CS 211 Notes

Introduction to Programming

Jaeden Bardati

September 22, 2021

1 Course Overview

September 10, 2021

1.1 Objectives

The course is intended to teach how to develop a computer program to solve a problem. C++ is a tools that will be used to develop these skills and logical thinking. These skills will be transferable to other languages.

2 Computer Organization

2.1 Hardware

September 11, 2021

2.1.1 Components

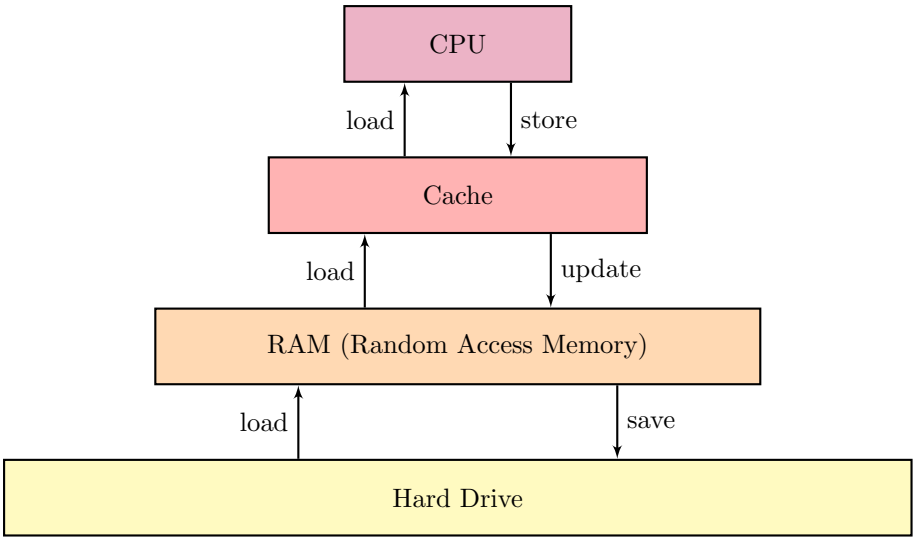
Modern computers are built using the **Von Neumann machine**. There are three aspects:

- **Architecture**: **I/O** (User interaction) + **Memory** (Storage) + **CPU** (CU: Control Unit, ALU: Arithmetic and Logic Unit). These are all connected by a shared bus.
- **Stored Programs**: All programs and data are stored in memory (binary).
- **Sequential Execution**: Also called the **fetch-decode-execute** cycle. Instructions are **fetched** from memory, **decoded** by the CU and then **executed** by the ALU. If there is a result, it is stored back in memory.

2.2 Memory

September 11, 2021

The memory is organized in a **hierarchy**. At the bottom of the hierarchy is the Hard Drive (in TB). At the top is the CPU. Since the hard drive is slow, when some data from the hard drive is needed, it is first loaded into **RAM (Random Access Memory)** (in GB). The RAM is still too slow for the RAM, so the data is stored in **cache** (in KB or MB). Yet still, this is not fast enough for the CPU, so **registers** (in Bytes) in the CPU itself are used to store variables.



As you **go up** the hierarchy, the **speed increases**, but the **size decreases** and the **cost increases**.

2.2.1 RAM

Random access memory is organized in an array of Bytes ("words").

Words in RAM are addressed with a byte themselves (e.g. 01101101 is an address). These are typically written in hexadecimal (e.g. 6D).

Words in RAM can be data or machine code instructions. Instructions contain a binary code for each operation (for example, addition). Instructions codes are dependent on the CPU.

2.3 First C++ Code

September 12, 2021

The following code is a hello world program in C++.

```
// helloworld.cpp
#include <iostream>   // include statement allows the use of C++ libraries
#include <stdio.h>    // this library contains getchar()

using namespace std; // a standard environment (input from keyboard, output is the screen)

int main() {

   cout << "Hello world!" << endl;  // Prints "Hello world!" to the screen

   getchar();     // wait for user to type a character

   return 0;     // 0 means that the execution was successful
}
```

2.4 Data types

September 12, 2021

Variables are referred to as identifiers. Identifiers are memory locations accessed and modified.

Inside a main function (as above), the following code declares a variable of integer type in C++.

```
int numYears;     /// allocated space in memory to contain num of years
```

You can also initialize the variable with a value on declaration:

```
int number = 5;   /// declare and initialize (give a value too)
```

Some rules to follow when naming variables are:

- Names have meanings
- Must be case sensitive (e.g. numYears is not numyears)
- Consists of letters, numbers and underscores
- First character cannot be a number

Some types of variables are:

- Integers (e.g. -5, 0, +2) [int]
- Real numbers (floating point numbers or doubles, e.g. 2.453, -4.1987e7) [float or double]
- Booleans (e.g. true, false) [bool]
- Characters [char]

You can assign a value to a variable after declaring it:

```
int width, height;
int area;

width = 5;
height = 3;

area = width * height;
```

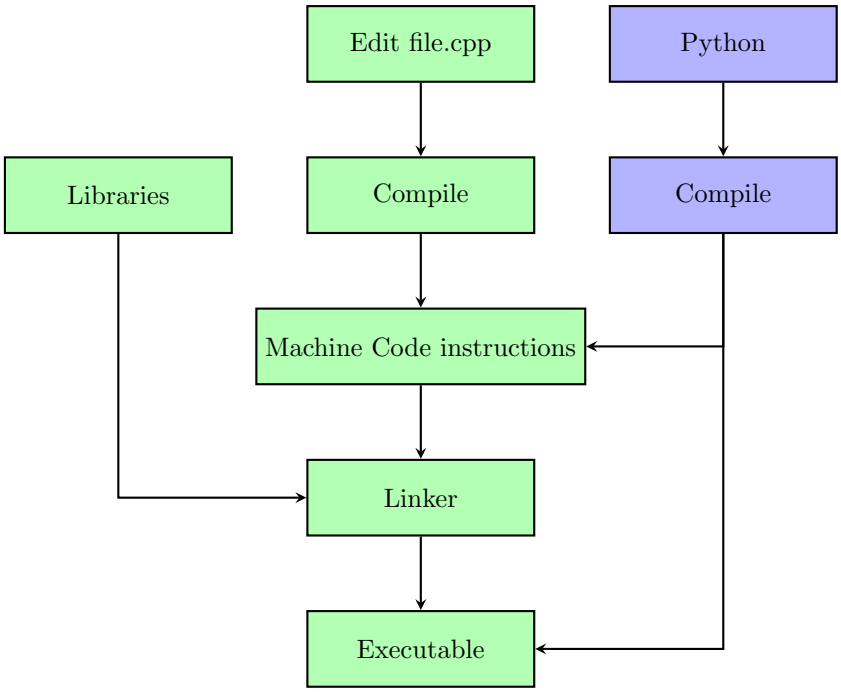
Constants (denoted with the keyword "const") cannot be changed throughout the program. The convention is to use capital letters for constants.

```
const double PI = 3.14159265;
cout << "pi is " << PI << endl;

int radius = 6;
double area2 = PI * radius * radius;
```

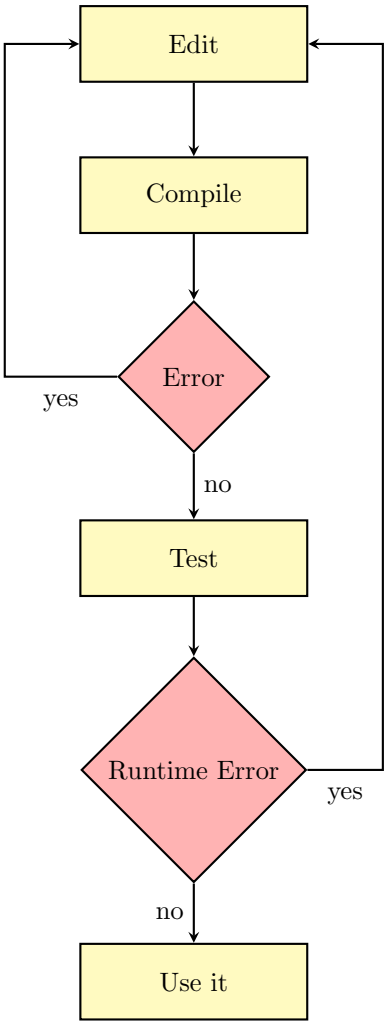
2.5 A Flow Chart: Program to Binary
September 12, 2021

This is a flow chart of what is done by the computer when compiling a C++ file. In blue is the Python equivalent.



Note that the bottom of the flow chart is the same for all programming languages, because in all languages, CPU-specific machine code is needed to execute code.

The process of catching errors is as according to the following flow chart.



In this context, **errors** are caught by the compiler. This is opposed to **runtime errors**, which are not caught by the compiler. These can be something like division by zero or infinite loops.