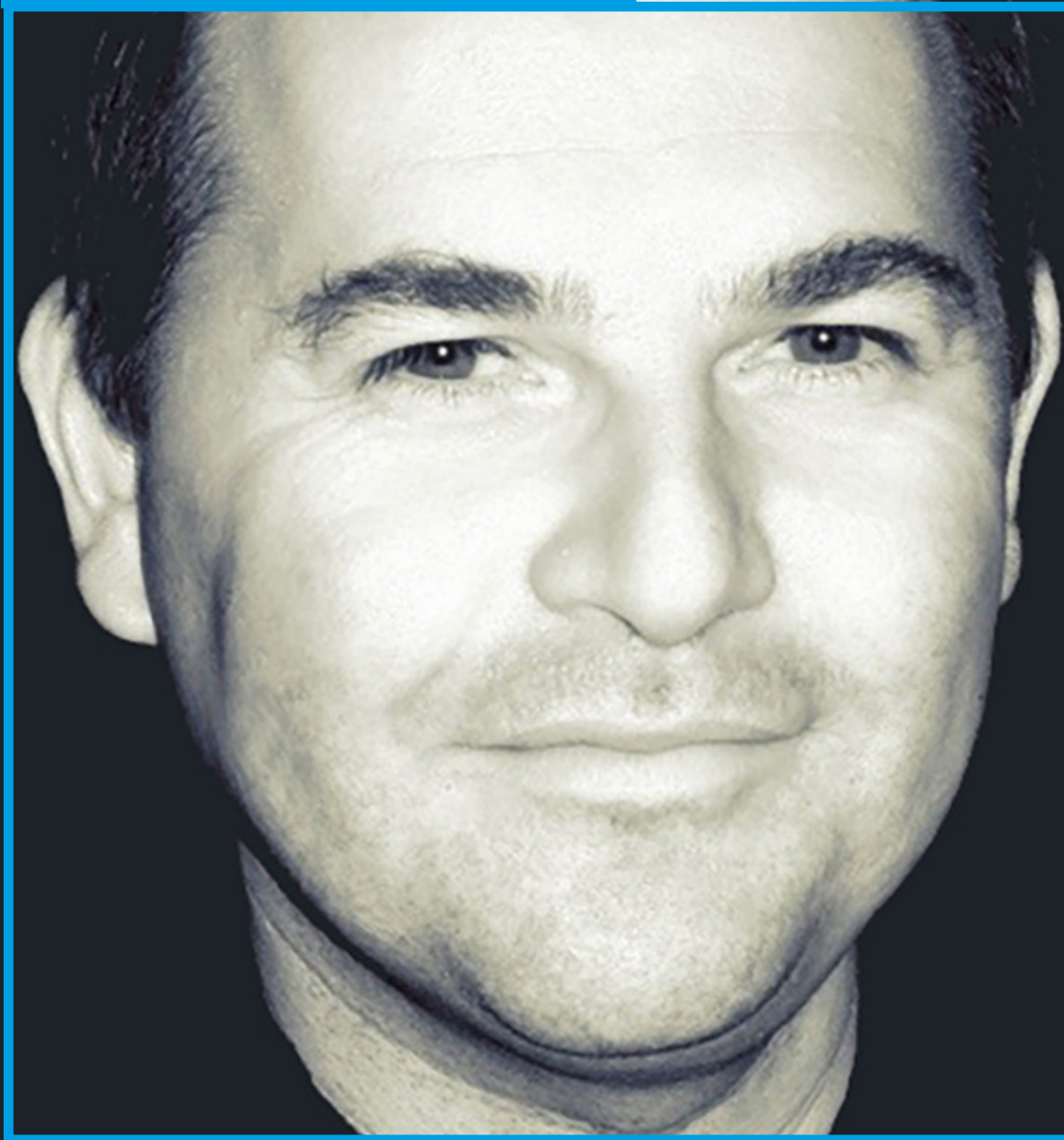


Welcome



oct. 2022

D. Ginhac



\$ ~ whoami

HI, I'M D. GINHAC

I will be your instructor for some [Computer Programming](#) lectures this year and next year.

I also do research in [AI Architectures for Edge Computer Vision](#).

 dginhac@u-bourgogne.fr

 [dginhac](#)

 [@dginhac](#)



Fundamentals of C++.

From **beginner** to **beyond**.

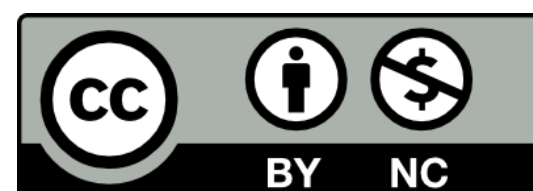
Visit <https://github.com/dginhac/esirem-itc313>

Everything available on GitHub.



The screenshot shows the GitHub repository page for 'dginhac/esirem-itc313'. The repository is public and has 2 branches and 0 tags. The main branch is 'master'. The repository contains a README, a license file (LICENSE-CC-BY-NC), and several folders: TD, TP, exams, lectures, samples, and utils. The commit history shows updates to the README, TD, TP, exams, lectures, samples, and utils folders, as well as updates to the .gitignore file and license files. A QR code is visible on the left side of the repository page.

Commit	Message	Time
cfdc2c1	update readme	6 minutes ago
	add TD2	11 months ago
	update TP	11 months ago
	add exams 2018-2019 with code	2 years ago
	add lecture 06	11 months ago
	add lecture 06	11 months ago
	Move github files from TP to utils	2 years ago
	update .gitignore	2 years ago
	add gplv3 license file	15 minutes ago
	update licenses	14 minutes ago
	update README with Licenses	21 minutes ago

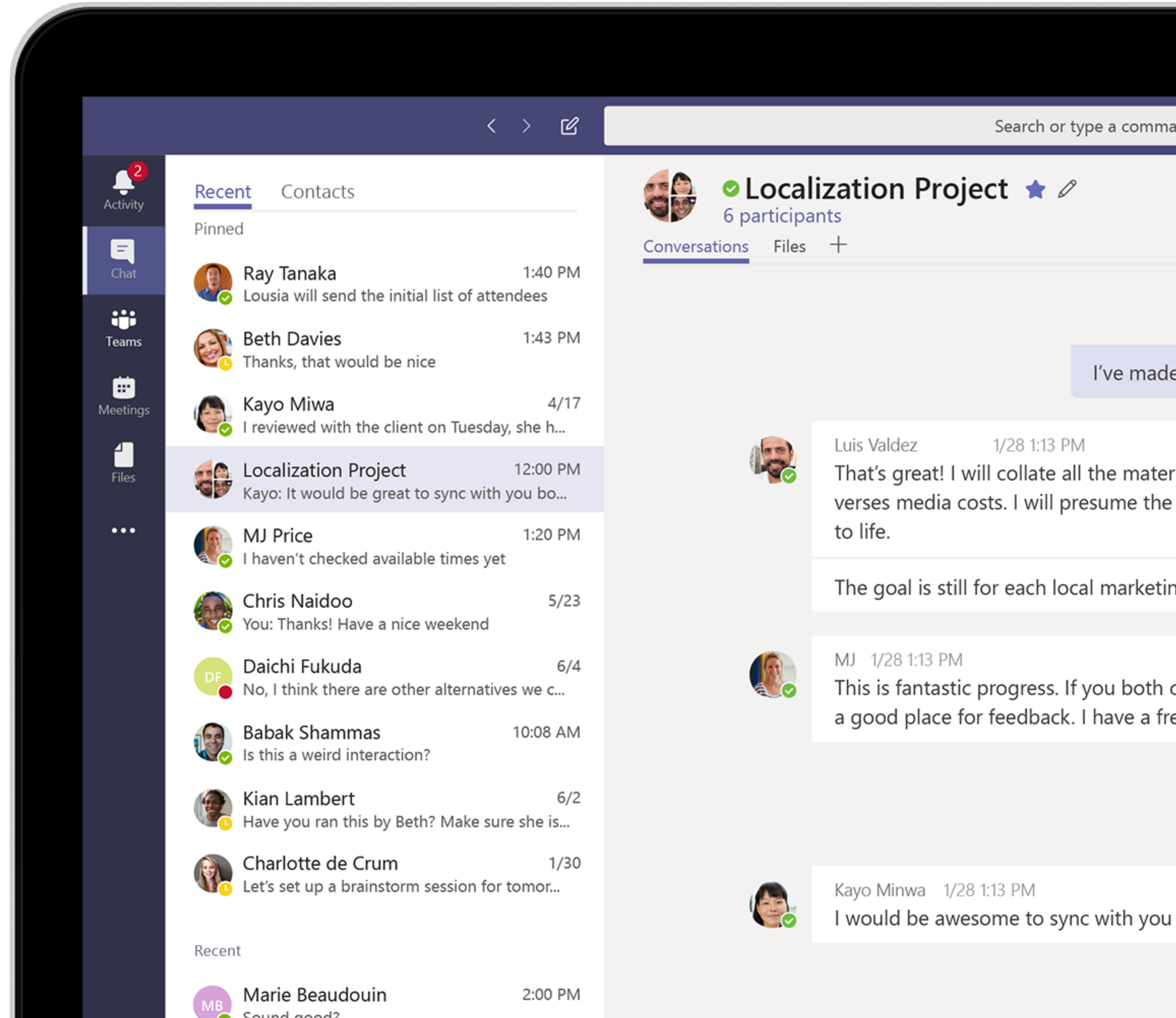


This work is licensed under [CC BY-NC 4.0](https://creativecommons.org/licenses/by-nc/4.0/) and [GPL version 3](https://www.gnu.org/licenses/gpl-3.0.html).



Microsoft TEAMS

for instant
messaging and
more if needed.



Some **quick surveys** to get to **know you!**



Connect to

www.wooclap.com/ITC313



A large commercial airplane is shown from a front-on perspective, flying over a runway. The sky is filled with golden, glowing clouds, suggesting a sunset or sunrise. The runway is a dark asphalt path with white dashed lines, leading towards the horizon. The overall scene is bright and atmospheric.

**This is your captain speaking
please put your phone in
airplane mode**



Today

Lecture #01
User-defined Data Types

Lecture #03
Polymorphism

Lecture #05
Templates



Lecture #00
Course Introduction



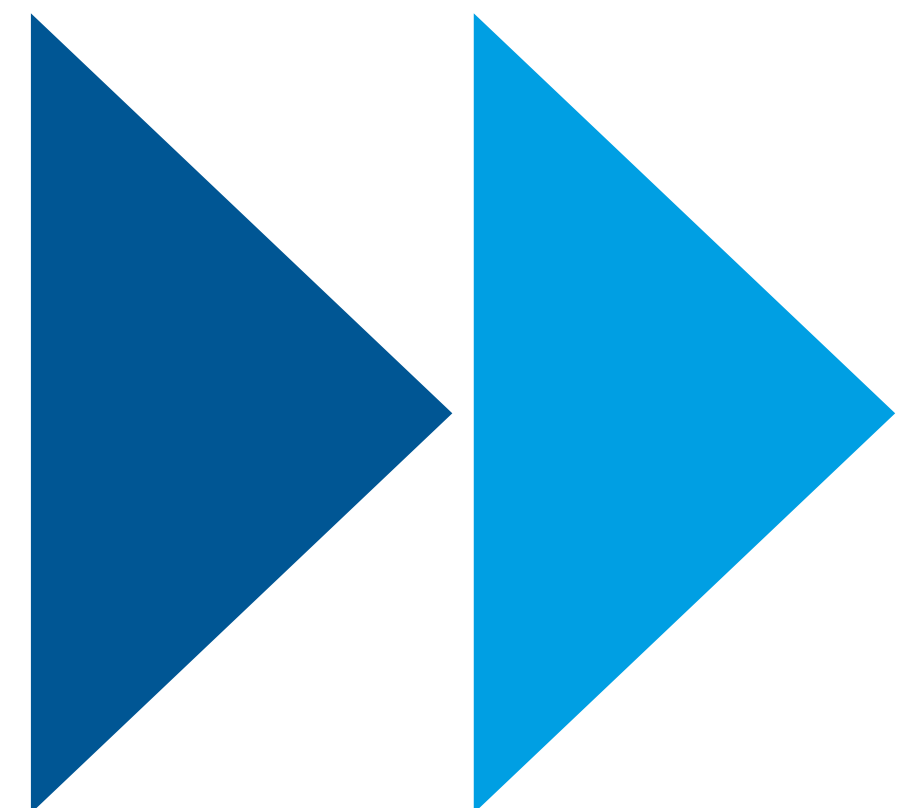
Lecture #02
Inheritance



Lecture #04
STL Containers



Lecture #06
Exceptions





<http://ginhac.com/teaching/ITC313/latest/00-introduction.pdf>

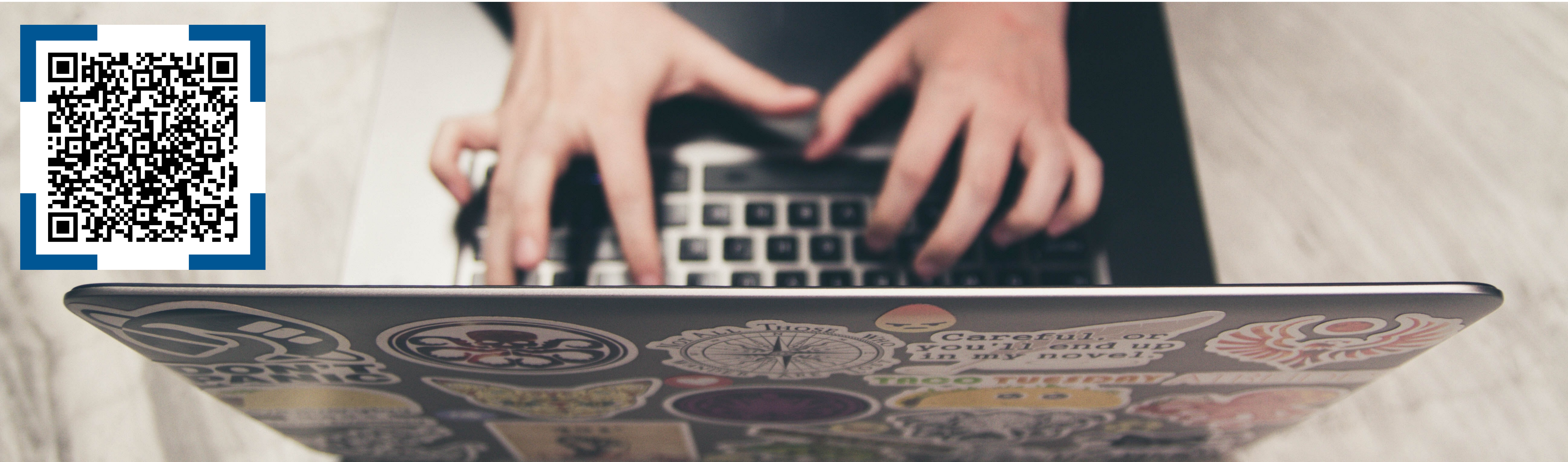


Photo by [NeONBRAND](#) on [Unsplash](#)

Give an overview of the **main topics** and introduce the **objectives** of the ITC313 course.

Enjoy! 😊

AGENDA

01 – About ITC313

02 – Why C++?

03 – The art of programming

04 – hello, world

Course introduction

The ITC313 course scheduling



Lectures (CM)

13 x 1.75 h
D. Ginhac



Tutorials (TD)

6 x 1.75 h
D. Ginhac



Labs (TP)

12 x 2 h
D. Ginhac
Other colleagues



Exams

Grading = Midterm + Final
+ Homework

Your learning objectives are

“What you should be able to do at the end of the course that you couldn't do before.”

1

Beginner
(aka Padawan)

Discover the basic features
to write good C++.

2

Intermediate
(aka Jedi knight)

Master the fundamentals
of C++ coding.

3

Expert
(aka Jedi Master)

Be comfortable with reading
and writing modern C++.

How to **achieve your learning objectives?**



Listen carefully.

Lecture slides, Code examples, Tutorials, Lab works, ... are all [available on **GitHub**](#).

You are strongly encouraged to [attend](#) and [participate actively](#) in the **lectures**, the **tutorials**, and the **labs**.



Be proactive.

My **slides** are always **very sparse**. So, you must [take notes](#) to keep track of my talk.

You can also use **your smartphone** to [take screenshots](#) of the blackboard.

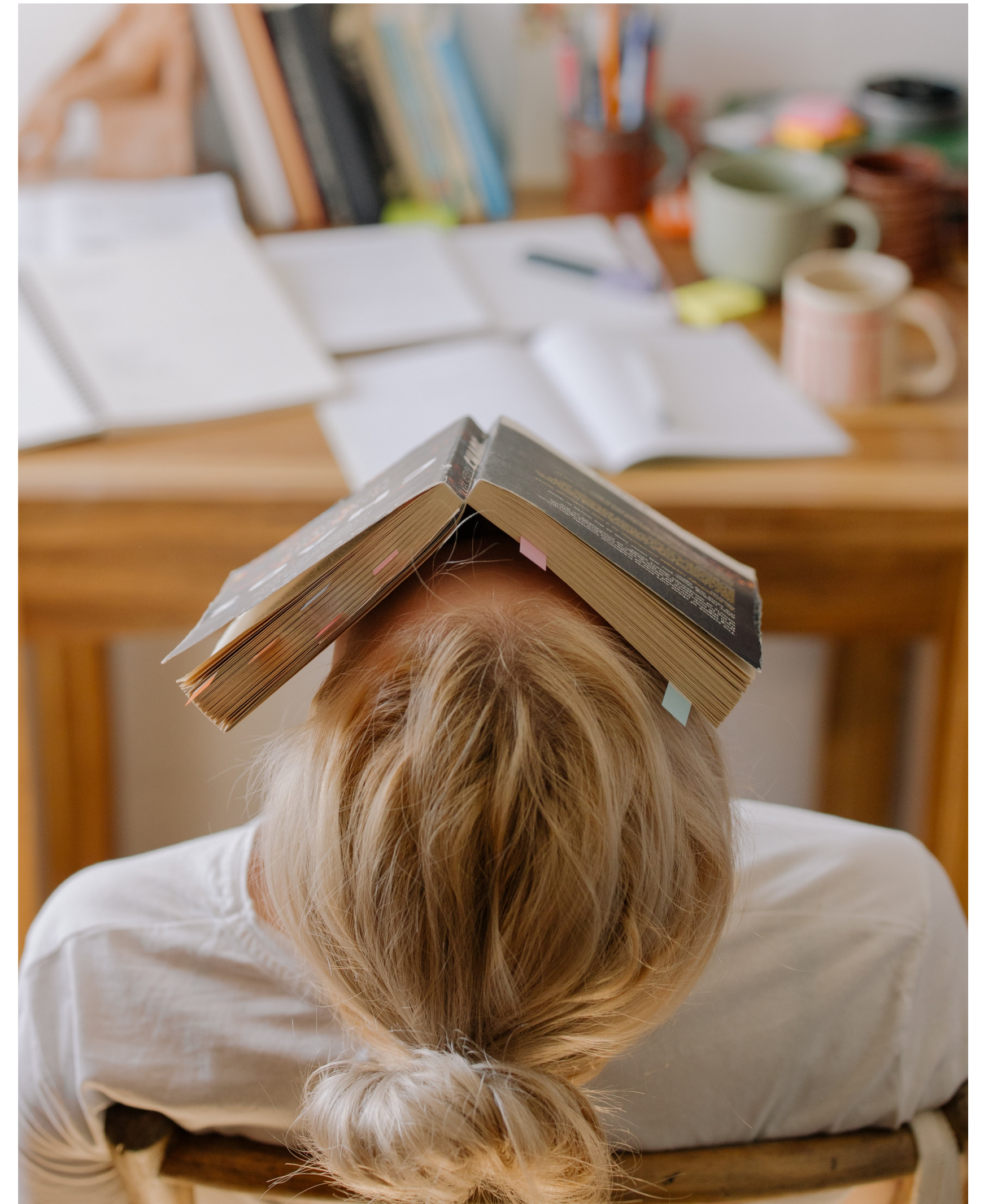
You can also use **your laptop** to download, review, modify and test the [C++ code examples](#).



Ask when unclear.

Learning / Teaching how to code in C++ is not really easy.

So, **be active** and [ask me](#) when anything appears unclear! I will spend time to [explain again](#) and [again](#).



Questions



AGENDA

01 – The ITC313 program

02 – Why C++?

03 – The art of programming

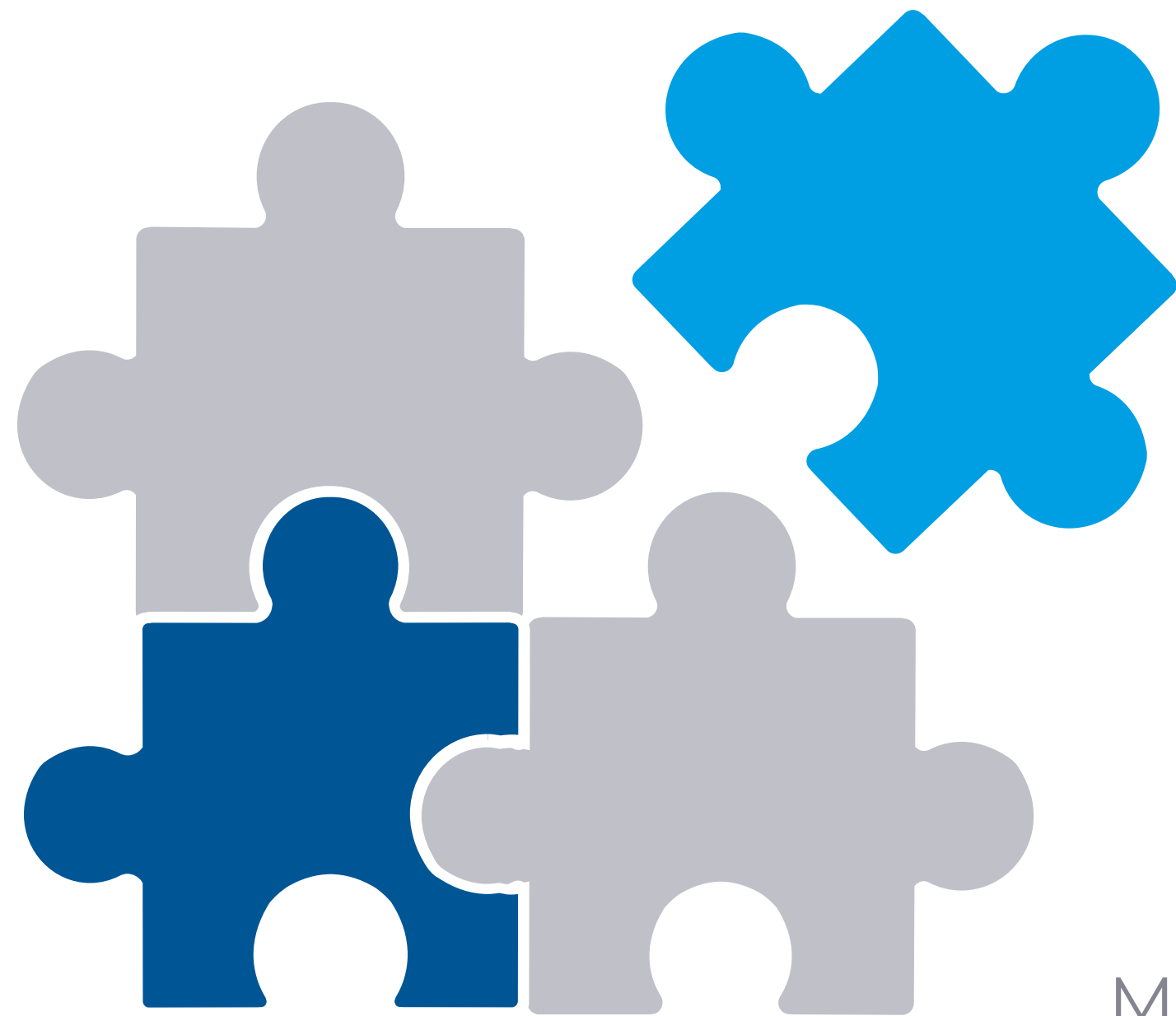
04 – hello, world

Course introduction

C++ = OOP language

Object Oriented programming (OOP) is a **programming paradigm** in which the programs are structured around **objects** rather than functions and logic

The four pillars of OOP



Abstraction

Mechanism of hiding the implementation details from the user, only the functionality will be provided to the user.

Encapsulation

Mechanism, also known as Data hiding, that refers to the bundling of data/methods into a single coherent unit.

Inheritance

Mechanism of basing an object upon another object that allows sharing of properties and behaviors and implementation of new functionalities.

Polymorphism

Mechanism of assigning a different meaning or usage to something in different contexts.
Includes Overloading and Overriding.

Why **C++** programming ?



Powerful



Relevant



Popular



Jobs

The C language was born in 1972...

Closely tied to **the dev of UNIX OS.**

The C programming language has been developed to move the UNIX kernel code **from assembly to a higher level language** while guaranteeing **high performance.**



Photo by [National Inventors Hall of Fame](#)

Dennis RITCHIE
/ Inventor of C

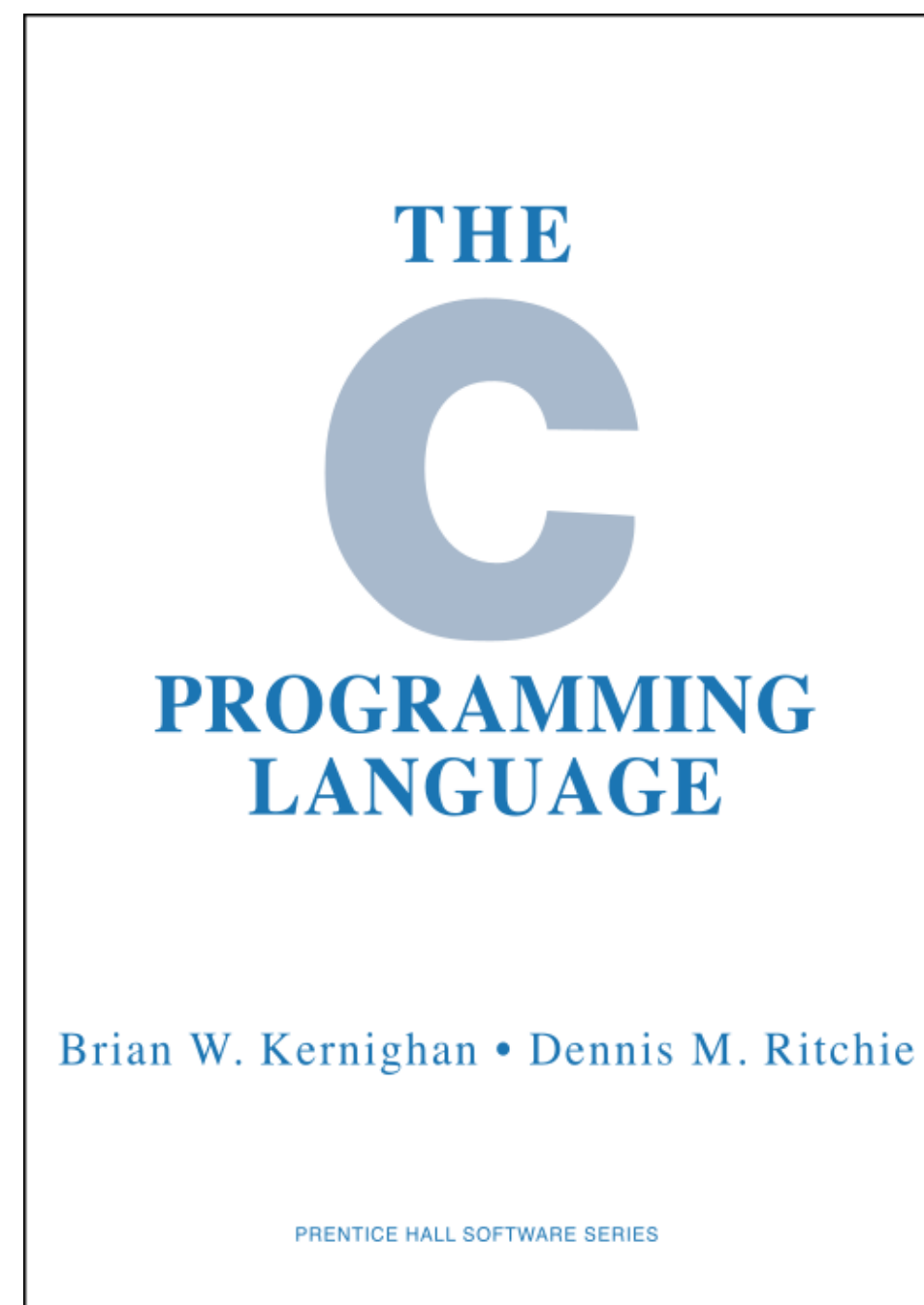


Photo by [National Inventors Hall of Fame](#)

Ken THOMSON
/ Designer of UNIX OS

... and C++ in 1983.

Originally defined as an [extension](#) of the C language called “**C++ = C with Classes**”.



Photo by [Bjarne Stroustrup](#)

Bjarne STROUSTRUP
/ Inventor of C++



Standardization

Updated standard [every 3 years](#).
C++20 is the current release (20/12).



C++11 was a revolution

C++ before 2011 was C with classes.
C++ since 2011 is a [new OOP language](#).



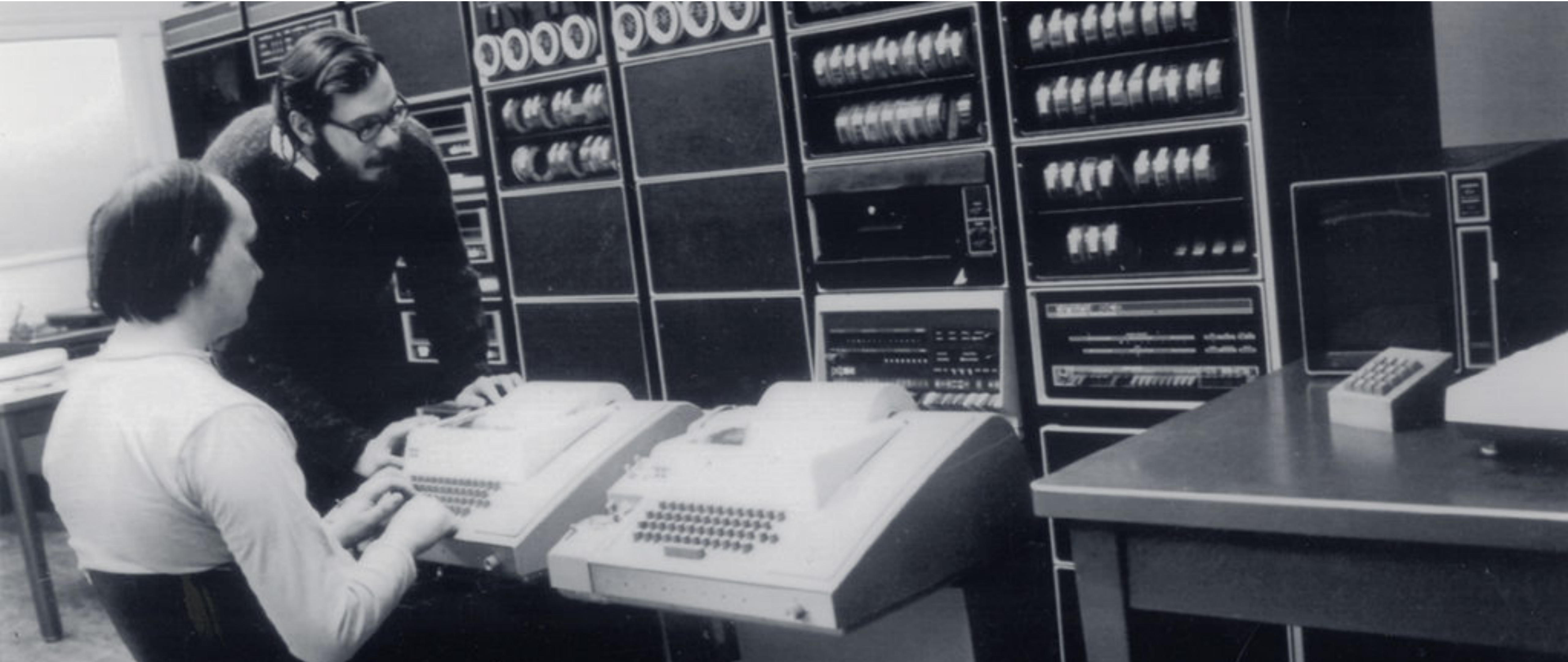
Modern C++

Modern C++ stands for C++ that is based on [C++11 and beyond](#).
Most C++ compilers support C++11 to C++17 (20?) features.

Today

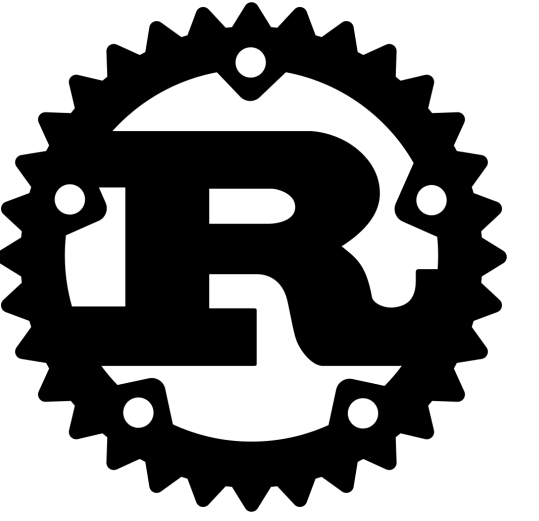
C/C++ still relevant?

Photo by [Computer History](#) - DEC - PDP-11- Ken Thompson and Dennis Ritchie





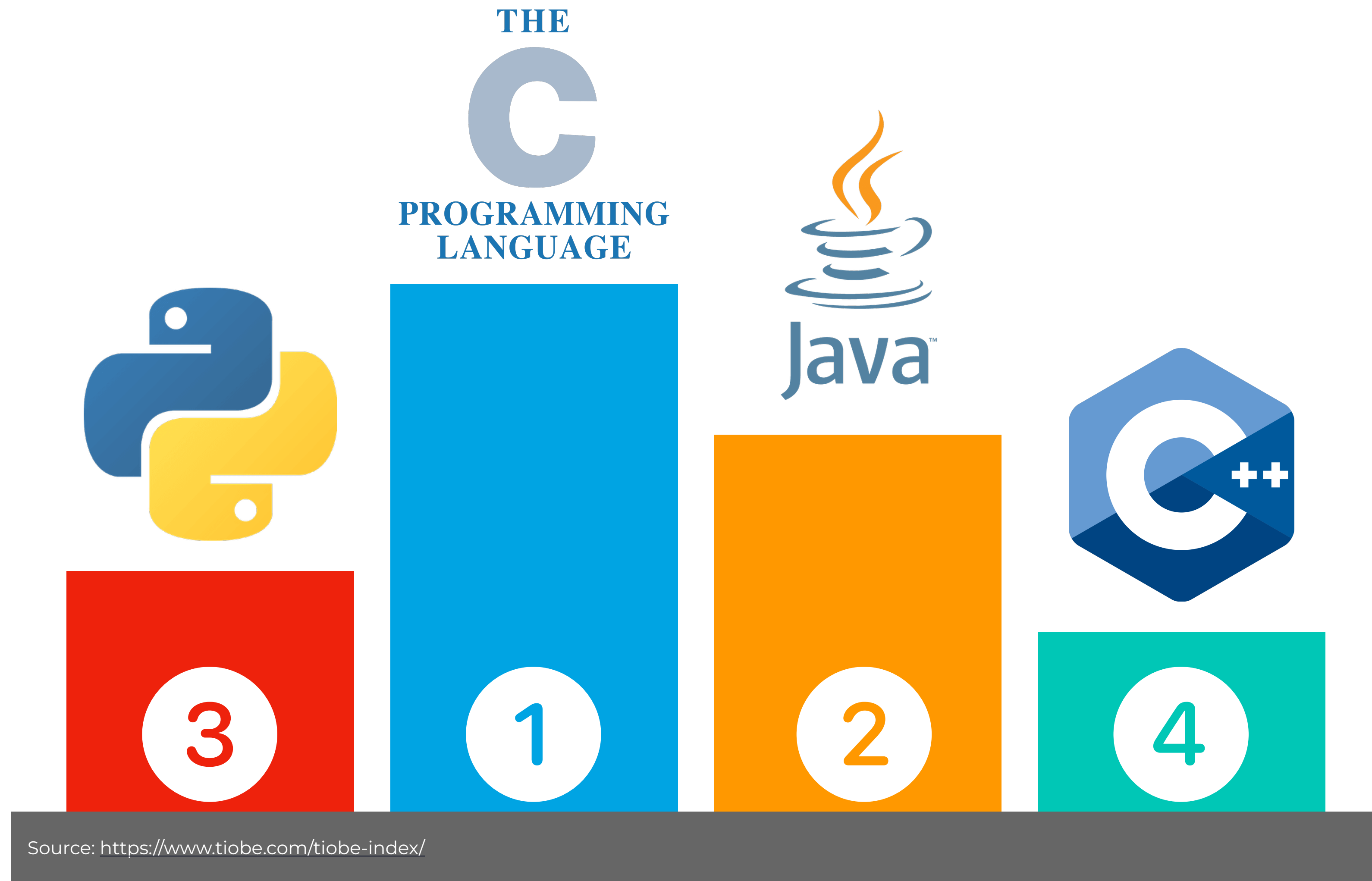
Java™



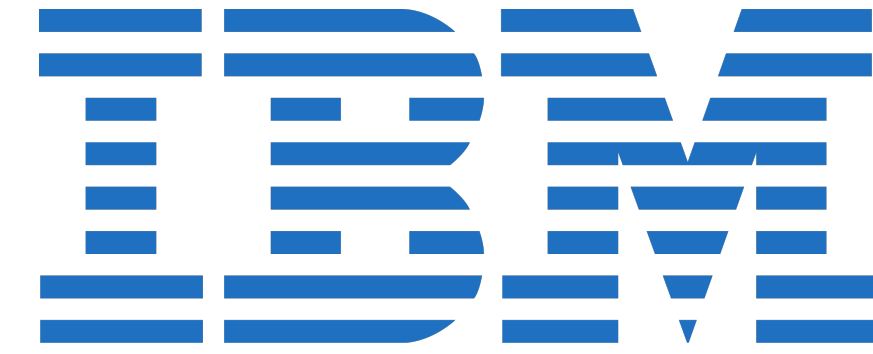
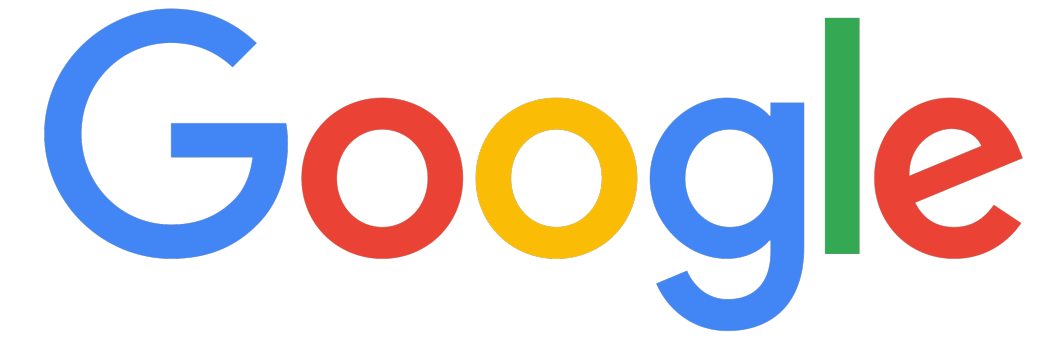
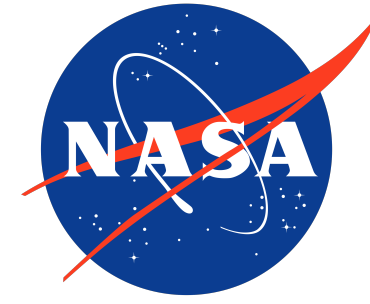
Welcome to the **jungle!**

About 700 notable languages (source [wikipedia](#))

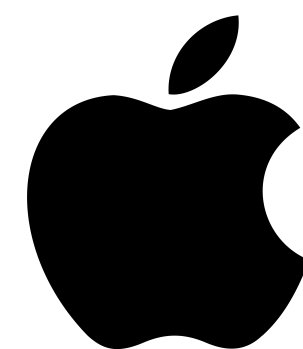


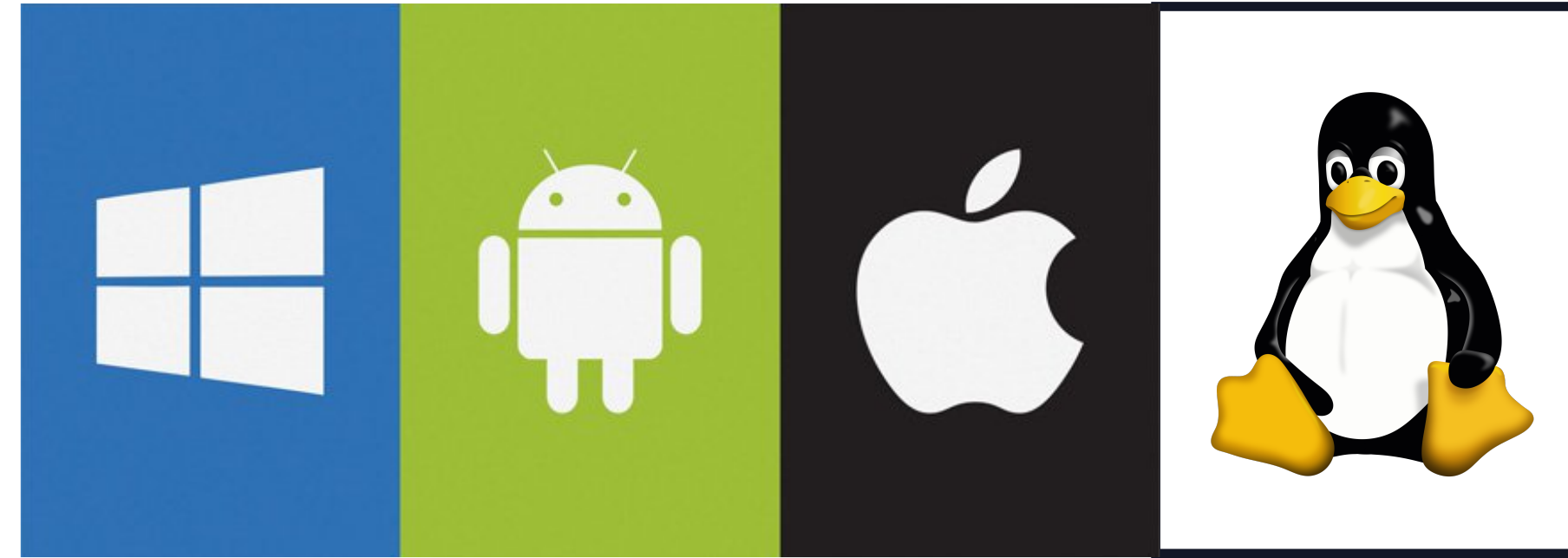


C/C++ are still among the **most popular languages!**

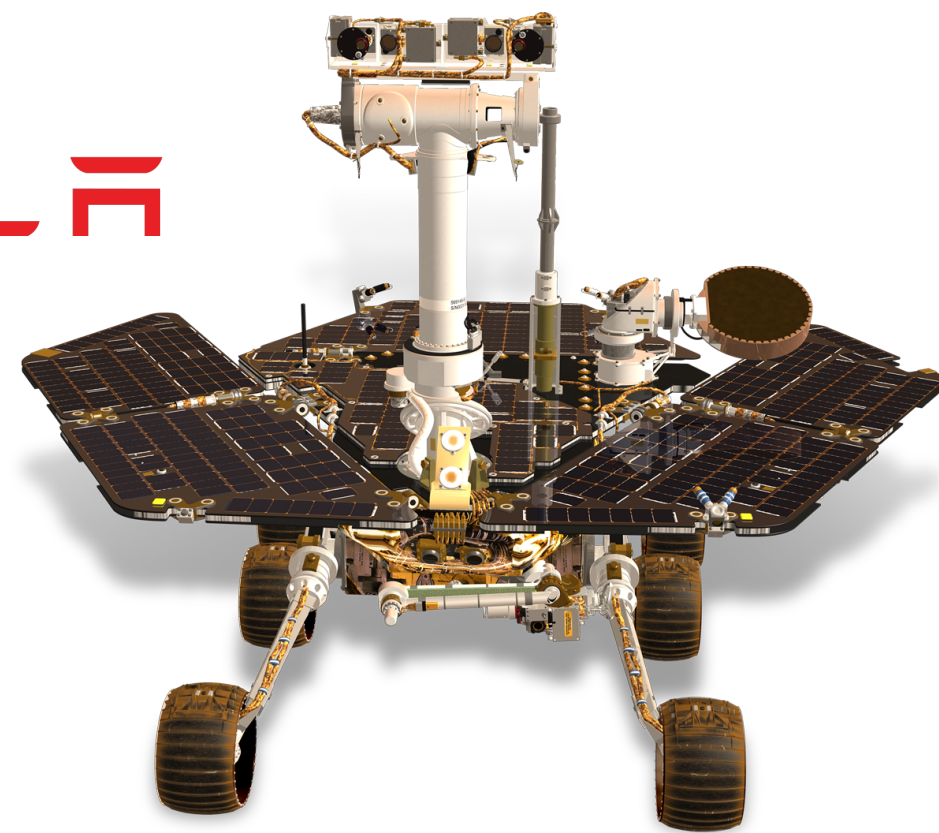


Used by **famous companies...**

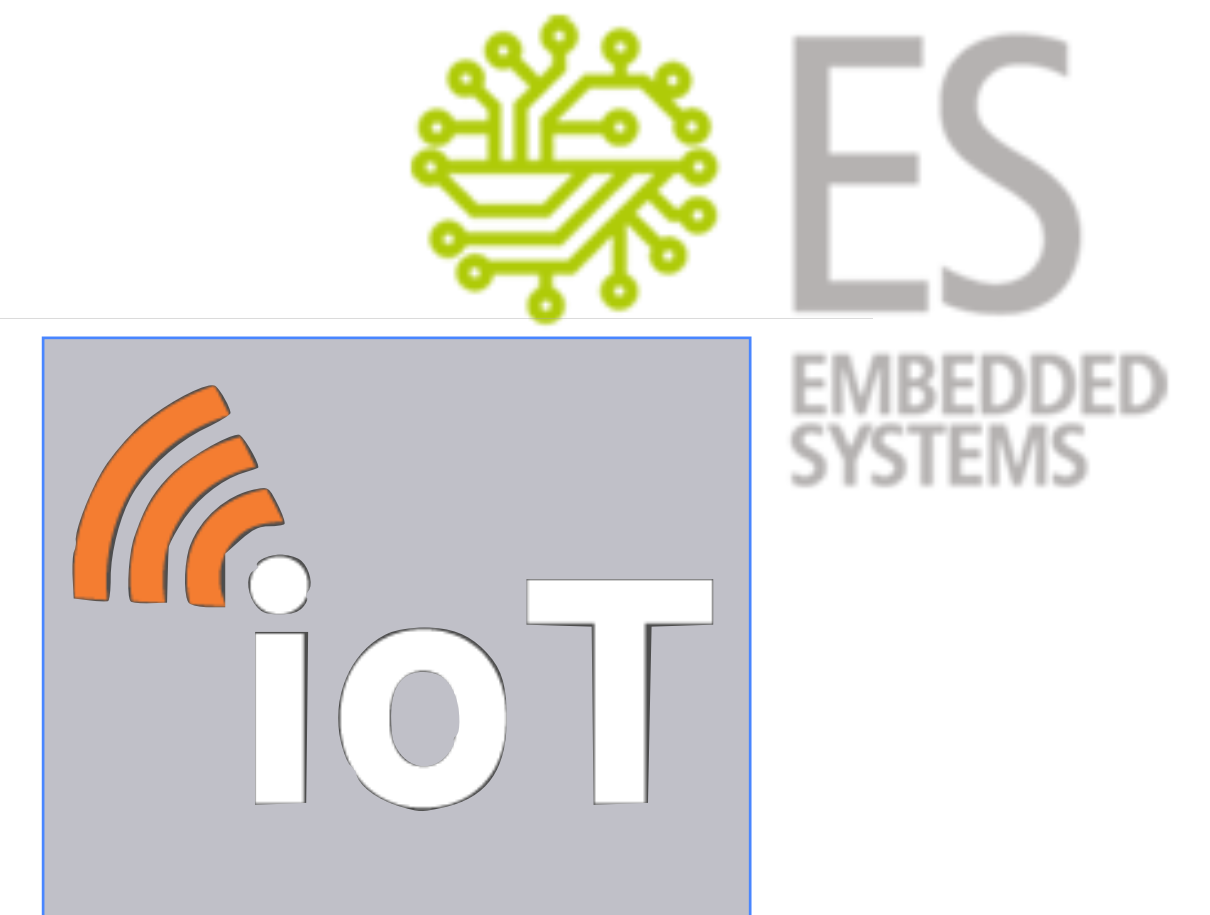




TESLA



... to **make great apps** with ❤️.



YES

Learning C++ is relevant.

Questions



AGENDA

01 – The ITC313 program

02 – Why C++?

03 – The art of programming

04 – hello, world

Course introduction

*Programming is the art of
telling another human what
one wants the computer to do.*

Donald KNUTH
/ The art of computer programming



Photo by [Vivian Cromwell](#) for [Quanta Magazine](#)



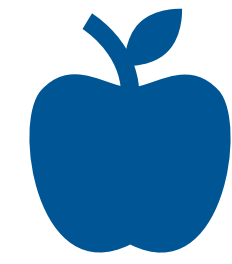
Correctness

It is just a matter of syntax to write a correct sequence of instructions.

“Does the code you wrote (or someone else) compile and run as intended?

Does the program behave correctly with no error ?”

01



Design

Even bad code can work!

“Does your code just do what it should?
How efficient/fast is your program?”

02

03



Style

Clean code is code that is easy to understand and easy to change.

“How understandable is your source code?
Can you make it more readable for humans?”

Make it **work**, make it **right**, make it **clean**



Correctness

Make your code run correctly.

Split your problem into short units and focus on one thing at a time.

Write well-organized classes containing only the required code.

Use the complete online reference for C++ and standard library <https://en.cppreference.com>

Use also C++ cheatsheets containing basic syntax <https://github.com/mortennobel/cpp-cheatsheet>



Design

Think code quality.

Test every piece of code.

Avoid duplication in the code - "Don't repeat yourself!"

Consider code refactoring to get more efficient code.



Style

Increase readability.

Use descriptive names for variables, classes, functions. Express ideas directly in code.

Follow naming conventions. <http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines.html>

Be careful: code says what is done, and comments what is supposed to be done.

Compilers don't read comments and neither do many programmers.

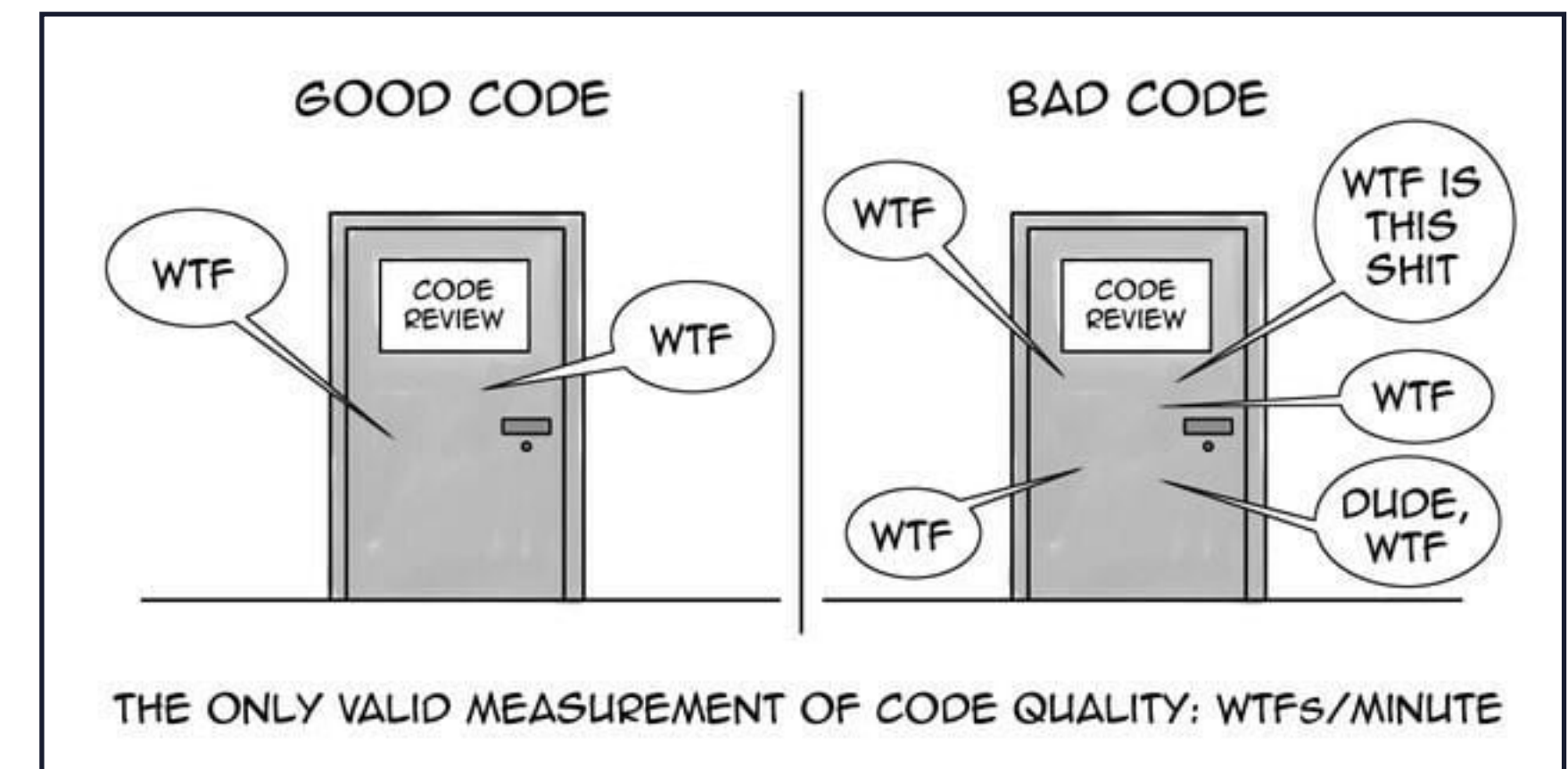


Photo by Elyess Eleuch on dev.to

Questions



AGENDA

01 – The ITC313 program

02 – Why C++?

03 – The art of programming

04 – hello, world

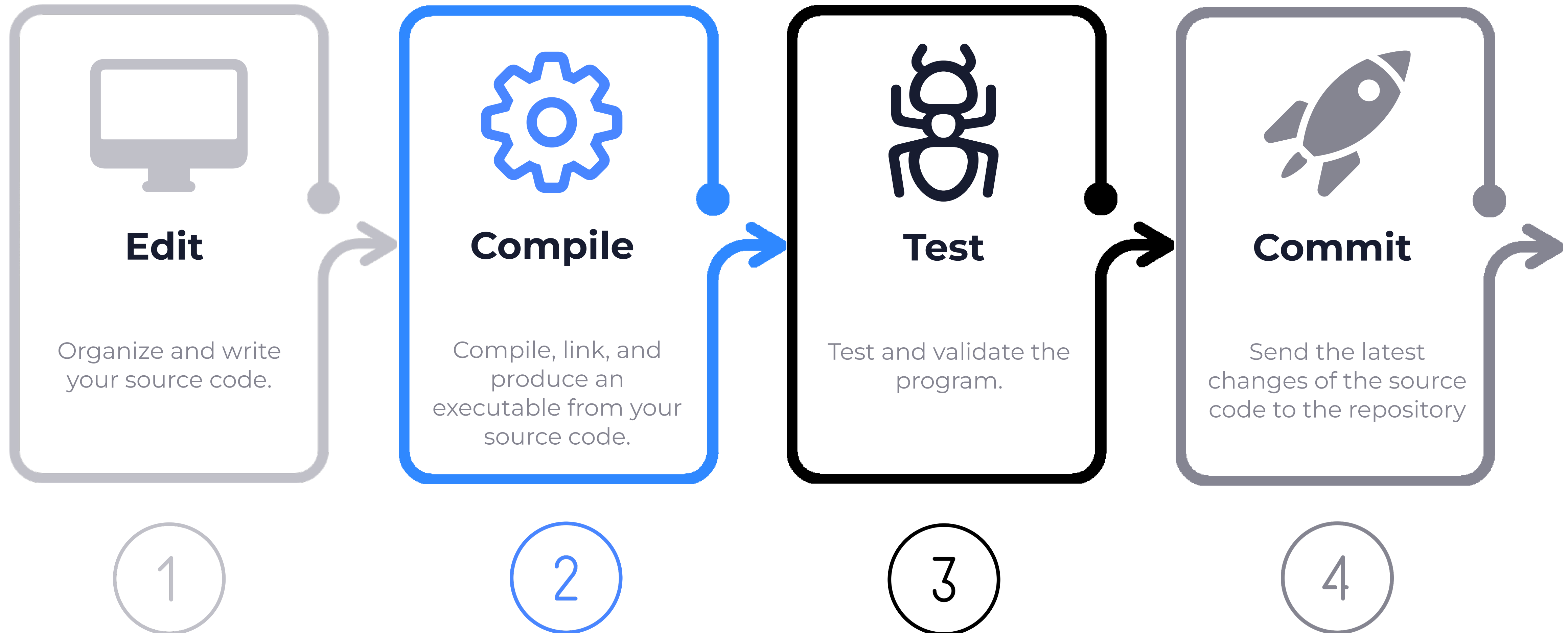
Course introduction



hello, world

needs more than just code.

Automate your workflow to write your first “hello, world” program



C++ online editor, IDE, compiler, interpreter, and REPL

Code, collaborate, compile, run, share, and deploy C++ and more online from your browser

Sign up for the full experience

C++

Run

Share

```

1 #include <iostream>
2
3 int main() {
4     std::cout << "Hello, world!";
5     return 0;
6 }

```

```

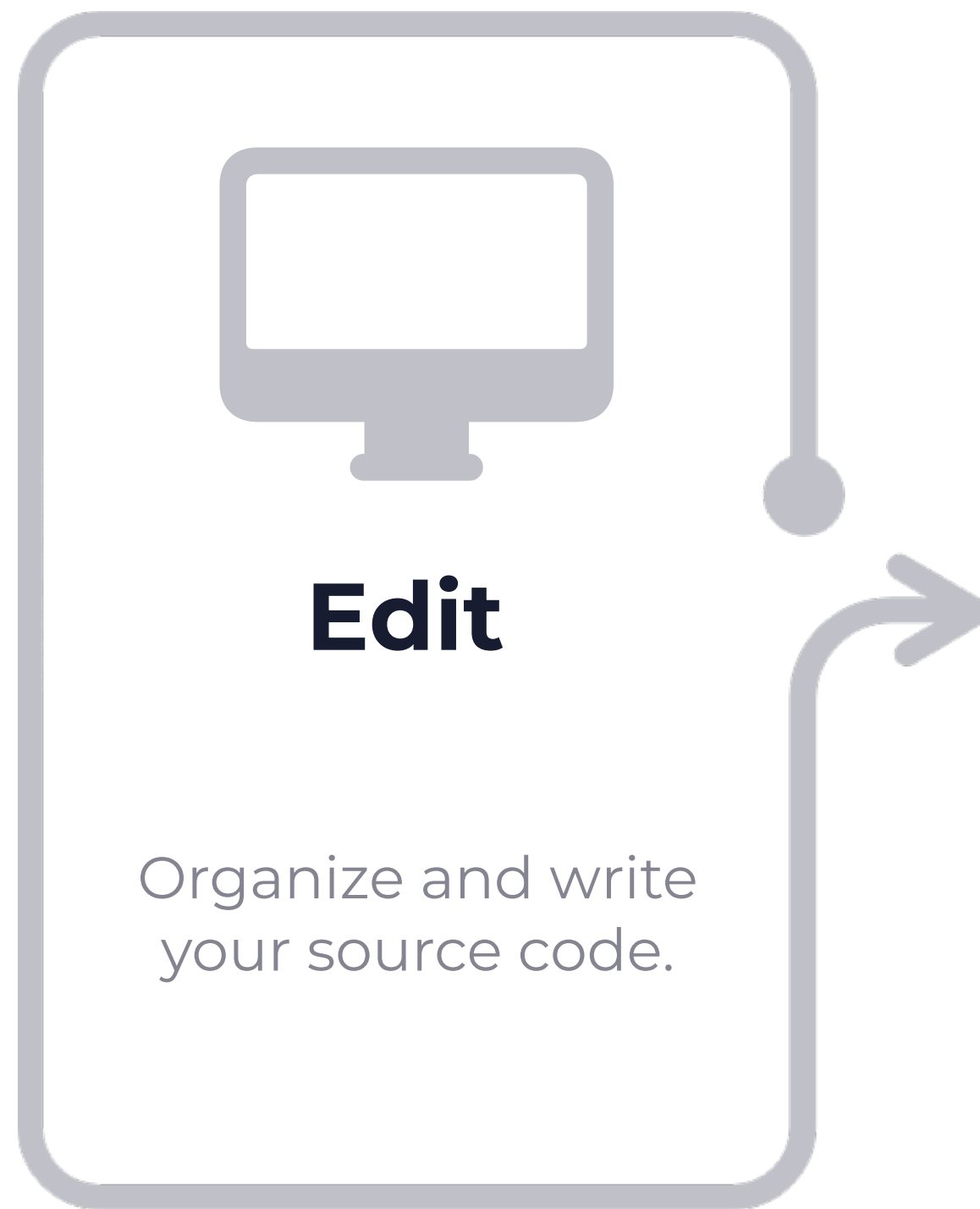
> clang++-7 -pthread -std=c++17 -o main main.cpp
> ./main
Hello, world!

```

Read-Eval-Print Loop and Online compilers

<https://repl.it/languages/cpp>

Write source files



1



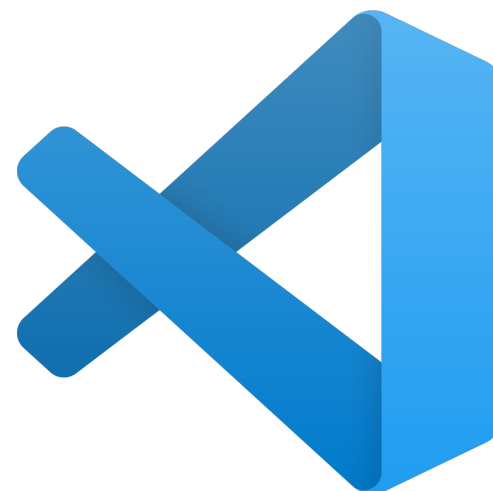
<https://www.sublimetext.com>



<https://atom.io>



<https://www.codeblocks.org>



<https://code.visualstudio.com>

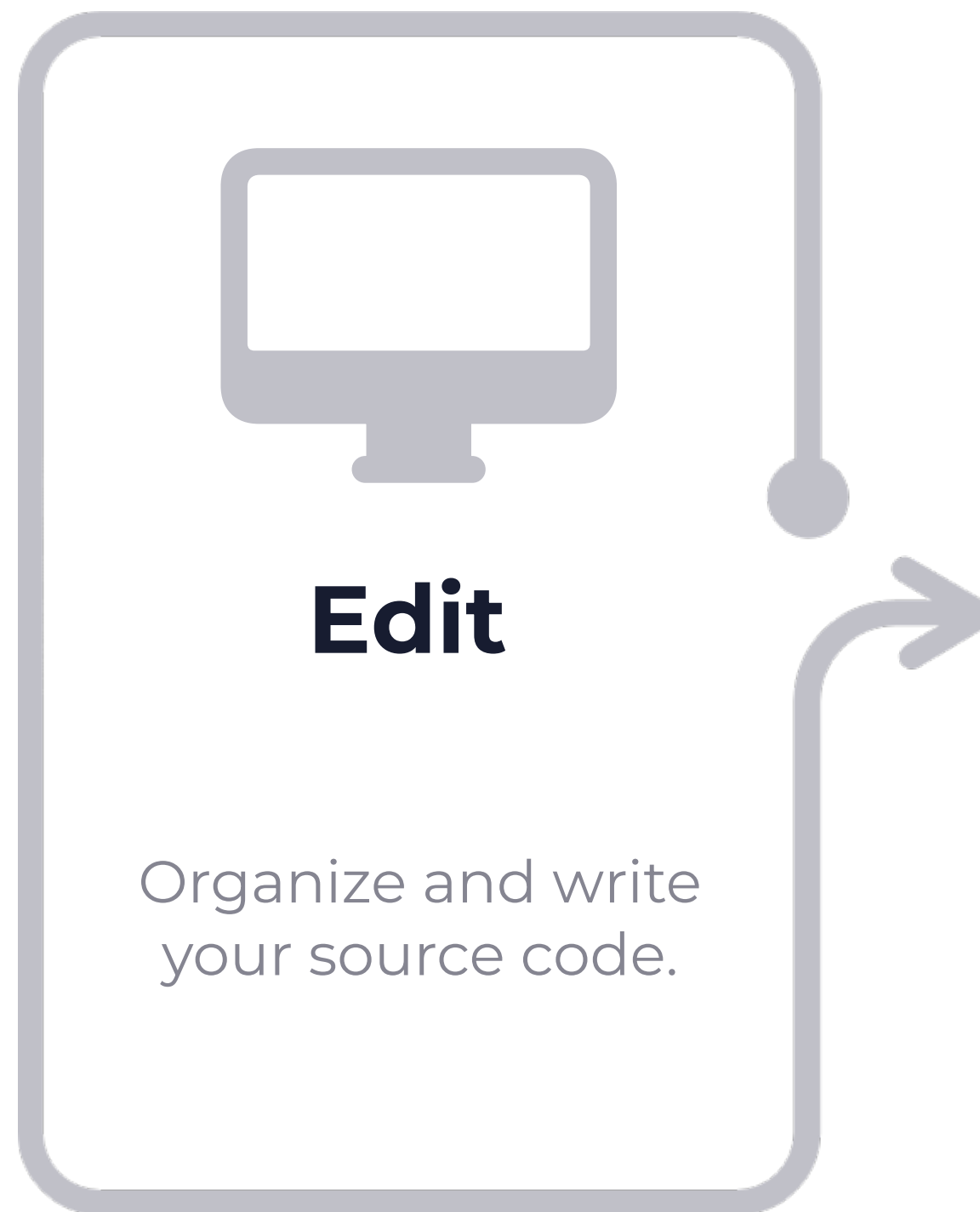


<https://www.jetbrains.com/clion/>



<https://replit.com>

Your first “hello, world” program



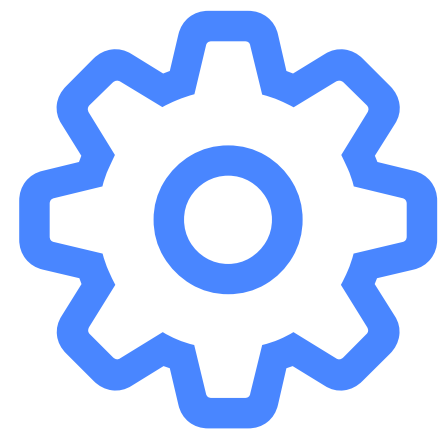
1

```
hello-world.cpp UNREGISTERED
OPEN FILES
x hello-world.cpp
1 #include <iostream>
2
3 // The main function
4 int main(int argc, char const *argv[]) {
5     // Print "hello, world"
6     // on the standard output stream (monitor)
7     std::cout << "hello, world" << std::endl;
8     // End of the program
9     return 0;
10 }
11
```

ok, © tabnine, Line 11, Column 1 Unix Spaces: 4 C++

Written with 

From Source files to Executable



Compile

Compile, link, and produce an executable from your source code.

2



GNU Compiler Collection

<https://gcc.gnu.org>



Minimalist GNU for Windows

<http://www.mingw.org>



LLVM Compiler Infrastructure

<https://clang.llvm.org>

The C++ compilation model



```
-zsh — d0m  1871
→ 00-helloworld ls
hello-world.cpp
→ 00-helloworld clang++ hello-world.cpp
→ 00-helloworld ls
a.out          hello-world.cpp
→ 00-helloworld ./a.out
hello, world
→ 00-helloworld
```

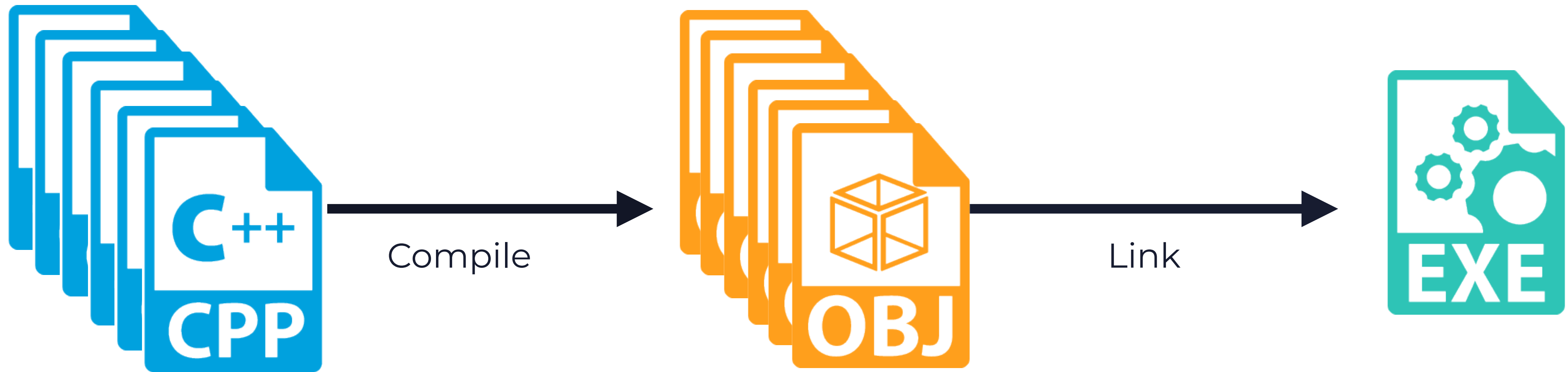
The C++ compilation model



COMPILATION
= **4 steps**

0. Code edition	-> .cpp	10 lines (246B)
1. Preprocessing	.cpp -> .cpp	45846 lines (1.6 MB)
2. Compilation	.cpp -> .s	1619 lines (56 KB)
3. Assembler	.s -> .o	12 KB
4. Linker	.o -> executable	39 KB

Dealing with multiple files



AUTOMATE
THE COMPILATION
with **MAKEFILES**

Make is a **build automation tool** that automatically builds executable programs and libraries from source code by reading files called [Makefiles](#).

You specify into the Makefile "What you want to make" and "How it goes about making it".

And then, you just run "[make](#)".



Upcoming demo

Track bugs

The [bug tracking process](#) involves detecting bugs during testing, assessing their impact, determining their causes, and [fixing](#) them.



Test

Test and validate the program.

3

Using a **debugger** to monitor your program.

An interactive debugger analyzes how a program flows and helps identifying incorrect running code that can cause unexpected behavior or crash.

Key concepts are breakpoints, watchpoints, stepping, viewing data, ...

The most used C++ debuggers are [gdb](#) from GNU and [lldb](#) from LLVM.

Writing and running **unit tests** to prevent bugs.

A unit test is an automated test that verifies a small piece of code in an isolated manner. It helps finding problems early in the development cycle and makes the final debug step easier.

Key concepts are assertions (boolean expressions).

The most used C++ testing frameworks are [Google Test](#), [Boost](#), [Catch2](#).



Upcoming demo

Software versioning

Software versioning is the process of numbering different releases of a particular code source. It allows programmers to know when changes have been made and to track changes applied in the code.



Commit

Send the latest changes of the source code to the repository

4

What is git?

Git is the most commonly used version control system.

Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to.

See <https://githowto.com> for a guided tour that walks through the fundamentals of Git.

Why using git is so crucial?

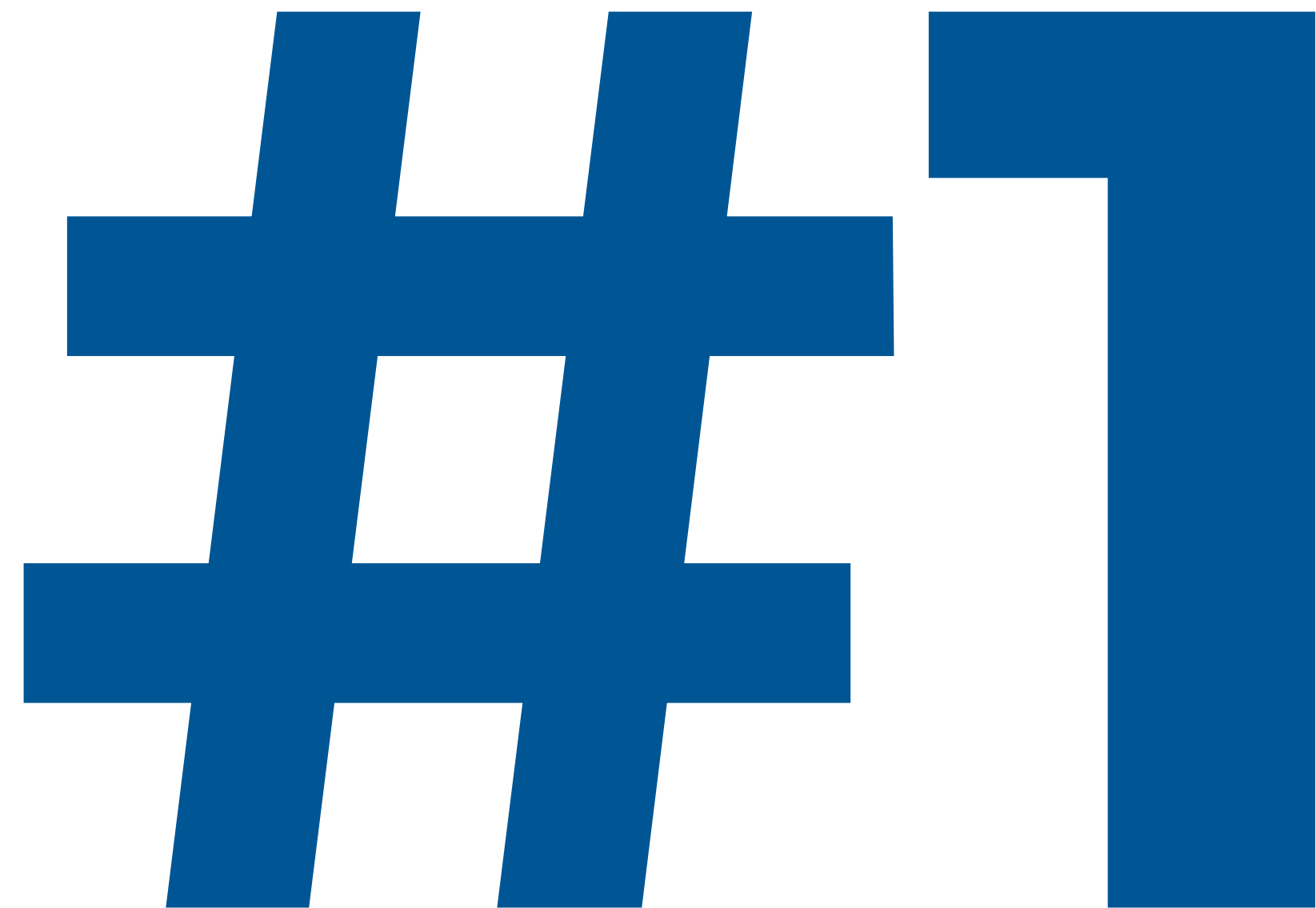
Git is the key to a successful and modern development workflow.

Git helps you to maintain the backup of your source code.

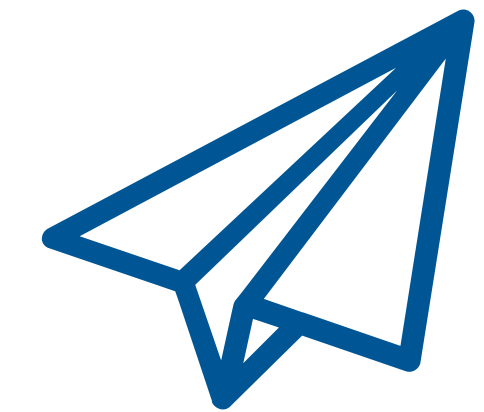
Git helps you to collaboratively work with other developers.



Upcoming demo



A FIRST TAKE HOME MESSAGE



Modern IDE can help you write code quickly.

But **high-quality C++ programming** is more about a deep understanding of OOP than mastering the language syntax.

Questions





Contacts

Pr. Dominique Ginhac

dginhac@u-bourgogne.fr

Come visit us at

<https://github.com/dginhac/esirem-itc313>

This work is **licensed** under a
Creative Commons Attribution-NonCommercial 4.0 International License.

<https://creativecommons.org/licenses/by-nc/4.0/>

