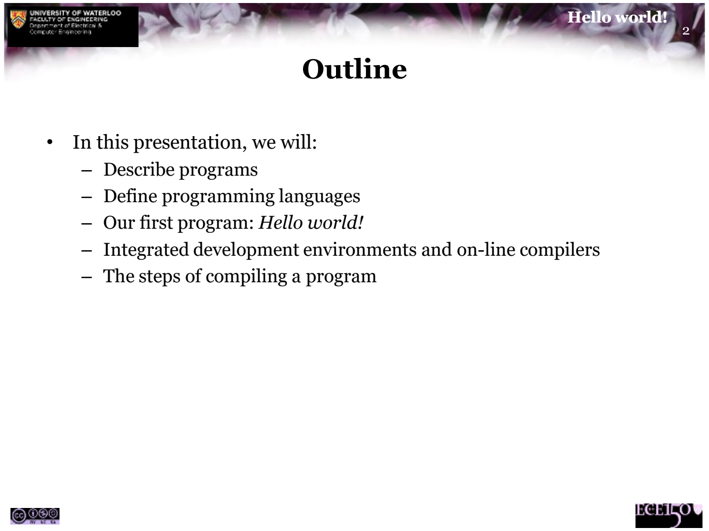
UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical &
Computer Engineering

ECE 150 *Fundamentals of Programming*

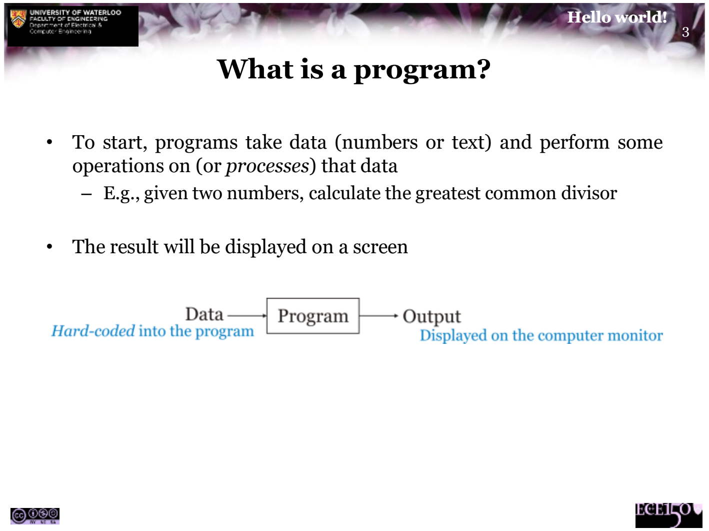
Hello world!

Douglas Wilhelm Harder, M.Math
Prof. Hiren Patel, Ph.D.
hdpatel@uwaterloo.ca dwharder@uwaterloo.ca



Outline

- In this presentation, we will:
  - Describe programs
  - Define programming languages
  - Our first program: *Hello world!*
  - Integrated development environments and on-line compilers
  - The steps of compiling a program



What is a program?

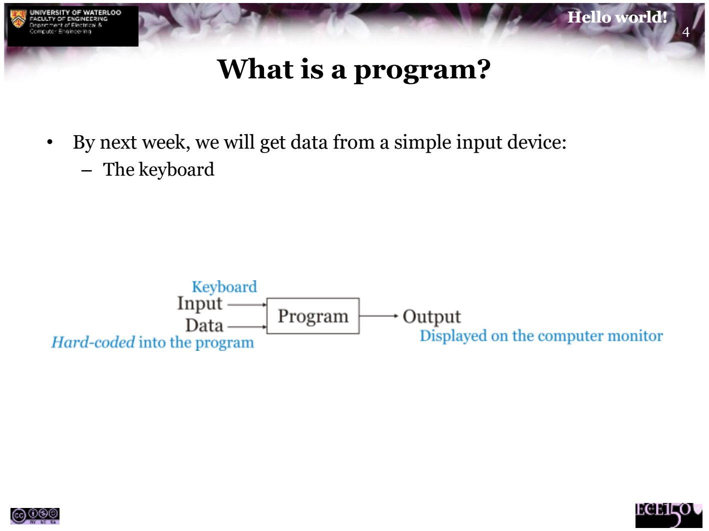
- To start, programs take data (numbers or text) and perform some operations on (or *processes*) that data
  - E.g., given two numbers, calculate the greatest common divisor

- The result will be displayed on a screen

Data → Program → Output

Hard-coded into the program

Displayed on the computer monitor



What is a program?

- By next week, we will get data from a simple input device:
  - The keyboard

Keyboard

Input → Program → Output

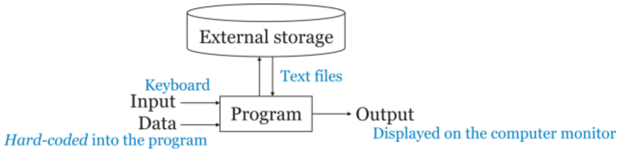
Data →

Hard-coded into the program

Displayed on the computer monitor

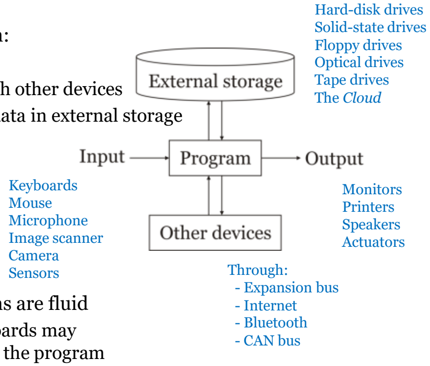
What is a program?

- After the mid-term examination, we will access from and store data in files on the hard drive


External storage
Text files
Keyboard
Input
Data
Hard-coded into the program
Program
Output
Displayed on the computer monitor

What is a program?

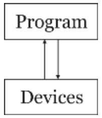
- In general, a program:
  - Receives input
  - Communicates with other devices
  - Reads and stores data in external storage
  - Produces output



- Note: these definitions are fluid
  - Printers and keyboards may communicate with the program
  - Any external storage is technically a device
    - Their dedication to long-term storage of data

What is a program?

- A simplified model:
  - A program communicates with devices to retrieve, process and store data



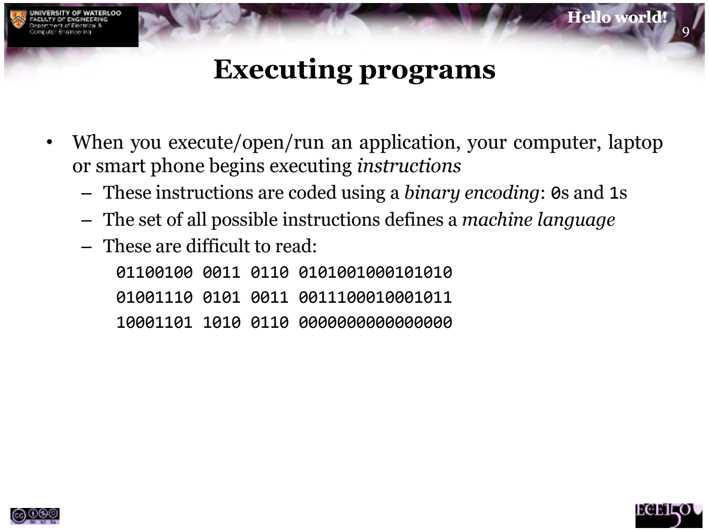
- It's still useful to consider the primary use of a device, be it for:
  - Input
  - Output
  - Storage

  or a device to be communicated with

Why learn to program?

- Why learn programming?
  - Programming is a systematic means of giving instructions to perform a task
  - If you are in electrical engineering, we have authored a web site to try to help you understand why the material in this course is relevant:

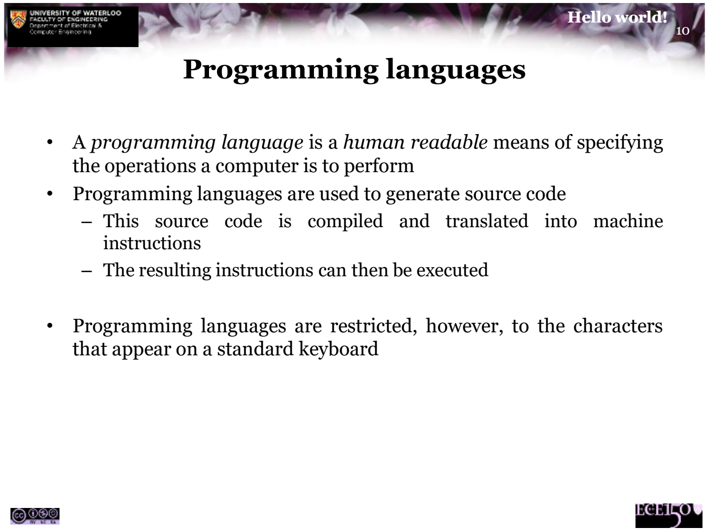
    Why learn programming for electrical-engineering students?



Executing programs

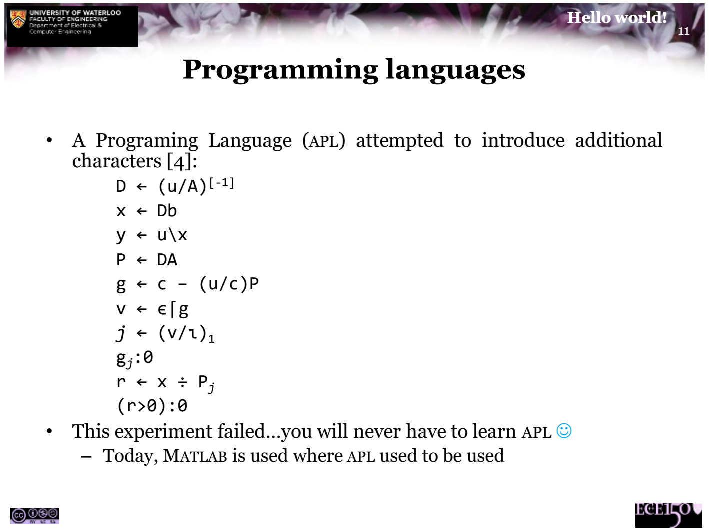
- When you execute/open/run an application, your computer, laptop or smart phone begins executing *instructions*
  - These instructions are coded using a *binary encoding*: `0`s and `1`s
  - The set of all possible instructions defines a *machine language*
  - These are difficult to read:

```
01100100 0011 0110 0101001000101010
01001110 0101 0011 0011100010001011
10001101 1010 0110 0000000000000000
```



Programming languages

- A *programming language* is a *human readable* means of specifying the operations a computer is to perform
- Programming languages are used to generate source code
  - This source code is compiled and translated into machine instructions
  - The resulting instructions can then be executed
- Programming languages are restricted, however, to the characters that appear on a standard keyboard

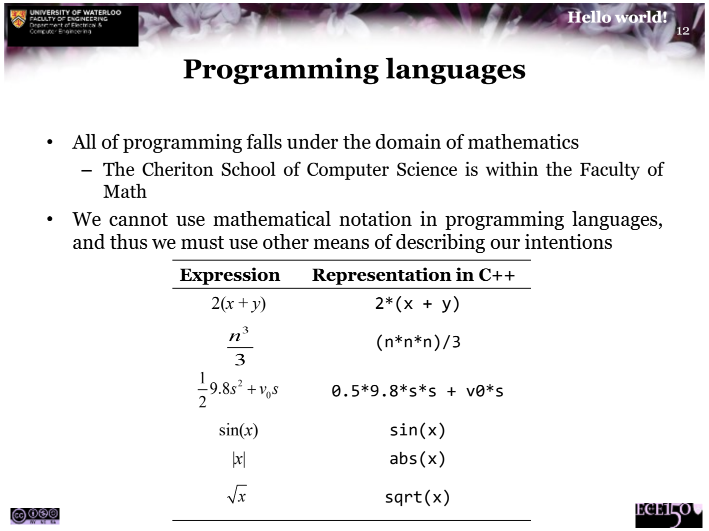


Programming languages

- A Programing Language (APL) attempted to introduce additional characters [4]:

```
D ← (u/A)^[-1]
x ← Db
y ← u\x
P ← DA
g ← c – (u/c)P
v ← ϵ⌈g
j ← (v/ι)_1
g_j:0
r ← x ÷ P_j
(r>0):0
```

- This experiment failed…you will never have to learn APL ☺
  - Today, MATLAB is used where APL used to be used



Programming languages

- All of programming falls under the domain of mathematics
  - The Cheriton School of Computer Science is within the Faculty of Math
- We cannot use mathematical notation in programming languages, and thus we must use other means of describing our intentions

Expression	Representation in C++
$2(x+y)$	`2*(x + y)`
$\frac{n^3}{3}$	`(n*n*n)/3`
$\frac{1}{2}9.8s^2 + v_0 s$	`0.5*9.8*s*s + v0*s`
$\sin(x)$	`sin(x)`
$\lvert x \rvert$	`abs(x)`
$\sqrt{x}$	`sqrt(x)`

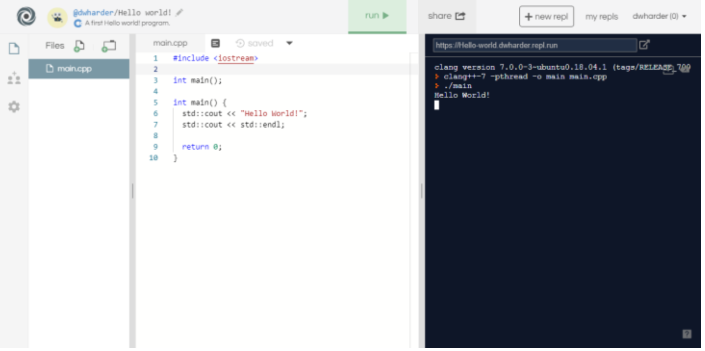
Our first program

```
#include <iostream>

int main();

int main() {
	std::cout << "Hello world!";
	std::cout << std::endl;

	return 0;
}
```

Our first program

- There are two approaches we can take to compiling and executing this code:
  - Using an Integrated Development Environment (IDE)
    - We will use Eclipse in the laboratories
  - Using an on-line compiler such as https://repl.it/
- On-line compilers, however:
  - May not always be available
  - Are useless for larger projects

Our first program

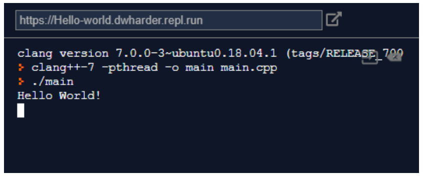


```
#include <iostream>

int main();

int main() {
  std::cout << "Hello World!";
  std::cout << std::endl;

  return 0;
}
```

```
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700
› clang++-7 -pthread -o main main.cpp
› ./main
Hello World!
```

Our first program

- When you select the **Run** button, text is printed to the console output



```
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700
› clang++-7 -pthread -o main main.cpp
› ./main
Hello World!
```

  - Question: What is happening behind the scene?

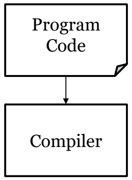
Steps in generating an executable program

- The program undergoes the following four steps in order to create an executable program that you can run
  - Step 1: Creating the program using a programming language, and writing it using an editor

Program Code

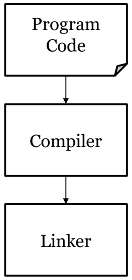
Steps in generating an executable program

- The program undergoes the following four steps in order to create an executable program that you can run
  - Step 1: Creating the program using a programming language, and writing it using an editor
  - Step 2: Compiling the program into machine-language code



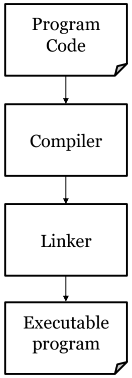
Steps in generating an executable program

- The program undergoes the following four steps in order to create an executable program that you can run
  - Step 1: Creating the program using a programming language, and writing it using an editor
  - Step 2: Compiling the program into machine-language code
  - Step 3: Linking together the program with other helper programs into a single executable program
    - E.g., printing to the screen



Steps in generating an executable program

- The program undergoes the following four steps in order to create an executable program that you can run
  - Step 1: Creating the program using a programming language, and writing it using an editor
  - Step 2: Compiling the program into machine-language code
  - Step 3: Linking together the program with other helper programs into a single executable program
    - E.g., printing to the screen
  - Step 4: Executing the program



Summary

- In this presentation, you now
  - Understand what a program is
  - Have an overview of how computers executing instructions
    - These are encoded in binary: `0`s and `1`s
  - Understand that programming languages allow us to define programs using a human-readable interface
    - The program must be compiled into an executable and run
  - Have written your first program: the ubiquitous *Hello world!*
  - Saw this output on https://repl.it
    - The first lab includes installing the Eclipse IDE
    - You are not required to use Eclipse, but it is the only IDE that is supported
  - Understand the steps of compilation

References

[1] https://en.wikipedia.org/wiki/APL_(programming_language)

Acknowledgments

Proof read by Dr. Thomas McConkey

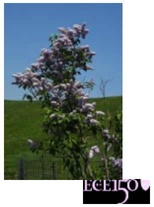
Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.



Disclaimer

These slides are provided for the ECE 150 *Fundamentals of Programming* course taught at the University of Waterloo. The material in it reflects the authors' best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.