ESIREM
ÉCOLE SUPÉRIEURE D'INGÉNIEURS
NUMÉRIQUE ET MATÉRIAUX

Download slides on
https://github.com/dginhac/esirem-itc313

ITC313 - Lesson 01

Fundamentals of programming
Learning C++: from beginner to beyond

Dominique Ginhac

#ESIREM

Lesson 01

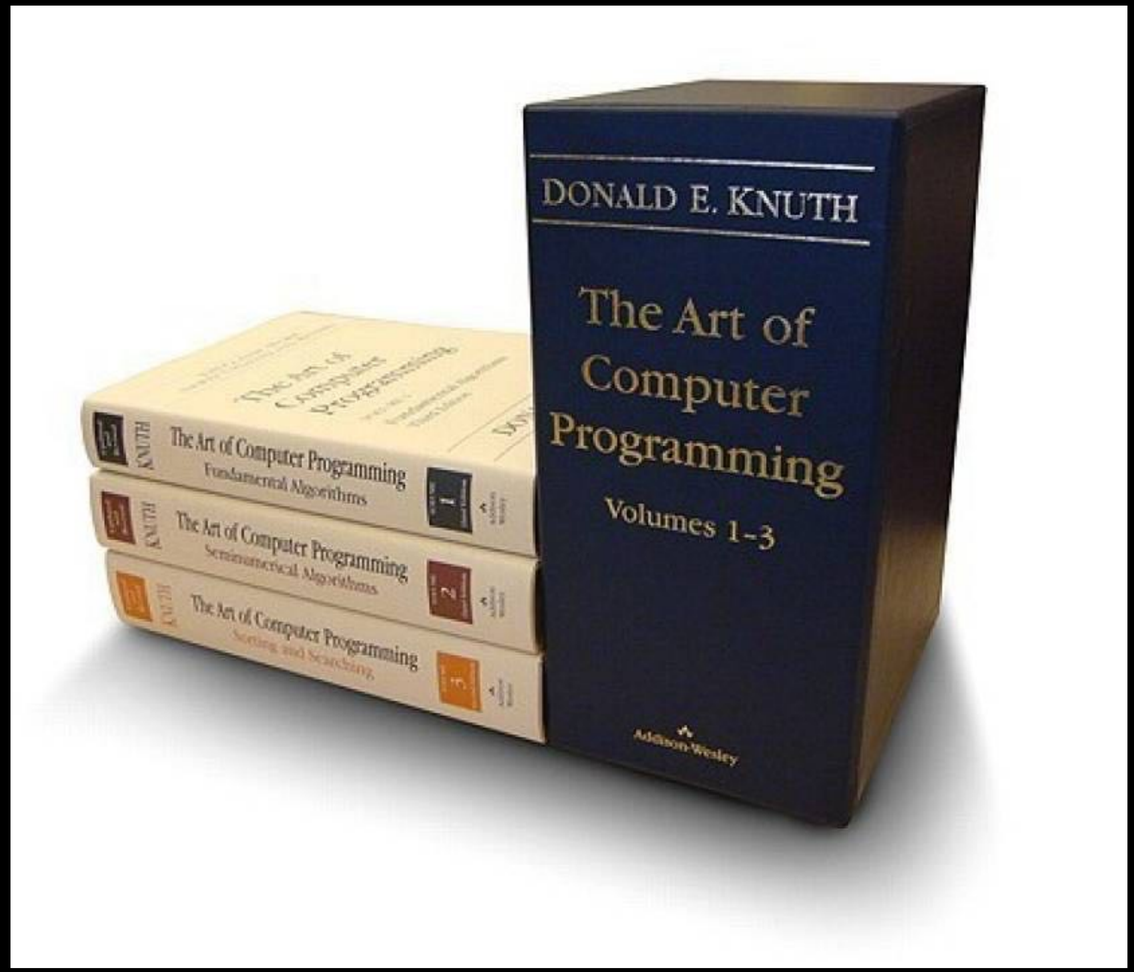
Hello, world!

Students are welcomed to the tradition of programming by editing, compiling and running the well-known “HELLO, WORLD!” code. They will also be introduced to the main concepts of WORKFLOW PROGRAMMING.

It’s not just about writing code

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

— D. Knuth



So, think about it and stop writing poor code!

5 rules for writing quality code

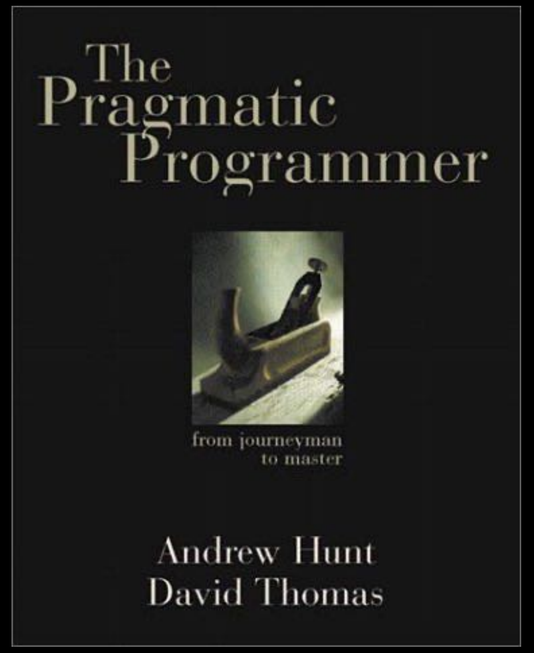
#1: Use descriptive names for variables, classes, functions, ...

#2: Follow naming conventions
See https://google.github.io/styleguide/cppguide.html for an example

#3: Split problem into short units and Focus on one thing at a time

#4: Provide comments that are critical for future understanding

#5: Include documentation on your app



See https://codeshare.co.uk/blog/10-golden-rules-for-becoming-a-better-programmer/

What do you need?

1

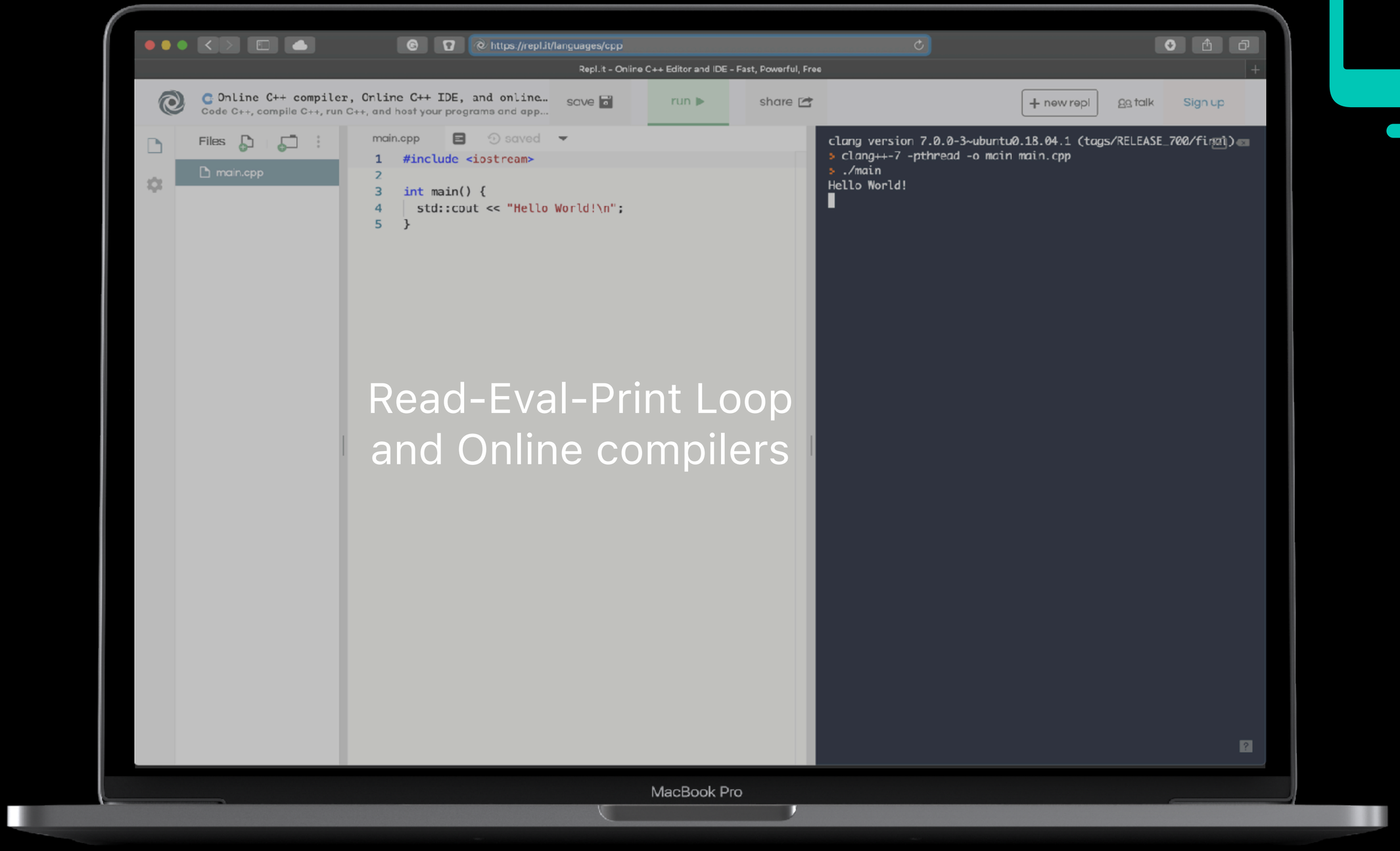
An editor to write code

2

A C++ compiler / linker

3

A debugger



Read-Eval-Print Loop
and Online compilers

See https://repl.it/languages/cpp

Get an editor

https://www.sublimetext.com

https://atom.io

https://code.visualstudio.com

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

hello-world.cpp UNREGISTERED

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;
}

```

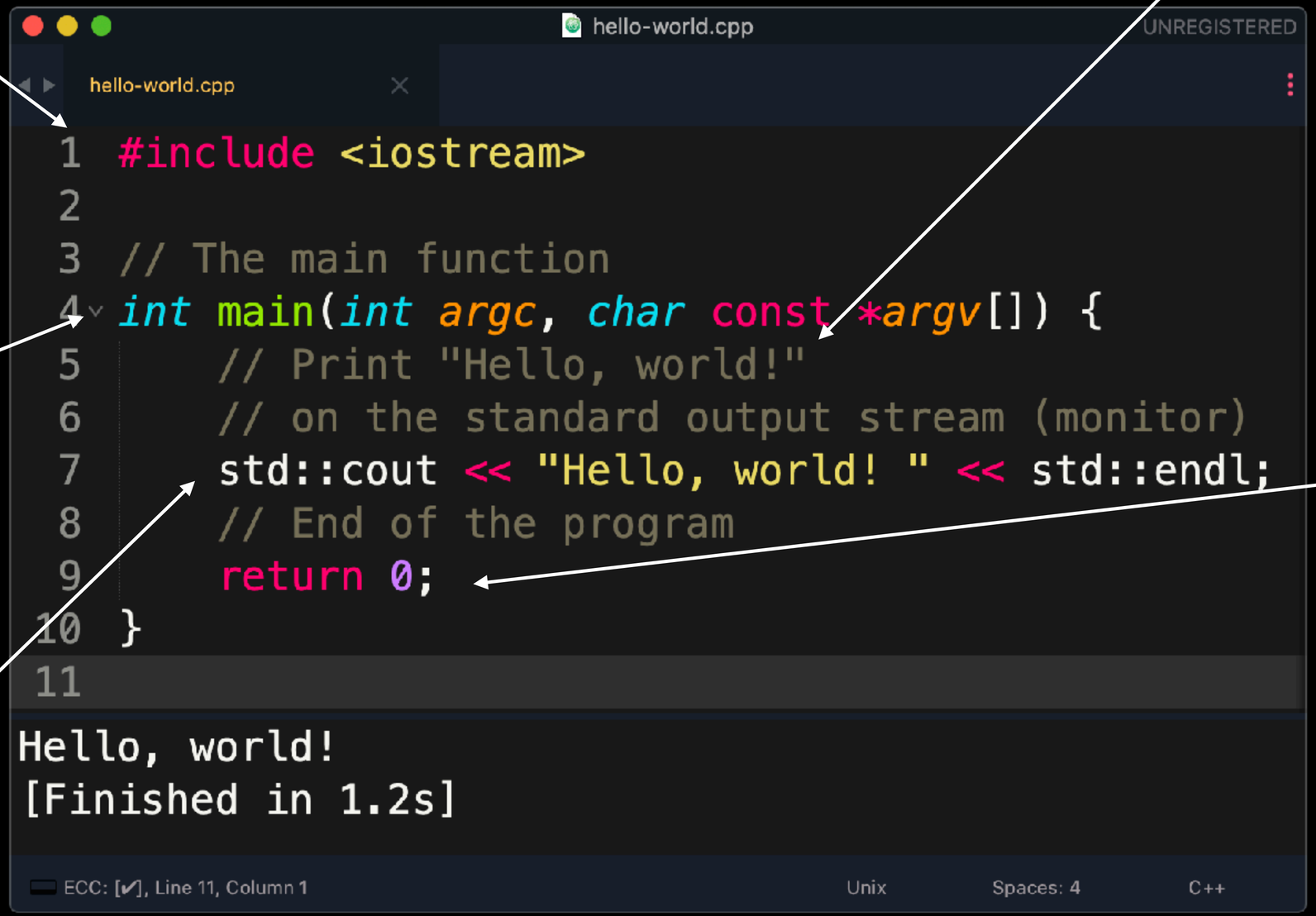
```
Hello, world!
[Finished in 1.2s]
```

ECC: [✓], Line 11, Column 1 Unix Spaces: 4 C++

Written with
Sublime Text

#include <iostream>
Input Output Stream library

// Print “Hello, world!”
A comment



int main()
Main function

return 0;
End of main() function

Returns status 0 if successful, 1 or higher for errors

std::cout (character output) is an object of standard output stream.

Using namespace std;
Explicit call of namespace std

hello-world-with-cin.cpp — ~/Documents/learning/teaching/esirem/3A/ITC313-Prog/2020-2021/code/lesson01

hello-world-with-cin.cpp

```cpp
#include <iostream>
using namespace std;

// The main function
int main(int argc, char const *argv[]) {
    // Creates a string variable
    string name;
    cout << "Enter your name: ";
    cin >> name;
    // Prints "Hello, !" + name
    // on the standard output stream (monitor)
    cout << "Hello, " << name << "!" << endl;
    // End of the program
    return 0;
}
```

/Documents/learning/teaching/esirem/3A/ITC313-Prog/2020-2021/code/lesson01/hello-world-v LF UTF-8 C

written with Atom

Call of cin, cout and endl without std::

d0m — hello-world-with-cin — 69×9

```
Last login: Mon Sep  7 09:13:48 on ttys005
/var/folders/kd/w8vdlm6j4xj4k1f7p26j13nm0000gn/T/hello-world-with-cin
 ; exit;
➜  ~ /var/folders/kd/w8vdlm6j4xj4k1f7p26j13nm0000gn/T/hello-world-wit
h-cin ; exit;
Enter your name: d0m
Hello, d0m!

[Process completed]
```

Get a compiler

GNU Compiler Collection

https://gcc.gnu.org

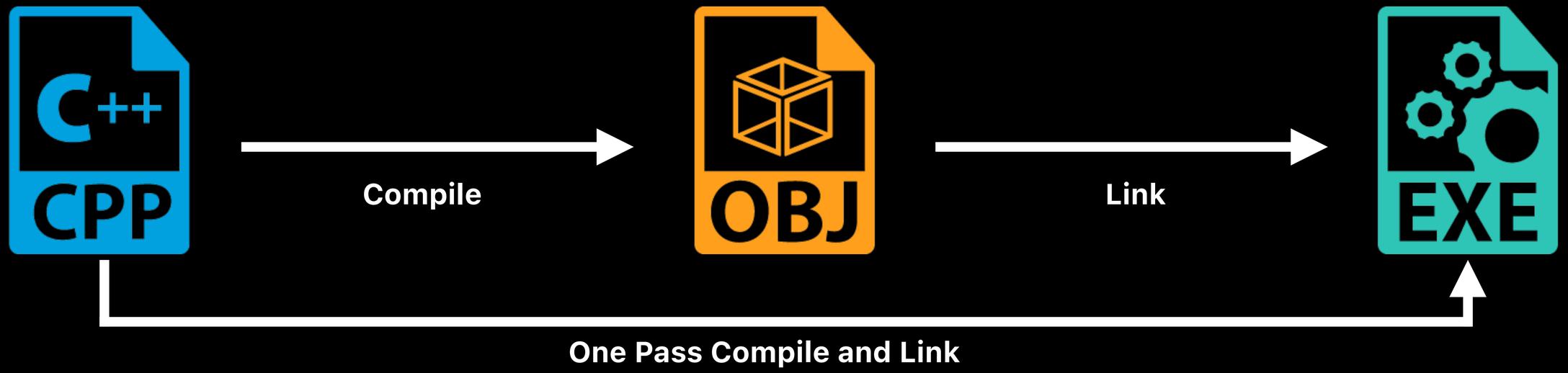
LLVM Compiler Infrastructure

https://clang.llvm.org

Minimalist GNU for Windows

http://www.mingw.org

Building and Running




```
lesson01 — -zsh — 80×8
→  lesson01 clang++ -c hello-world.cpp
→  lesson01 clang++ -o hello-world hello-world.o
→  lesson01 ./hello-world
Hello, world!
→  lesson01 clang++ -o hello-world hello-world.cpp
→  lesson01 ./hello-world
Hello, world!
→  lesson01
```

Building and Debugging

```
lesson01 — -zsh — 140×43
➜  lesson01 clang++ -g hello-world-with-cin.cpp -o hello-world-with-cin
➜  lesson01 lldb hello-world-with-cin
(lldb) target create "hello-world-with-cin"
Current executable set to 'hello-world-with-cin' (x86_64).
(lldb) list
   5    int main(int argc, char const *argv[]) {
   6       // Creates a string variable
   7       string name;
   8       cout << "Enter your name: ";
   9       cin >> name;
   10      // Prints "Hello, !" + name
   11      // on the standard output stream (monitor)
   12      cout << "Hello, " << name << "!" << endl;
   13      // End of the program
   14      return 0;
(lldb) b 10
Breakpoint 1: where = hello-world-with-cin`main + 84 at hello-world-with-cin.cpp:12:9, address = 0x00000001000014c4
(lldb) run
Process 2756 launched: '/Users/d0m/Documents/learning/teaching/esirem/3A/ITC313-Prog/2020-2021/code/lesson01/hello-world-with-cin' (x86_64)
Enter your name: d0m
Process 2756 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 1.1
    frame #0: 0x00000001000014c4 hello-world-with-cin`main(argc=1, argv=0x00007ffeefbff908) at hello-world-with-cin.cpp:12:9
   9       cin >> name;
   10      // Prints "Hello, !" + name
   11      // on the standard output stream (monitor)
-> 12      cout << "Hello, " << name << "!" << endl;
   13      // End of the program
   14      return 0;
   15   }
Target 0: (hello-world-with-cin) stopped.
(lldb) p name
(std::__1::string) $0 = "d0m"
(lldb) frame variable
(int) argc = 1
(const char **) argv = 0x00007ffeefbff908
(std::__1::string) name = "d0m"
(lldb) continue
Process 2756 resuming
Hello, d0m!
Process 2756 exited with status = 0 (0x00000000)
(lldb) quit
➜  lesson01
```

GDB: The GNU Project Debugger

The LLDB Debugger

https://lldb.llvm.org/use/map.html

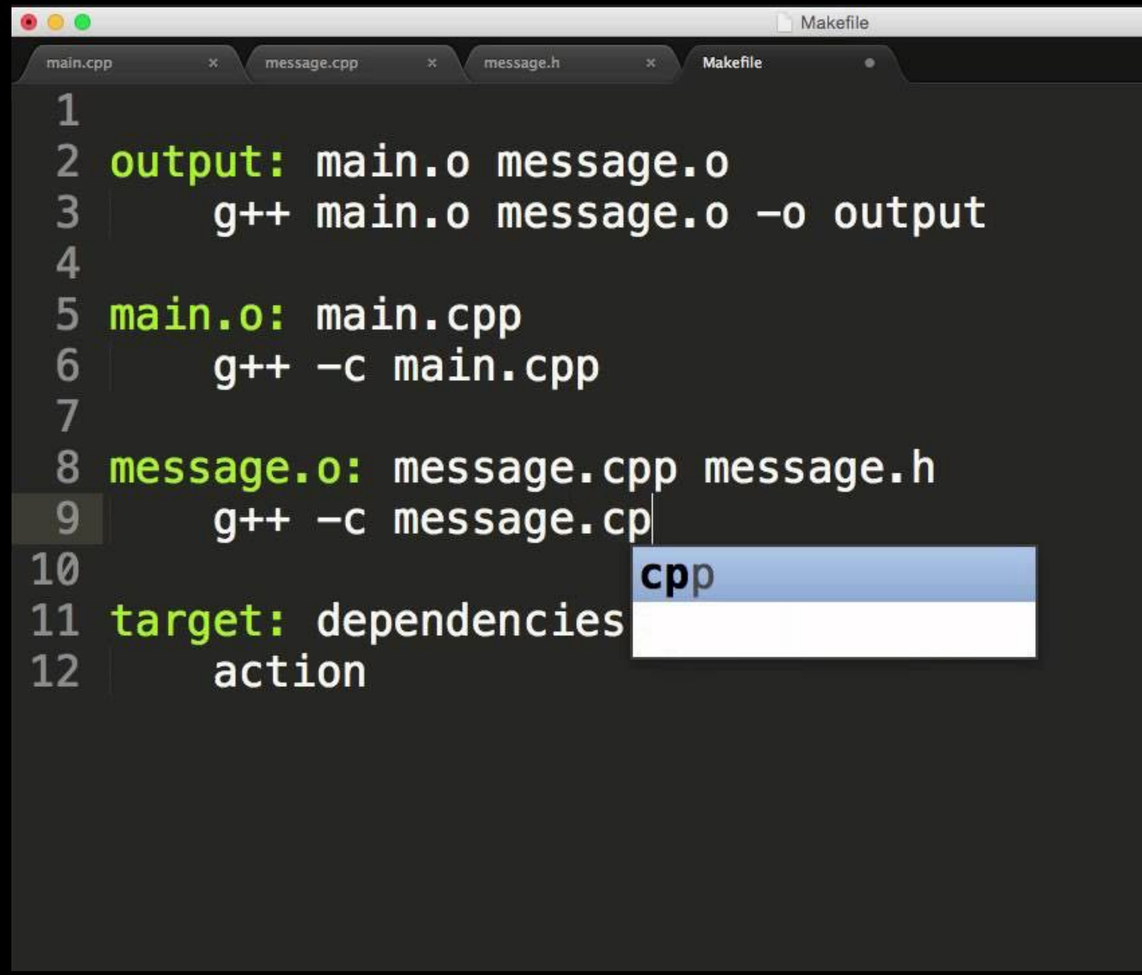
Building from multiple files

Automate the compilation process with Makefiles

You tell the Makefile "What you want to make" and “How it goes about making it”.

And then, you just run “make”

Upcoming tutorial



See https://www.youtube.com/watch?v=_r7i5X0rXJk

Your daily tasks

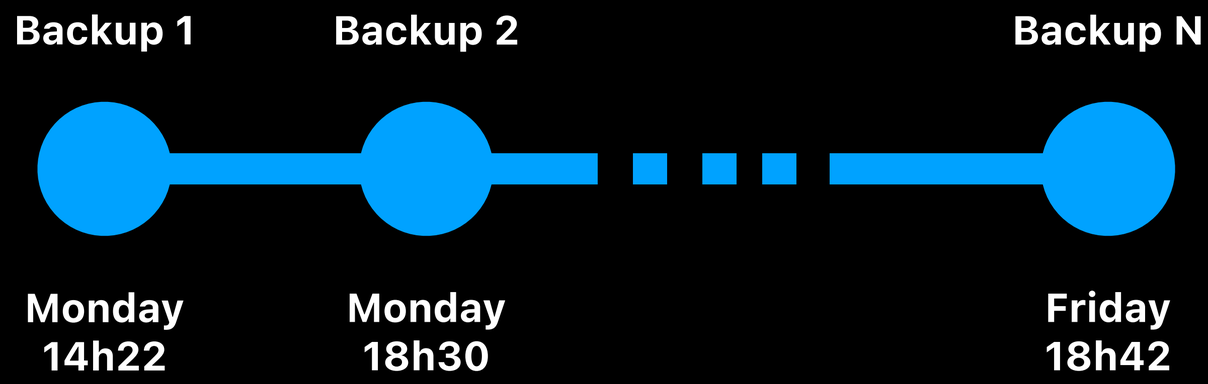
Write code

Compile code

Debug code

Update code

Run code



Not so easy to remember the difference between each backup!

How keeping TRACK OF CHANGES?

History tracking

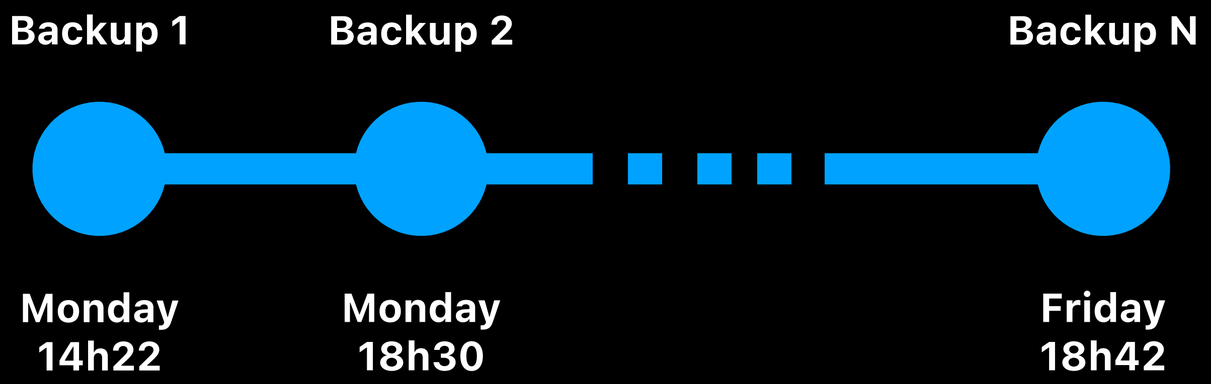
Write code

Compile code

Debug code

Update code

Run code



Collaborative History tracking

Write code

Compile code

Debug code

Update code

Run code

Relatively straightforward for a solo developer

More COMPLEX with a TEAM

See https://git-scm.com/video/what-is-version-control

Source Code Management

SCM are absolutely critical to a successful and modern development workflow

Upcoming tutorial

#01

Take Home Message

Modern IDE and Source Code Management tools can help you increase your PRODUCTIVITY.

But high-quality C++ is more a matter of OOP deep understanding than mastering language Syntax.
(See https://github.com/mortennobel/cpp-cheatsheet for a complete C++ cheatsheet)

Next Lectures

~~Lesson 00: Introduction~~

~~Lesson 01: Hello, world!~~

Lesson 02: User-defined types

Lesson 03: Inheritance

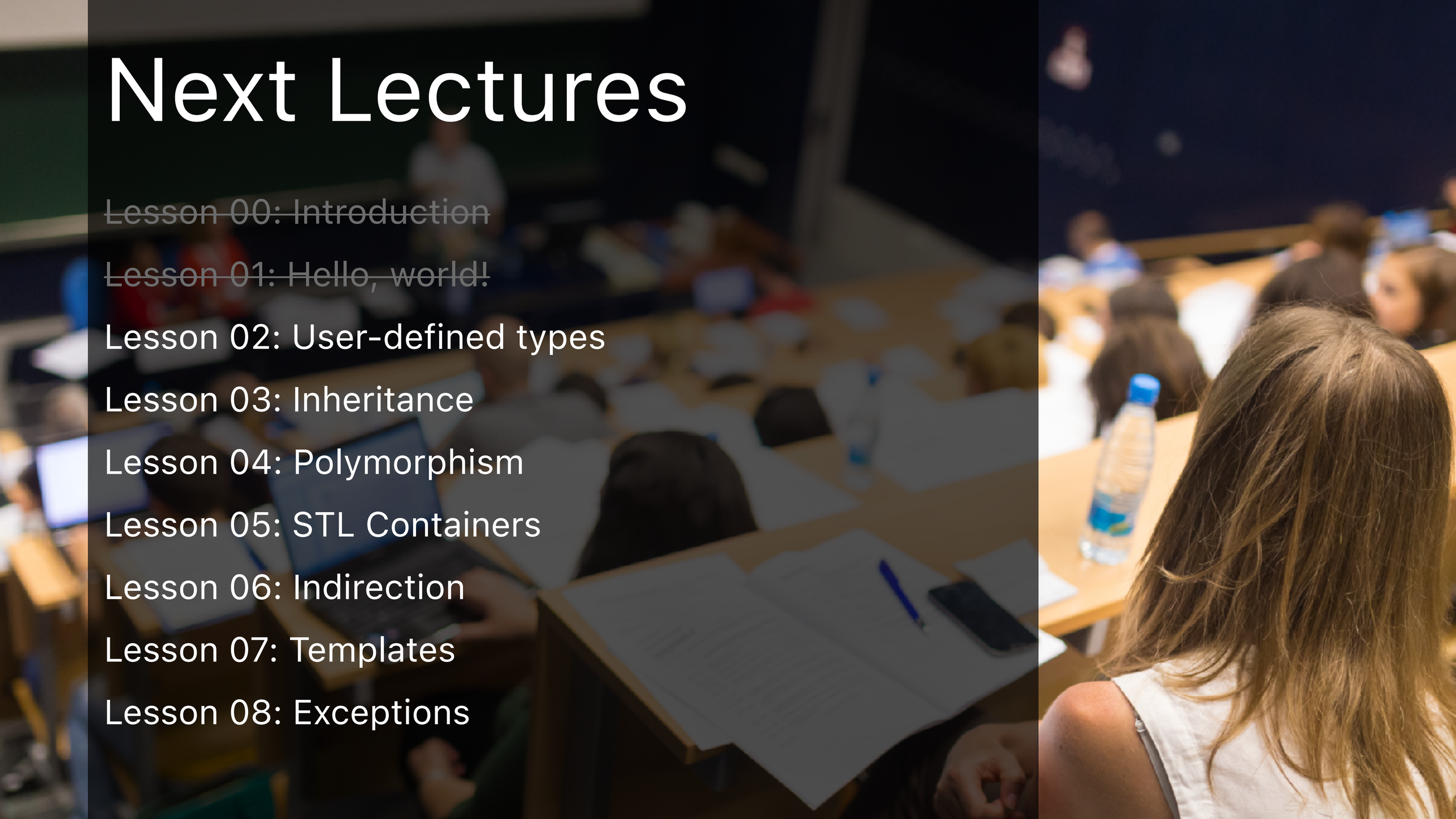
Lesson 04: Polymorphism

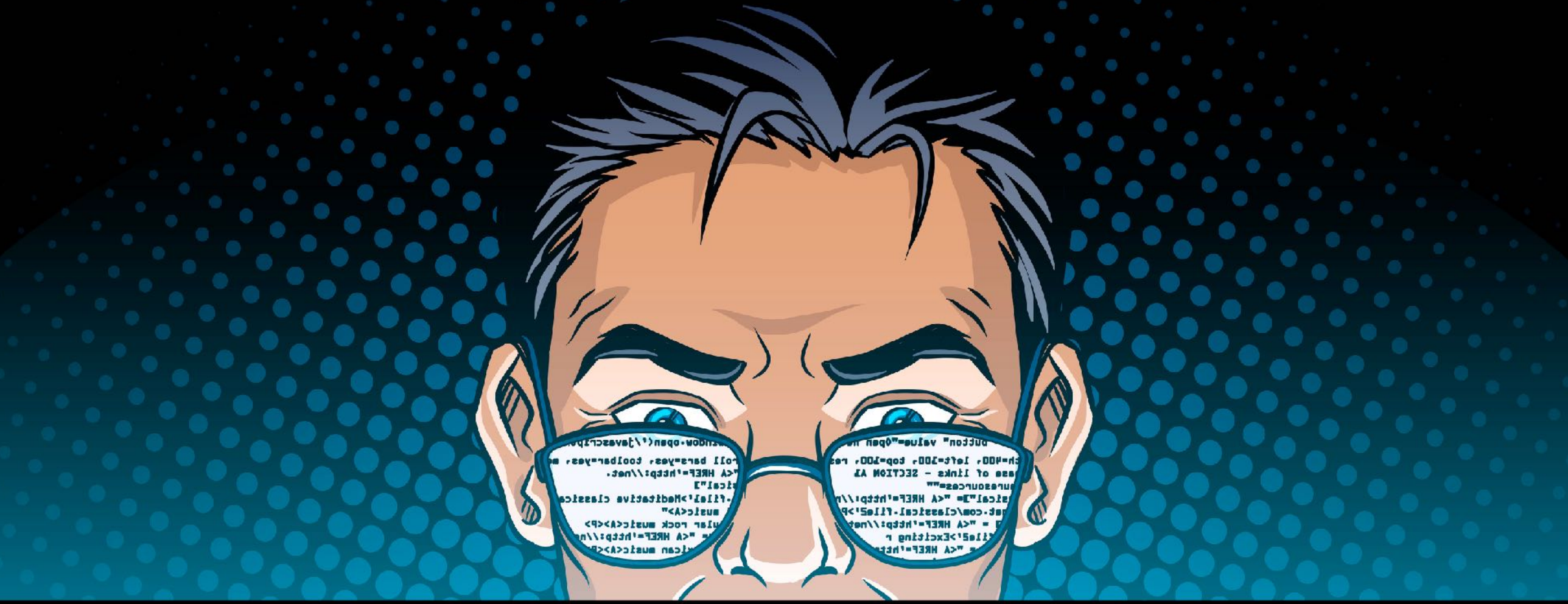
Lesson 05: STL Containers

Lesson 06: Indirection

Lesson 07: Templates

Lesson 08: Exceptions





So let’s go on C++ training with ❤️

(c) 2020/2021 - dominique.ginhac@u-bourgogne.fr - @dginhac