Outline

- In this presentation, we will:
  - Describe programs
  - Define programming languages
  - Look at our first program: Hello world!
  - Introduce:
    - Integrated development environments
    - On-line compilers
  - Describe the steps of compiling a program

What is a program?

- To start, programs give instructions to a processor to take data (numbers, text, or more generally, information) and perform some operations on (or processes) that data to solve a problem

- Initially, the result will be displayed on a screen

What is a program?

- By the end of this week, we will get data from a simple input device: the keyboard

- Definition: a console is the combination of a keyboard and screen
What is a program?

• By the middle of the course, we will access from and store data in files stored within a file system

![Diagram](image)

External storage

- A file system on a hard-disk drive or solid-state drive

Input

Data

Processor

Output

What is a program?

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

![Diagram](image)

External storage

- File systems or databases on
  - Hard-disk drives
  - Solid-state drives
  - Floppy drives
  - Optical drives
  - Tape drives
  - The Cloud

Input

Program

Processor

Output

What is a program?

• In upper years, you will investigate embedded systems that are much simpler

![Diagram](image)

External storage

- Sensors

Processor

Actuators

Input

Program

Output

What is a program?

• A simplified model:
  - A program provides instructions for the processor to communicate with devices to retrieve, process, store and send data

![Diagram](image)

Program

Processor

Output

Input

Devices

Through:
- Expansion bus
- Internet
- Bluetooth
- CAN bus

Devices

Monitors
Printers
Speakers
Actuators

Keyboards
Mouse
Microphone
Image scanner
Camera
Sensors
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? https://ece.uwaterloo.ca/~ece150/Why_programming_for_EE/

Executing programs

• When you execute/open/run/launch an application, your computer, laptop or smart phone begins executing instructions
  – These instructions are coded using a binary encoding:
    • 0 V or 5 V or 0s and 1s
    – 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 00000000000000

Programming languages

• A programming language is a human readable means of specifying the operations a computer is to perform
• Programming languages are used to author source code
  – This source code is compiled and translated into machine instructions
  – The resulting instructions can then be executed
• Note that some programming languages are interpreted
  – Thus, C++ will be very different from Python or Maple or MATLAB
• Programming languages are restricted to the characters that appear on a standard keyboard
  – These are derived from ASCII
  • The American Standard Code for Information Interchange

Expression | Representation in C++
--- | ---
2(x + y) | 2*(x + y)
\( \frac{n}{3} \) | (n*n*n)/3
\( \frac{1}{2} 9.8s^2 + v_0s \) | 0.5*9.8*s*s + v0*s
\( \sin(x) \) | sin(x)
|d| | abs(x)
\( \sqrt{x} \) | sqrt(x)
Hello world!

Our first program

```cpp
#include <iostream>

// Function declarations
int main();

// Function definitions
int main()
{
    // Display this text to the console
    std::cout << "Hello world!" << std::endl;
    return 0;
}
```

Comments allow for the programmer to add commentary in English—these are not part of the program.

There are three approaches we can take to authoring, compiling, executing and testing 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/](https://repl.it/)
- Using a text editor (vi) and a command-line compiler (gcc)

On-line compilers, however:
- May not always be available
- Become increasingly useless for larger projects

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

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
  - 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: 0s and 1s
  – 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


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.