





















RPI Computer Science > Submitti > Data Structures



-  [Course Home](#)
-  [Gradeables](#)
-  [Notifications](#)
-  [Office Hours Queue](#)
-  [Submini Polls](#)
-  [Course Materials](#)
-  [Discussion Forum](#)

-  **Syllabus**
-  [In-Person / Remote Learning](#)
-  [Calendar](#)
-  [Weekly Office Hours and Lab Schedule](#)
-  [Crash Course in C++](#)
-  [Development Environment](#)
-  [Programming Information](#)
-  [Collaboration Policy & Academic Integrity](#)
-  [Getting Help](#)

-  [Rainbow Grades](#)
-  [My Late Days/Extensions](#)

-  [My Courses](#)
-  [My Profile](#)

Syllabus

Syllabus

[Course Grades](#)

[Submini Lecture Polls](#)

[Homework Policies](#)

Course Description

This course covers fundamental data structures and their use in programming. This includes both class design and features of the C++ programming language. Much of our discussion will be built around the design and use of the C++ standard library (STL). By using the standard library, students will be able to write reasonably sophisticated programs quickly.

Learning Outcomes

Students who have successfully completed this course will be able to:

- Demonstrate strong problem solving skills in constructing complete C++ programs to tackle exercises inspired by real-world problems.
- Analyze the performance of algorithms and data structures.
- Select and use the most appropriate data structure from the C++ standard library (STL) for a particular programming task.
- Design and implement efficient customized data structures.

Prerequisites

We allow students at their own discretion to skip CSCI 1100 Computer Science I and register for CSCI 1200 Data Structures. Here are the concepts that we assume you have

learned from CSI or equivalent coursework or other programming experience.

- **Programming:**

- Arithmetic expressions, if/else statements
- Writing your own functions, including the following:
 - Passing arguments to the function from the calling function
 - Reference variables (call by value vs. call by reference)
 - Returning a value from a function to the calling function
 - Scope and lifetime of variables, local vs. global variables
- One and two dimensional arrays and/or vectors
- `while` and `for` loops, including nested loops
- Reading data from files, and writing data to files
- Some basic understanding and use of classes
 - Creating your own simple classes
 - Calling member functions

- **Problem Solving:** Given a problem you should be able to design an algorithm or algorithms to solve the problem, and implement and debug an efficient solution. You should have written a number of programs of 100 lines or more consisting of several different functions. Some examples:

- Count the number of times each letter appears in a file
- Find the maximum value in an array or vector
- Insert a new element into a sorted array in its correct place
- Find the value closest to the average in a vector
- Find the two closest values in a vector

- **Algorithmic concepts:**

You should have seen and have a basic understanding of the following concepts:

- Recursion
- Run time analysis of algorithms (big O notation)
- Elementary searching and sorting algorithms

Programming Languages: C++ vs. Java vs. Python

The language used in this course is C++ but you do not need to know C++ before taking this class.

The Computer Science I class at RPI is taught in the Python programming language. Transitioning a solid foundation in problem solving, computational thinking, and implementation to a new programming language is an important skill for all computer scientists. Once you are comfortable with your new programming environment (code editor, compiler/interpreter, reference material, etc.) you will soon be busily and productively coding again!

Similarly, many students enter this course having studied Java in their high school AP Computer Science class. If you are a reasonably proficient Java programmer, you should easily adapt to the differences between the C++ and Java.

See also: [Crash Course in C++ Syntax](#)

Warning

This course moves at a rapid pace and will likely be substantially more difficult than your previous programming classes. The homework assignments are challenging and students should start the assignments as soon as they are posted, so there is plenty of time to ask the instructor and TAs questions on the [Discussion Forum](#) and in office hours. *Students should not get behind at any point in the semester.* Students should work practice problems and study examples from lecture. Working with other students and working with tutors and TAs are both encouraged, but students need to be certain they understand the material and can do problems on their own.

See also advice from [Spring 2014 CSCI 1200 Data Structures students](#).

Be sure to click through the other pages linked in the upper right menu.