INTRODUCTION TO COMPUTER PROGRAMMING

Fabio Stroppa

Why is software important?

Almost all electronic devices run some software:

- Personal computers
- Smartphones
- Automobile engine control systems
- Medical devices (e.g., DaVinci surgeon robot, Exoskeletons for Rehabilitation)
- Office machines (e.g., photocopiers)
- Domestic devices (e.g., smart TVs, washers/dryers, dishwashers)
- Many more…

Why software-based solutions are implemented instead of only hardware?

- More cost effective to implement software than hardware
- Software bugs are easier to fix than broken or faulty hardware components
- As systems become increasingly complex, bugs are unavoidable
- Software allows new features to be added later (e.g., patches for videogames)
- It allows manufacturers to implement only the minimal functionalities on hardware and do the rest in software

Which are the software-related jobs?

There are many more software jobs than hardware ones:

- Application developer
- Cyber security analyst
- Game developer
- Information system manager
- IT consultant
- Multimedia programmer
- Web developer/designer
- Software engineer

Which language should you learn?

Wikipedia claims there are approximately 700 programming languages, while others say that number is closer to 9,000!

Most relevant programming languages: C, C++, Java, C#, MATLAB, Python, Fortran

You can check the Programming Language Popularity Normalized Comparison at http://ww1.langpop.com/

Some information about C/C++

C

- Created by Dennis Ritchie, Bell Labs in 1970s
- International standard ISO/IEC 9899:2011 (informally known as “C11”)

Dennis Ritchie

C++

- Created by Bjarne Stroustrup, Bell Labs in 1983
- International standard ISO/IEC 14882:2014 (informally known as “C++14”)

Bjarne Stroustrup

Why we will learn C and C++

- Vendor neutral
- International standard
- General purpose
- Powerful yet efficient
- It allows developers to directly manage memory
- Easy to move from C/C++ to other languages, but often not in the other direction
- Many other popular languages were inspired by C/C++
- When something lasts in computer industry for more than 40 years, outliving its creator, it must be good!

Why not, for example, Java?

Besides the motivations listed in the previous slides

- Java implements everything as a class. One must know what a class is before to approach Java. In C++ you can do both.
- C/C++ are more efficient than Java.
- Next year, you will learn how to manage memory dynamically.
- Every programmer should start learning from C/C++.

Why not, for example, Java?

Besides the motivations listed in the previous slides

- Java implements everything as a class. One must know what a class is before to approach Java. In C++ you can do both.
- C/C++ are more efficient than Java. ← Are they though?
- Next year, you will learn how to manage memory dynamically.
- Every programmer should start learning from C/C++.

Why not, for example, Java?

B

- to
-
-
-

SUPER WISE ABSOLUTE TRUTH:

In the long run, the programming language does not really matter. One can learn a programming language in one night. What it really matters is to learn how to code and how to solve problems.

Overview of programming languages

Type of Execution:

Compiled Languages

- Translated to the target machine's native language by a program called *compiler*.
- Compilation happens only once; execution does not require re-compilation.
- Very fast code.
- Not portable across operating systems.

Interpreted Languages

- Read by a program called an *interpreter* and directly executed by it.
- Interpretation and execution happen at the same time.
- Usually much slower than an equivalent compiled program.
- Portable across operating systems.

Just-In-Time Compiled Languages

- Involves compilation during execution of a program (at run time) rather than before execution.
- Offers balance between performance and portability.

Overview of programming languages

Nature of the language:

Low-Level Languages

- Generally, quite similar to machine code.
- Subjected to hardware.
- Difficult to port to other platforms.
- They are always compiled.

High-Level Languages

- Easy to understand by the human mind (e.g., mathematical functions).
- Takes less time to develop a program.
- As a trade-off, it sacrifices some degree of control over what the resulting program actually does.

Overview of programming languages

Type system:

Strongly Typed

Variables are bound to specific data types

Weakly Typed

Variables are not bound to specific data types

Static Checking

Types are checked during compilation or interpretation

Dynamic Checking

Types are checked at run time

Safe

Prohibits operations on typed variables that might lead to undefined behavior or errors

Unsafe

Gives more responsibilities to the developer