Fpt University

TRƯỜNG ĐẠI HỌC FPT

Introduction to Programming Fundamentals using C

Objectives

After studying this section, you should be able to:

- Define some concepts related to programming
- Explain how to make a good software
- Understand steps to develop a software
- Answer why C is the first language selected
- Understand how a C program can be translated and execute
- Understand a C program structure

Contents

- Definitions
- How to make a good software?
- Steps to develop a software?
- Overview Computer hardware
- Data Units
- Addressing Information
- Program Instructions
- Programming Languages
- Compiler
- Why C is the first language selected?
- Some notable features of C
- Structure of a simple C Program.

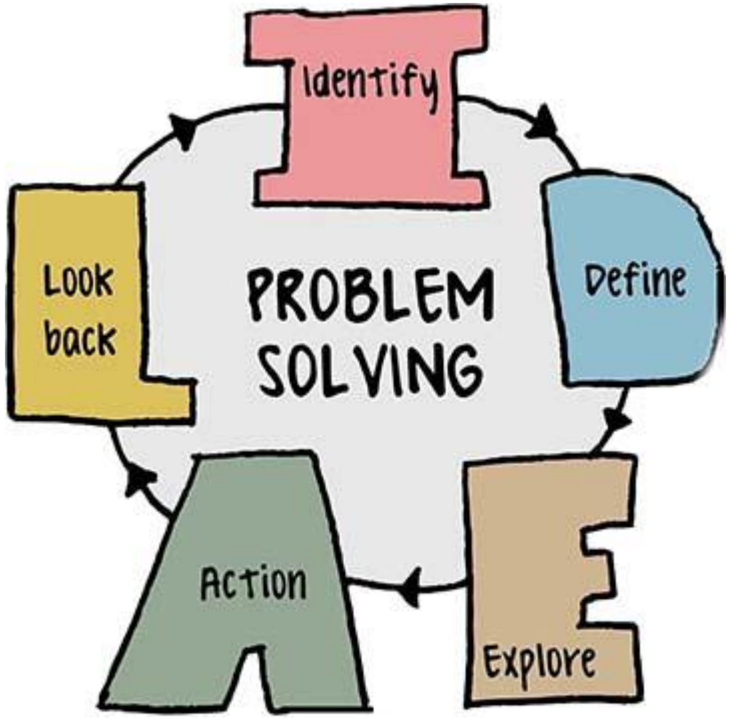
Definitions

Information & Data

- In computer science, **Information** is any data that can be **stored**, **processed**, **analyzed**, and **transmitted** through digital means (such as files, databases, or the Internet).
- Information in this context can include **text**, **images**, **videos**, or **any form of data** that can be encoded and processed.
- **Data**: Values are used to describe information. So, information can be called as the mean of data

Problem, Solve a problem and Solution

- **Problem** is a challenging situation that you face.
- **Solve a problem** is the process of finding and applying a solution to fix or handle that problem.
- **Solution** is an option proposed to solve a problem or achieve a goal.



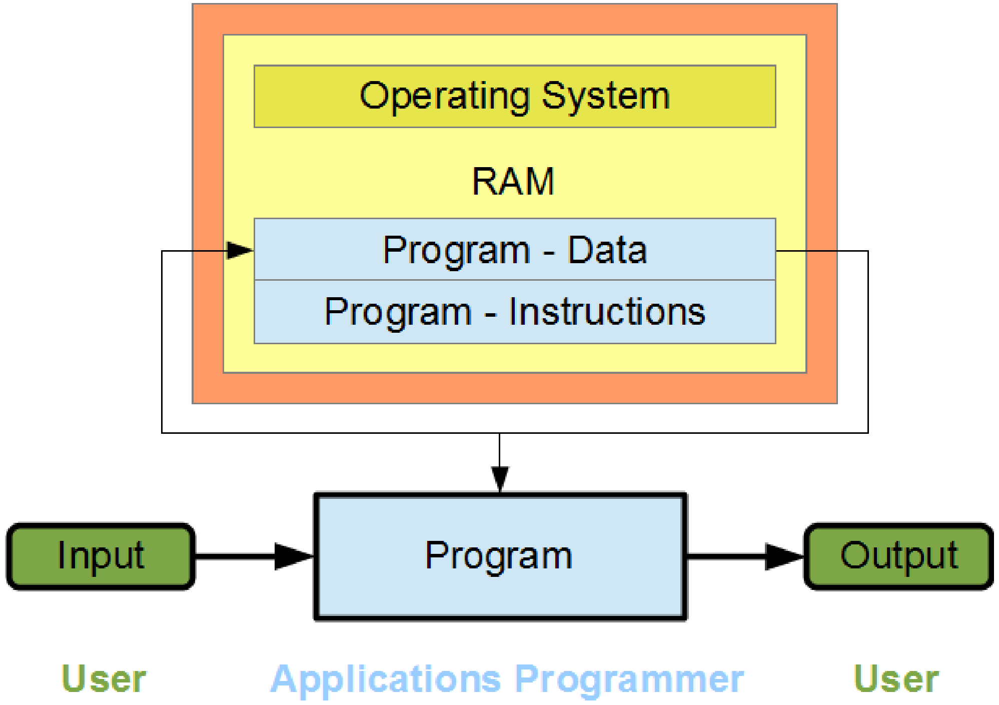
Algorithm, Program, Computer program

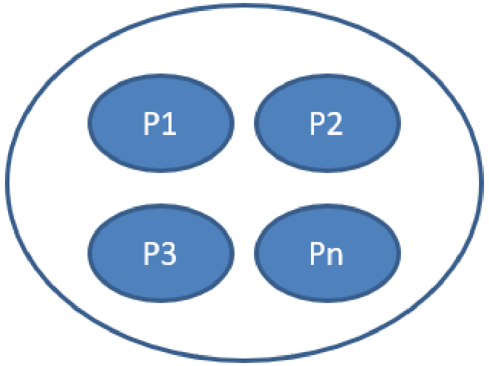
- **An algorithm** is a procedure used for solving a problem or performing a computation.
- **Program** is a sequence of steps to find out the solution of a problem.
- **Computer program** is a sequence or set of instructions in a programming language for a computer to execute

Computer program (cont.)

- **Computer program = data + instructions**
  - A simulation of solution
  - Is a set of instructions that computer hardware will execute
  - ⇒ **Increase the performance of standard workflow**

- **Computer software**: A set of related programs





How to make a good software?

Issues for a program/ software

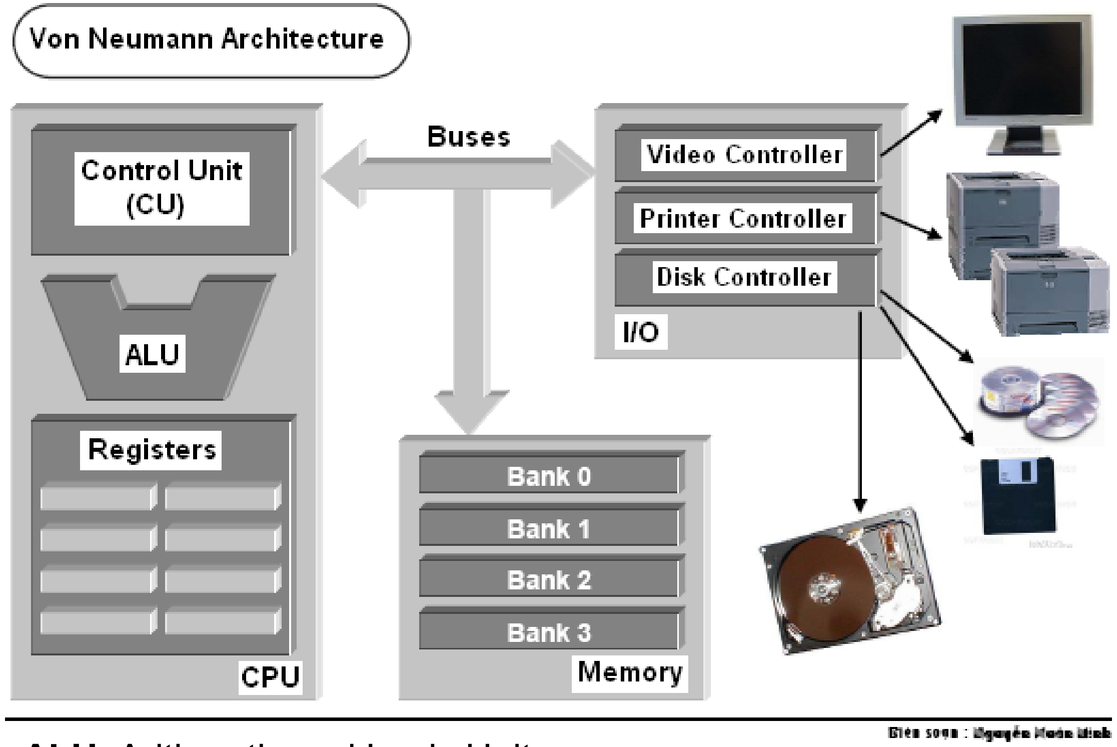
- **Usability**: Users can use the program to solve the problem
  - Robust and user-friendly interfaces
- **Correctness**: The solution must be correct
  - Comprehensive testing
- **Maintainability**: The program can be modified easily
  - Understandability: Structured programming; Document the code and overall design to help others (and yourself) understand and maintain the software.
  - Modifiability: Standards compliance
- **Portability**: The program can run on different platforms with minimal modification
  - Standards compliance → Needed modifications are minimum (platform: CPU + operating system running on it)

Issues for a program/ software (cont.)

- **Modularity**:
  - Break the program into smaller, self-contained modules or functions.
  - Each module should perform a specific task or represent a logical grouping of related functionality.
- **Scalability:**
  - Ensure the software can handle increased loads (e.g., more users, larger datasets) without significant performance degradation.
  - Opt for efficient algorithms and data structures.
- **Robustness:**
  - Write software that handles errors and unexpected inputs gracefully.
  - Incorporate error handling, logging, and validation mechanisms.

Overview Computer Hardware

Computer Hardware - Review



ALU: Arithmetic and Logic Unit

3 steps to read a memory cell:

(1) CPU puts the memory address to address bus

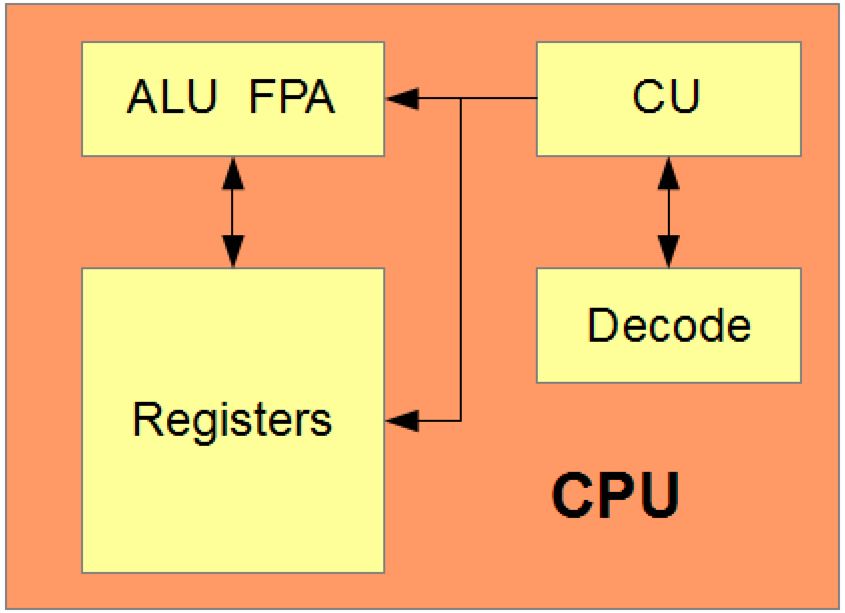
(2) CPU puts the read-signal to control bus.

(3) Data in memory cell is transferred to a register in CPU

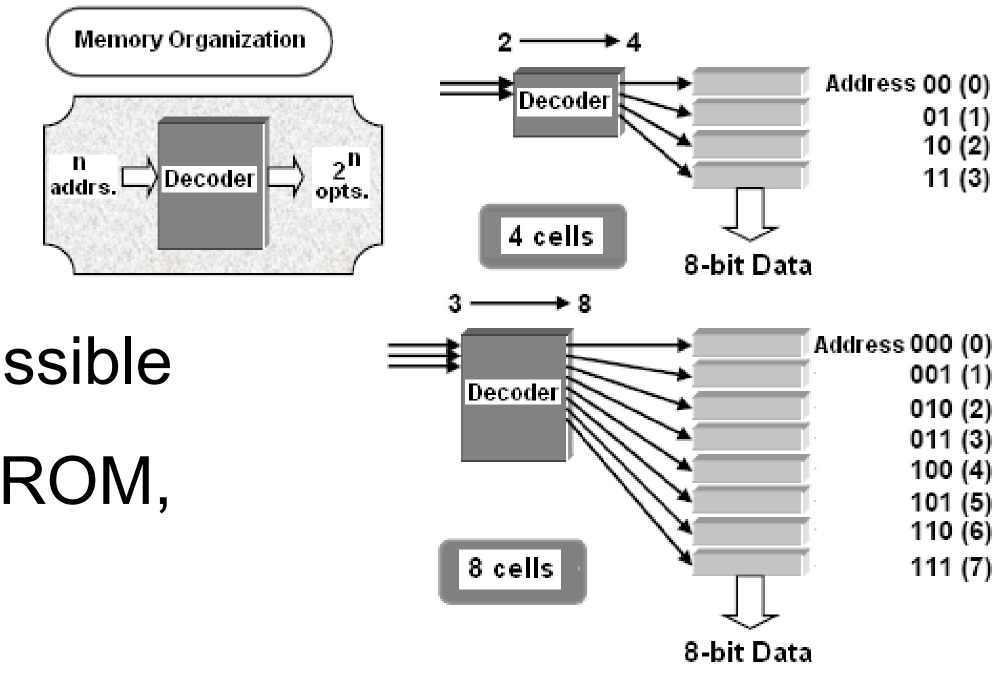
Bus	Used to
Address bus	Determine the IO peripherals, position of accessed memory.
Data bus	Transmit data
Control bus	Determine operation on peripherals, read peripheral ‘s states

Central Processing Unit (CPU)

- The CPU executes program instructions serially (one at a time). A modern CPU consists of:
  - Registers
  - A decode unit
  - A control unit (CU)
  - An arithmetic and logic unit (ALU)
  - A floating-point accelerator (FPA)



Primary Memory

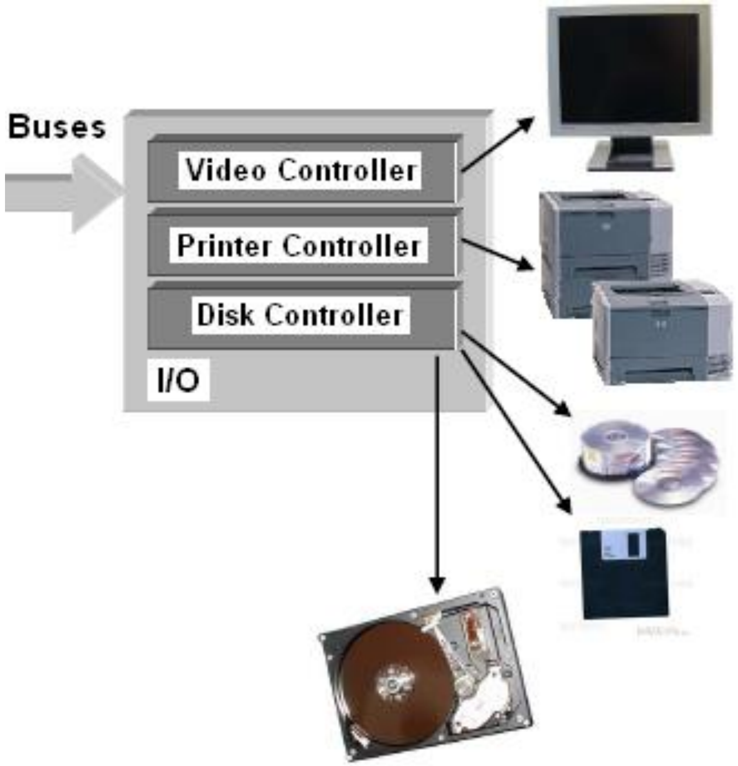


Biên soạn : Nguyễn Hoàn Minh

- Primary memory is memory directly accessible by the CPU. Primary memory includes: ROM, RAM
- **ROM** holds the instructions for starting the system. ROM is not volatile.
- **RAM** holds the program instructions and the program data. RAM is volatile

Devices

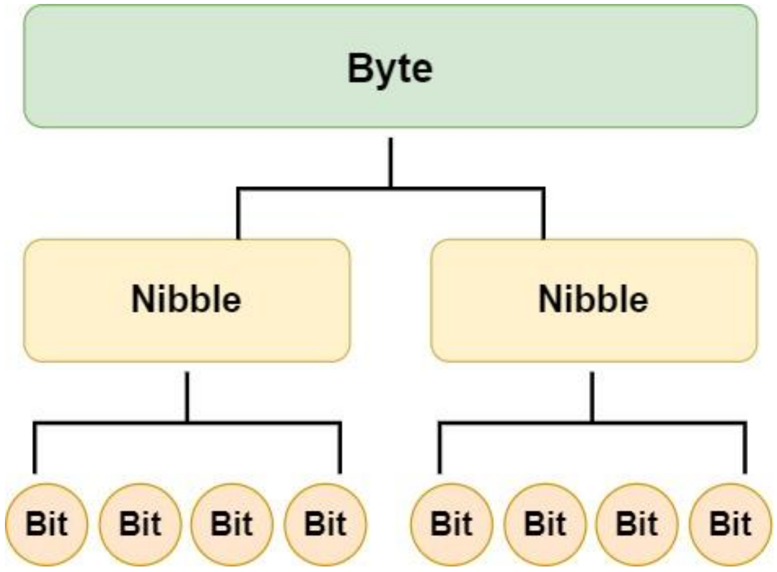
- Include basic **I/O devices** such as a keyboard, a monitor and a mouse, …
- **Storage devices** such as a floppy drive, a hard drive and a CD-ROM drive (secondary storage).
- All device interfaces connect to the system buses through a central controller.



Data Units

Data Units

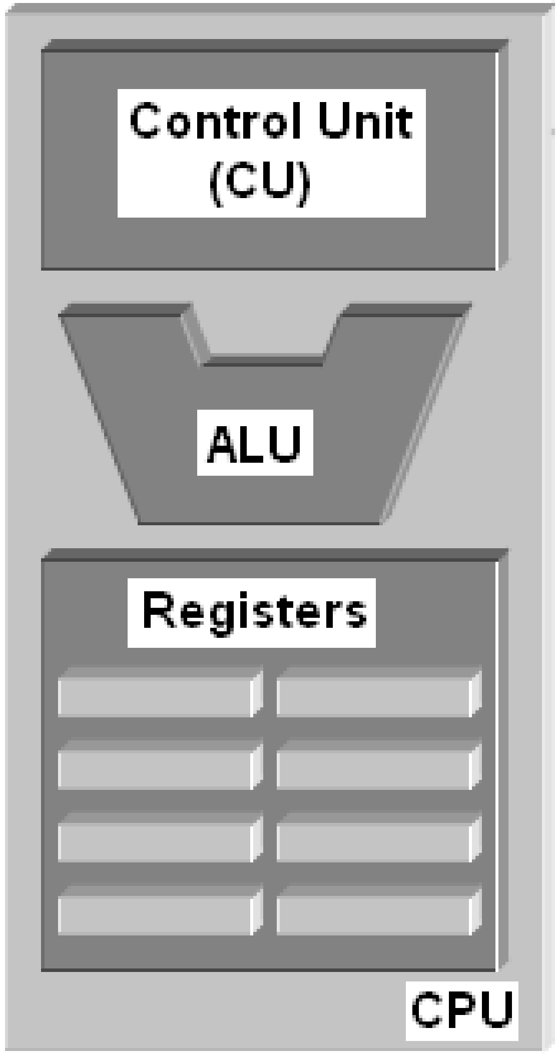
- Transistor is the basic physical unit for storing data → Binary format
- **John von Neumann** selected binary (base 2) digits as the EDVAC's fundamental unit.
- The vast majority of modern computers process and store information in binary digits.
- We call a **bi**nary digi**t** as a **bit**.
- Nibble = 4 consecutive bits.
- Byte = 8 consecutive bits = 2 nibbles
- Unit of memory is BYTE



00000000	← possibility 0
00000001	← possibility 1
00000010	←possibility 2
00000011	← possibility 3
00000100	← possibility 4
…	
00111000	← possibility 104
…	
11111111	← possibility 255

Data Units (cont.)

- The natural unit of the CPU is a **word**.
- The word length is number of bits of a general register within CPU (CPU memory).
- Word length can be 8, 16 (old CPUs), 32, 64 (current CPUs)



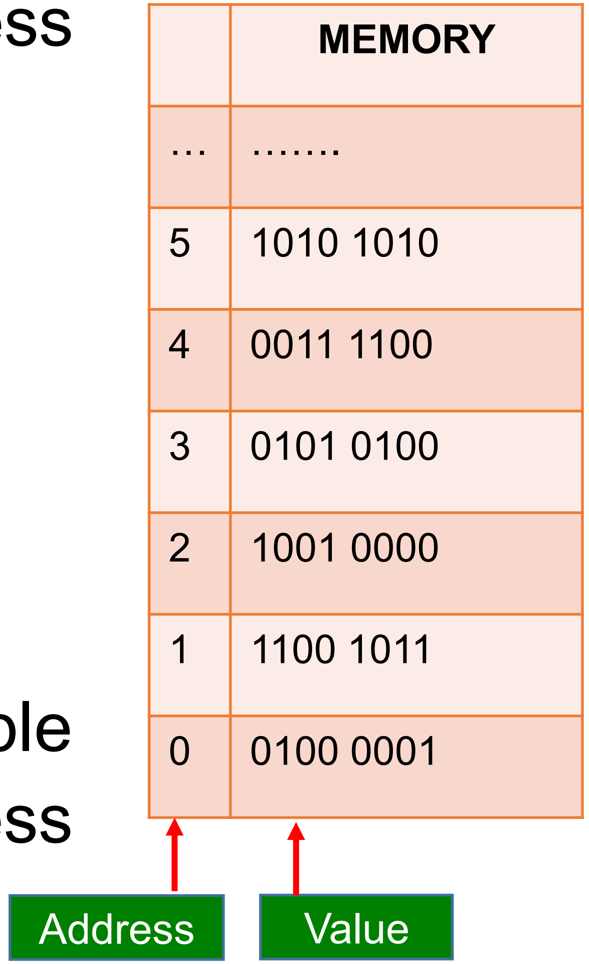
Addressing Information

Addressing Information

- Each byte of primary memory has a unique address (order number), starting from zero.

Name	Symbol	Size
KiloByte	KB	1,024 Byte
MegaByte	MB	1,024 KB
GigaByte	GB	1,024 MB
TeraByte	TB	1,024 GB
PetaByte	PB	1,024 TB
ExaByte	EB	1,024 PB

- Addressible Memory: The maximum size of addressable primary memory depends upon the size of the address registers



	MEMORY
…	…….
5	1010 1010
4	0011 1100
3	0101 0100
2	1001 0000
1	1100 1011
0	0100 0001

Program Instructions

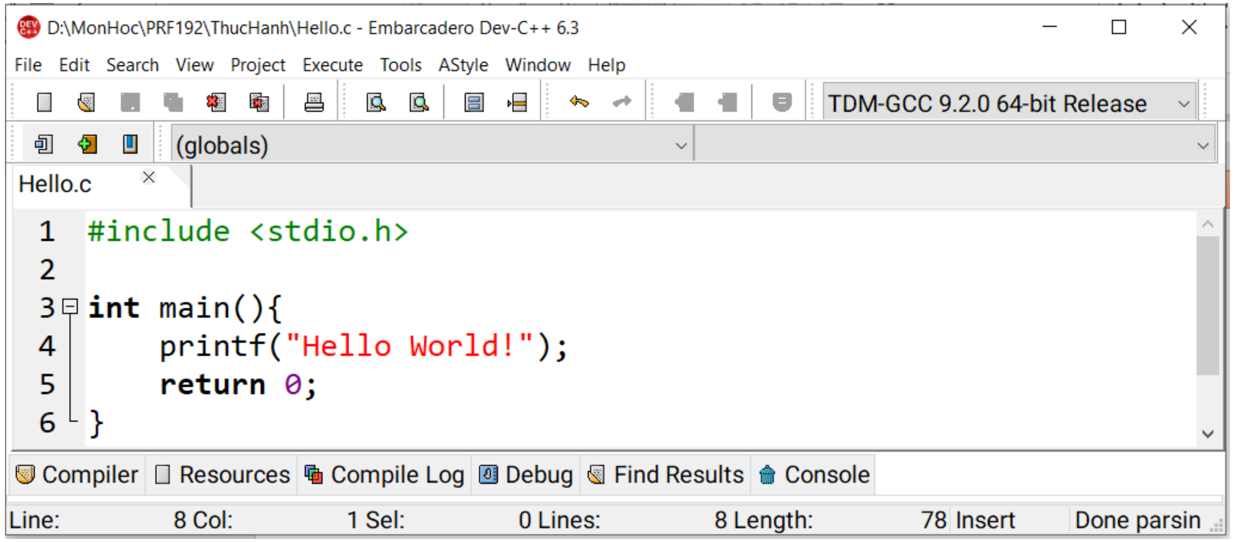
Program Instructions

01001011	100110110110	011011010111
Opcode	Operand 1	Operand 2

- Each program instruction consists of an **operation** and **operands**
- The CPU performs the operation on the values stored as operands or on the values stored in the operand addresses.
- **Operands**: Constants, registers, primary memory addresses

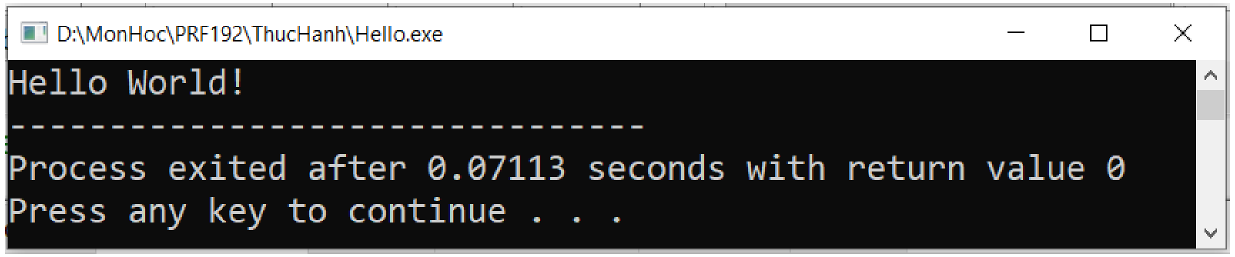
Let's create our first C file.

- Open Dev-C++ → File → New → Source File to create: **Hello.c**


D:\MonHoc\PRF192\ThucHanh\Hello.c - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
Hello.c
Compiler Resources Compile Log Debug Find Results Console
Line: 8 Col: 1 Sel: 0 Lines: 8 Length: 78 Insert Done parsin

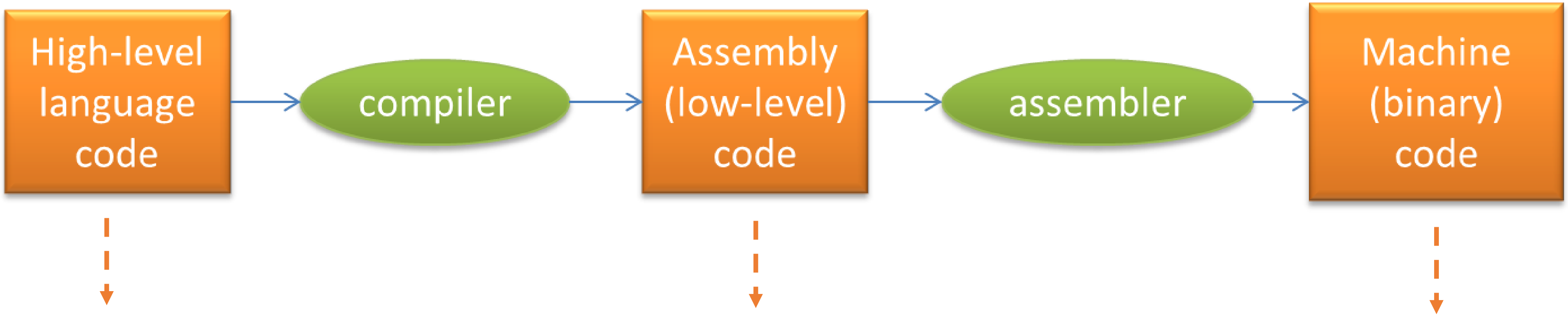
```c
#include <stdio.h>

int main(){
	printf("Hello World!");
	return 0;
}
```

- Execute → Compile & Run or F11 key press to execute the program:



```
Hello World!
--------------------------------
Process exited after 0.07113 seconds with return value 0
Press any key to continue . . .
```

Program Instructions (cont.)



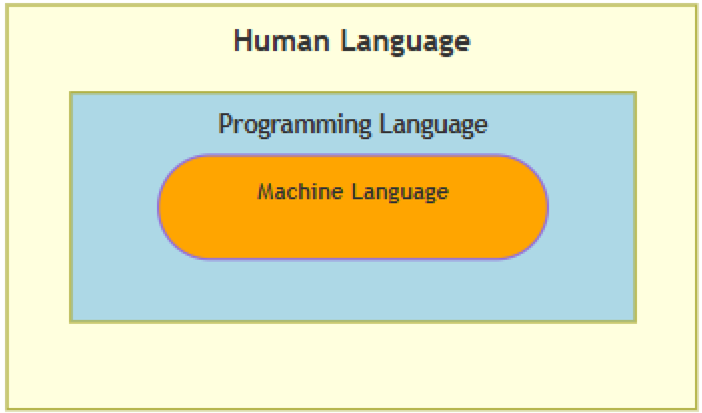
```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

```
swap:
      muli $2, $5,4
      add  $2, $4,$2
      lw   $15, 0($2)
      lw   $16, 4($2)
      sw   $16, 0($2)
      sw   $15, 4($2)
      jr   $31
```

```
00000000101000010000000000011000
00000000100011100001100000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Programming Languages

Programming Languages



- Programs that perform relatively simple tasks and are written in assembly language contain a large number of statements.
- **Machine Language → Assembly language → High-level languages.**
- To make our programs shorter, we use higher-level languages.

5 Generations of Programming Languages

1) **Machine languages**.
2) **Assembly languages**.
3) **Third-generation** languages. These are languages with instructions that describe how a result is to be obtained (C, Pascal, C++, Java…).
4) **Fourth-generation** languages. These are languages with instructions that describe what is to be done without specifying how it is to be done (SQL).
5) **Fifth-generation** languages are the closest to human languages. They are used for artificial intelligence, fuzzy sets, and neural networks (Prolog, Matlab)

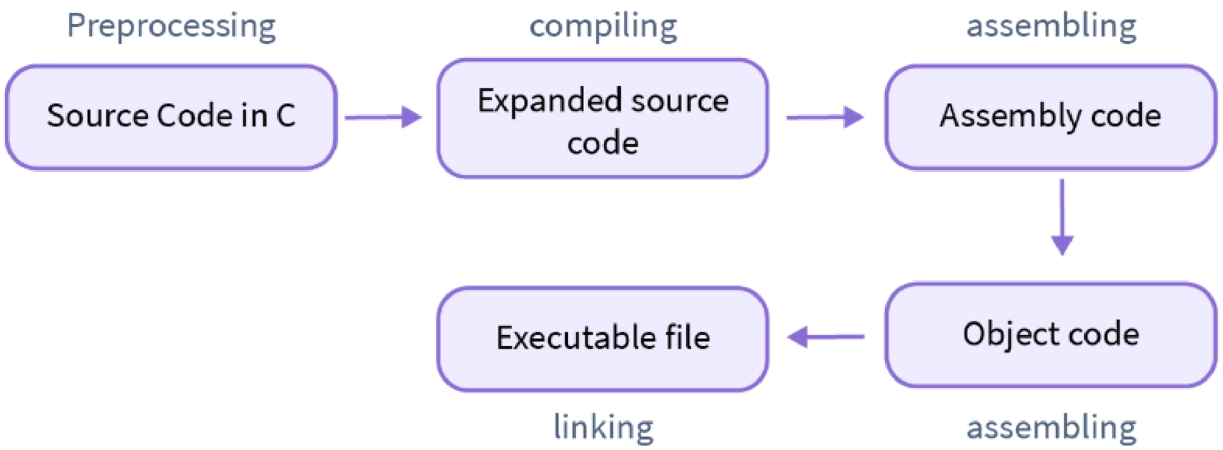
Compiler

Translating and Executing a Program

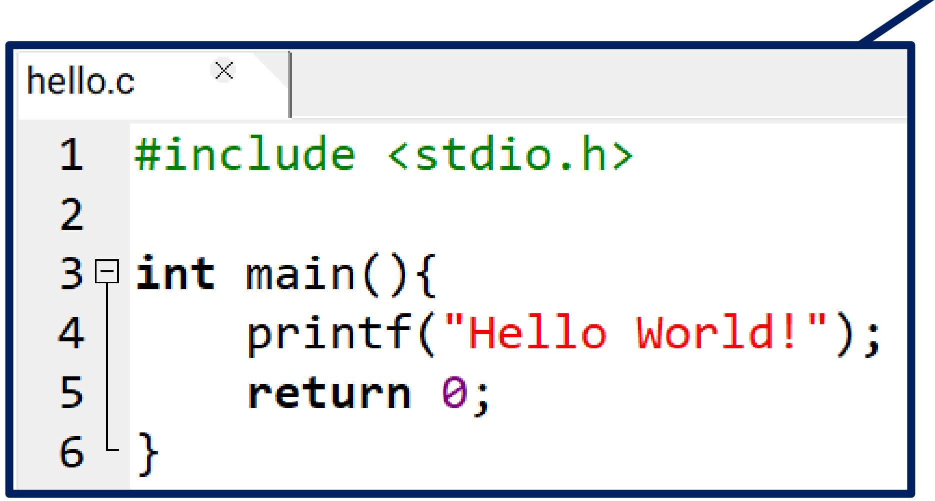
- **Program code** in a high level language can not run. It must be translated to binary code (machine code) before running.
- 2 ways of translations:
- **Interpreting**: One-by-one statement is translated then run → **Interpreter**
- **Compiling**: All statements of program are translated then executed as a whole → **Compiler**
- **C translator is a compiler**

Compiler

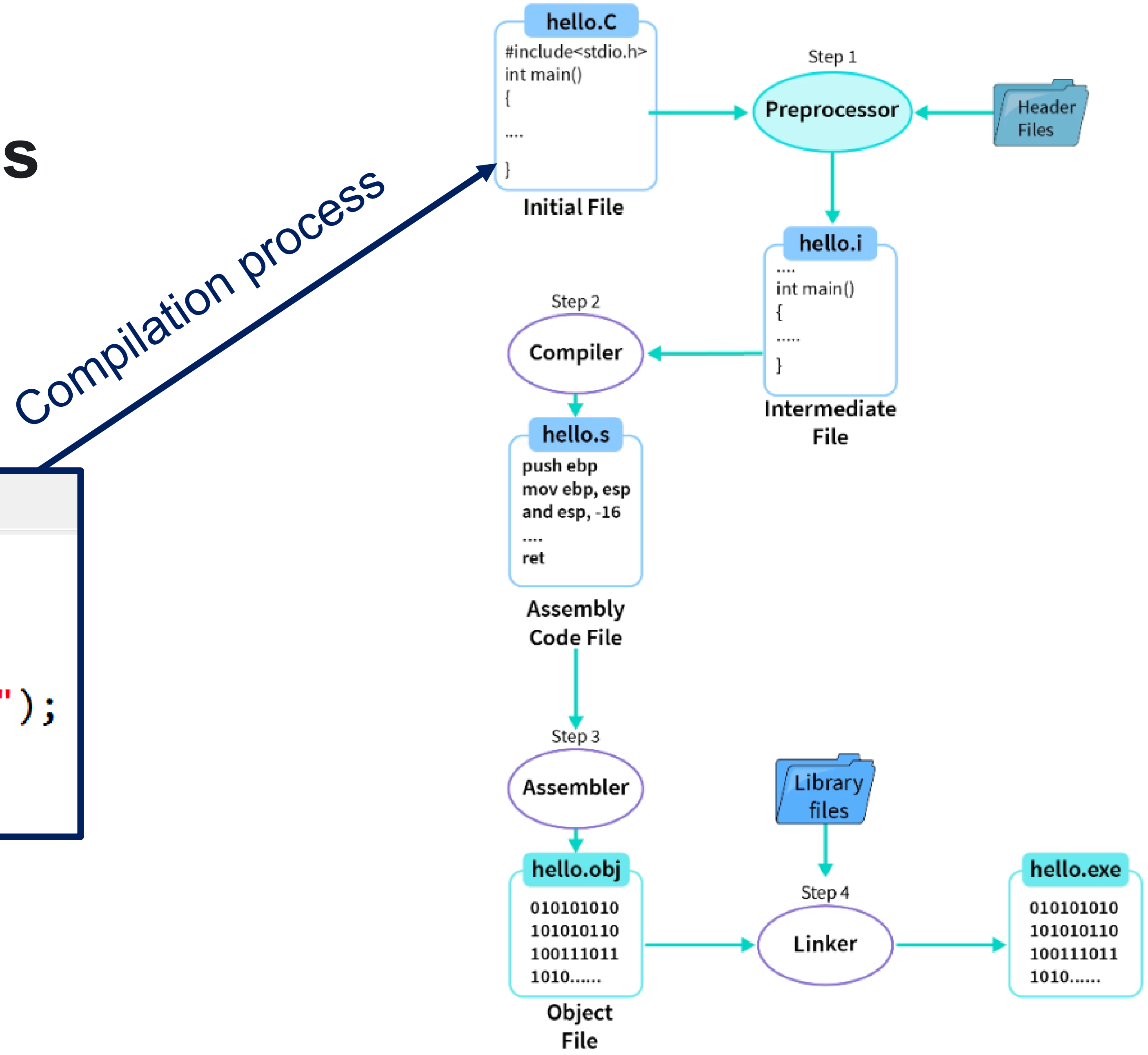
- **The compilation process in C is**:
  - Converting an understandable human code into a machine understandable code
  - Checking the syntax and semantics of the code to determine any syntax errors or warnings present in our C program.
- **Compilation process in C involves four steps:**
  - Preprocessing
  - Compiling
  - Assembling
  - Linking



Compilation process



```c
#include <stdio.h>

int main(){
	printf("Hello World!");
	return 0;
}
```



Compilation process (cont.)

The C program file **hello.c**

- **Step 1**: Preprocessing of header files, all the statements starting with **#** symbol and comments are replaced/removed. Generates an intermediate file **hello.i**
- **Step 2**: Compiler software translates the **hello.i** file to **hello.s** with assembly level instructions (**low-level code**).
- **Step 3**: Assembly-level code instructions are converted into machine-understandable code (binary/hexadecimal form) by the assembler. The file generated is known as the object file with an extension of **.obj/.o** i.e. **hello.obj/hello.o** file.
- **Step 4**: Linker is used to link the library files with the object file to define the unknown statements. It generates an executable file with **.exe/.out** extension i.e. a **hello.exe/hello.out** file.

Next, run the **hello.exe/hello.out** executable file to get the desired output on our output window, i.e., **Hello World!**.

Why C is the first language selected?

Why C is the first language selected?

- C is one of the most popular languages in use globally
- Some reasons for learning programming using the C language includes:
  - C is English-like,
  - C is quite compact - has a small number of keywords,
  - A large number of C programs need to be maintained,
  - C is the lowest of high-level languages,
  - C is faster and more powerful than other high-level languages,
  - The UNIX, Linux and Windows operating systems are written in C and C++.
  - The most common languages, such as Java, C#, are similar to C.
  - C supports basic ways which help us understanding memory of a program. These can be hidden in higher languages.

Language	Time to Run
Assembly	0.18 seconds
C	2.7 seconds
Basic	10 seconds

Comparative times
for a Sieve of
Eratosthenes test

Some Notable C Features

Some Notable C Features

- **Comments**

  /*   */

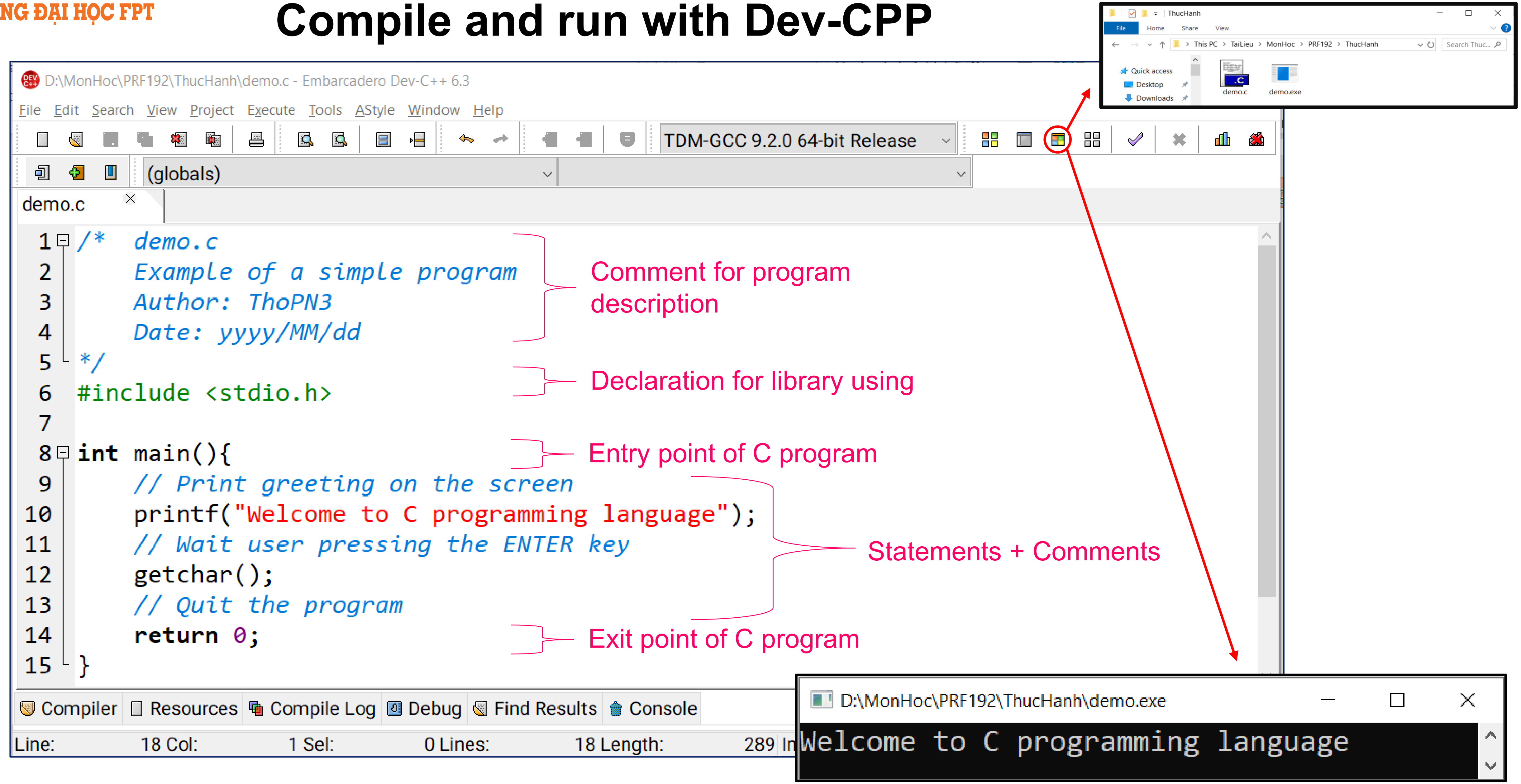
  - We use comments to document our programs and to enhance their readability. C compilers ignore all comments.

- **Whitespace**
  - We use whitespace to improve program readability and to display the structure of our program's logic. C compilers ignore all whitespace

- **Case Sensitivity**
  - C language is case sensitive.
  - C compilers treat the character 'A' as different from the character 'a' .

Structure of a Simple C Program

Compile and run with Dev-CPP



D:\MonHoc\PRF192\ThucHanh\demo.c - Embarcadero Dev-C++ 6.3
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
demo.c
Compiler
Resources
Compile Log
Debug
Find Results
Console
Line: 18 Col: 1 Sel: 0 Lines: 18 Length: 289 In
ThucHanh
File Home Share View
This PC › TaiLieu › MonHoc › PRF192 › ThucHanh
Search Thuc...
Quick access
Desktop
Downloads
demo.c
demo.exe
D:\MonHoc\PRF192\ThucHanh\demo.exe

```c
/*  demo.c
    Example of a simple program
    Author: ThoPN3
    Date: yyyy/MM/dd
*/
#include <stdio.h>

int main(){
    // Print greeting on the screen
    printf("Welcome to C programming language");
    // Wait user pressing the ENTER key
    getchar();
    // Quit the program
    return 0;
}
```

Comment for program description

Declaration for library using

Entry point of C program

Statements + Comments

Exit point of C program

```
Welcome to C programming language
```

Compile and run a program using the command prompt

Step 1: Install a C Compiler

- Download and install MinGW or TDM-GCC.

  (Link download MinGW: https://sourceforge.net/projects/mingw/)

- During installation, ensure the **gcc** tool is included.

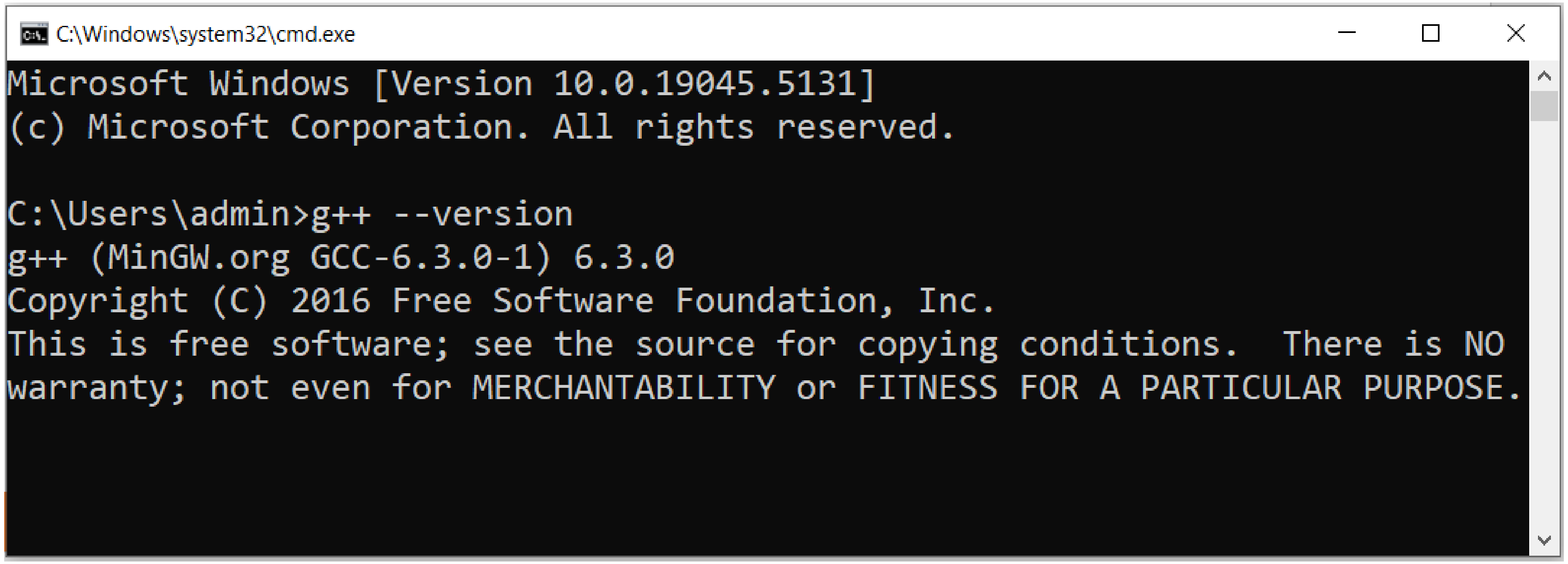
Step 2: Set Up Environment Variables

- Add the bin directory of MinGW (e.g., C:\MinGW\bin) to your **PATH** environment variable:
  - Go to **Control Panel** → **System and Security** → **System** → **Advanced System Settings** → **Environment Variables**.
  - Find the **Path** variable in **System Variables**, click **Edit**, and add the **bin directory path** (C:\MinGW\bin)

Compile and run a program using the command prompt (cont.)

Step 3: Test Environment:

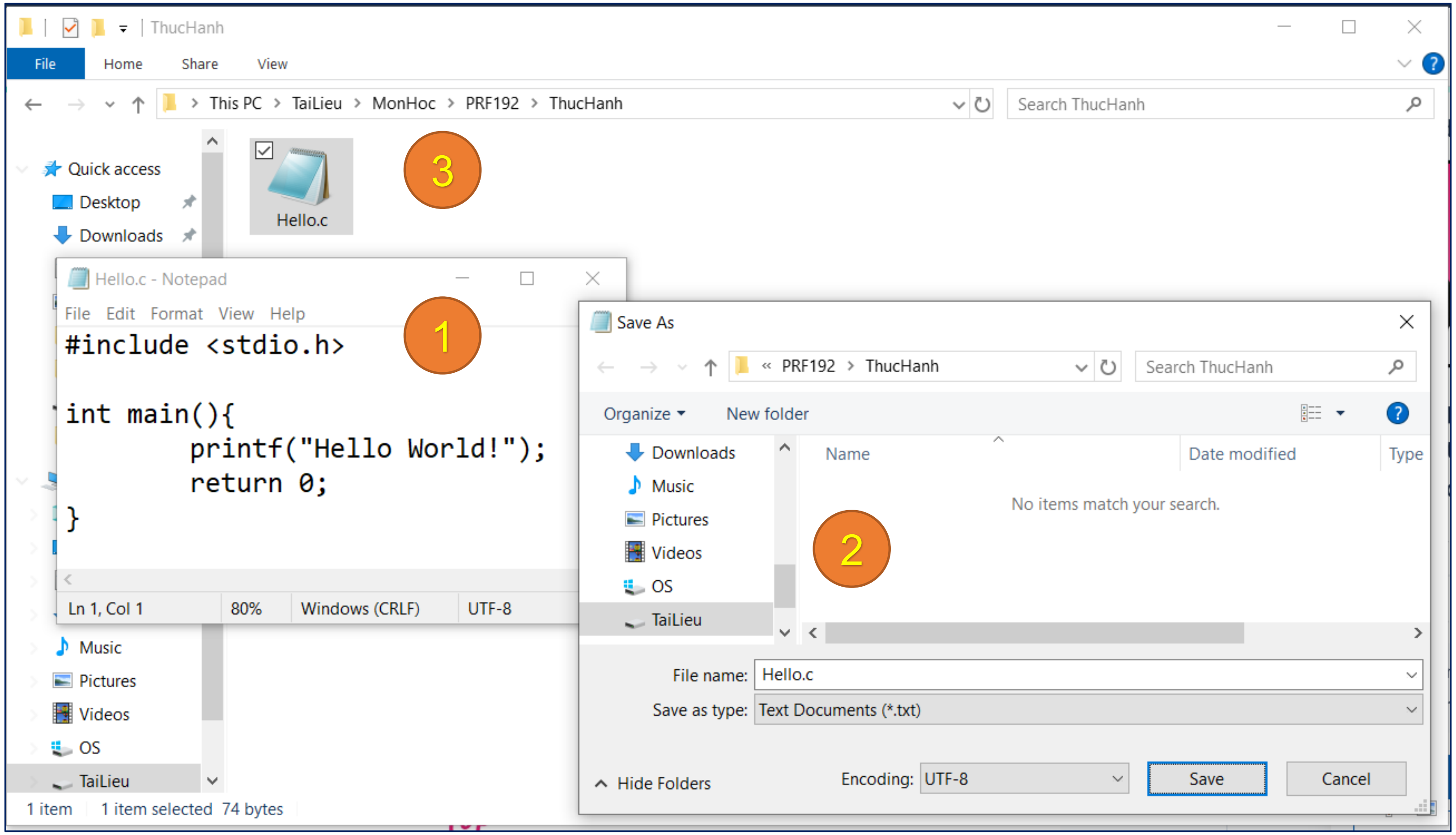
Open Command Prompt and type: **g++ --version**



```
Microsoft Windows [Version 10.0.19045.5131]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin>g++ --version
g++ (MinGW.org GCC-6.3.0-1) 6.3.0
Copyright (C) 2016 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Compile and run a program using the command prompt (cont.)

Step 4: Write a program Hello.c



```
#include <stdio.h>

int main(){
        printf("Hello World!");
        return 0;
}
```

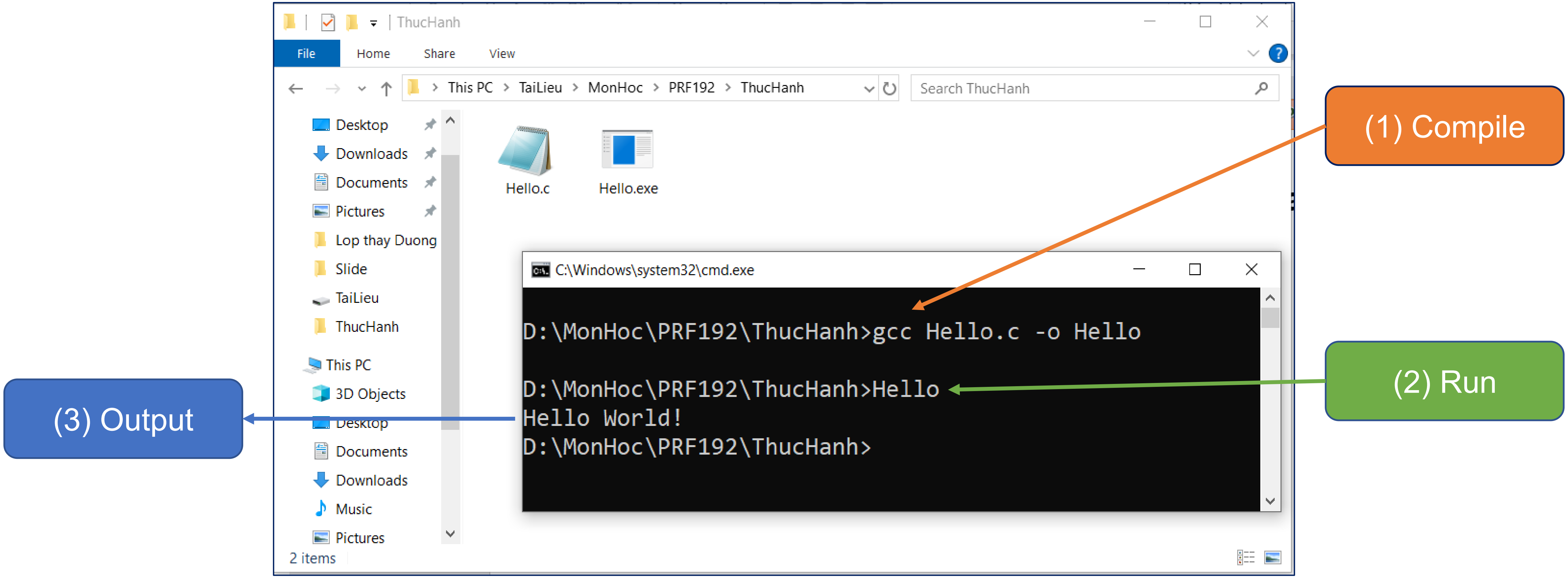
File name: Hello.c

Save as type: Text Documents (*.txt)

Encoding: UTF-8

Compile and run a program using the command prompt (cont.)

Step 5: Compile & Run



```
D:\MonHoc\PRF192\ThucHanh>gcc Hello.c -o Hello

D:\MonHoc\PRF192\ThucHanh>Hello
Hello World!
D:\MonHoc\PRF192\ThucHanh>
```

(1) Compile

(2) Run

(3) Output

Structure…: C program Entry Points

- Entry point: The point where a program begins.
- Entry points of C-programs:

Syntax:

```
[int] main( [void] )
{
          <statements>
          [ return number; ]
}
```

Summary

- Definitions related to programming
- How to make a good software?
- Steps to develop a software?
- Computer hardware.
- Fundamental Data Units
- Program Instructions
- Programming Languages
- C Compilers
- Why C is the first language selected?
- Some notable features of C
- Structure of a simple C Program.