

# Introduction to Programming II

CS101 - Catholic Polytechnic University - Spring 2026

Dr. Marcus Birkenkrahe

February 17, 2026

## Contents

<b>1 General Course Information</b>	<b>2</b>
<b>2 Objectives</b>	<b>3</b>
<b>3 Target audience</b>	<b>3</b>
<b>4 Student Learning Outcomes</b>	<b>3</b>
<b>5 Course requirements</b>	<b>4</b>
<b>6 Weekly schedule</b>	<b>4</b>
<b>7 Grading system</b>	<b>5</b>
<b>8 How to use AI</b>	<b>6</b>
<b>9 Rubric</b>	<b>6</b>
<b>10 CPU policies</b>	<b>6</b>



## 1 General Course Information

- Course title: Introduction to Programming II (with C++)
- Course number and section: CS 101
- Meeting Times: Friday, 17:00-20:00 hrs Pacific Standard Time
- Meeting place: Microsoft Team room (TBC)
- Professor: Marcus Birkenkrahe, PhD
- Phone: (501) 422-4725
- Office hours: By appointment (email: [mbirkenkrahe@catholicpolytechnic.org](mailto:mbirkenkrahe@catholicpolytechnic.org))
- Books (optional + free + online):
  1. **Think C** by Scheffler/Downey (2019): Starts easy then gets harder fast. [tinyurl.com/think-C-book](http://tinyurl.com/think-C-book) (also available in German!)
  2. **The Rook's Guide to C++** (2013): [tinyurl.com/rooksguide-cpp](http://tinyurl.com/rooksguide-cpp) - with my fixes to the exercises: [tinyurl.com/rooksguide-cpp-notes](http://tinyurl.com/rooksguide-cpp-notes)
  3. **Beej's Guide to C Programming (2024)**: [beej.us/guide/bgc/](http://beej.us/guide/bgc/) - assumes that you already know another programming language.
  4. Zed Shaw's **Learn C the Hard Way** is tough but very applied, no-nonsense, exercises only: [archive.org](http://archive.org).

## 2 Objectives

This course on **CS 101 – Introduction to Programming II** is the second semester of the introductory programming sequence and builds directly on CS 100. While CS 100 introduces programming fundamentals and computational problem-solving techniques using a high-level language, this course shifts the emphasis from **control** to **data**.

Students learn how programs organize, manage, and reason about data using abstraction. Core topics include abstract data types (ADTs), fundamental data structures, encapsulation, and informal reasoning about performance. The course emphasizes conceptual understanding, practical use of library components, and disciplined program design, preparing students for later study in data structures, algorithms, and systems.

## 3 Target audience

This course is intended for students who have completed CS 100 or an equivalent introductory programming course. It is appropriate both for students continuing in computer science or related fields and for students with prior programming experience who are ready to engage with data-centric problem solving and abstraction.

Students with substantial prior programming background may enter this course directly with departmental approval.

## 4 Student Learning Outcomes

Students who complete this course will be able to:

- Apply computational problem-solving techniques to data-oriented problems.
- Explain and use abstract data types (ADTs) to structure programs.
- Use and reason about classic data structures such as arrays, lists, stacks, queues, and associative containers.
- Describe and apply recursion in problem solving.
- Use introductory tree-based structures at a conceptual level.
- Organize programs using encapsulation and basic object-oriented principles.

- Compare alternative solutions using informal notions of complexity and performance trade-offs.
- Describe classic searching and sorting algorithms at a conceptual level.
- Read, adapt, and reuse library components and existing code.
- Enter a formal data structures course with greater programming maturity and conceptual clarity.

## 5 Course requirements

Formal prerequisites: CS 100 – Introduction to Programming, or equivalent experience with departmental approval.

Students are expected to be comfortable with basic programming concepts, including variables, control structures (loops and conditionals), functions or methods, and simple input/output. No prior exposure to formal data structures or algorithm analysis is assumed. Success in this course requires regular practice, careful reasoning about data and abstraction, and incremental program development.

## 6 Weekly schedule

The course meets **once per week on Fridays, 5:00–8:00 PM (Pacific Standard Time)** for **16 weeks**, beginning **January 30**. This schedule reflects actual course delivery, with a narrower and more realistic scope emphasizing foundational data structures, STL usage, and program organization.

Week	Content	Date (Fri)
1	Course overview, expectations, tooling	Jan 30
2	C++ review; motivation for data structures	Feb 6
3	Data structures overview; arrays as a first structure	Feb 13
4	C-style arrays and indexed data	Feb 20
5	STL <code>vector</code> : usage and comparison with arrays	Feb 27
6	<code>vector</code> continued; informal time complexity	Mar 6
7	Pointers and memory basics	Mar 13
8	Pointers in use; arrays, functions, and memory	Mar 20
9	Introduction to classes and encapsulation	Mar 27
	NO CLASS (Good Friday)	Apr 3
10	Constructors, destructors; intro to STL containers	Apr 10
11	STL <code>vector</code> revisited; iteration and idioms	Apr 17
12	STL <code>map</code> : associative containers	Apr 24
13	<code>map</code> continued; keys, values, and use cases	May 1
14	Conceptual overview: lists, stacks, queues	May 8
15	Summary, integration, and outlook toward Data Structures	May 15

## 7 Grading system

You should be able to see your current grade at any time using the Canvas gradebook for the course.

WHEN	DESCRIPTION	IMPACT
Weekly	Programming assignments	25%
Weekly	Participation & Practice	25%
Weekly	Multiple choice tests	25%
TBD	Final exam (optional)	25%

Notes:

- **To pass:** 60% of all available points.
- **Tests:** weekly online quizzes based on classroom lectures and practice.
- **Final exam:** random selection of the known test questions. **Note:** You only have to write the final exam if you want to improve your grade at the end of the course. If the final exam result is below your final grade average up to this point, it will be ignored.

## 8 How to use AI

In this course, AI is treated neither as a shortcut nor as an oracle, but as a collaborator whose usefulness depends on how it is used. You are encouraged to use AI systems to explore ideas, clarify goals, generate drafts, and test workflows, provided you remain responsible for understanding, checking, and explaining the results. AI systems can produce convincing but incorrect output, so verification, reflection, and revision are essential parts of the work. The goal is not to avoid AI, but to learn how to use it thoughtfully, transparently, and critically—so that it supports your reasoning rather than replacing it. Successful use of AI in this course means you can explain what it did, why it worked (or didn't), and how you would improve or correct it.

## 9 Rubric

Component	Weight	Excellent	Good
Participation & Attendance	25%	Consistent, active, prepared	Regular, engaged
Programming Assignments	25%	Correct, clear, well-documented	Mostly correct, minor issues
Tests	25%	Conceptual mastery	Good understanding
Final Exam	25%	Integrative, precise reasoning	Mostly correct

## 10 CPU policies

For policy matters and procedures, especially concerning attendance, grading tables, withdrawals etc. please consult the Catholic Polytechnic University's CATALOG OF COURSES (2025-2026).