

## Stanford C++ Options

---

The Computer Science department currently offers several classes that deal with the C++ programming language. This plethora of classes reflects the sheer enormity of the C++ language and each class shows you the language through a difference lens.

This handout clarifies the differences between the C++ options so that you can make an informed decision about which classes to take. Please note that this handout only deals with the *content* of the classes. You're responsible for determining which ones fulfill your particular academic requirements.

### Class Descriptions

Here are the summaries of each class, as given in the Stanford Bulletin:

**CS 106B. Programming Abstractions**—Abstraction and its relation to programming. Software engineering principles of data abstraction and modularity. Object-oriented programming, fundamental data structures (such as stacks, queues, sets) and data-directed design. Recursion and recursive data structures (linked lists, trees). Introduction to time and space complexity analysis. Uses the programming language C++ covering its basic facilities. Prerequisite: 106A or consent of instructor.

**CS 106X. Programming Methodology and Abstractions (Accelerated)**—Intensive; 106A,B in one quarter. Students who complete 106A should enroll in 106B; 106X may be taken after 106A only with consent of instructor. Uses the C++ programming language. How programming concepts are expressed in C++. Abstraction and its relation to programming. Software engineering principles of data abstraction and modularity. Object-oriented programming, fundamental data structures (such as stacks, queues, sets) and data-directed design. Recursion and recursive data structures (linked lists, trees). Introduction to time and space complexity analysis. Prerequisite: substantial programming experience that allows ready understanding of concepts presented in 106A.

**CS 107. Programming Paradigms**—Advanced memory management features of C and C++; the differences between imperative and object-oriented paradigms. The functional paradigm (using LISP) and concurrent programming (using C and C++). Brief survey of other modern languages such as Python, Objective C, and C#.

**CS 107L. Programming Paradigms Laboratory**—Advanced C++ topics beyond the scope of 107. Topics: advanced memory management; placement new; manual destruction; operator overloading; STL template containers; algorithms; iterators; single and multiple inheritance; class hierarchy design; and C++ pitfalls.

**CS 193D. Professional Software Development with C++**—C++ programming techniques and methodologies. Language concepts including object-oriented design, memory management, and the standard library. Modern software development concepts such as design patterns, test-driven development, extreme programming, and XML. Prerequisites: basic C++ or significant experience in C or Java.

**CSNaN. Modern C++ Techniques** -- This class is not listed in the Bulletin. It is a student-taught course in advanced C++ concepts.

## What Will I Learn?

The following chart compares your options side by side.

	<b>CS106B</b> Programming Abstractions	<b>CS106X</b> Programming Methodology & Abstractions	<b>CS107</b> Programming Paradigms	<b>CS107L</b> Programming Paradigms Laboratory	<b>CS193D</b> Professional Software Development with C++	<b>CSNaN</b> Modern C++ Techniques
<b>Language Features</b>						
Introductory C++	Yes	Yes	No	Refresher Only	Refresher Only	No
C++ For Java Programmers	Yes	No	No	No	Yes	No
Data Structures and Algorithms	Yes	Yes	Some	No	No	No
Object Oriented Programming	Yes	Yes	Yes	Yes	Yes	Yes
Advanced Language Features	No	No	No	Much	Some	Yes
Standard Template Library	No	No	Some	Some	Yes	Yes
<b>Related Technologies</b>						
Concurrency	No	No	Yes	No	No	No
XML and SOAP	No	No	No	No	Yes	No
<b>Design and Methodologies</b>						
C++ Coding Patterns	No	No	Yes	Yes	Yes	Yes
Design Patterns	No	No	No	No	Yes	Some
Programming Style	Some	Some	Some	No	Yes	No
Object Oriented Design	No	No	No	Yes	Yes	Yes
<b>Real World</b>						
Real World Testing and Debugging	Some	Some	Some	No	Yes	No
Software Dev Methodologies	No	No	No	No	Yes	No

## How Do I Decide?

With all of these wonderful choices, you may be struggling to find the right C++ class for you. Here's a slightly arcane flowchart you can use as a rule of thumb.

