Introduction to C++ Part 1: Getting set up, cin, cout, vectors

Overview

- C++ can run differently depending on what machine you are using
- We will be using a standard set up that everyone will have to use
- To do this we will be running Linux and using C++11
  - More Specifically, Ubuntu (Lubuntu) 16.04

Options for Running Ubuntu

- Virtual Machine (Recommended)
  - Easiest to set up
  - Have a pre built Virtual Machine you can download and run
  - All software you need for CSE 1325 is pre-installed
  - Can set up your own if you want
- Run Ubuntu on your machine
  - Install on a spare computer you have
  - Dual boot (requires partitioning hard drive)
- Purchase a machine with Ubuntu Installed

Virtual Machine Set up (prebuilt)

- Download and Install Virtual Box (https://www.virtualbox.org/wiki/Downloads)
- Download the prebuilt Virtual Machine (2GB) (https://drive.google.com/open?id=0B2kXlVDGGdp1OGhDUFF0aVBESVk)
- In Virtual Box
  - File -> Import Appliance -> select what you just downloaded
  - Appliance Settings
    - Change RAM to 2048 (if you have 4GB or more RAM) else use half of your RAM
    - Make sure “Reinitialize the MAC address: is enabled

Virtual Machine Set up (prebuilt)

- Once all set up, click the CSE1325 Lubuntu then click start.
- If you receive an error,
  - go into your BIOS and make sure Intel Virtualization Technology is turned on
  - Make sure Hyper-V is either disabled or uninstalled
- Password for account is “student”
- Go to “Sharing a Folder Slide”

Virtual Machine Set Up (your own)

- Download and install Virtual box
- Download Ubuntu Desktop 16.04 iso (turn off all donations)
  - Lubuntu 16.04 is also acceptable
- In Virtual Box
  - Machine -> New
  - Select Linux and Ubuntu 64 bit. Name the machine
  - Set memory size (same guidelines as above)
  - Create Virtual Hard Disk
    - 10GB minimum (40 is preferred)

Virtual Machine Set Up (your own)

- In Virtual Box Still
  - Machine -> Settings -> Storage
    - Click icon far right of “Optical Drive”, then click Choose Virtual Optical Disk File”
    - Choose the Ubuntu iso
  - Machine -> Start -> Normal start
    - Follow Prompts to set up Ubuntu

- Go to “Sharing a Folder” slide

Sharing a Folder (Optional)

- Allows you to have a folder accessible by your computer and the virtual computer
- If you used your own set up, additional steps must be taken (see Additional Steps slide)
- Virtual Box (with Linux not booted)
  - Machine -> Settings -> Shared Folders
  - Select Add Shared Folder icon
  - Select Folder you want to share (Documents, Class specific folder, etc)
  - Enable Auto-mount in add share dialog
  - Click ok in shared folders dialog

Shared Folder Additional Steps

- If you plan on sharing a folder with VirtualBox and your host operating system, you have a few additional steps
- Open bash, e.g., Ctrl-Alt-t
- Install Guest Utils, typing your password when prompted:
  sudo apt-get install virtualbox-guest-utils
- Add yourself to the vboxsf group:
  sudo usermod -a -G vboxsf student
- Shutdown:
  sudo shutdown now
- Complete “Sharing a Folder with VirtualBox” slide
- Complete “Installing Tools under Ubuntu” slide

Virtual Box Tips

- Go / exit full-screen by pressing RIGHT Ctrl-f
- Change your password in bash via passwd
- Take snapshots occasionally, for more info see http://news.filehippo.com/2014/06/use-snapshot-virtualbox/
- NEVER close VirtualBox while a machine is running within it!
  - This is like pulling the cord out of the wall for a desktop computer!
  - Instead, shut down via the menu or via bash's
    sudo shutdown now

Native Ubuntu

- Install natively or via dual boot
  - https://www.ubuntu.com/download/desktop/install-ubuntu-desktop
  - rEFInd (https://sourceforge.net/projects/refind/) seems to be highly regarded for managing boot images on a Mac
- Purchase a machine with Ubuntu pre-installed
  - Try e.g., http://dell.com/developers, http://system76.com, or http://emperorlinux.com/

Installing Tools (for your own VM or Native)

- Open Terminal

```
$ #Do NOT type the $ - that's a prompt!
$ sudo apt-get update
$ sudo apt-get install build-essential
$ sudo apt-get install ddd
$ sudo apt-get install libgtkmm-3.0-dev
$ sudo apt-get install libgstreamermm-1.0-dev
$ sudo apt-get install libgtkmm-3.0-doc
$ sudo apt-get install libgstreamermm-1.0-doc
$ sudo apt-get install devhelp
$ sudo apt-get install gtk-3-examples
$ sudo apt-get install git-all
$ sudo apt-get install umbrello
$ sudo apt-get install kio
$ sudo apt-get install oxygen-icon-theme
$ sudo apt-get install gedit - NOT INSTALLED
```

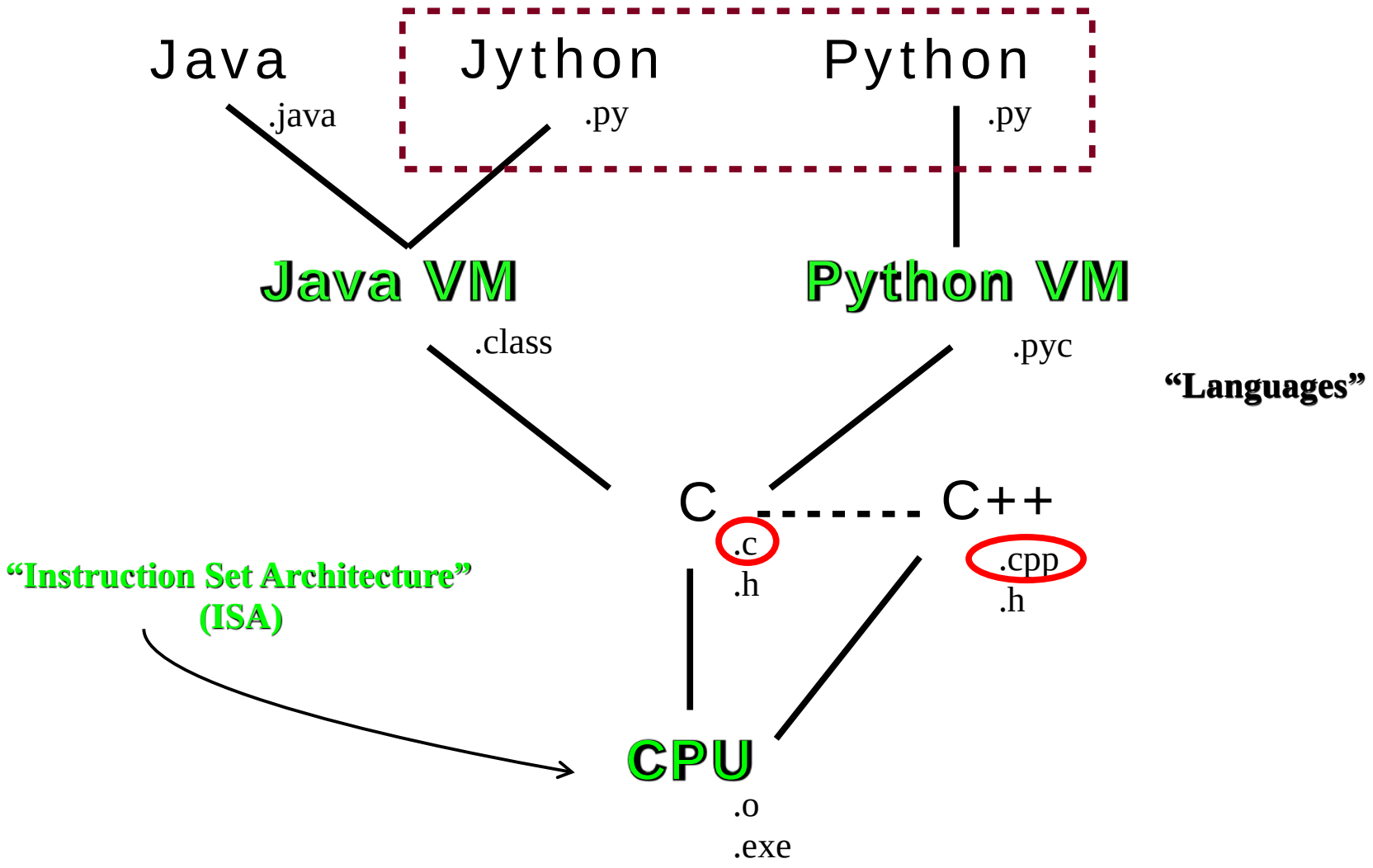
Resources for C++

- No Standard Online Documentation
  - Cplusplus.com is a good start
  - Has identifiers for what is specific to different versions
- Stack Overflow
- O’Reilly books
- Cplusplus.com’s forum for beginners.

Bash or Terminal

- Most everything in Linux relies on bash
  - Command Line Interface (CLI)
- Programmers sometimes find these more efficient than GUIs
  - Easier to type out a command you know than digging through menus
- Hand out summarizing bash will be posted on blackboard

Language Hierarchy



Writing the Canonical 1st Program

Python:

Structured
Object-Oriented

```
print("Hello, World")
```

C:

Structured

```
#include <stdio.h>
main() {
   printf("Hello World");
}
```

Java:

Object-Oriented

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

C++:

Structured
Object-Oriented

```
#include <iostream>
using namespace std;
int main() {
  cout << "Hello World!" << endl;
}
```

Don't worry about these yet

Comparisons between Compiled code

Python

```
>>> dis.dis(hello)
>>> 2     0 LOAD_CONST
          1 ('Hello, world!')
          3 PRINT_ITEM
          4 PRINT_NEWLINE
          5 LOAD_CONST
          0 (None)
          8 RETURN_VALUE
```

Python Virtual Machine Code

Java

```
ricegf@pluto:~/dev/cpp/1$ javap -c HelloWorld.class
Compiled from "HelloWorld.java"
public class HelloWorld {
  public HelloWorld();
    Code:
       0: aload_0
       1: invokespecial #1
                  // Method java/lang/Object."<init>":()V
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #2
              // Field java/lang/System.out:Ljava/io/PrintStream;
       3: ldc           #3
                  // String Hello, World
       5: invokevirtual #4
      // Method java/io/PrintStream.println:(Ljava/lang/String;)V
       8: return
}
ricegf@pluto:~/dev/cpp/1$
```

Comparisons between Compiled code

- C++ (just a snippet)

```
ricegf@pluto:~/dev/cpp/1$ objdump -d a.out

a.out:     file format elf64-x86-64


Disassembly of section .init:

0000000000400600 <_init>:
  400600:        48 83 ec 08              sub    $0x8,%rsp
  400604:        48 8b 05 ed 09 20 00     mov    0x2009ed(%rip),%rax        # 600ff8
<_DYNAMIC+0x1e0>
  40060b:        48 85 c0                 test   %rax,%rax
  40060e:        74 05                    je     400615 <_init+0x15>
  400610:        e8 1b 00 00 00           callq  400630 <__gmon_start__@plt>
  400615:        48 83 c4 08              add    $0x8,%rsp
  400619:        c3                       retq

Disassembly of section .plt:

0000000000400620 <__gmon_start__@plt-0x10>:
  400620:        ff 35 e2 09 20 00        pushq  0x2009e2(%rip)        # 601008
< GLOBAL_OFFSET_TABLE_+0x8>
```

Hello World Code Examples

Differences from the book

- The book uses “std_lib_facilities.h”
- Sometimes causes problems, so we wont be using this.
- Standard C++11 libraries
- Keep this in mind if you decide to use the book.
- If you want to use it: http://stroustrup.com/Programming/std_lib_facilities.h

I already expect you to know:

- Functions
- Arrays
- Boolean logic (&&, ||, !, ==)
- Basic types (bool, int, string, double)
- Basic operations (+, -, *, /, =, %)
- Loops
- Files
- Include statements
- Header files (.h files)
- Scope
- Pointers

Special Notes About cin and >>

- cin only reads until a whitespace
  - “Shawn Gieser” is read as two separate inputs
  - Following code example to handle this

```
string first_name;
string last_name;
cin >> first_name >> last_name;
```

  - Can also use the method getline(cin, variable)

- >> converts input to what type you are storing it in
  - Storing “22” will work with a string or an int
  - Storing “Name” will work with a string, but not an int

Vectors

- Vectors are basically a list of data
- Like arrays but different
  - Do not have to define the size of the list
- All elements have to be the same time.
  - Type decided when you create vector
- Must #include <Vector> to use

Vectors

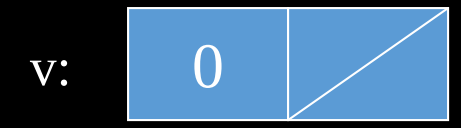
- To create a vector
- Vector<vector_type> vector name;
- Vector<int> v1;
- Vector<int> v2 = {1, 2, 3, 4, 5}

Vectors

- To determine how many elements there are
  - size()
- To add elements to the end of the vector
  - push_back(data);
- Find beginning and/or the end of the vector
  - begin(), end()
- Accessing an element
  - vector_name[#]

Vectors

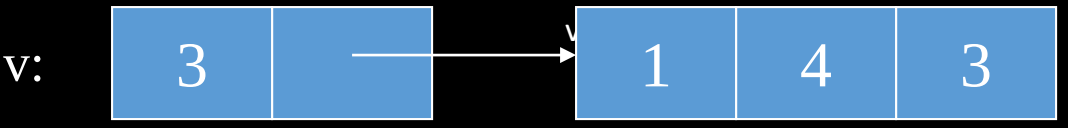
vector<int> v; // start off empty



v.push_back(1); // add an element with the value **1**

v.push_back(4); // add an element with the value **4** at end (“the back”)

v.push_back(3); // add an element with the value **3** at end (“the back”)



Vector Example – Sorting

```
// compute mean (average) and median temperatures:
#include <iostream>  // for cin and cout
#include <vector>    // for vector
#include <algorithm> // for sort
using namespace std;

int main()
{
    vector<double> temps;      // temperatures in Fahrenheit, e.g. 64.6
    double temp;

    while (cin>>temp)
        temps.push_back(temp); // read and put into vector

    double sum = 0;
    for (int i = 0; i<temps.size(); ++i) sum += temps[i];  // sums temperatures

    cout << "Mean temperature: " << sum/temps.size() << '\n';

    sort(temps.begin(), temps.end());
    cout << "Median temperature: " << temps[temps.size()/2] << '\n';
}
```

Homework 1

- Set up your workspace
- Basic input and output
  - Hints Chapter 3 in book
  - If cout is output what is input?
- Bonus section
- Take screen shots of code working
- Upload code and screen shots in a zip file to blackboard

Git

- What is Git?

Git

- Version Control – The task of keeping software system consisting of many version and configurations well organized.
- Basically Backups for each version you make
- Useful if:
  - You accidently delete your code
  - You change something and nothing works
  - Your team mate changes something and you have no idea what

Git

- Bitbucket is a good resource to store your git on the cloud.
- GitHub is the most popular, but Bitbucket allows student accounts to create free private accounts. Github forces you to pay for private accounts.
- I’ll post resources how to use Git