

Practical Sheet N° 1

The Machine, the Keyboard and the Compiler

Duration: 3 h

Aims of this session

This first session contains **no C programming to invent**. One program is given to you, complete, and you will type it without being asked to understand it. Everything else is about becoming fast and comfortable with the tools. By the end you will be able to:

- **Recognise** the units of your machine and organise your files and folders.
- **Identify** your keyboard layout and **produce every character that C needs**, on the physical keyboard or on the on-screen keyboard.
- **Edit** text without the mouse: select all, copy, undo, jump to a line.
- **Compile** and **run** a source file with Dev-C++, and tell the `.c` file from the `.exe` file.
- **Read** a compiler error message and find the line it points to.

Note

Tools required. Dev-C++ (<https://www.embarcadero.com/free-tools/dev-cpp>) or Code::Blocks (<https://www.codeblocks.org/>). Both are free and both include the gcc compiler, so nothing else is needed.

Part A — Your Machine and Your Files

A.1 — The identity card of your machine

Chapter 1 described a computer as a processor, a memory and peripherals. Those are not abstractions: they are written on the machine in front of you. Press `Windows` + `Pause`,

or open **Settings > System > About**.

Exercise 1 — ★ Reading your own machine

Fill in this table for the machine you are working on.

Item	Value on my machine
Processor (model and frequency)	
Installed memory (RAM)	
Operating system and version	
Free space on disk D:	
Screen resolution	

Then answer in one sentence each:

- Your RAM is given in gigabytes. How many *bytes* is that, roughly?
- When you switch the machine off, what disappears: the contents of the RAM, or the contents of the disk? Which of your two answers explains why you must save your work?

A.2 — Making the machine tell you the truth about files

By default Windows **hides file extensions**. A file that looks like `hello` may really be `hello.txt`, and a file that looks like `hello.c` may really be `hello.c.txt`. You cannot work like that, and this single setting is the cause of a large share of the problems beginners meet in week one.

- Open the file explorer (`Windows + E`).
- Windows 11: **View > Show > File name extensions**.
Windows 10: tab **View**, tick **File name extensions**.
- Create the folder `D:\ASD1\TP1`. Work there for the whole session. **Never work on the Desktop or in Downloads**: files are lost there, and lost files cannot be marked.

Warning!

Do not create your C files with Notepad. Notepad silently adds `.txt` to the name, producing `hello.c.txt`, which the compiler will refuse to see. Always create source files from inside Dev-C++.

Part B — The Keyboard

C is written with characters that ordinary French or English text almost never uses: **#**, **{**, **}**, **[**, **]**, ****, **%**, **<**, **>**. If you hunt for each one with your eyes, a ten-line program takes twenty minutes. If you know where they are, it takes two. That difference is what this part buys you.

B.1 — Which layout is under your fingers?

1. Open Notepad and press the six keys immediately to the right of **Tab**.
2. If you read **azerty**, your layout is **AZERTY** (French). If you read **qwerty**, it is **QWERTY** (English/US).
3. Look at the right-hand end of the taskbar: a small indicator shows **FRA**, **ENG** or **ARA**. Click it to see the layouts installed.
4. Press **Windows** + **Space** (or **Alt** + **Shift**) several times and watch that indicator change. **This is the accident that makes a keyboard “stop working”: the layout was switched by mistake.**

B.2 — One key, three characters

A key is not one character. It carries up to three, and which one you obtain depends on the modifier you hold down: nothing, **Shift**, or **AltGr** (Figure 1). On an AZERTY keyboard the characters C needs live mostly on the **AltGr** level, which is why they seem to be missing.

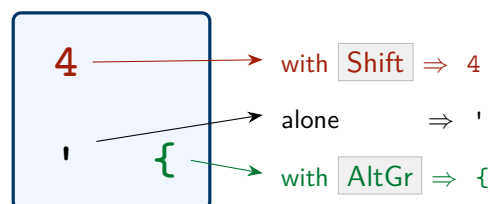


Figure 1: One key of a French AZERTY keyboard carries three characters. Note that the *digit* needs **Shift**, and that the brace needs **AltGr**.

Two consequences you will meet today

On a French AZERTY keyboard the **digits require** **Shift** (press **Caps Lock** if you have many numbers to type), and the **full stop requires** **Shift** as well. On a QWERTY keyboard both are direct.

B.3 — Reference card: every character C needs

The two tables below give the usual keystroke on each layout. **Verify every line on your own machine** and write what actually works in the last column: keyboards are not all identical, and this card is yours for the rest of the semester.

	AZERTY	QWERTY	Mine
#	AltGr + 3	Shift + 3	
{	AltGr + 4	Shift + [	
}	AltGr + =	Shift +]	
[	AltGr + 5	[alone	
]	AltGr +)	] alone	
(	5 alone	Shift + 9	
)	) alone	Shift + 0	
<	left of W	Shift + ,	
>	Shift on it	Shift + .	
"	3 alone	Shift + '	
'	4 alone	key alone	
\	AltGr + 8	key alone	
	AltGr + 6	Shift + \	

	AZERTY	QWERTY	Mine
;	; alone	; alone	
:	: alone	Shift + ;	
,	, alone	, alone	
.	Shift + ;	. alone	
%	Shift + ù	Shift + 5	
&	1 alone	Shift + 7	
*	* alone	Shift + 8	
-	8 alone	Shift + -	
!	! alone	Shift + 1	
=	= alone	= alone	
+	Shift + =	Shift + =	
-	6 alone	- alone	
/	Shift + :	/ alone	

Note

The dead-key trap. On AZERTY, the accent keys $\hat{\text{~}}$ and ¨ are *dead keys*: they wait for the next letter in order to build $\hat{\text{a}}$ or ë . To obtain a bare $\hat{\text{~}}$, press the key **then the space bar**. The same applies to ~ , which is **AltGr + 2** followed by a space.

Exercise 2 — ★ Filling and checking the card

Open Notepad and type, on one line, every character of the two tables above, separated by a space. Compare what appears with the tables and **correct the last column** wherever your machine disagrees. Report the characters, if any, for which the given keystroke did not work, and the keystroke you found instead.

B.4 — The on-screen keyboard: your map when you are lost

When you cannot find a character, do not search key by key. Ask the machine to show you the map.

1. Press **Windows + R**, type **osk** and press **Enter**. The on-screen keyboard appears. (Alternative: **Settings > Accessibility > Keyboard > On-Screen Keyboard**.)
2. Click **Shift** on the on-screen keyboard and **watch every key face change**. Click it again to release.
3. Click **Alt Gr**. The faces change again: these are the characters hidden on the third level.
4. You may also click the keys with the mouse to insert a character directly into Notepad or into Dev-C++.

Exercise 3 — ★ Hunting with the map

Using *only* the on-screen keyboard, find and note the physical key that produces each of these four characters on your layout. Then produce each one on the real keyboard to confirm.

Character	Which physical key	Which modifier
\		
~		
^		

Finally: give one situation, other than a missing character, in which the on-screen keyboard is the only way to work.

B.5 — Editing without the mouse

Reaching for the mouse to select a line costs about two seconds. Doing it four hundred times in a semester costs an afternoon. Learn these once, use them for ever.

Keys	Action	When you will need it
Ctrl + A	Select all	Copy or replace a whole program
Ctrl + C / V / X	Copy / paste / cut	Duplicate a line you must repeat
Ctrl + Z / Y	Undo / redo	After breaking something on purpose
Ctrl + S	Save	Every two minutes, without thinking
Home / End	Start / end of line	Add the semicolon you forgot
Ctrl + Home / End	Start / end of file	Jump to the top of a long file
Shift + arrows	Select while moving	Select exactly one word or one line
Tab / Shift + Tab	Indent / unindent a block	Re-indent an unreadable program
Ctrl + F	Find	Locate a variable in a long program
Ctrl + G	Go to line	Jump to the line an error message names
Insert	Toggle overwrite	Only to <i>escape</i> it: if your typing eats the next characters, you pressed it by accident

B.6 — The speed drill

Speed at the keyboard is not a talent. It is a measurement, and it improves when you repeat. The three lines below contain almost every awkward character in C.

C Program — The three drill lines

```

1 #include <stdio.h>
2 printf("%d\n", x);
3 { i = i + 1; }
```

Exercise 4 — ★ Measuring yourself

In Notepad, type each line three times in a row, timing each attempt with the stopwatch of your telephone. Record your times in seconds.

Line	Attempt 1	Attempt 2	Attempt 3
#include <stdio.h>			
printf("%d\n", x);			
{ i = i + 1; }			

Then answer:

- By what percentage did your third attempt improve on your first?
- Which single character slowed you down the most? Where is it on your keyboard?
- A target for the end of the semester is each line in under 10 seconds without looking at the keyboard. How far are you today?

Part C — The Compiler

C.1 — Preparing Dev-C++

- Launch Dev-C++. If a language dialogue appears, choose English.
- Tools > Editor Options > Display:** tick **Line numbers**. Every error message names a line, so you must be able to see them.
- Locate the three panes: the **editor** in the middle, the **compile log** and the **message pane** at the bottom. The message pane is where the compiler answers you.
- Find in the menu **Execute** the three entries **Compile** (F9), **Run** (F10) and **Compile & Run** (F11).

C.2 — The given program

Here is your first C program. **You are not asked to understand it today**; Chapter 2 will explain every line. You are asked to reproduce it exactly, using the reference card of Part B. Do not copy and paste: typing it is the exercise.

C Program — hello.c — given, type it exactly

```

1  #include <stdio.h>
2
3  int main(void)
4  {

```

```

5   int year = 2026;
6
7   printf("Hello! My name is ...\n");
8   printf("University of Oum El Bouaghi - %d\n", year);
9
10  return 0;
11 }

```

1. File > New > Source File, or **Ctrl + N**.
2. Type the program, replacing the dots with your own name.
3. File > Save As, name it `hello.c`, in `D:\ASD1\TP1`. **The extension matters:** saved as `.cpp` the file would be compiled as C++, which is a different language.
4. Press **F9** (compile only). Read the message pane.
5. Press **F10** (run only).
6. Press **F11** (compile and run).

Expected result

```

Hello! My name is BENALI Ahmed
University of Oum El Bouaghi - 2026

```

Exercise 5 — ★ Compiling and running are two different acts

Answer in your report.

- a. After **F9** alone, did anything appear on the screen? What did the compiler produce instead?
- b. After **F10** alone, was the program compiled again? How can you tell?
- c. Change the year to 2027, save, then press **F10** *without* pressing **F9** first. Which year is displayed? Explain what this proves about the difference between the file you edit and the program that runs.

C.3 — Two files, two different objects

Exercise 6 — ★★ The source and the executable

Open `D:\ASD1\TP1` in the file explorer.

- a. Which files are now present besides `hello.c`? Note the name, the extension and the size of each.
- b. `hello.exe` is far larger than `hello.c`, although it holds the same program. Propose an explanation.
- c. Open `hello.c` with Notepad: you can read it. Open `hello.exe` with Notepad: what appears, and who is that text written for?

- d. Close Dev-C++ completely and double-click `hello.exe`. Does it still run?
- e. Copy `hello.exe` alone onto a USB key and run it on a neighbouring machine that has no compiler installed. Does it run? Now do the same with `hello.c` alone. What is your conclusion?

What you have just proved with your own hands

Chapter 1 states that an algorithm is not executable, that a program is only a text, and that the compiler alone produces something the processor can run. Exercise 6 is that statement, measured: the text `hello.c` needs a compiler, while the executable `hello.exe` needs nothing but the machine.

C.4 — Breaking the program on purpose

Reading error messages is a skill, and the only way to acquire it is to produce errors deliberately in a program you already know is correct.

Exercise 7 — ★★ Five deliberate mistakes

Starting each time from the *working* `hello.c`, make one single change, compile with `F9`, and fill one row of the table. Use `Ctrl+G` to jump to the line the compiler names, then `Ctrl+Z` to undo before the next change.

The change I made	The first message	Line named	Real line
Remove the ; of the first <code>printf</code>			
Remove the line <code>#include <stdio.h></code>			
Write <code>Printf</code> with a capital P			
Remove the last closing brace }			
Remove one " of the first message			

Then answer: *does the compiler always point at the line where the mistake really is?* Use your two last columns to justify your answer, and say which of the five cases was the most misleading.

Note

Golden rule. Always correct the **first** error of the list and recompile immediately. The errors that follow are frequently consequences of the first one, and they disappear on their own.

Part D — Optional

Finish Parts A to C first. These are for the time that remains, and for home.

- **From memory.** Close Dev-C++, keep only this printed sheet, and retype `hello.c` in a new file. Time yourself and compare with the time it took the first time.
- **The other tool.** Install Code::Blocks and compile the same file. Note two differences with Dev-C++, in particular the notion of *project*.
- **Without any tool.** Open a command prompt (`cmd`), move to your folder with `cd D:\ASD1\TP1` and compile by hand:

Command Prompt

```
gcc hello.c -o hello.exe
hello.exe
```

What does the compiler display when everything goes well?

- **Comfort.** In **Tools > Editor Options**, find the option that highlights the brace matching the one under the cursor. Why will it save you time from Chapter 3 onwards?
- **Speed.** Repeat the drill of B.6 once a day for a week and draw the curve of your best times.

What to hand in

Note

One folder named `TP1_Surname_Name` containing:

- your file `hello.c` **and** the `hello.exe` it produced;
- a report of two pages at most, holding the completed tables of exercises 1, 3, 4 and 7, your corrected reference card of B.3, and the short answers asked in exercises 1, 3, 4, 5 and 6.

The tables are the important part. Keep the reference card of B.3, corrected for your own machine: you will use it every week until the end of the semester.