CSCI 335
Software Design and Analysis III
Lecture 1: Introduction

Professor Anita Raja
Aug 25, 2022

1

Agenda

- Course Logistics
- C++ Introduction

2

2

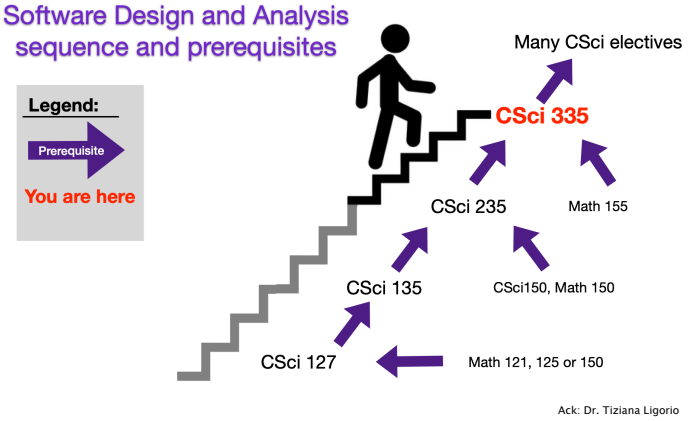
Why 335?

- Sequel to CSCI 235.
- Deepen and enhance your algorithm analysis and programming skills.
- Gain working knowledge of various advanced algorithms.
- Have fun doing this!

3

3

Software Design and Analysis sequence and prerequisites



Ack: Dr. Tiziana Ligorio

4

4

Course Logistics

- Instructor: Dr. Anita Raja
- Office Hours:
  - Instructor: Mondays 11:00-12:00pm, Thursday 12:00-1:00pm or by appointment
  - TAs: information will be posted on blackboard by first week of classes
- Contact: anita.raja@hunter.cuny.edu
- Class Schedule: Mondays and Thursdays 2:30pm-3:45pm
- Room: HW 714
- Course Website: Blackboard
- Textbook: Required: ***Data Structures and Algorithm Analysis in C++*, 4th Edition, Mark Allen Weiss.**

5

5

What is this course about?

- Selection problem
- C++
- Textbook
- Topics covered: Algorithm Analysis, Advanced Trees, Special Priority Queues, Sorting Algorithms, Disjoint Sets, Graph Algorithms, Dynamic Programming, Randomized Algorithms, and Amortized Analysis.

6

6

Grading Rubric and Academic Integrity

Participation	7%
Assignments (5; lowest grade dropped)	48 %
Midterm Exam	20%
Final Exam	25%

- Participation: includes in-class exercises, surveys, blackboard discussion, assignment solution presentation.
- Read syllabus posted on blackboard carefully.
- Read programming rules.pdf – addendum to syllabus.
- Schedule on blackboard.
- Keep up with the reading - lectures will be posted before class.
- Academic Integrity.

7

7

How to succeed in this course?

If you want to succeed in this course, do *all* of the following:

- Do the assigned readings *before* the lecture, *not after* it.
- Make a list of questions before the class.
- Submit all assignments on time.
- Solve a set of problems at the end of each chapter.
- Study for exams.
- Do all assignments yourself! College's Academic integrity Policy will be strictly enforced!

8

8

Communication

- Questions on lecture/assignment/exam:
  - Check blackboard first.
  - If not, post to blackboard (follow blackboard rules).
- If you prefer to email the question
  - Lecture/exam/course logistics email me at anita.raja@hunter.cuny.edu
  - Questions on assignments email and cc me.

  - Acknowledgements:
  - Materials for this course were adapted from materials provided by Mark Weiss' textbook and website, Prof. Stamos and other Hunter Computer Science resources.

9

9

Announcements

- Blackboard Survey
- Tech Prep in Fall Club Hours
- HW1 released 8/29

10

10

C++ review with emphasis on C++11 concepts

- Programming Style is very important
  - Code Readability
  - Code Reuse
  - Code Maintenance
- Good source: Google C++ style guide.

11

11

C++ review with emphasis on C++11 concepts

- Examples
  - Class names *CamelCase* : class PointFinder
  - Function names *CamelCase*: FindClosestPoint(…)
  - Local variables and parameters (lower case, separate words with _):
    int number_of_neighbors;

12

12

C++ review with emphasis on C++11 concepts

- Example function:

```
double AbsDistance(const array<double, 3> &point1, const
array<double, 3> &point2, bool use_two_params_only = false)
{
  const double result2 = fabs(point1[0] – point2[0]) +
fabs(point1[1] – point2[1]);
  return use_two_params_only ? result2 : result2 +
fabs(point1[2] – point2[2]);
}
```

13

13

C++ review with emphasis on C++11 concepts

```
// @point1: a 3D point.
// @point2: a 3D point.
// @use_two_params_only: if true, use only first two coordinates.
// Compute and return the absolute distance between the points.
double AbsDistance(const array<double, 3> &point1,  const
array<double, 3> &point2, bool use_two_params_only = false){
  const double result2 = fabs(point1[0] – point2[0]) +
fabs(point1[1] – point2[1]);
  return use_two_params_only ? result2: result2 +
fabs(point1[2] – point2[2]);
}
```

14

14

C++ review with emphasis on C++11 concepts : Const

- Accessor vs Mutator
- Variables
  - E.g. `const array<double, 3> &point1`
- Member function
  - E..g int read() const
- Parameter passing
  - Call-by-constant reference
  - String randomItem()(const vector<string> & arr); //returns random item in arr
- Const iterator
  - the element which is being pointed to by a const_iterator can't be modified.

15

15

C++ review with emphasis on C++11 concepts : public

```
//Class names / types: CamelCase.
//Public/private class data members (lower case and underscore at end): a_variable_
//Constants as class members: const double kInitialValue = 10.0;
//Functions: CamelCase.
//Place public before private.
    class PointFinder {
      public:
        const double kInitialValue = 10.0;
        PointFinder() {};
        double AbsDistance(const std::array<double, 3> &a_point);
        std::array<double, 3> FindClosestPoint(std::array<double, 3> input_point,
                                               double max_distance_from_point);
      private:
        std::array<double, 3> initial_point_;
    }
//Use full names that describe what the variable is doing. Example:
double minimum_distance_from_start;
// Do not use cryptic and small names: double fg;
Indices in loops can be i, j, k, etc.
```

No public data is allowed.
Only constants and member functions can be public

16

16

C++ review with emphasis on C++11 concepts : Comments

- At the **beginning of each file**, provide a brief description of the contents of the file and the author's name.
- On top of the **class names**, names provide a brief description.
- Provide a comment on top of each **function** (unless functionality is obvious). In the case of class functions, put comment only in the header (do not replicate).
- Inside the **implementations**, only if it is hard to understand from the code.
-

17

17

C++ review with emphasis on C++11 concepts : Comments

- Example of class comments:

```
// A class that searches for the closest point from a given set of 3D points.
// Sample usage:
// PointFinder a_finder;
// a_finder.AddPoint(std::array<double, 3>{1.0, 3.0, 10.0});
// auto closest_point=a_finder.FindClosestPoint(std::array<double, 3>{0,1,0}, 30.0);
class PointFinder { … }
```

18

18

C++ review with emphasis on C++11 concepts: Comments

Example of function comments:

```
// @input_point: A given 3D point.
// @max_distance_from_point: Do not look beyond that distance for a closest  point.
// @return the closest point to input_point.
std::array<double, 3> FindClosestPoint(std::array<double, 3> input_point, double max_distance_from_point);
```

19

19

Summary

- C++ review, C++11 concepts
  - Programming style
- Next class
  - IntCell class – initialization, explicit
  - Lvalues, Rvalues
  - Rvalues
  - std::swap; std::move
  - The Big Five

20

20

For Next Class

- Read Chapter 1
- Complete Blackboard survey by Friday night
- Check linux accounts; if you have lost access to that account, email cstechsp@hunter.cuny.edu
- Will use gradescope and github
- Optional: Go over self-assessment questions

21

21