CS 106L Course Information

Overview
What makes C++ one of the most popular, complex, and expressive programming languages in 2020? Over the course of 9 weeks, we will explore the language and library features of C++, with a particular focus on modern developments that have revitalized interest in C++. By studying the STL and exploring ways to extend it, we hope that you will be able to build similarly powerful libraries for your own projects, research, internships, and beyond.

Lecturers
Avery Wang (averywang@stanford.edu)
Anna Zeng (aszeng@stanford.edu)
Communicate with us through Piazza (private post) rather than email
Office Hours: see Piazza master post.

Prerequisites
Corequisite is CS 106B or equivalent. This class can be taken with or after taking CS 106B. Lectures are aligned with CS 106B schedule, and may build on the most recent CS 106B lectures.

Lectures
Due to the COVID-19 outbreak, Stanford classes will be completely remote. During the lecture time (MW 4:30 - 5:50 PM), we will give online lectures through Zoom. All lectures will be recorded and available to only registered students and approved auditors on Canvas.

Website
The course website is http://cs106l.stanford.edu and you should check this site regularly for updates.

Piazza + Email
We will be using Piazza for this class. Please post any questions you have on Piazza, and we will respond. Just like in CS 106B, do not post assignment code on Piazza publicly. If you have private comments, questions, or concerns, need special accommodations, or just want to talk privately, you can make private Piazza posts instead, which only Avery and Anna will see. Do not send us emails - use public or private Piazza posts.

Github
All lecture code will be hosted on GitHub. Lecture code will be provided as direct cpp and header files, and you can paste them into any IDE or text editor, and run it through Qt Creator or whatever compiler you like.

Office Hours
See Piazza master post for times/details. Held over Zoom.
**Grading**

This is a one unit class, taken Satisfactory/No Credit (S/NC) as mandated by the Stanford Spring quarter grading guidelines. The requirements for passing are to satisfactorily complete the two assignments. All submissions should be uploaded to Paperless. The workload is certainly above average for a typical one-unit class, so keep that in mind as you sign up for the class - though we promise that you will also learn much more than average from a one-unit class!

**Partners**

Assignment 1 is to be done individually. Assignment 2 can be done in groups of two.

**Late Policy**

Each late day is a 24 hour extension, which can be earned by filling out a survey. There are 3 surveys through the quarter. In addition, should any emergencies arise, please email us and we will do our best to accommodate you.

**Honor Code**

All of the Honor Code policies from CS 106B apply to CS 106L. Don’t look at anyone else’s code, and don’t share your code with anyone else. For projects, you may share code within your pair.

**Important Deadlines**

- April 6    Week 1    Intro survey released
- April 13   Week 2    Intro survey due
- April 22   Week 3    Assignment 1 released
- April 29   Week 4    Assignment 1 screenshots due
- May 4      Week 5    Mid-quarter survey released
- May 11     Week 6    Mid-quarter survey due
- May 13     Week 6    Assignment 1 due
- May 18     Week 7    Assignment 2 released
- May 20     Week 7    End-quarter survey released
- May 27     Week 8    End-quarter survey due
- Jun 1      Week 9    Assignment 2 checkpoint due
- Jun 10     Week 10   Assignment 2 due

**Exercises and Assignments**

*Only assignments 1 and 2 are required. All other exercises are optional.*

- Week 1-2: Optional Project 1 - GraphViz (lectures 1-4)
- Week 3: Cracking the Coding Interview in C++ (lectures 5-6)
- Week 4-5: Assignment 1 - Wiki Racer (lectures 1-9)
- Week 6-7: Optional Project 2 - GapBuffer (lecture 8-15)
- Week 8-9: Assignment 2 - STL HashMap (lectures 1-15, focus on 10-15)
- Week 10: Optional Project 3 - K-d Tree
### Tentative Lecture Schedule

<table>
<thead>
<tr>
<th>#</th>
<th>Date</th>
<th>Lecture Topics</th>
<th>Lecturer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>PART 1: How do we use the C++ language and libraries as a client?</strong></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Apr. 6</td>
<td><strong>Introduction</strong>&lt;br&gt;logistics, philosophy, compilation, first program</td>
<td>Anna</td>
</tr>
<tr>
<td>2</td>
<td>Apr. 8</td>
<td><strong>Structures</strong>&lt;br&gt;pair/tuple, struct. binding, multiple return, auto</td>
<td>Avery</td>
</tr>
<tr>
<td>3</td>
<td>Apr. 13</td>
<td><strong>Initialization and References</strong>&lt;br&gt;uniform initialization, returning references</td>
<td>Avery</td>
</tr>
<tr>
<td>4</td>
<td>Apr. 15</td>
<td><strong>Streams</strong>&lt;br&gt;i/o streams, state bits, getline/&gt;&gt;</td>
<td>Anna</td>
</tr>
<tr>
<td>5</td>
<td>Apr. 20</td>
<td><strong>Containers</strong>&lt;br&gt;sequence and associative containers: Stanford to STL</td>
<td>Avery</td>
</tr>
<tr>
<td>6</td>
<td>Apr. 22</td>
<td><strong>Iterators</strong>&lt;br&gt;usage, for-each, iterator categories</td>
<td>Avery</td>
</tr>
<tr>
<td>7</td>
<td>Apr. 27</td>
<td><strong>Templates</strong>&lt;br&gt;template usage, instantiation, concept-lifting</td>
<td>Avery</td>
</tr>
<tr>
<td>8</td>
<td>Apr. 29</td>
<td><strong>Functions</strong>&lt;br&gt;implicit interface and lambda functions</td>
<td>Avery</td>
</tr>
<tr>
<td>9</td>
<td>May 4</td>
<td><strong>Algorithms</strong>&lt;br&gt;algorithms library, stylometry application</td>
<td>Anna</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>PART 2: How do we use the C++ language to design our own libraries?</strong></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>May 6</td>
<td><strong>Template Classes</strong>&lt;br&gt;basis, typename, member types</td>
<td>Avery</td>
</tr>
<tr>
<td>11</td>
<td>May 11</td>
<td><strong>Const Correctness</strong>&lt;br&gt;const references, const_iterator, const members</td>
<td>Avery</td>
</tr>
<tr>
<td>12</td>
<td>May 13</td>
<td><strong>Operator Overloading</strong>&lt;br&gt;friend, member, POLA</td>
<td>Anna</td>
</tr>
<tr>
<td>13</td>
<td>May 18</td>
<td><strong>Special Member Functions</strong>&lt;br&gt;ctor vs. assignment, copy operations, rule of 3/4.5/5/0</td>
<td>Avery</td>
</tr>
<tr>
<td>14</td>
<td>May 20</td>
<td><strong>Move Semantics</strong>&lt;br&gt;value category, move semantics, std::move</td>
<td>Avery</td>
</tr>
<tr>
<td>15</td>
<td>May 27</td>
<td><strong>RAII</strong>&lt;br&gt;exception safety, smart pointers</td>
<td>Anna</td>
</tr>
<tr>
<td>16</td>
<td>Jun. 1</td>
<td><strong>Multithreading</strong>&lt;br&gt;threads, race conditions, mutexes</td>
<td>Anna</td>
</tr>
<tr>
<td>17</td>
<td>Jun. 3</td>
<td><strong>Final Lecture</strong>&lt;br&gt;wrap up previous lectures, C++20 topics!</td>
<td>TBD</td>
</tr>
</tbody>
</table>
Bonus Topics Schedule
Time TBA based on survey, topics will probably change based on interest

Week 1: references and value category
Week 2: stringstream and implementing the Stanford I/O library (eg. getInteger())
Week 3: containers best practices, iterator invalidation, erase-remove, stream iterators
Week 4: variadic templates and SFINAE
Week 5: template metaprogramming
Week 6: lambda and iterator implementation
Week 7: the perfect forwarding problem, maybe CRTP
Week 8: implementing a smart pointer, advanced smart pointer usage
Week 9: custom allocators or C++20 stuff