING LANGUAGE

31

- **QUESTION** : Since the computer only understands the binary language (machine language), how will it execute a program written in a high-level programming language?
- **RESPONSE** : Indeed, while the computer can only comprehend the binary language consisting of 0s and 1s, it needs an intermediary to translate a program written in a high-level programming language (human-readable language) into machine-understandable language.
- There are two main approaches to achieve this: Compiler or Interpreter.



# PROGRAMMING LANGUAGE

32

**Compiler:** A compiler is software that takes the source code written in a high-level programming language and translates it entirely into machine code (binary language) all at once. The resulting binary code can be directly executed by the computer's processor.



# PROGRAMMING LANGUAGE

33

**Compiler:** A compiler is software that takes the source code written in a high-level programming language and translates it entirely into machine code (binary language) all at once. The resulting binary code can be directly executed by the computer's processor.

**Interpreter:** An interpreter is another type of software that reads and executes the source code line by line. It translates each line into machine code on-the-fly and immediately executes it. The interpreter continues this process until the entire program is executed.

# PROGRAMMING LANGUAGE

34

**Compiler:** A compiler is software that takes the source code written in a high-level programming language and translates it entirely into machine code (binary language) all at once. The resulting binary code can be directly executed by the computer's processor.

**Interpreter:** An interpreter is another type of software that reads and executes the source code line by line. It translates each line into machine code on-the-fly and immediately executes it. The interpreter continues this process until the entire program is executed.

In both cases, the goal is to convert the high-level programming instructions into machine code so that the computer can perform the specified operations as intended by the programmer.

# C Programming Language

35

## Brief History of C

- The C programming language was developed in 1972 by Dennis Ritchie at bell laboratories of AT&T (American Telephone & Telegraph), located in the U.S.A..
- It was initially designed for programming UNIX operating system. Now the software tool as well as the C compiler is written in C.
- In 1978, Dennis Ritchie and Brian Kernighan published the first edition “The C Programming Language” and is commonly known as K&R C.
- In 1983, the American National Standards Institute (ANSI) established a committee to provide a modern, comprehensive definition of C. The resulting definition, the ANSI standard, or “ANSI C”, was completed late 1988.

# C Programming Language

36

## Why is C popular

- C is reliable, simple and easy to use.
- C is a block-structured programming language.
- C is a portable language, which means that C programs written on one system can be run on other systems with no modification.
- C has one of the largest assortments of operators, such as those used for calculations and data comparisons.
- C programs are fast and efficient. This is because C uses a powerful set of data types and operators. Major parts of popular operating systems like Windows, UNIX, Linux are still written in C.

# What do I need to program in C

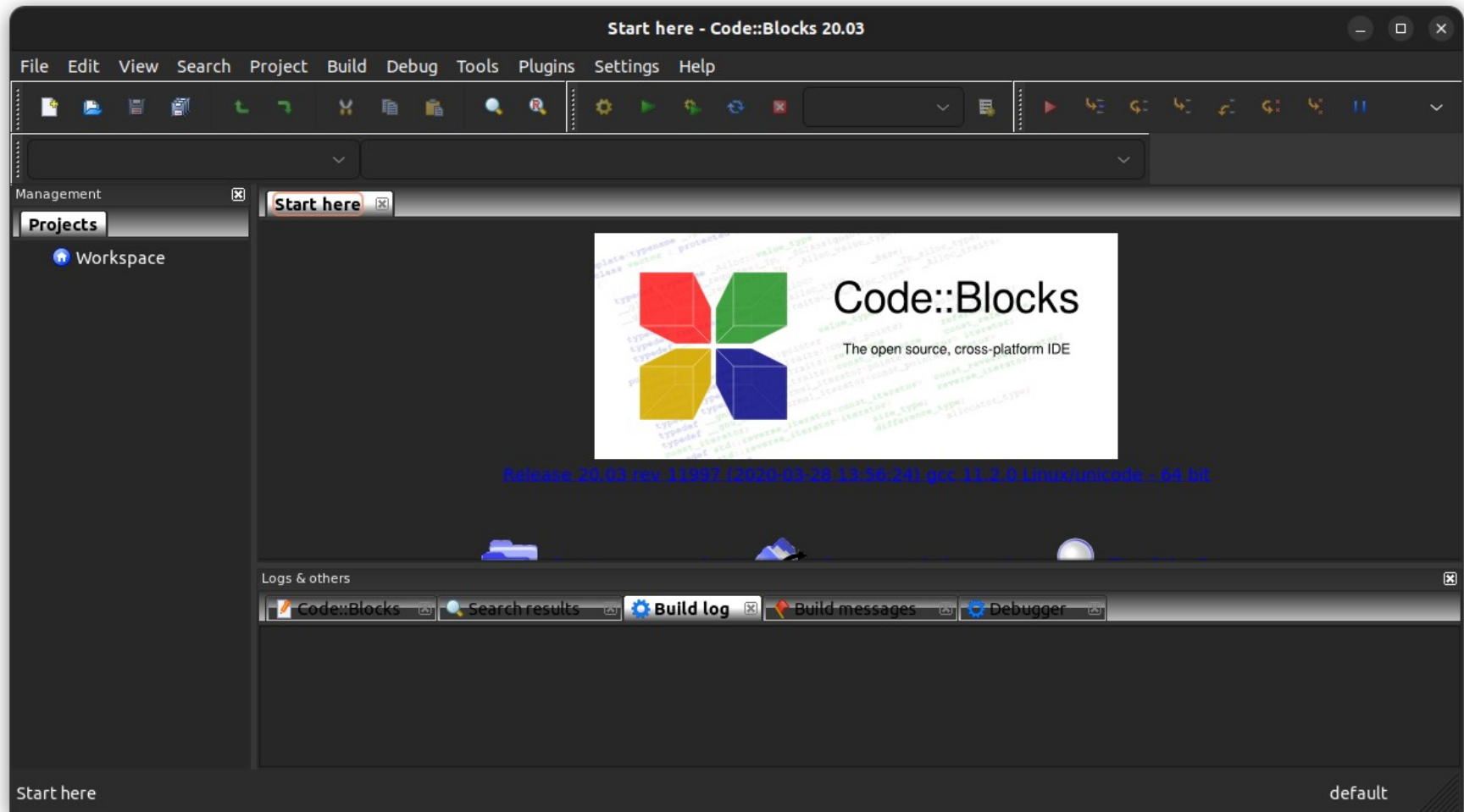
37

- To program in C, we need at least to 2 (or 3) tools: A text editor; a compiler; (and optionally a debugger).
- It's possible to install separately these tools, but it exist also a 3-in-1 packages called IDE (Integrated Development Environment).
- The most used IDEs for C programming are: **CodeBlocks**, Visual Studio, Clion, Eclipse, CodeLine, Xcode
- Under ANDROID one can uses: CXXDroid, CppDroid, Termux

# What do I need to program in C

38

## Code::Blocks



09/26/23

# First program in C

39

## Example of a C program

```
/*  
    This is my first C program  
    It prints the message "Hello world"  
*/  
  
#include <stdio.h>  
#include <stdlib.h>  
  
int main()    // here start the main function  
{  
    printf("Hello world!\n");    // instruction to print the message "Hello world"  
    return 0;  
}    // end of the program
```

# Comments

40

- **Comments** are added to give additional information (explanation) in the program.
- **Comments** are ignored by the compiler.
- One line comments can be added after **//** symbol
- Multiple lines comments should be enclosed within the decimeters (delimiters) **/\* ... \*/**



# Preprocessor **#include**

41

- **#include** is a pre-processor directive. It is an instruction to ask the C compiler to include the contents of a file (in this case the system file called **stdio.h**).
- The **stdio.h** (standard input output header file) contains definition & declaration of system defined function such as `printf( )` (function used to display).

# Main function

42

- The **main()** function is the entry point of any C program. It is the point from where the execution of a program is started.
- The working parts of the function main are enclosed within braces (or curly brackets) .

```
{  
    ..... /* working parts */  
}
```

# Statement

43

- A statement is a complete instruction to the computer.
- A program is a series of statements. In C, statements are indicated by a semicolon at the end.

**Ex:** `printf("Hello world! \n");` is a statement;

- A compound statement is two or more statements grouped together by enclosing them in braces; it is also called a block.