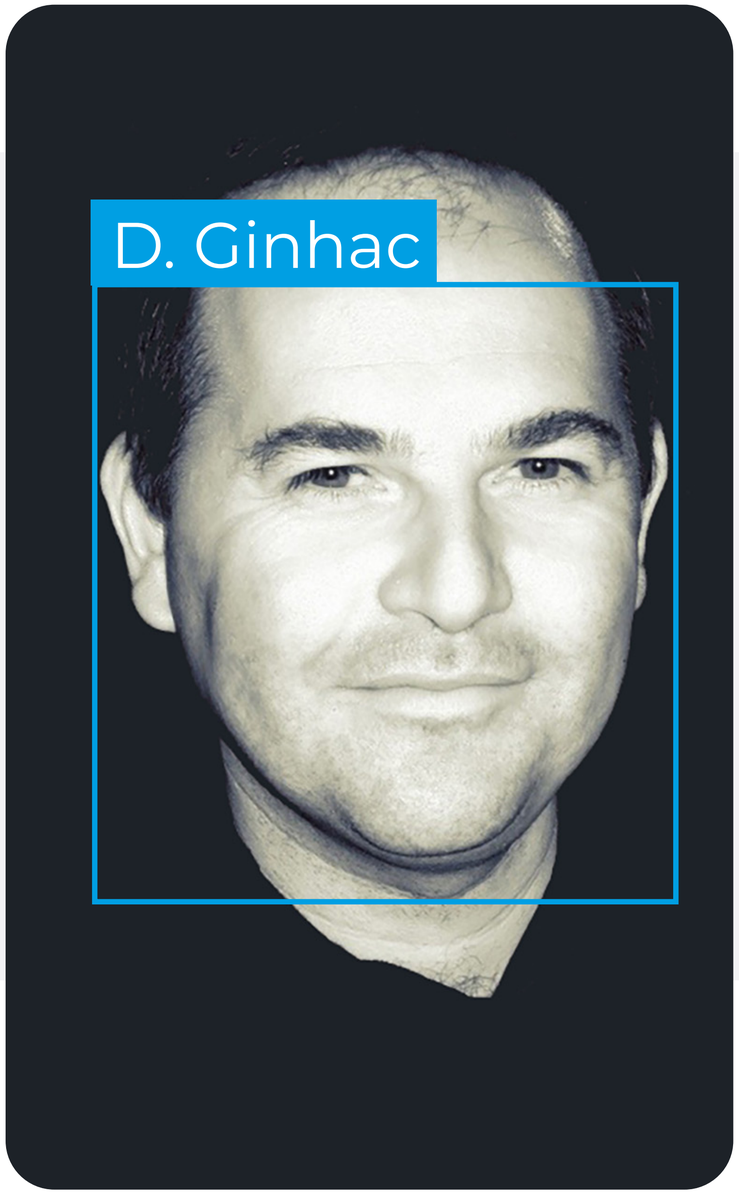
ESIREM
ÉCOLE SUPÉRIEURE D'INGÉNIEURS
NUMÉRIQUE ET MATÉRIAUX

École associée Polytech

std::cout << "Welcome!";

Last updated on September, 25, 2023 by dG



$ ~ whoami

HI, I'M D. GINHAC

I will be your instructor for some Computer Programming lectures this year and next year.

I'm conducting research in Computer Vision, and more specifically on how to embed artificial intelligence in real-time systems.

dginhac@u-bourgogne.fr dginhac @dginhac

Slides are available on

https://ginhac.com/teaching/ITC313/latest/00-introduction.pdf

ESIREM
ÉCOLE SUPÉRIEURE D'INGÉNIEURS
NUMÉRIQUE ET MATÉRIAUX

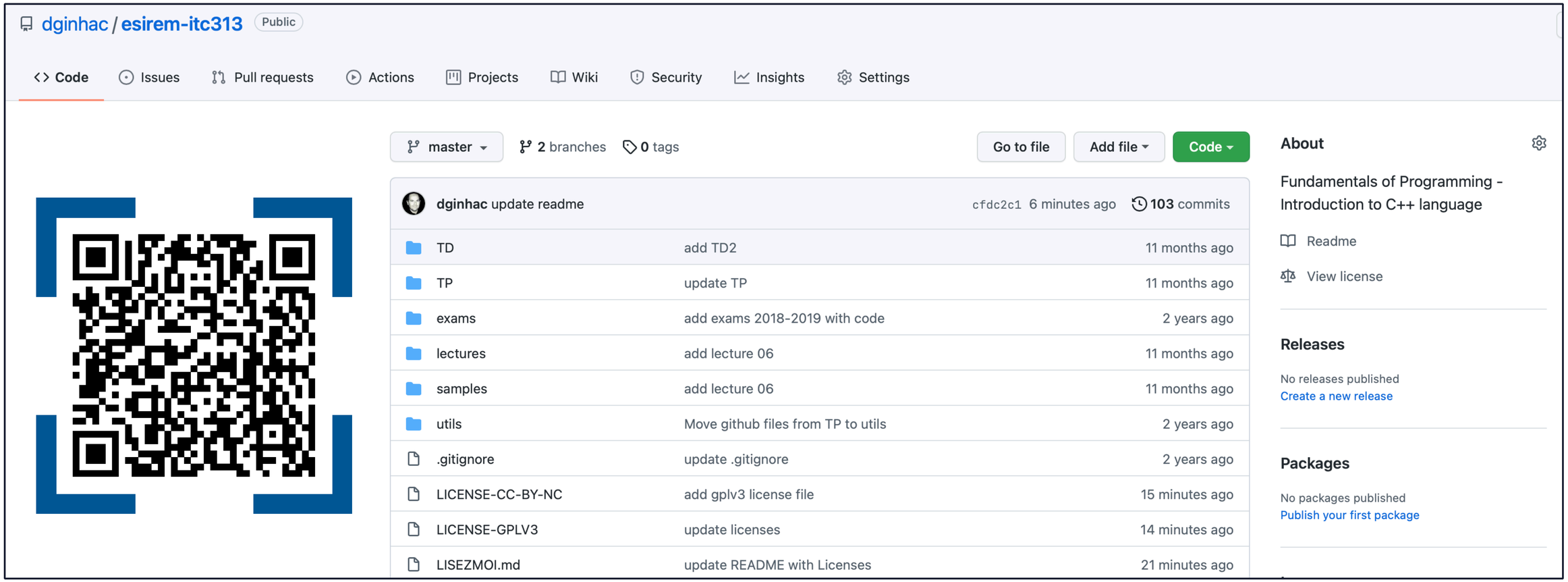
École associée Polytech

Fundamentals of C++

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

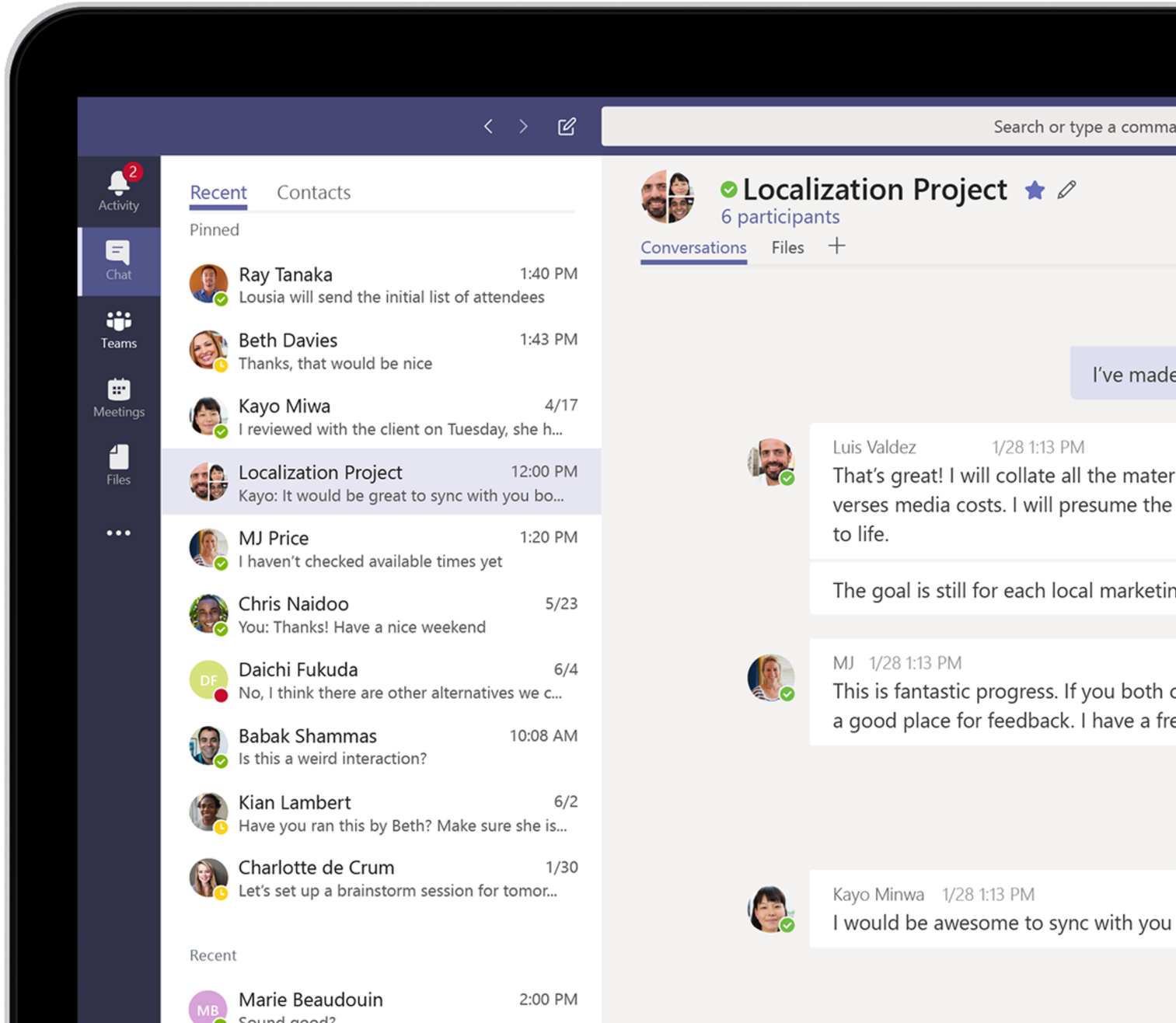
Visit https://github.com/dginhac/esirem-itc313

Everything else available on GitHub.



Microsoft TEAMS

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



Some quick surveys to get to know you!

Connect to

https://app.wooclap.com/ITC313


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

Today

Lecture #00
Course Introduction

Lecture #01
User-defined Data Types

Lecture #02
Inheritance

Lecture #03
Polymorphism

Lecture #04
STL Containers

Lecture #05
Templates

Lecture #06
Exceptions

Lecture #00

Course introduction

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

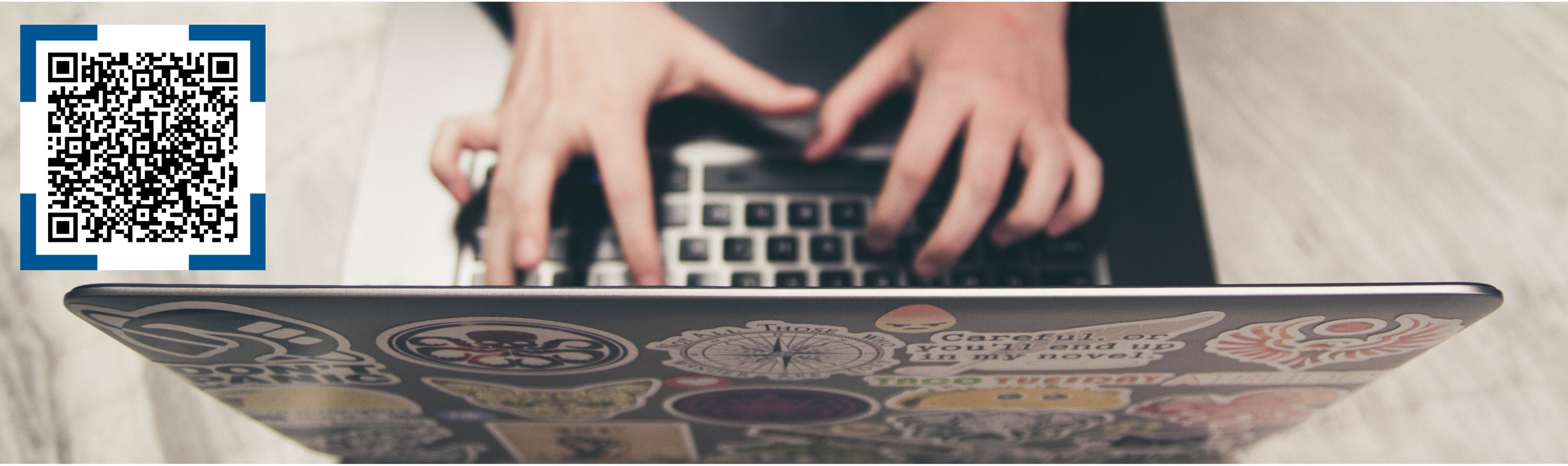


Photo by NeONBRAND on Unsplash

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

The ITC313 course scheduling

Photo by Xin Wang on Unsplash

Lectures (CM)

Starting Week 39

13 x 1.75 h
D. Ginhac

Tutorials (TD)

Starting Week 41

6 x 1.75 h
D. Ginhac

Labs (TP)

Starting Week 48

12 x 2 h
D. Ginhac
Other colleagues

Exams

11/29 & 01/23

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?

MORE
AUTONOMY

GAINED
SKILLS

CONSTANT
MOTIVATION

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.

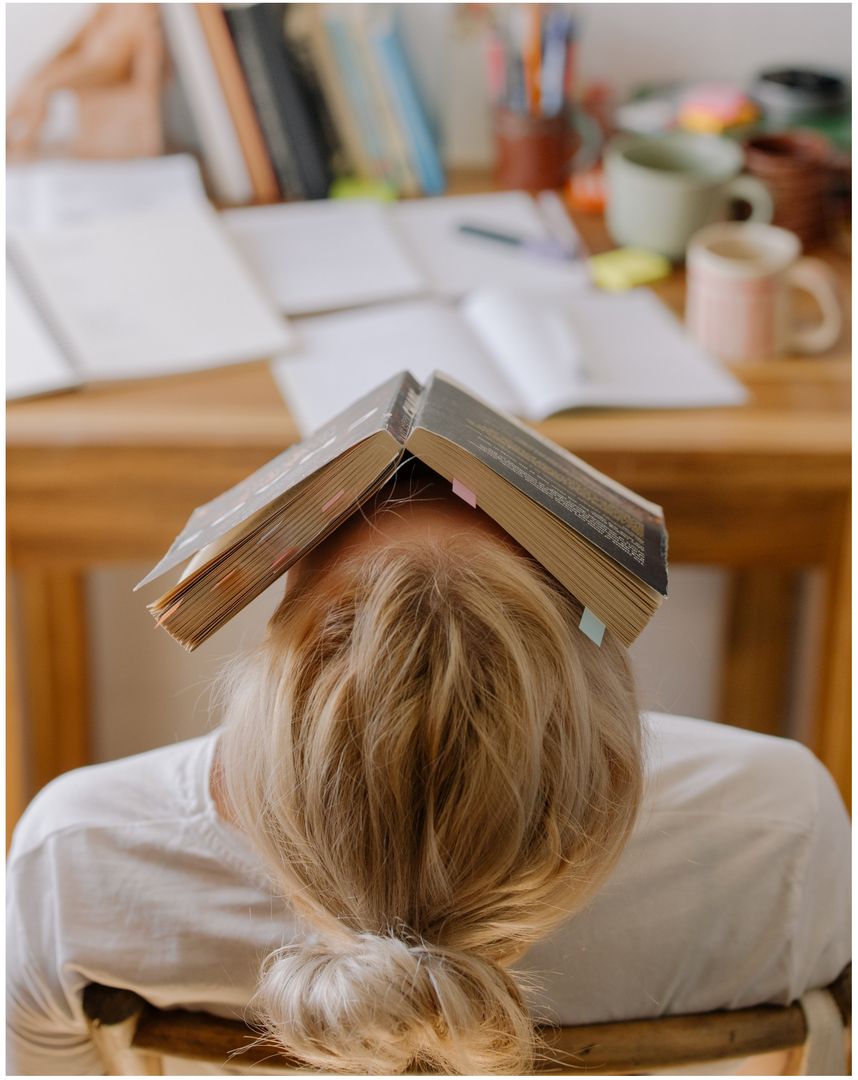
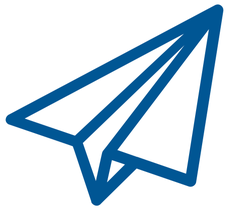


Photo by cottonbro from Pexels

#0

AN INITIAL TAKE HOME MESSAGE



Learning Computer Science has never been so accessible thanks to the broad range of resources available!

But, nobody became a software developer by following a MOOC or watching tutorials on YouTube.

“Programming is a skill best acquired by practice and example rather than from books.” – Alan Turing

Questions

AGENDA

01 – The ITC313 program

02 – Why C++?

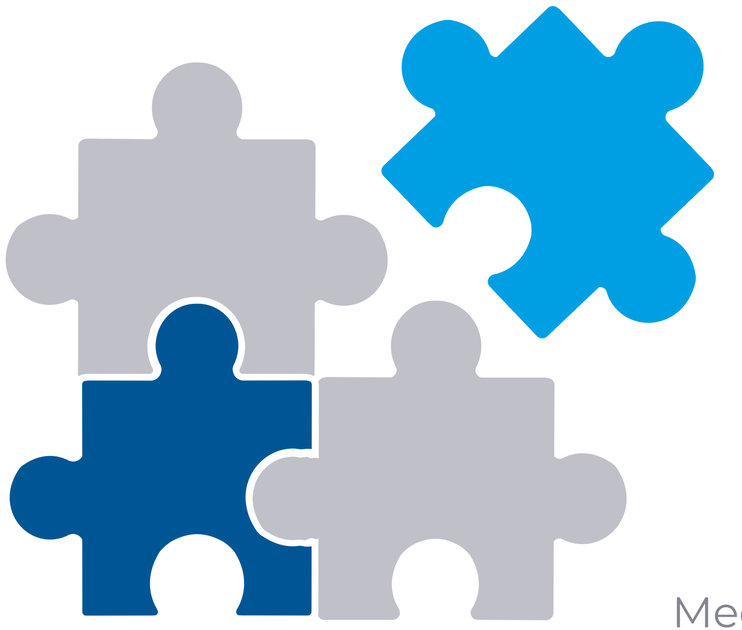
03 – The art of programming

04 – hello, world

C++ = one 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.

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

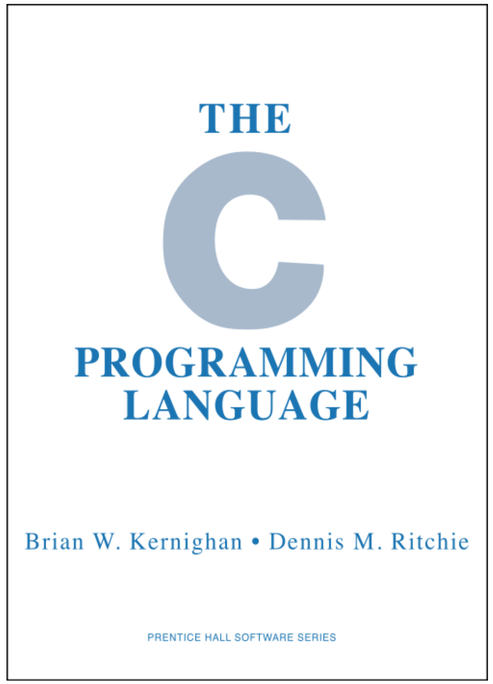


Photo by National Inventors Hall of Fame

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++23 is in dev.

C++11 was a revolution

C++ before 2011 was C with classes.
C++ since 2011 is a new OOP language.

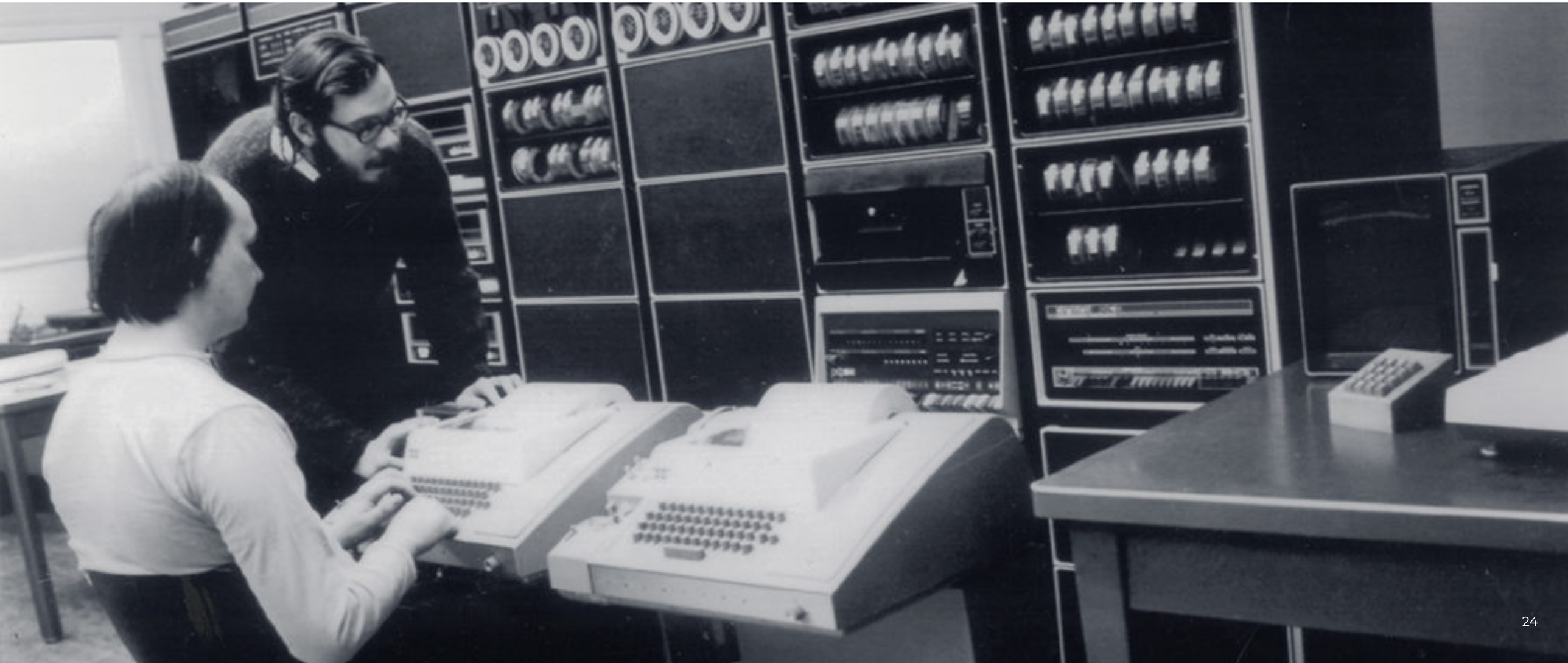
Modern C++

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?

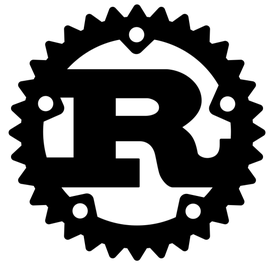
Photo by Computer History - DEC - PDP-11- Ken Thompson and Dennis Ritchie









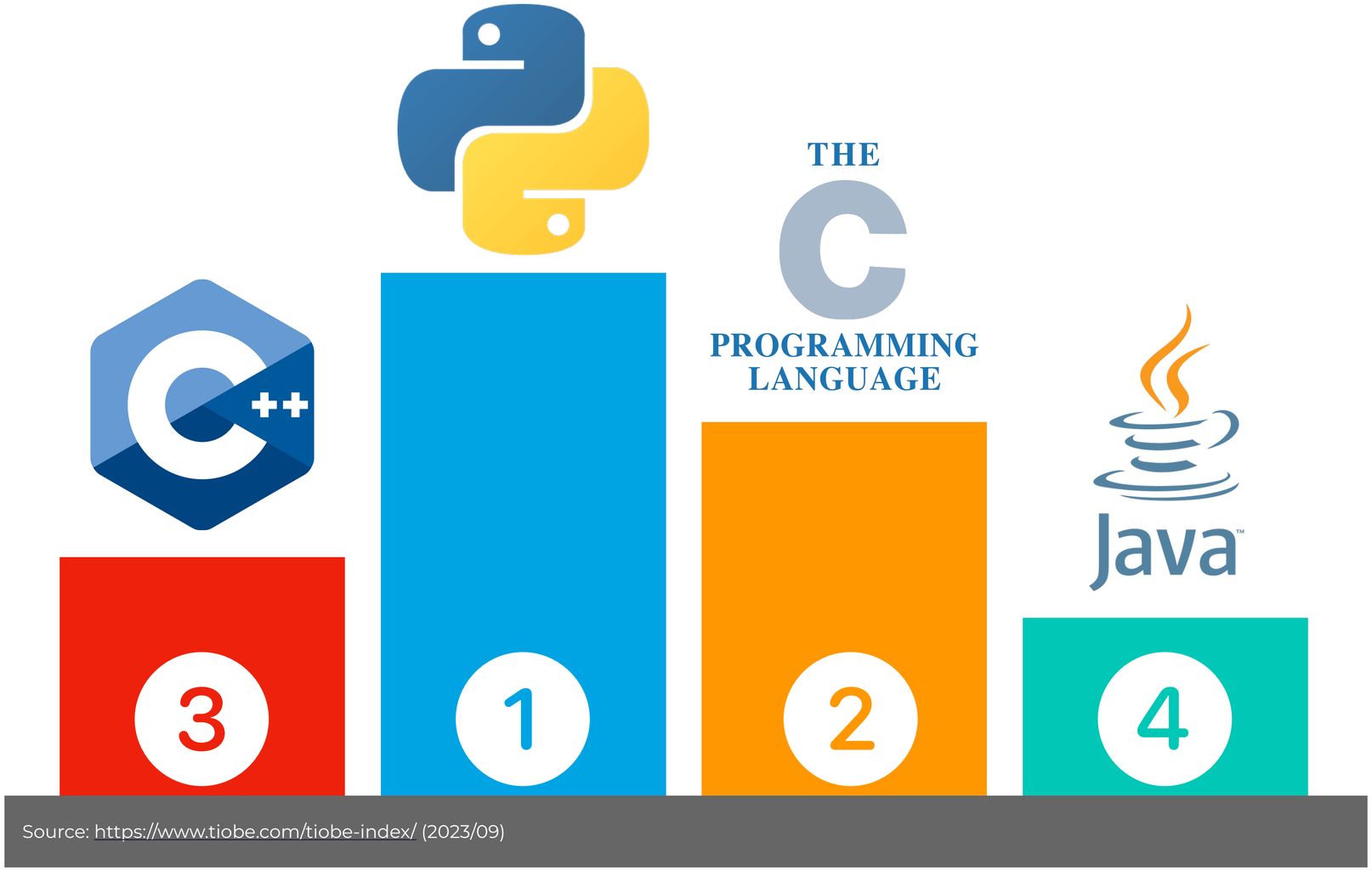



Welcome to the jungle!

About 700 notable languages (source wikipedia)



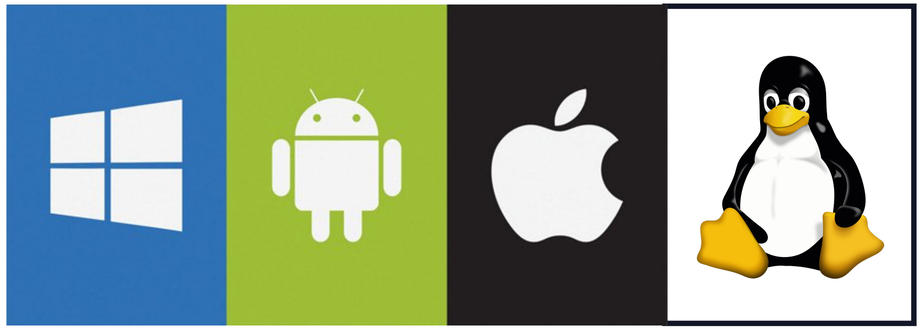




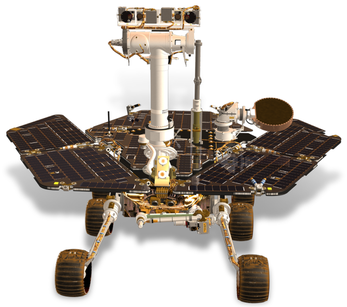
Source: https://www.tiobe.com/tiobe-index/ (2023/09)

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

Used by famous companies...



… to make great apps with ❤️



UNREAL ENGINE

ES EMBEDDED SYSTEMS

IoT

YES

Learning C++ is still **relevant** in 2023.

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

Art of 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?

Does the program behave correctly?”

01

Art of Design

Even bad code can work!

“How efficient is your program?”

02

03

Art of 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.

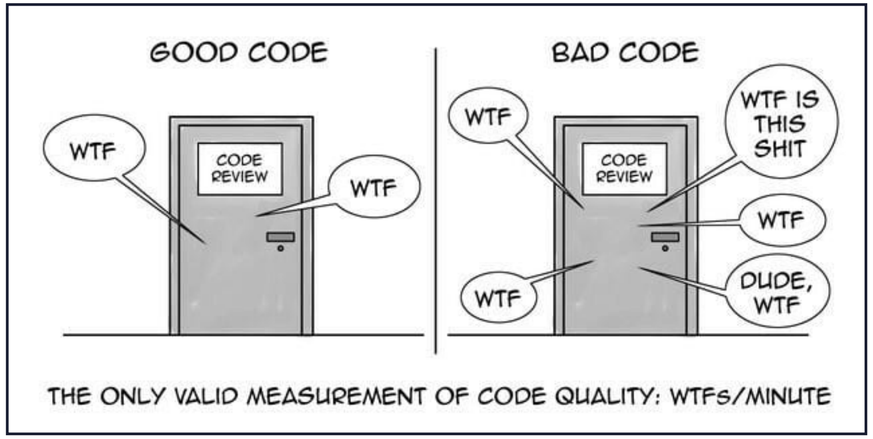


Photo by Elyess Eleuch on dev.to

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++?

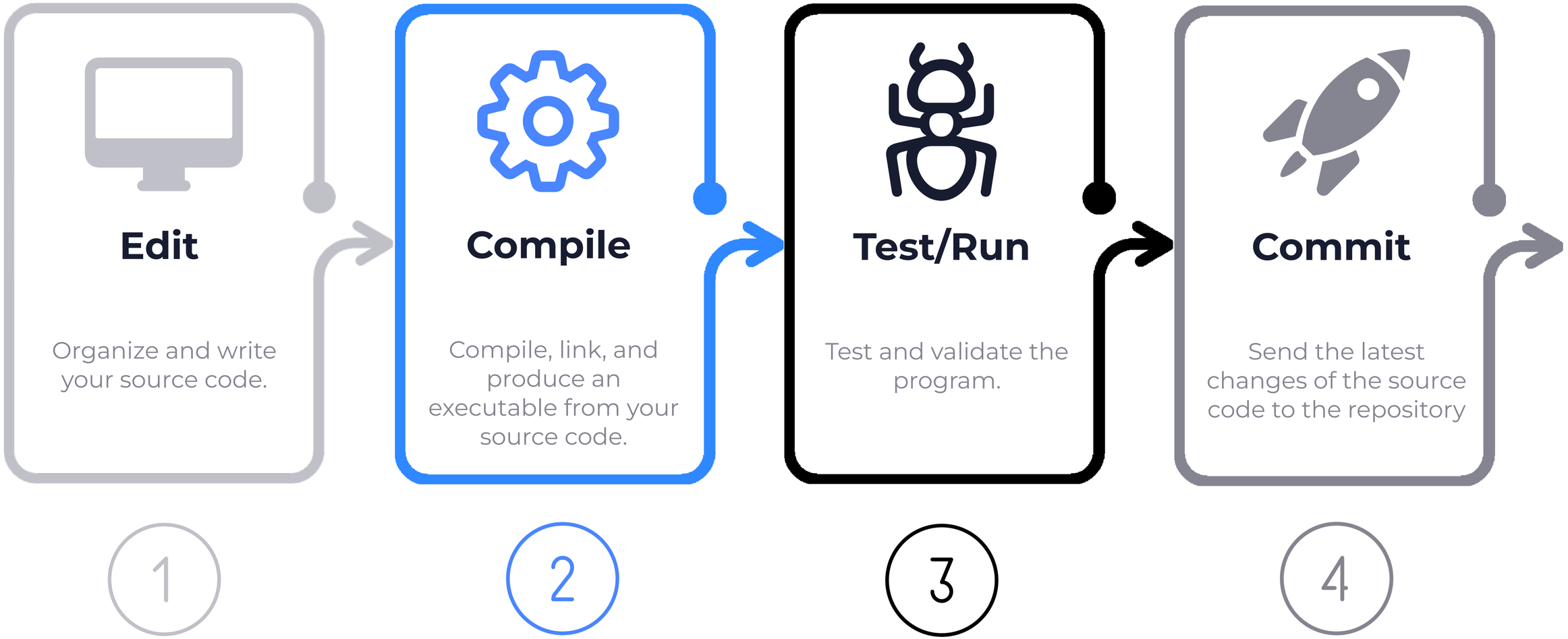
03 – The art of programming

04 – hello, world

hello, world

needs more than just code.

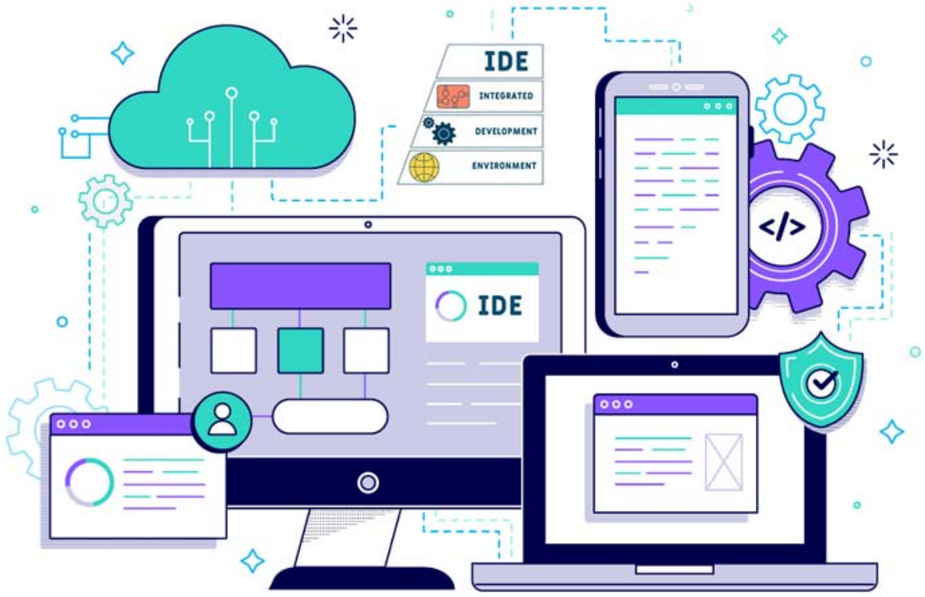
Automate your workflow to write your first “hello, world” program

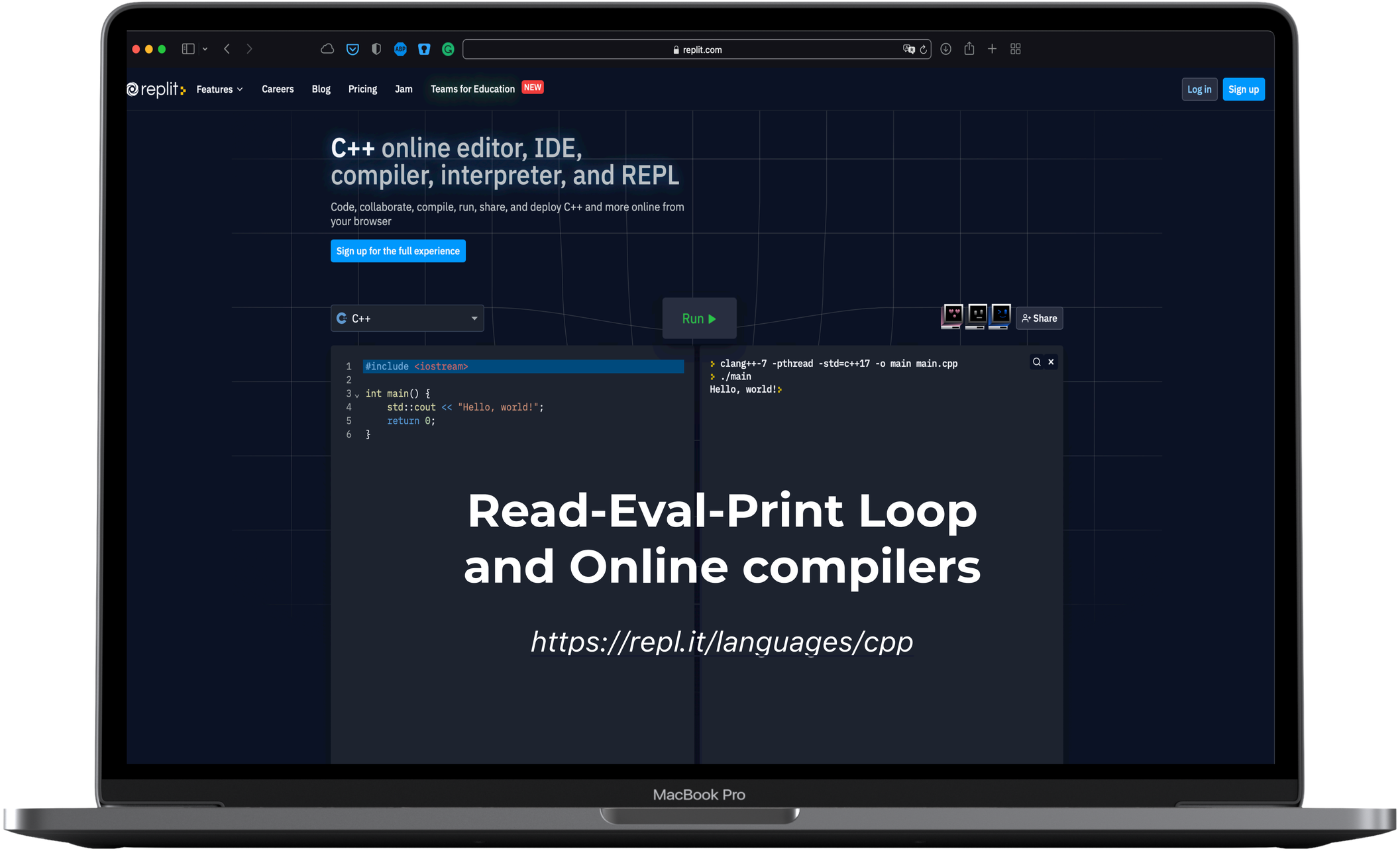


Use IDE to “rule them all”

An integrated development environment (IDE) is a software application that helps programmers write code efficiently.

It increases developer productivity by combining capabilities such as software editing, building, testing, and packaging in an easy-to-use application.




replit.com
replit
Features
Careers
Blog
Pricing
Jam
Teams for Education
NEW
Log in
Sign up
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
#include <iostream>
int main() {
std::cout << "Hello, world!";
return 0;
}
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
MacBook Pro

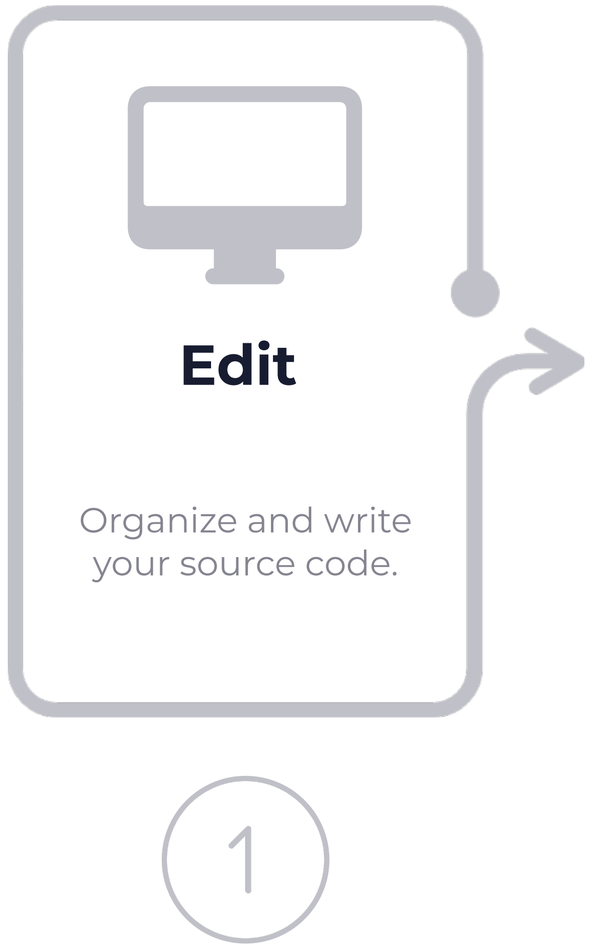
IDE on your own computer

https://www.sublimetext.com

https://code.visualstudio.com

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

Write source files



https://www.sublimetext.com

https://atom.io

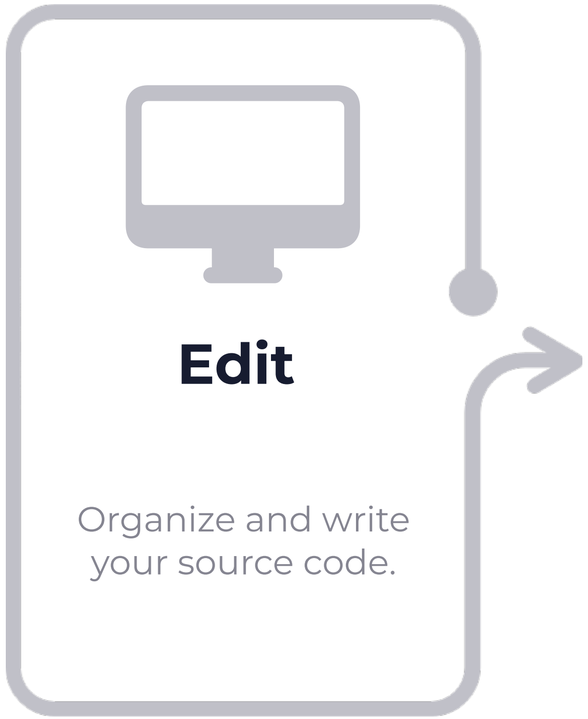
https://www.codeblocks.org

https://code.visualstudio.com

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

https://replit.com

Your first “hello, world” program



Edit

Organize and write your source code.

1

hello-world.cpp — UNREGISTERED

OPEN FILES

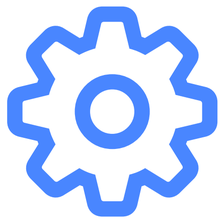
hello-world.cpp

```cpp
#include <iostream>

// The main function
int main(int argc, char const *argv[]) {
    // Print "hello, world"
    // on the standard output stream (monitor)
    std::cout << "hello, world" << std::endl;
    // End of the program
    return 0;
}

```

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

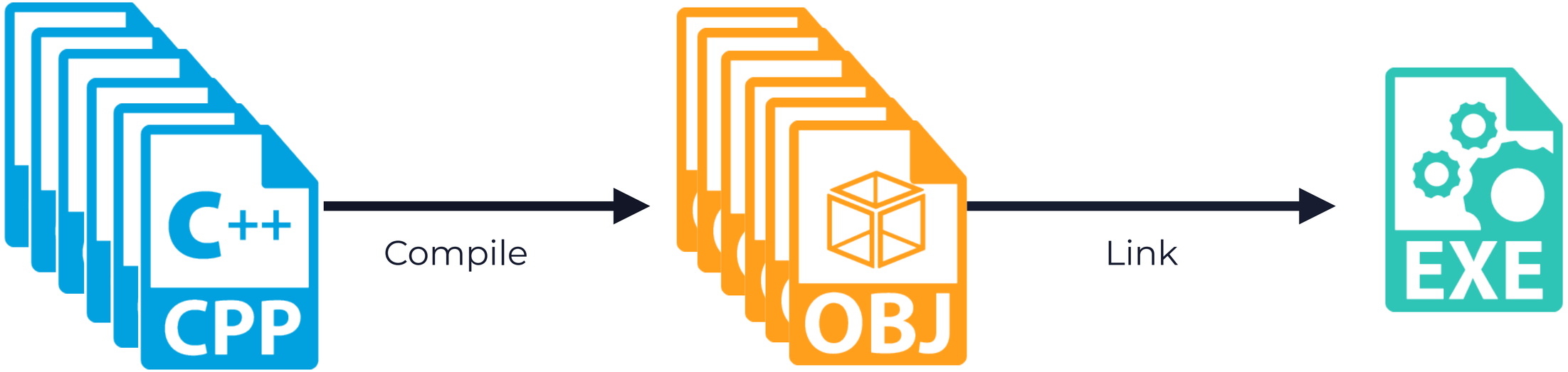

```
-zsh — d0m                                                    ⌥⌘1
➜  00-helloworld ls
hello-world.cpp
➜  00-helloworld clang++ hello-world.cpp
➜  00-helloworld ls
a.out           hello-world.cpp
➜  00-helloworld ./a.out
hello, world
➜  00-helloworld
```

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

3

The bug tracking process involves detecting bugs during testing, assessing their impact, determining their causes, and fixing them.

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**.

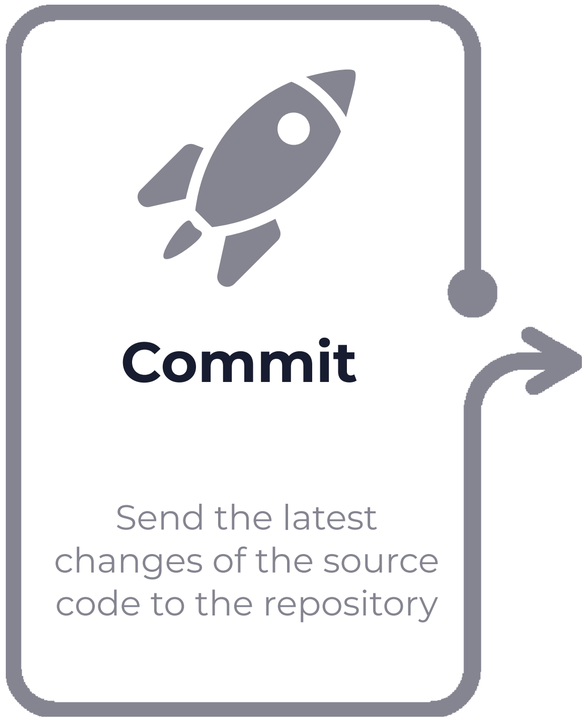
A unit test is an automated test that verifies a small piece of code in an isolated manner. It helps finding problems early in the development cycle and makes the final debug step easier.

Key concepts are assertions (boolean expressions).

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

Upcoming demo

Software versioning



Commit

Send the latest changes of the source code to the repository

4

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.

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

What about Artificial Intelligence for coding?

ChatGPT

MutableAI

Copilot

AskCodi

Codium

Codiga

Tabnine

CodeWhisperer

Will ChatGPT/Copilot replace developers?

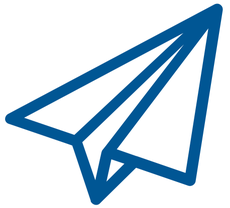
The short answer is NO

- However ChatGPT/Copilot has the potential to automate some aspects of programming, such as code generation, bug fixing, and documentation.
- ChatGPT/Copilot can learn from vast amounts of code and data, making it possible to generate new code similar to existing code.
- ChatGPT/Copilot cannot replace the human creativity and critical thinking required to design and develop complex software applications.

chatGPT/Copilot should become your perfect Pair Programming Partner.

#1

A FIRST TAKE HOME MESSAGE



Modern IDE and AI tools can help you write code quickly.

But high-quality C++ programming is more about a deep understanding of OOP principles 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