PROGRAMMING IN C-LANGUAGE

COURSE CONTENTS

- Chapter 1: INTRODUCTION TO PROGRAMMING
- Chapter 2: DATA TYPES
- Chapter 3: EXPRESSIONS AND OPERATORS
- Chapter 4: DECISION CONTROL STATEMENT
- Chapter 5: LOOPING
- Chapter 6: ARRAYS
- Chapter 7: FUNCTIONS
- Chapter 8: STRUCTURES AND POINTERS

UNIVERSITY of KIGALI

CHAPTER ONE: INTRODUCTION TO PROGRAMMING

1.1 INTRODUCTION:

First of all we should remind ourselves the meaning of a language, such as English, French or any other and what is a computer language in general.

A language is a method of human communication, consisting of words used in a structured and conventional way. **Computer language** is just a set of instruction or set of programs or predefined application developed to make an application and to understand its related software, using computer language people can communicate with different electronic devices.

UNIVERSITY of KIGALI

C is a successor of B language, which was introduced around 1970.

- The language was formalized in 1988 by the American National Standard Institute. (ANSI).
- The UNIX OS was totally written in C by 1973.
- Today, C is the most widely used and popular System Programming Language.
- Most of the state-of-the-art softwares have been implemented using C.
- Today's most popular Linux OS and RBDMS MySQL have been written in C.
- Useful for all applications,
- Easy to interface with system devices / assembler routines.

UNIVERSITY OF KUCAK

☐ Supports modular programming.

Why to use C?

C was initially used for system development work, in particular the programs that make up the operating system. C was adopted as a system development language because it produces code that runs nearly as fast as code written in assembly language.

PROGRAMMING LANGUAGE DESCRIPTION

A programming language is a language that a computer can understand and provides for programmer an environment to write and execute programs in it. A number of programming languages exist but the choice mainly depends on the nature of the problem at hand and the programmer's ability to use the language.

UNIVERSITY Of KIGALI

- There has been a great revolution in programming languages right from the time when programming started.
- The first generation of programming languages involved the use of machine language.
- The second involved the use of assembly language.
- The third generation involved high level programming languages like COBOL, PASCAL, FORTAN, C, C++, Java etc,
- The fourth generation involved languages mainly used for database manipulation like Structured Query Language (SQL).
- The fifth generation involved languages mainly used for artificial intelligence.
- **a computer program** can be defined as a set of step-by-step computer instructions that are designed to tell a computer what and how to accomplish a task or achieve particular expected results.

- **Computer programming** (often shortened to **programming** or **coding**) is the process of **designing, writing, testing, debugging** , and **maintaining** the source code of computer programs.
- A computer program's **source code** is the collection of files needed to **convert** from human-readable form to some kind of computer-executable form. The **source code** may be converted into an executable **file (object code)** by a compiler.
- A **programming language** is an artificial language designed to express computations that can be performed by a machine, particularly a computer.
- Programming languages can be used to create programmes that control the behavior of a machine, to express algorithms precisely, or as a mode of human communication.

PROGRAMMING LANGUAGES

categories :

- **LOW LEVEL LANGUAGE**: is a programming language that is machine dependent, it runs on only one particular type of computer.

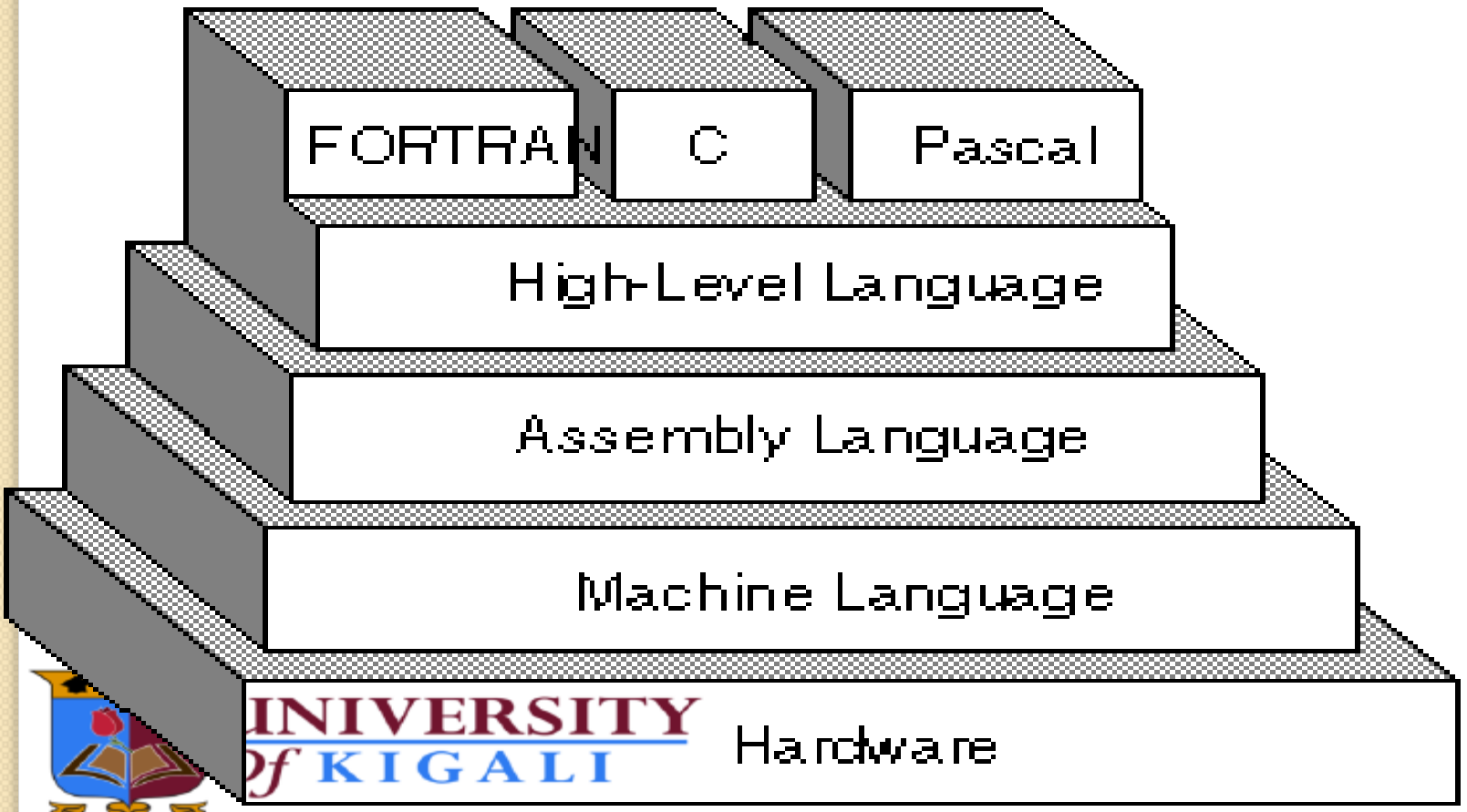
There are 2 types:

-Machine language: known as the first generation of programming languages, is the only language the computer recognizes.

Machine language is a collection of very detailed instructions that are meant to control the internal circuit (hardware) of a particular computer.

Machine language instructions uses a series of binary (0's and 1's)

-**Assembly language:** the second generation of programming languages, a programmer writes instructions using symbolic codes, these are meaningful abbreviations and codes.



- HIGH-LEVEL LANGUAGE

- A programming language such as C, FORTRAN, or Pascal that enables a programmer to write programs that are more or less independent of a particular type of computer.
- Such languages are considered high-level because they **are closer to human languages** and further from machine languages. In contrast, assembly languages are considered low-level because they are very close to machine languages.

How C is applied in our daily basis:

- It is used in making computer games.
- Used for developing an operating systems.
- Used in commercial company for robot control mechanism using micro controller system.
- Used to control traffic light system. Using loop structures.
- Used in motor system control for increase the speed and velocity of a motor.

1.3 Advantages of high level programming

The main advantage of high-level languages over low-level languages is that they are easier to read, write, and maintain.

Programs written in a high-level language must be translated into machine language by a compiler or interpreter.

- **C is portable**
- **C is Powerful and Flexible**
- **C has few keywords**
- **Wide acceptability (Popularity)**

UNIVERSITY of KIGALI

C is portable

C program written for one computer system (an IBM PC, for example) can be compiled and run on another system with a little or no modification,

Machine language or assembly language varies from computer to computer and hence, program written in these languages are not portable.

NB:

means C-language platform dependency where C application will run only on the same specific platform, because such device will convert the source code depending on O/S used while developing it.

C is Powerful and Flexible

- What you can accomplish is limited only by your imagination. The language itself places no constraints on you.
- C is used for project as diverse as operating system, word processors, spreadsheet and even compiler of other language.
- C combines the convenience and portable nature of a high-level language with the flexibility of low-level language.

C has few keywords

The following names are reserved by the C language. Their meaning is already defined, and they cannot be re-defined to mean anything else.

auto	break	case	char	continue	default
do	double	else	enum	extern	float
for	goto	if	int	long	register
return	short	sizeof	static	struct	switch
typedef	union	unsigned	void	volatile	while

Wide acceptability (Popularity)

- C is a language known by the majority of programmers around the world.
- C is a popular language preferred by professional programmers, it has a powerful compiler.

1.4 COMPILATION TERMINOLOGY

- A program that is written in high-level language, must be translated into machine language before it can be executed.

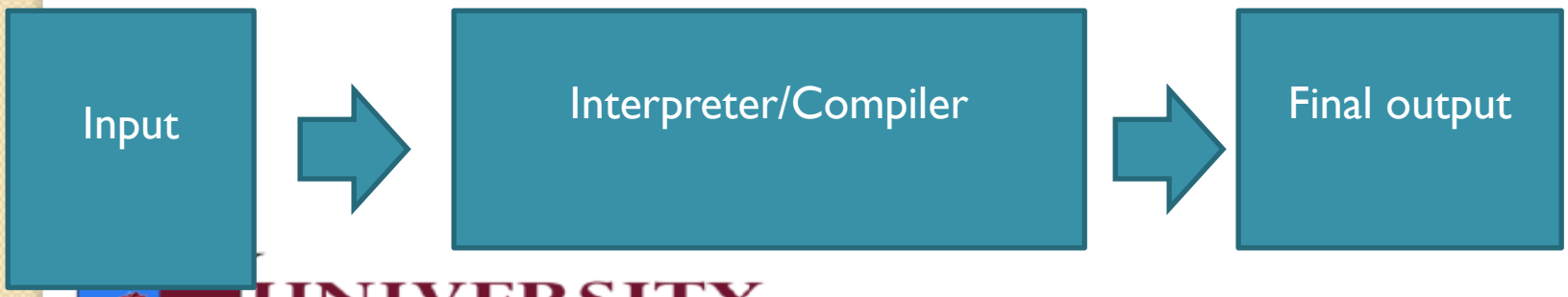
Compilers

- The **compiler** translates the entire program (written in high-level language) into machine language before executing any set of the instructions.
- A compiler reads the entire program and converts it to the **object code**. It provides errors not of one line but errors of the entire program. Only errors free programs are executed.
- It consumes little time for converting a **source program** to an **object program**. When the program length for any application is large, compilers are preferred.

Interpreters

An interpreter reads only **one line of a source code** at a time and converts it to object codes.

- In case of any errors, the same will be indicated instantly.
- The program written with an interpreter can easily be read and understood by the other users as well.



UNIVERSITY Of KIGALI

COMPARISON BETWEEN A COMPILER AND AN INTERPRETER:

- **Interpreter**
- It reads only one line of a source program at a time and converts it to object code.
- It is easier.
- It consumes more time for converting a source program to an object program.
- **Compiler**
- It reads the entire program and converts it to the object code.
- It is not easier.
- It consumes little time for converting a source program to an object program.

1.5 STAGES IN PROGRAM DEVELOPMENT:

- **Define or understand the problem**: First of all, you must understand a problem, if you don't know the problem you can't find a solution as desired output.
- **Analyze the problem:** Once you know what the problem is, you can analyze it and make a plan to resolve it by thinking about the input data.
- **Develop an algorithm and flowchart:** This is a process whereby a set of instructions are used to produce a solution to a given problem.
- **Writing the code:** This is the next step where you write the codes for the program to make it work.
- **Compiling and debugging the code:** Once the program coding is completed, you compile your program means you translate the source code to object code and if there are errors, you debug them. **Debugging** means to correct or remove errors (bugs) if there are in a program.
- **Run the program:** To run an application means to execute it and check, if using some data, it is working with the correctness of the program.

DEBUGGING THE COMPUTER PROGRAM

- **Errors** are mistakes which we the programmers make.
- Programming errors are called bugs.
- the process of tracking them down and correcting them if they are in a computer program is therefore, called Debugging.
- **Programming errors come in 3 categories**:
- Syntax (compile error) error.
- Run-time error.
- Logic errors.

CONT,…

- *a)* ***Compile (syntax) error:*** *you get it when you've broken the rules of computer* programming language.

  e.g: spelling printf as prinntf or printif
- You also receive a compiler error, if accidentally use the wrong punctuation or place a punctuation in the wrong place.
- *b)* ***Run-time error:*** *it can be caused by attempting to do impossible arithmetic* operations, such calculating non-numeric data, dividing a number by zero, or find the square root of a negative number.
- *c)* ***Logic error:*** *with this, your application runs but produces incorrect result. Perhaps* the results of calculation are incorrect or the wrong text appears, or the text is ok but appears in the wrong location.

UNIVERSITY
Of KIGALI

1.6 FLOWCHART

- A **flowchart**: A flow chart is a graphical or symbolic representation of a process.
- Each step in the process is represented by a different symbol and contains a short description of the process step.
- The flow chart symbols are linked together with arrows showing the process flow direction.
- And is also a graphical representation of an algorithm.

1.7 Program Development Cycle

1. Editing(or writing) the program

   Eg: Dev C++ (it is running under graphic interface)

   Turbo C++ (it is running under command line)

2. Compiling it

(we should expect zero error to execute our program)

3. Linking it (connects source code with object code)

4. Executing it (generates the output/results)

Step 1

- Use an **editor to write** your source code(the program).
- By tradition C source code files have the extension .c
- That means that you have to save the source code with .c extension. ex: Hello.c

Step 2

- **Compile** the program using a compiler.
- If the compiler doesn't find any errors in the program, it produces an object file.
- The compiler produces object files with an .obj extension and the same name as the source code file(for example Hello.c to Hello.obj).
- If the compiler finds errors, it responds(reports) them, you must return to step 1 to make the corrections in your source code

Step 3

- **Link** the program with a Linker.
- If no errors occur, the Linker produces an executable program with an .exe extension and the same name as the object file(for example Hello.obj is linked to create Hello.exe).

Step 4

- **Execute** the program to see the results.
- You should test to determine whether it functions properly, if no start again with step I and make modification and addition to your source code.

1.8 STRUCTURE OF A C PROGRAM

- Include header file section
- Global Declaration Section
- /* comments */
- main() Function name
- {
- /* comments*/
- Declaration part
- Executable part
- }
- User-defined functions
- {
- }

- **STRUCTURE OF A C PROGRAM** consist of the following key parts:

1. Include header file section.
2. Global declaration.
3. Function main.
4. Declaration part.
5. Executable part.
6. User- defined function.
7. Comments.

1. Include header file section: (Link Section)

- C program depends upon some **header files** for function definition that are used in program. Each **header file** by default is extended with **.h**. The file should be included using **# include directive** as given below.
- For example if you want to use some mathematical function then you have to define link for math.h file in link section. For Example
- # include<stdio.h>
- # include<math.h>

UNIVERSITY Of KIGALI

- **Global declaration**: This section declares some variables that are used in more than one function. These variables are known as global variables. This section must be declared outside of all the functions.
- **Function main**: Every program written in C language must contain main () function. Empty parentheses after main are necessary. **The function main () is a starting point of every 'C' program.** The execution of a program always begins with the function **main ().** (A function is a self-contained block of statements that perform coherent task).
- **The program execution starts from the opening brace ({) and ends with the closing brace (}).** Between these two braces the program should declare the declaration and the executable part.

- **Declaration part: The declaration part declares the entire variables that are used in executable part.**
- The initialization of variables are also done in this section.
- The initialization means providing initial values to the variables.
- **Executable part**: This part contains the statements following the declaration of the variables.
- This part contains a set of statements or a single statement. These statements are enclosed between the braces. Eg. Printf("….."), scanf("…")

- **User-defined function:** The functions defined by the user are called **user-defined functions**. These functions are generally defined after the **main ()** function. They can also be defined before **main ()** function.
- **Comments:** Comments are not necessary in the program. However, to understand the flow of programs the programmer can include comments in the program. **Comments are used for documentation**.
- Comments are nothing but some kind of statements which are placed between delimiters /* and */.
- The compiler does not execute comments.

1.9 PROGRAMMING RULES

A programmer while writing a program should follow the following rules:

1. All statements should be written in **lower case letters**. Upper case letters are only used for **symbolic constants**.
2. **Blank spaces** may be inserted between words to improve the readability of statements.
3. It is not necessary to fix the position of **statement** in the program

I.e. the programmer can write the statement anywhere between the **two braces** following the declaration part. The user can also write one or more statements in one line separating them with a semicolon (;).

The following statements are valid.

```
a=b+c;      d=b*c;
```

Or `a=b+c;  d=b*c;`

4. The opening and closing braces should be balanced i.e. for example, if opening braces are **four**; then closing braces should also be **four.**
5. Every **C** statement should be ended with **Semi-colon (;)**

The Simplest C Program

```
#include <stdio.h>
 main()
{
    printf("WELCOME TO UNIVERSITY OF KIGALI!\n");
}
```

Program Output:

WELCOME TO UNIVERSITY OF KIGALI!

Second program

```
#include <stdio.h>
main()
{
    printf("Hello,\n software\n development\n");
}
```

Output

Hello
Software
development

Other special notation

- \a "bell" – i.e a beep
- \b Backspace
- \f Form Feed (new page)
- \n New Line (line feed)
- \t Tab
- \v Vertical Tab
- \\ Backslash
- \' Single Quote
- \" Double Quote
- \? Question Mark

Comment example

```
#include<stdio.h>
main() /* main function heading */
{
    printf("\n Hello,World! \n");
  /*  Display message on */
  /*  the screen  * /
}
```

UNIVERSITY of KIGALI