Do Now Exercise

Welcome to Comp 15, Data Structures, Summer 2019! To prepare you for the lecture today, please do the following exercise.

List name of data structures that you have heard of.

COMP15: Data Structures

Week 1, Summer 2019

Course Webpage

https://www.cs.tufts.edu/comp/15/

Course Staff

Tomoki Shibata

tshibata@cs.tufts.edu

Office Hours:
Monday 5-6pm and Wednesday after class

Location:
Halligan Room (TBA)

Matthew Russell

mrussell@cs.tufts.edu

Office Hours:
Tuesday 5-7pm and Thursday 2-4pm

Location:
Halligan Room 209

* Please check the course webpage for any updates.

Course Textbook

No required textbook.

(Yet, there are useful books online for your references.)

Course Language

C++

Course Language

C++

Course Programming Environment

- Linux
- Terminal
- clang++
- Text Editor

Course Workflow

1 class meeting / week

Course Workflow

3.5 hours / class meeting

Course Workflow

3.5 hours

=> 1-hour lecture + short break

+1-hour lecture + short break

+ lab session

Questions?

Course Workflow

- Lectures
- In-Class Activities
- Labs
- Programming Projects
- Teach Yourself
- Midterm Exam
- Final Exam

Programming Project Planning

Goal: Help yourself to manage your time for your project.

Deliverable: A sheet of paper that indicates:

- Your name and last updated date
- Important milestones along your timeline.

e.g. 5/22: Read the spec.

5/23: Setup the work space for the project.

5/24: Implement XXX.

5/25: Test XXX.

...

(Add **buffers** if you like, in order to deal with unexpected events!)

"If I had an hour to solve a problem, I'd spend 55 minutes thinking about the problem and 5 minutes thinking about solutions."

by Albert Einstein

Recommend to spend enough time to find what you are being asked to achieve in the project

before start writing a program.

Academic Integrity Policy

https://students.tufts.edu/student-affairs/student-code-conduct/academic-integrity-policy

Course Feedback

https://forms.gle/rrWtS57spwAiiguF9

(Anonymous. The code is on Canvas.)

Questions?

What does it mean by problem-solving?

(Introducing Tomoki's and Matt's HCI researches.)

Example 1

Problem:

Typing on a small touchscreen with a software keyboard is challenging because each character key is too tiny to tap with a finger.

A Solution:

https://www.youtube.com/watch?v=mPfktzYsVZI

Example 1

Comp 15 Question:

I need to keep track of the user's finger drag motions to visualize the trajectory later on.

What kind of data structures should I use?

Example 2

Problem:

Communication channels between human and computers are limited.

A Solution:

https://now.tufts.edu/articles/load-your-mind

Example 2

Comp 15 Question:

Every second, I need to analyze brain data measured in the last X seconds while the data is kept being sampled at Y [Hz].

What kind of data structures should I use?

An engineer is to solve a problem
by selecting the best solution
from a set of feasible solutions.

To become a better engineer,

- Acquire knowledges and skills to be able to select a solution.
- Increase the size of the potential solution set.

=> We want to have a lots of experiences!

Through the semester, keep being

A **learner** and **self-reliant**.

Our Grad Teaching Assistant!

Questions?

Introduce yourself!

Do Now Exercise

Welcome to Comp 15, Data Structures, Summer 2019! To prepare you for the lecture today, please do the following exercise.

List name of data structures that you have heard of.

Do Now Exercise

Students' answers:

linked lists, (dynamic) arrays, (binary) trees, maps, graphs, stacks, queues, heaps, vector, dictionary, hash tables, tries

Why do we have so many
Data Structures?

Why do we concern ourselves with **Data Structures**?

Because we do care about

performances of software program.

(This is based on Tomoki's view. You will be able to find various answers on the Internet.)

What are **performances** of program?

Time and **Space** (and more...)

A Sample Exam Question

Given data structures A, B, and C,

(a set of plausible solutions)

I would like to achieve X, Y, and Z.

(challenges to be addressed)

Which data structure should I use and why?

(how to select a solution)

By the end of this semester, try finding an answer to the following question.

What do **Data Structures** mean to you?

Let's take a short break!

Arrays

In-Class Activity

(Note from the activity)

Some keywords we heard in class: array, contiguous, stack diagram,

Class

instance

How about **struct** keyword?

Class:

“A class is the blueprint from which individual objects are created.”

The Java™ Tutorials
https://docs.oracle.com/javase/tutorial/java/concepts/class.html

instance:

“... your bicycle is an instance of the class of objects known as bicycles.”

The Java™ Tutorials
https://docs.oracle.com/javase/tutorial/java/concepts/class.html

Let's **capture** an array of (something) in the real world and **translate** it into a class in the programming world.

Live Coding

Goal of Live Coding:

Learn a procedure of designing a class, not memorize code.

(Notes from the live coding)

```
1//Array.hpp
2#ifndef ARRAY_HPP //(4)
3#define ARRAY_HPP //(4)
4
5class Array{ //(1)
6public: //(5)
7  Array(); //default constructor //(10)
8  Array(int capacity); //non-default (user defined) constructor //(13)
9  Array(const Array& other); //copy constructor //(14)
10  Array& operator=(const Array& other); //assignment operator //(19)
11  ~Array(); //destructor //(11)
12
13  void add(int i); //(6)
14  int at(int index) const; //(6), (18)
15
16  int getCapacity() const; //(15), (16)
17  int getSize() const; //(15), (16)
18
19private: //(5)
20  int* numbers; //(8)
21  int size; //(8)
22  int capacity; //(8)
23}; //(2)
24
25#endif //(4)
```

Please do NOT assume the code is complete.

(1) class keyword
(2) semicolon
(3) prepared for test
(4) header guard
(5) public, private section
(6) added operations to support
(7) prepared how to test the functionality
(8) chose this design. (We had the fixed size design first, but changed to this. Why? We had at least 2 example cases to answer this question.)
(9) implemented
(10) implemented
(11) implemented
(12) delete keyword with []

(Notes from the live coding)

```
1//test.cpp
2#include <cassert> //(24)
3#include "Array.hpp"
4
5int main(){
6  Array a; //(3)
7  a.add(0); //(7)
8  a.add(1); //(7)
9  a.add(2); //(7)
10
11  Array a2 = a;
12
13  Array a3;
14  a3 = a2;
15
16  assert(a3.getSize() == a2.getSize()); //(23)
17  return 0;
18}
```

Please do NOT assume the code is complete.

(13) implemented
(14) implemented
(15) implemented for (14)
(16) added const keyword
(17) deep copy
(18) added const keyword
(19) implemented
(20) dereference
(21) discussed why this is important
(22) needs to release resources
(23) assert()
(24) for assert()

(Notes from the live coding)

```cpp
//Array.cpp
#include "Array.hpp"

Array::Array(){ //(10)
  capacity = 5;
  numbers = new int[capacity];
  size = 0;
}

Array::Array(int capacity){ //(13)
  this->capacity = capacity;
  numbers = new int[capacity];
  size = 0;
}

Array::Array(const Array& other){ //(14)
  capacity = other.getCapacity();
  size = other.getSize();
  numbers = new int[capacity]; //(17)
  for(int i = 0; i < size; i++){ //(17)
    numbers[i] = other.at(i); //(17)
  }
}

Array& Array::operator=(const Array& other){ //(19)
  if(&other != this){ //(21)
    delete [] numbers; //(22)
    capacity = other.getCapacity();
    size = other.getSize();
    numbers = new int[capacity];
    for(int i = 0; i < size; i++){
      numbers[i] = other.at(i);
    }
  }
  return (*this); //(20)
}
```

```cpp

Array::~Array(){ //(11)
  delete [] numbers; //(12)
}

int Array::getCapacity() const{ //(15), (16)
  return capacity;
}

int Array::getSize() const{ //(15), (16)
  return size;
}

void Array::add(int i){ //(9)
  //ToDo: check if there is a slot for the new one
  numbers[size] = i;
  ++size;
}

int Array::at(int index) const{ //(9) (18)
  //ToDo: check if the index is within the boundary
  return numbers[index];
}
```

Please do NOT assume the code is complete.

(Notes from the live coding)

- **Please let the instructor know if you find any errors.**
- **Please do NOT expect that program codes from the live coding session will be posted in the future lectures.**
- **I rather encourage you to type in the program together in class.**

Project 1: Card Deck

Some keywords from today's lecture:

- arrays
- class and instance
- "class", "this", "delete" keyword
- header guards
- public, private section
- default constructor, non-default (or user defined) constructor, copy constructor, assignment operator, destructor
- deep copy, (shallow copy)
- (const-ness)
- testing with assertion()

To the lab!