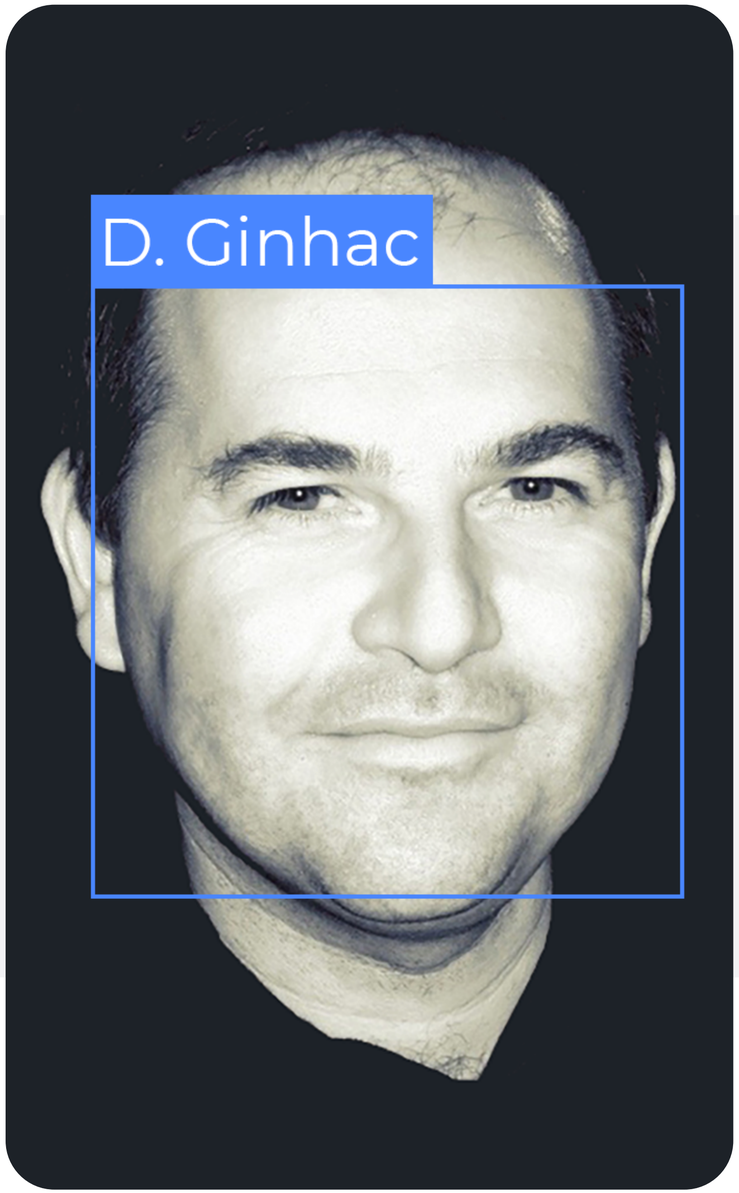
Welcome

oct. 2021



HI, I'M D. GINHAC

I will teach you some Computer Programming modules this year and next year.

I am also a researcher in Artificial Intelligence and Computer Vision on the edge.

dginhac@u-bourgogne.fr

@dginhac

ESIREM
ÉCOLE SUPÉRIEURE D'INGÉNIEURS
NUMÉRIQUE ET MATÉRIAUX

École associée Polytech

Fundamentals of C++.

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

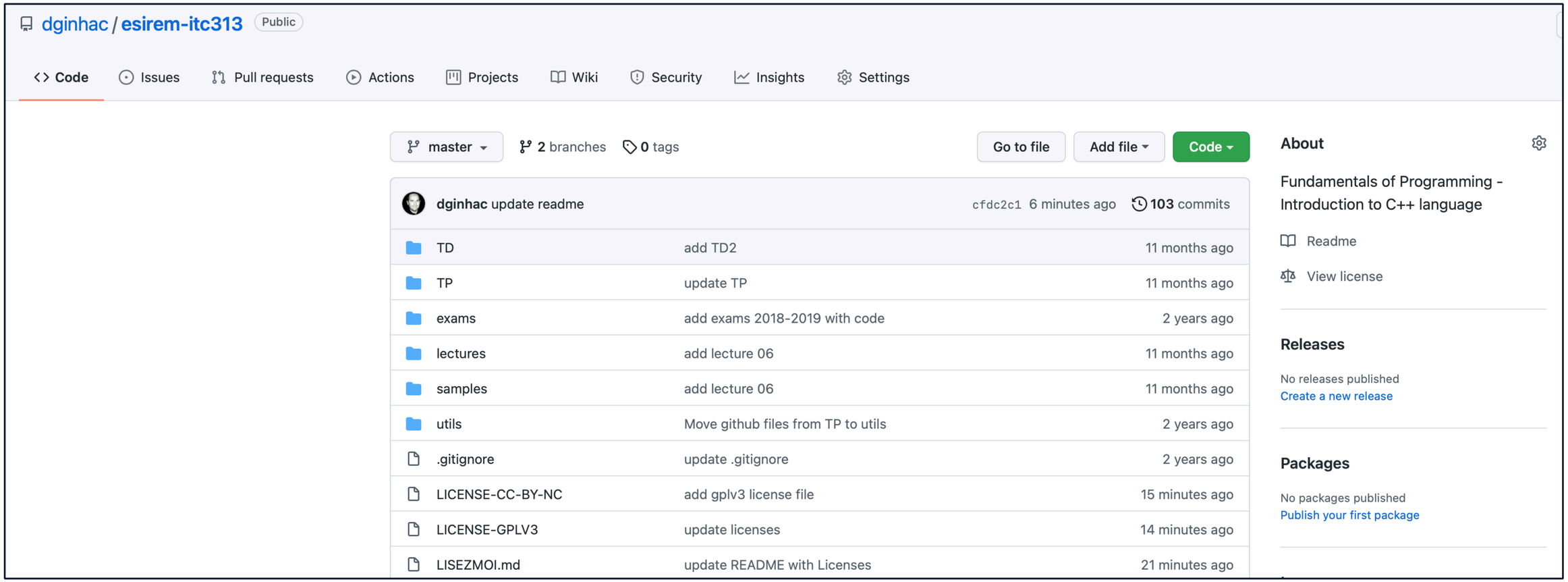
Some quick surveys to get to know you!

Connect to

www.wooclap.com/ITC313

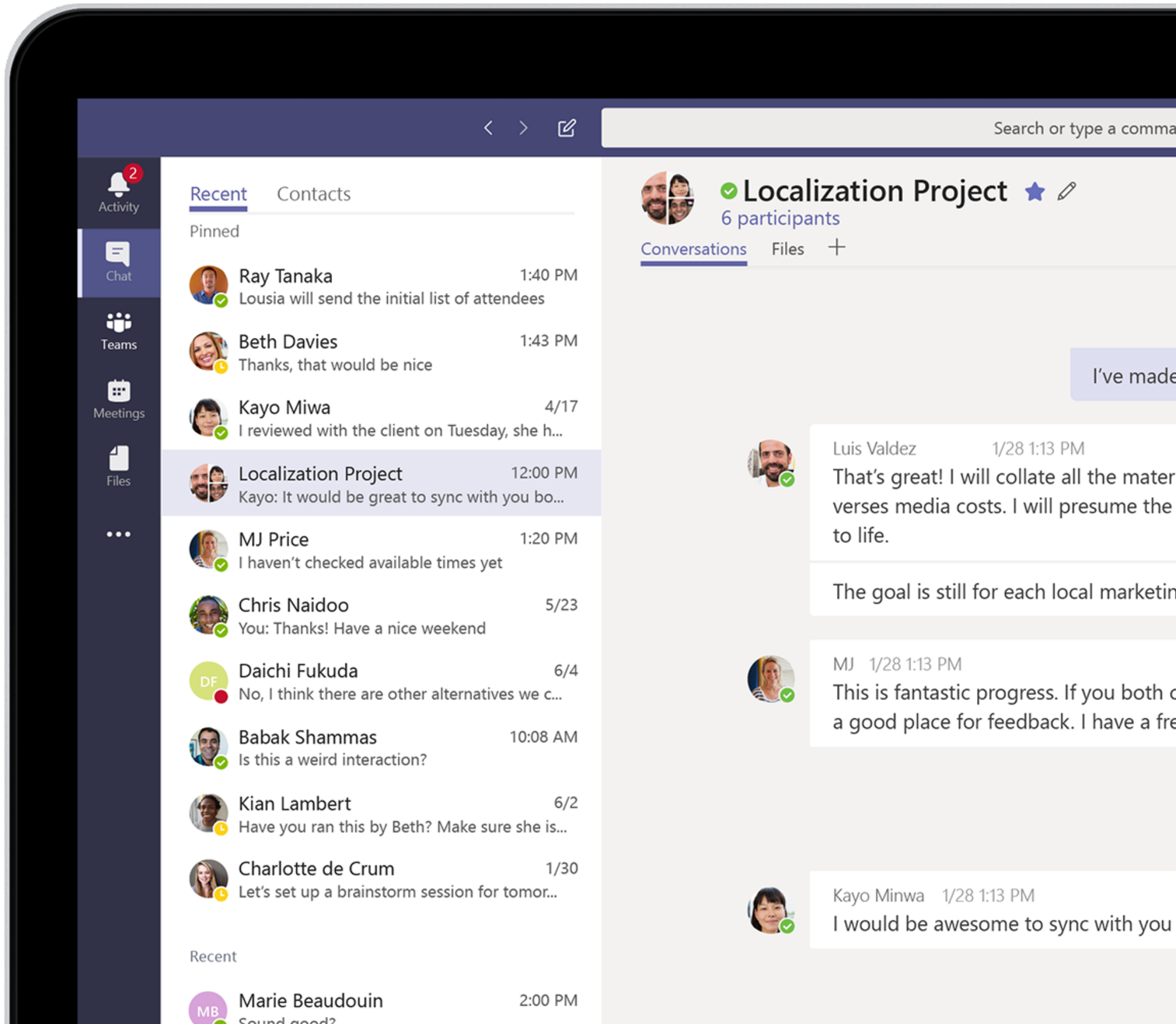
Visit https://github.com/dginhac/esirem-itc313

Everything available on GitHub.



Microsoft TEAMS

for **instant messaging** and more if needed.



Today

Lecture #00
Course Introduction

Lecture #01
User-defined Data Types

Lecture #02
Inheritance

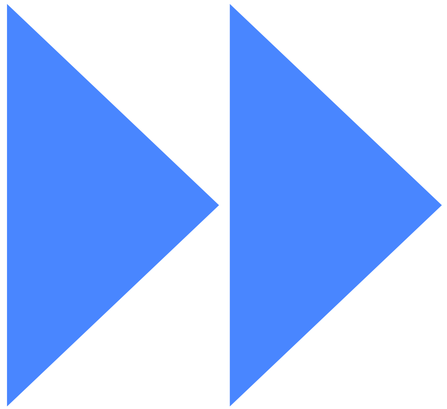
Lecture #03
Polymorphism

Lecture #04
STL Containers

Lecture #05
Indirection

Lecture #06
Templates

Lecture #07
Exceptions



Lecture #00

Course introduction

http://ginhac.com/teaching/ITC313/2021-2022/00-introduction.pdf

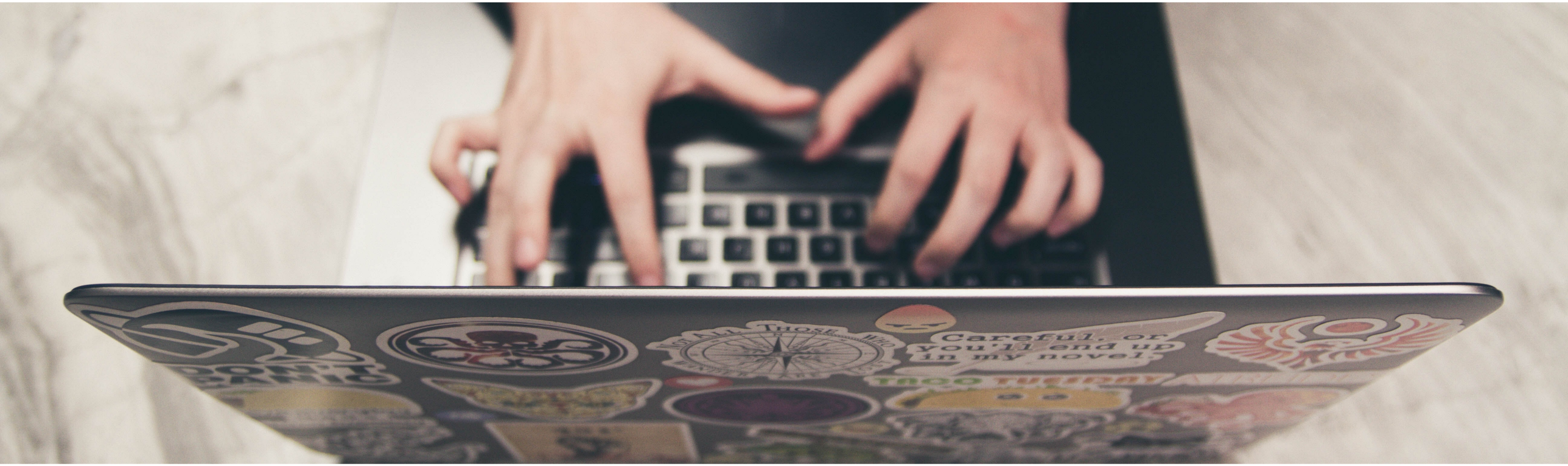


Photo by NeONBRAND on Unsplash

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

Enjoy! ☺

AGENDA

01 – The ITC313 program

02 – Why C++?

03 – The art of programming

04 – hello, world

The ITC313 course scheduling

Photo by Xin Wang on Unsplash

Lectures (CM)

13 x 1.75 h
D. Ginhac

Tutorials (TD)

6 x 1.75 h
D. Ginhac

Labs (TP)

12 x 2 h
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.”

Beginner
(aka Padawan)

Discover the basic features to write good C++.

Intermediate
(aka Jedi knight)

Master the fundamentals of C++ coding.

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 often clean and do not contain everything I say. So you have total notes.

You can also use your smartphone to take screenshots of the blackboard.

You can also use your laptop to download, understand/update, and run 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.

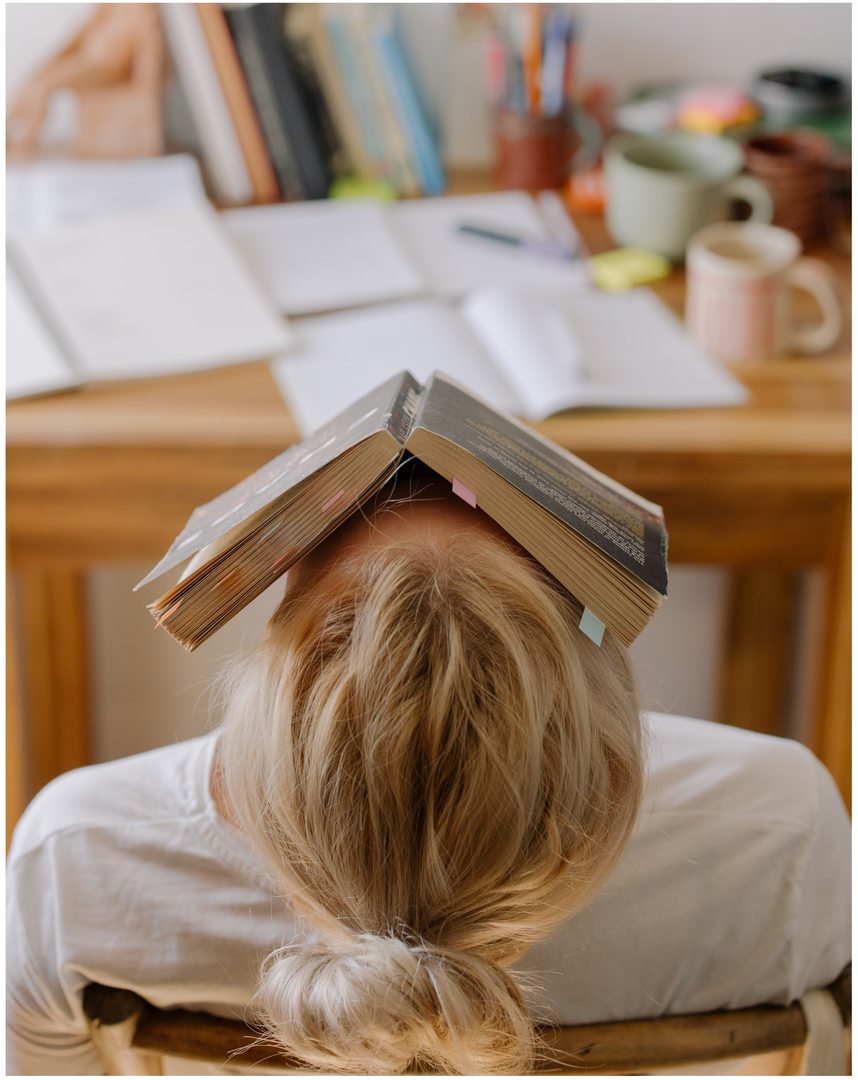


Photo by cottonbro from Pexels

Questions

AGENDA

01 – The ITC313 program

02 – Why C++?

03 – The art of programming

04 – hello, world

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

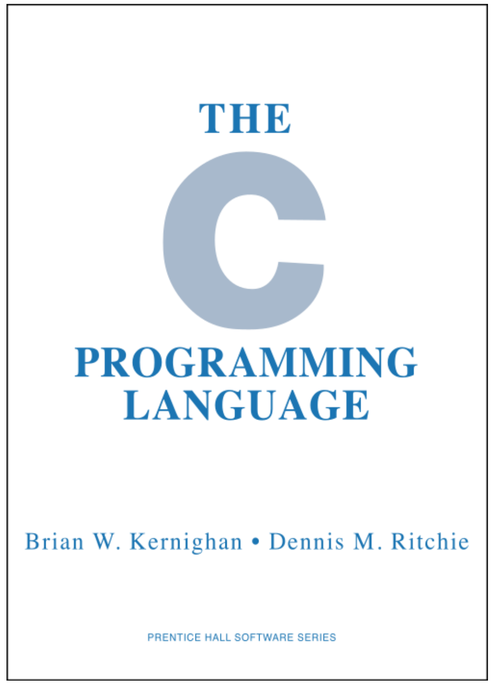


Photo by National Inventors Hall of Fame

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.

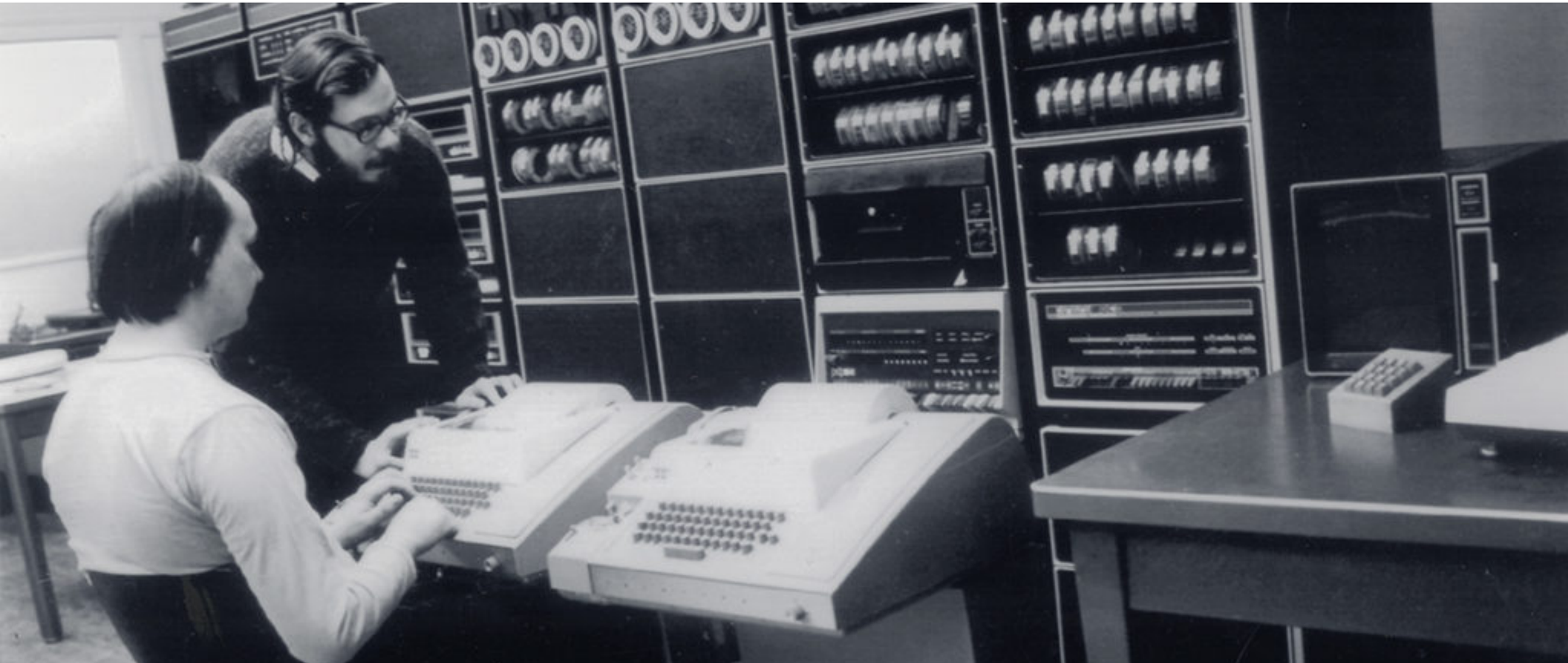
Modern C++

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?

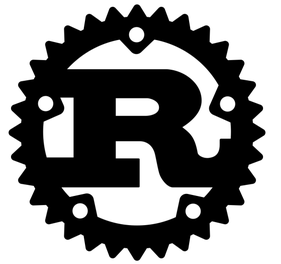
Photo by Computer History - DEC - PDP-11- Ken Thompson and Dennis Ritchie









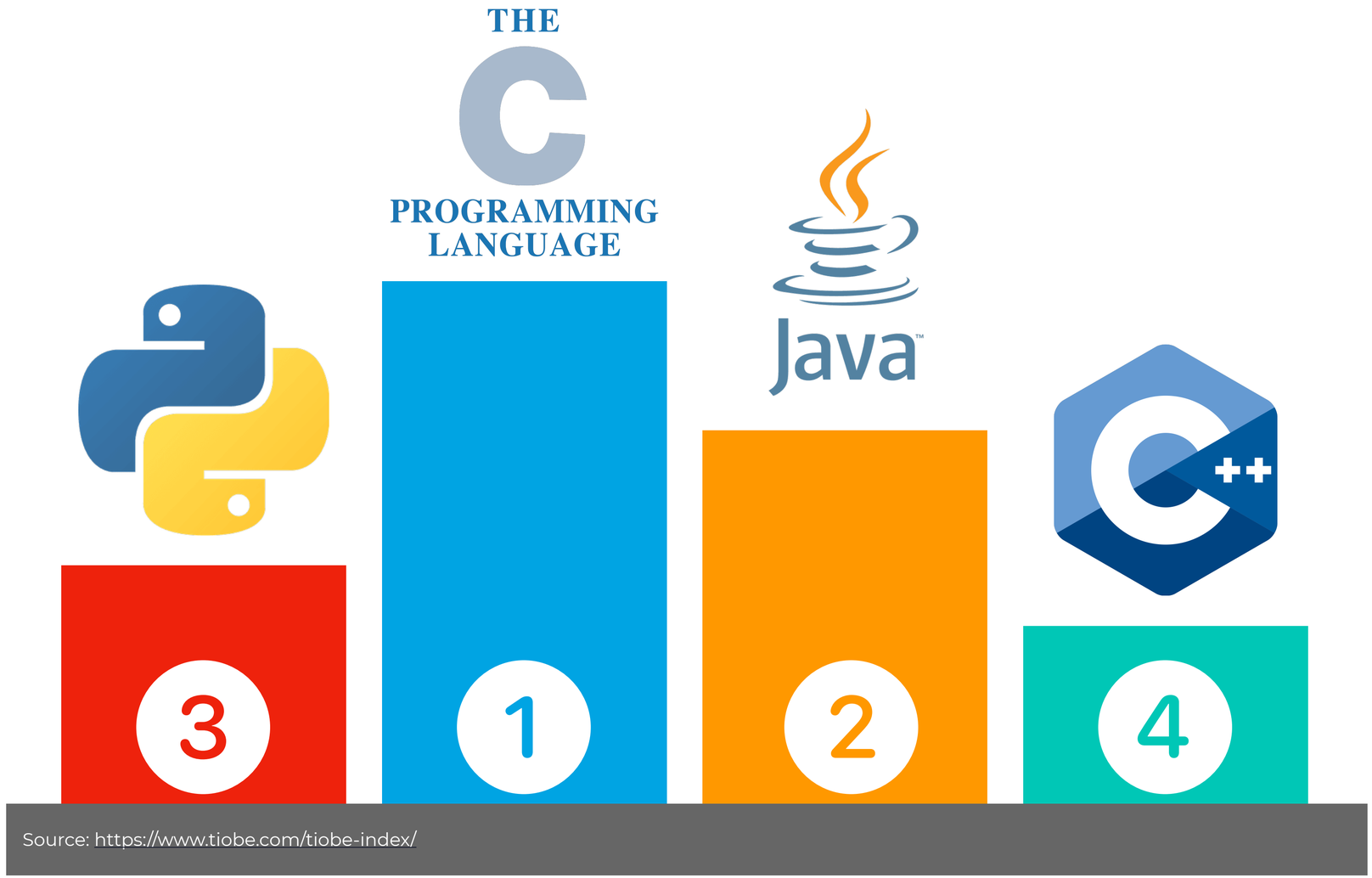



Welcome to the jungle!

About 700 notable languages (source wikipedia)

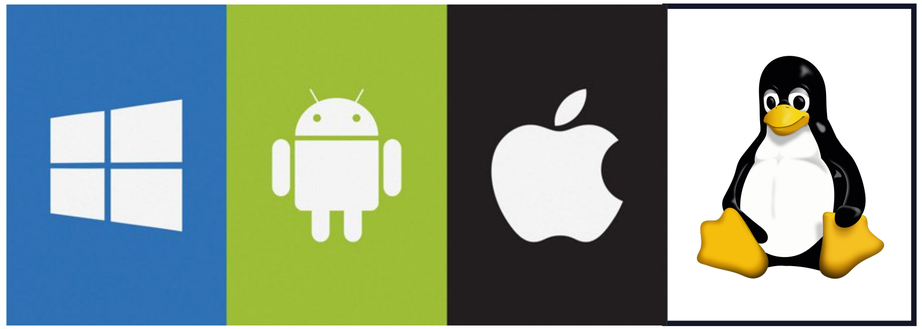






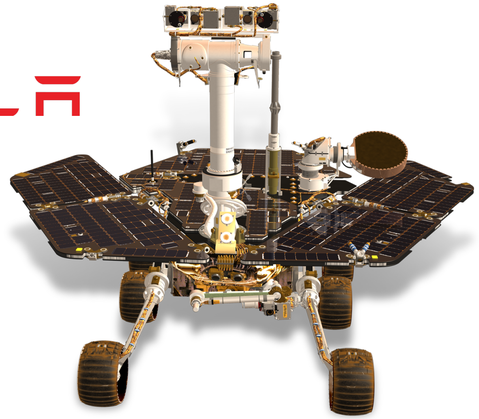
C/C++ are still among the most popular languages!

Used by famous companies...



… to make great apps with ❤️.

TESLA



UNREAL ENGINE

ES
EMBEDDED SYSTEMS

YES

Learning C++ is relevant.

Questions

AGENDA

01 – The ITC313 program

02 – Why C++?

03 – The art of programming

04 – hello, world

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.

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

Does the program behave correctly with no error ?”

01

Design

A more qualitative measure related to efficiency.

“How efficient is your program?

Does your code just do what it should while ensuring low resource consumption and fast execution time?”

02

03

Style

A subjective measure related to the aesthetics.

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

Make it work, make it right, make it fast

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

Clean your code.

Clean Code is Code that can be read and directly understood.

Avoid duplication in the code - “Don’t repeat yourself!”

Test every piece of code. Once it works, consider code refactoring to get cleaner code.

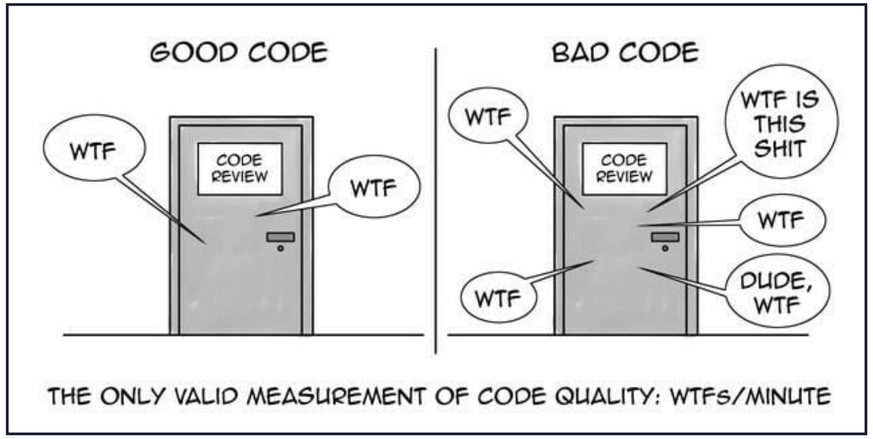


Photo by Elyess Eleuch on dev.to

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.

Questions

AGENDA

01 – The ITC313 program

02 – Why C++?

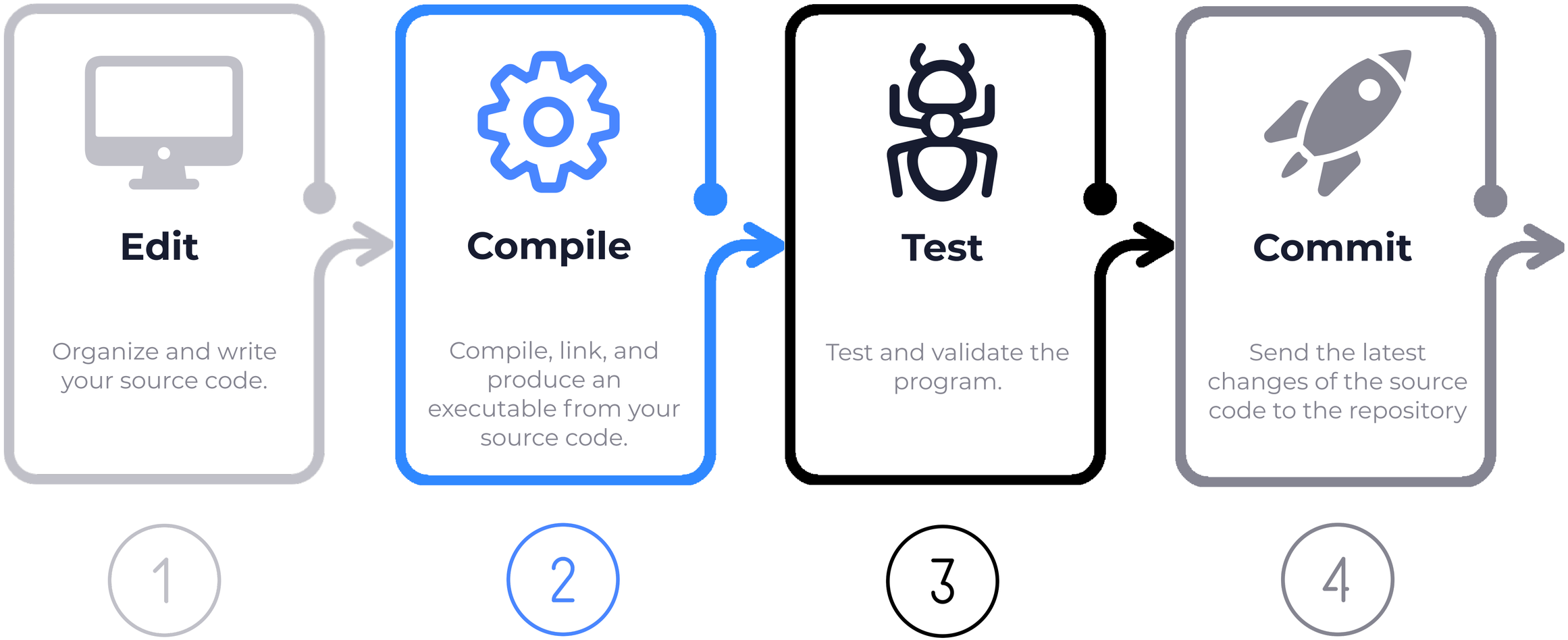
03 – The art of programming

04 – hello, world

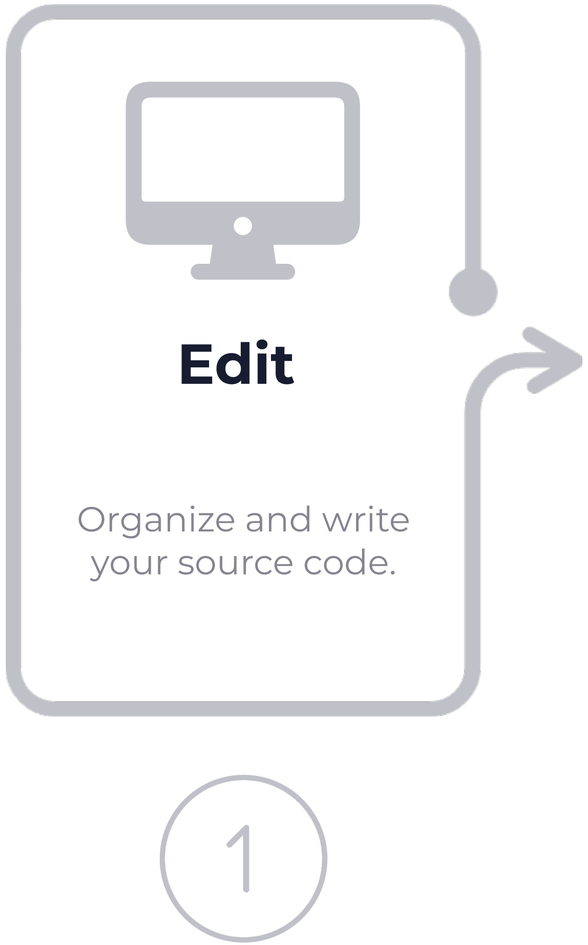
hello, world

needs more than just code.

Tools to write your first “hello, world” program



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



Edit

Organize and write your source code.

1

https://www.sublimetext.com

https://atom.io

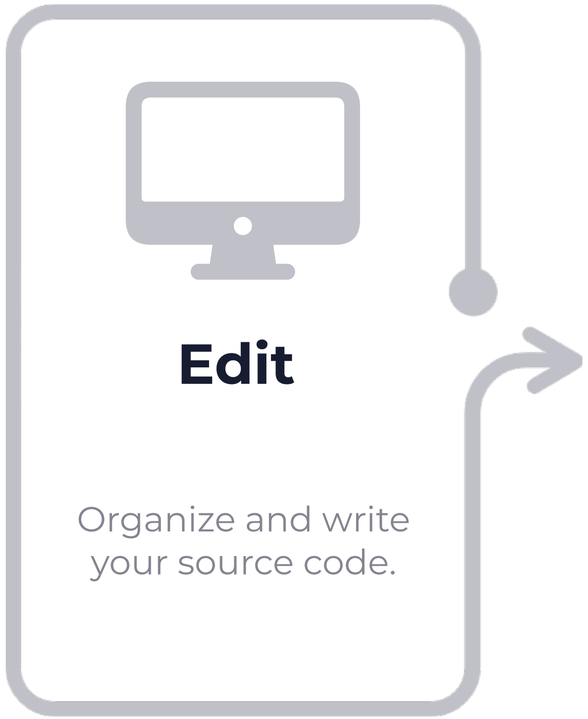
https://www.codeblocks.org

https://code.visualstudio.com

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

https://replit.com

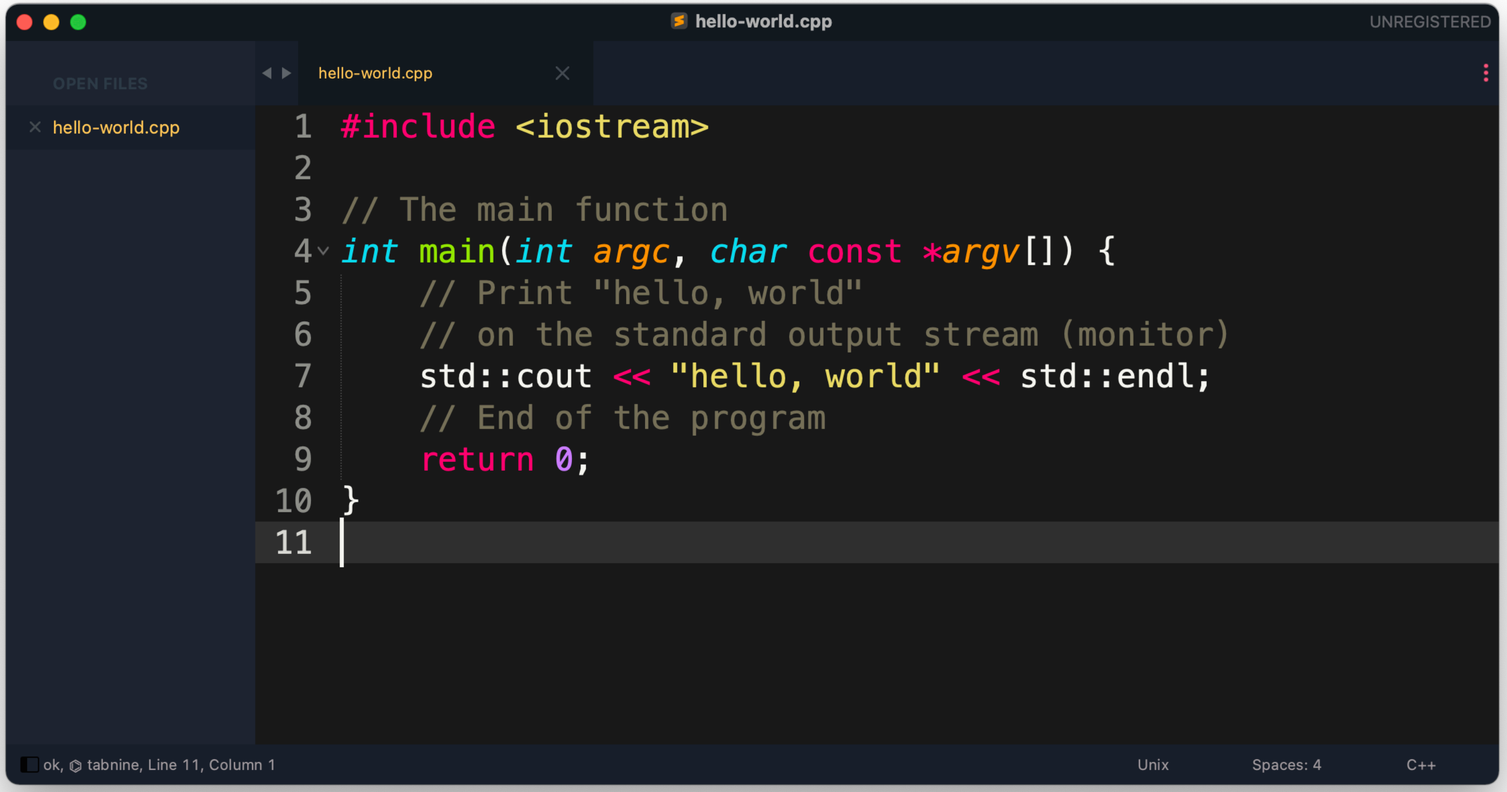
Automate your workflow to write your first “hello, world” program



Edit

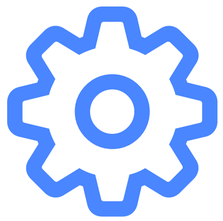
Organize and write your source code.

1



Written with

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



Compile

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

2

GNU Compiler Collection

https://gcc.gnu.org

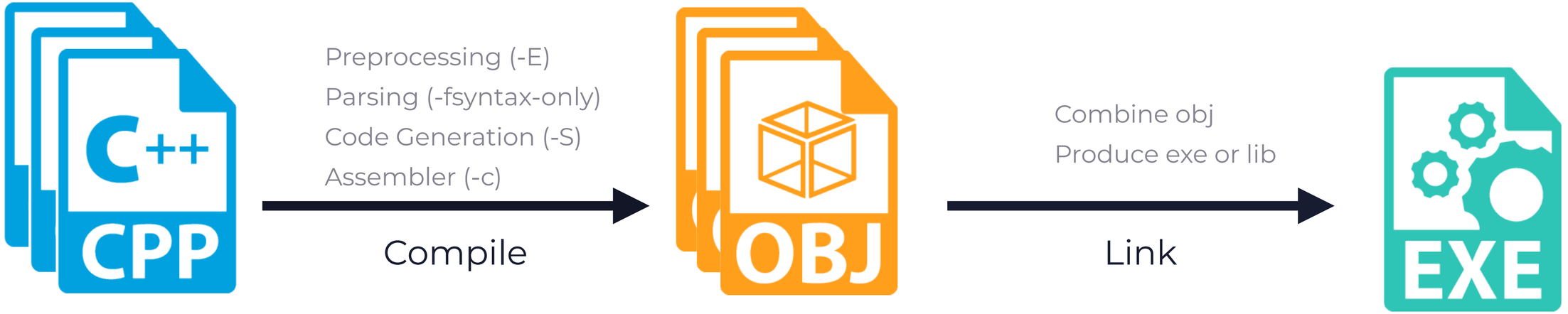
Minimalist GNU for Windows

http://www.mingw.org

LLVM Compiler Infrastructure

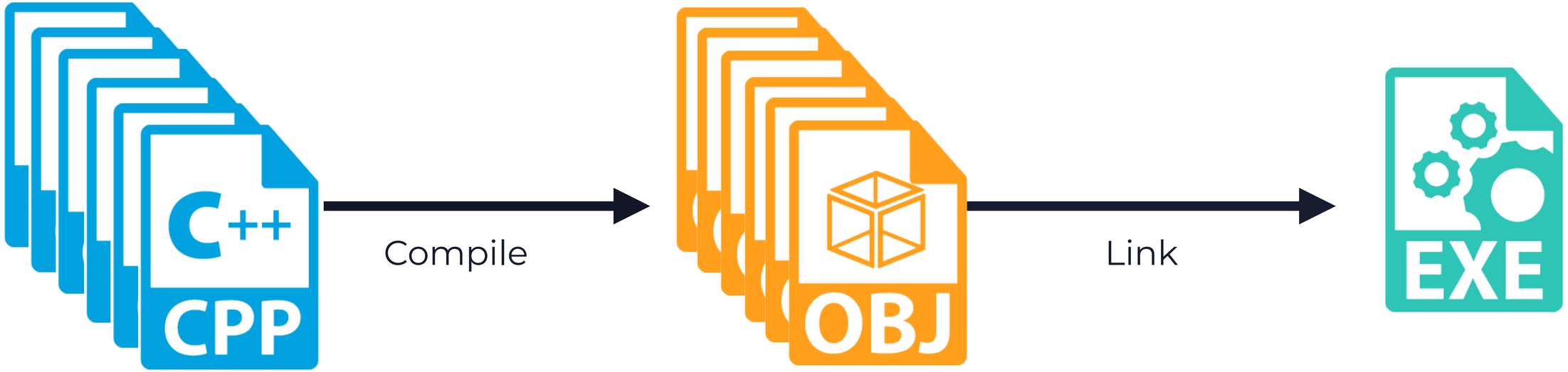
https://clang.llvm.org

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




```
-zsh                                                    ⌥⌘1
➜  helloworld ls
hello-world.cpp
➜  helloworld clang++ -std=c++17 -c hello-world.cpp
➜  helloworld ls
hello-world.cpp hello-world.o
➜  helloworld clang++ -std=c++17 hello-world.o -o hello-world
➜  helloworld ls
hello-world     hello-world.cpp hello-world.o
➜  helloworld ./hello-world
hello, world
➜  helloworld
```

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



AUTOMATE

THE COMPILATION with MAKEFILES

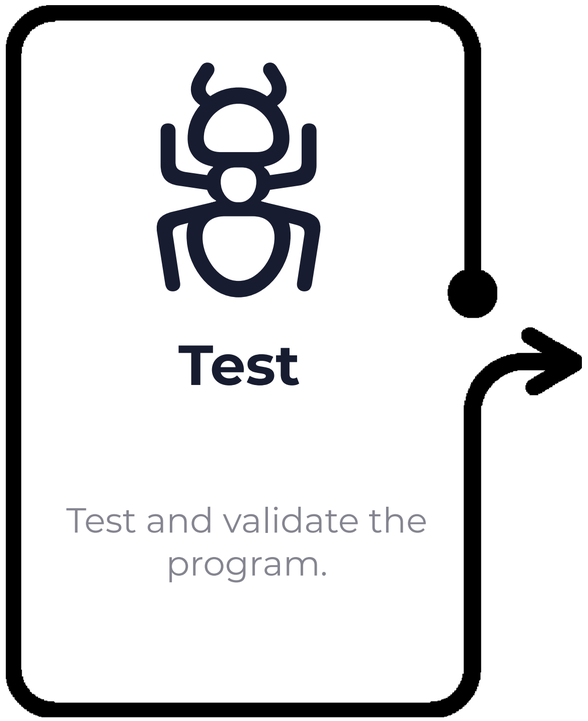
Make is a build automation tool that automatically builds executable programs and libraries from source code by reading files called Makefiles.

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

Automate your workflow to write your first “hello, world” 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 (aka the unit) in an isolated manner. It helps finding problems early in the development cycle and makes the final debug step easier.

Unit tests can be written after code (standard approach) or before code (Test Driven Development approach).

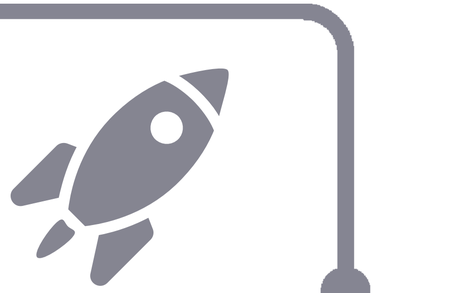
Unit testing allows you to refactor code at any time and make sure the code still works correctly.

Key concepts are assertions (boolean expressions).

The most used C++ testing frameworks are Google Test, Boost, Catch2.

Upcoming demo

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



Commit

Send the latest changes of the source code to the repository

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.

Git also makes collaboration easier, allowing changes by multiple people to all be merged into one source.

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.

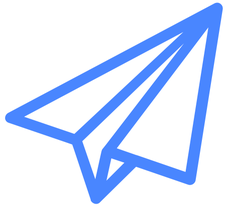
Git helps you to maintain the backup of your source code.

Git helps you to collaboratively work with other developers.

Upcoming demo

#1

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

Photo by Alexander Andrews on Unsplash

Contacts

Pr. Dominique Ginhac

dginhac@u-bourgogne.fr

Come visit us at

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