Data Structures

CSCI 2270-202: REC 01

Sanskar Katiyar

About Me

Sanskar Katiyar

sanskar.katiyar@colorado.edu

Master’s in *Computer Science* (‘21)

Interests: Complex Systems, Robotics (Planning & Perception)

Fall ‘19: GSS for CSCI 1320: CS1 - Engineering Applications

Logistics: Overview

Recitation

Thursday, 8 am – 9:15 am at MUEN E113

To earn credit: Show your work before you leave

Office Hours at ECAE 128 (Aerospace Lobby) – Table 1

Tuesday: 11:30 am – 1:30 pm

Friday: 2:30 pm – 4:30 pm

Logistics: Office Hours

Office Hours conflict with class schedule?

1. Attend **any*** CSCI 2270 TA’s Office Hours.
2. I can change my hours to suit the class, if necessary.
3. Email me, we can set up a Zoom meeting, etc.

Google Calendar (via Moodle)

1. (All) Instructors, TAs, CAs, CMs
2. Available on Moodle, under **Course Logistics.**

Recitation Outline

1. Moodle & Piazza
2. VS Code (Setup)
3. VS Code and C++
4. Functions in C++
5. File I/O in C++
6. Exercise

Moodle & Piazza

Moodle

***For**: Course material, Assignments, Quizzes, Recitations, Midterms, Announcements, Google Calendar, etc.*

moodle.cs.colorado.edu

Login with your CU (Identikey, Password)

CSCI 2270 - Zagrodzki, Ashraf, Trivedi - CS2: Data Structures

Enrolment Key: **<Placeholder: Email me for key>**

Piazza

***For**: Discussion with classmates on lecture material, sharing project ideas, announcements etc.*

Anonymous to classmates; **NOT** to teaching staff.

Do **NOT** share answer code snippets on Piazza **[Honor Code]**

piazza.com/colorado/spring2020/csci2270

Login/Register with your ***colorado.edu*** Email

VS Code (Setup)

VS Code

Follow Instructions: “**VS Code Setup Guide**” on Moodle

Why VS Code?

Local development environment & Consistency

Terminal commands:

`mkdir`: create a new directory (ex: mkdir lab1)

`cd`: change directory (ex1: cd lab1, ex2: *cd* ..)

`ls`: list all items in current terminal directory

`<Tab>` key: autocompletes terminal commands, lists out options (if > 1)

VS Code (Common Issues)

MacOS

Install Xcode-tools: `xcode-select –install`

Windows 10 (WSL)

- Install Windows 10 updates
- `sudo apt-get update` » `sudo apt install g++`
- [Ctrl + Shift + P] » Terminal: Select Default Shell » WSL Bash
- Drives are mapped as – [C:] » **/mnt/c/**

VS Code (JupyterHub)

Not recommended unless:

You don’t have a Linux, Mac or Windows 10 machine

Go to: https://coding.csel.io/

Login with your identikey -> Choose CSCI 2270 (Workspace)

Can get overloaded at times -> **No access** (Around Submissions)

VS Code and C++

Hello World: Code

File: hello.cpp

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

Hello World: Compile

g++ <file1.cpp> <file2.cpp> … –<arg_name> <arg_value>

$$\textbf{g++}\ \ \underbrace{\text{hello.cpp}}_{\textbf{Path of file to be compiled}}\ \ \underbrace{\textbf{\textit{-o}}\ \textit{hello}}_{\textbf{Path/Name of }\underline{\textbf{output}}\textbf{ file}}\ \ \underbrace{\textbf{\textit{-std}}\textit{=c++11}}_{\textbf{C++ }\underline{\textbf{standard}}\textbf{ to compile with}}$$

Hello World: Execute

./hello

Write output file name/path in terminal and hit <RETURN>

./a.out

Default: If you don’t use the `-o` argument

Command Line Arguments in C++

```cpp
#include <iostream>

int add (int a, int b)
{ return a + b; }

int main ()
{
  int a = 2, b = 3;
  std :: cout << "a+b=" << add(a, b) << std :: endl ;
  return 0 ;
}
```

What about different values of a, b?

Command Line Arguments in C++

Different values of a, b?

Can change the variable values?

Problem: Need to recompile every time we make a change.

Solution: Passing arguments while executing

Command Line Arguments in C++

```
int main (int argc, char const *argv[])
```

argc: Number of arguments

argv: (Informally) Array that holds the actual arguments

First argument is always the filename that is being executed.

Different from arguments passed to g++.

Command Line Arguments in C++

File: commandLine.cpp

```
#include <iostream>
int main ( int argc, char const *argv[])
{
   std :: cout << "Number of arguments: " ;
   std :: cout << argc << std :: endl ;
   std :: cout << "Program arguments: " << std :: endl ;
   for ( int i = 0 ; i < argc; i++) {
      std :: cout << "Argument #" << i << ": " ;
      std :: cout << argv[i] << std :: endl ;
   }
}
```

Command Line Arguments in C++

```
./commandLine arg1 arg2 arg3
```

4	“commandLine”	“arg1”	“arg2”	“arg3”
argc	**argv[0]**	**argv[1]**	**argv[2]**	**argv[3]**

Can you fix the addition program to accept multiple arguments?

- **Check:** argc == 2
- **Typecast** to integers! [stoi()]

Functions in C++

Functions

Functions:
- Are **complete*** code snippets
- Provide **reusability**
- **Modularize** the code

Recall: add() function we saw previously

Functions: Multiple Source Files

What if we need to use the same function in multiple files?

Copy over the function to each file?

- What happens when we make changes in the function?
- Will need to make changes in each file

Recall: Function Prototype **vs** Function Definition

Recall: Header Files

Functions: Header

File: function.h

```
int add ( int a, int b);
```

Function Prototype

File: funcdef.cpp

```
#include "function.h"
int add ( int a, int b)
{
  return a + b;
}
```

Function Definition

Functions: Program

File: main.cpp

```
#include <iostream>
#include "function.h"

using namespace std ;
int main ()
{
  cout << "2+3=" << add( 2 , 3 ) << endl ;
  return 0 ;
}
```

Functions: Compiling

```
g++ main.cpp funcdef.cpp -o func
```

func: corresponds to the file with the main() function

File I/O in C++

File Operations: Basics

```
#include <fstream>
```

Header File for File I/O

```
ifstream iFile ( "filename" );
```

File Object for Reading

```
ofstream oFile ( "filename" );
```

File Object for Writing

File Operations: Operation Modes

```
ios::app -- Append to the file
ios::ate -- Set the current position to the end
ios::trunc -- Delete everything in the file
```

There exist more such options: What to do if file not found, etc.

```
ofstream ofile ( "test.txt" , ios::app );
```

File Operations: Output/Write

File output example - oFile.cpp

```
#include <fstream>
#include <iostream>

using namespace std ;

int main ()
{
  //Creates instance of ofstream and opens the file
  ofstream oFile ( "filename.txt" );
  oFile << "Inserted this text into filename.txt" ;
  oFile.close(); // Close the file stream
}
```

Allows rewriting **std::cout** as **cout**

File Operations: Input/Read

File input example - iFile.cpp

```
...
int main ()
{
  char str[ 10 ];
  ifstream iFile ( "filename.txt" );

  iFile >> str; //Reads one string from the file
  cout << str << "\n" ; //Outputs the file contents

  cin.get(); // waits for a keypress
  iFile.close();
}
```

Exercise

Exercise

Open **Recitation 1 Writeup** on Moodle

Complete **3a, 3b**

Show your work & Sign the Attendance Sheet

VS Code Setup Issue?

- Use **https://coding.csel.io/**