

CS106 - Introduction to Computer Programming II

Last Updated - 01/25/02

Course Manager - Matthew Bauer, Senior Lecturer

2 credit hours; required for CS & CPE (or CS200); 75 min. lecture & 75 min. lab each week

Current Catalog Description - Continuation of CS 105. Introduces more advanced elements of C++ programming — including pointers, recursion, classes, and object-oriented programming techniques.

Prerequisite: CS 105 or consent of instructor. (2-1-2)

Textbook

- Roberge/Bauer/Smith, *Engaged Learning for Programming in C++: A Laboratory Course*, Jones and Bartlett Publishers, 2nd Edition, ©2001, ISBN-0763714232
- Deitel/Deitel, *C++ How To Program*, Prentice-Hall, Inc., 3rd Edition, ©2001, ISBN-01308957171

References - other textbooks or materials

- none

Course Goals - Students should be able to:

- Analyze and explain the behavior of simple programs involving the following fundamental programming constructs: assignment, I/O (including file I/O), selection, iteration, functions, pointers
- Write a program that uses each of the following fundamental programming constructs: assignment, I/O (including file I/O), selection, iteration, functions, pointers
- Break a problem into logical pieces that can be solved (programmed) independently.
- Develop, and analyze, algorithms for solving simple problems.
- Use a suitable programming language, and development environment, to implement, test, and debug algorithms for solving simple problems.
- Write programs that use each of the following data structures (and describe how they are represented in memory): strings, arrays, structures, and STL class libraries including strings and vectors
- Explain the basics of the concept of recursion.
- Write, test, and debug simple recursive functions and procedures.
- Explain and apply object-oriented design and testing involving the following concepts: data abstraction, encapsulation, information hiding, sub-classing, inheritance, templates
- Use a development environment to design, code, test, and debug simple programs, including multi-file source projects, in an object-oriented programming language.
- Solve problems by creating and using sequential search, binary search, and quadratic sorting algorithms (selection, insertion)
- Determine the time complexity of simple algorithms.

Prerequisites by Topic

- Experience writing programs that uses each of the following fundamental programming constructs:
 - assignment, I/O (including file I/O), selection, iteration, functions, arrays

Major Topics Covered in Course

1. Character Arrays, Strings, Structures	6 hours
2. Searching, Sorting	6 hours
3. Classes, Constructors, Destructors, Copying, Scope, Development Environment	7 hours
4. Using Objects, Object Oriented Design	6 hours
5. Project Design	6 hours
6. Operator Overloading, Inheritance	4 hours
7. Templates	3 hours
8. Recursion, Pointers	3 hours
Quiz #1, Midterm Exam, Quiz #2	4 hours

Laboratory projects (specify number of weeks on each)

- 8 labs (1-2 labs each week, each lab contains multiple programming assignments, some with shells, and analysis work)
 - Arrays and Using the Vector Class; Strings and Using the String Class; Structures; Searching and Sorting; Creating Classes; Using Objects; Functions II; Advanced Class Concepts
- 1 object-oriented programming project (individual or 2 person teams, 3 weeks, requiring at least 3 interacting classes, design and implementation)

Estimate CSAB Category Content in Credit Hours

	CORE	ADVANCED		CORE	ADVANCED
Data Structures	.3		Computer Organization and Architecture	0	
Algorithms	.3		Concepts of Programming Languages	1	
Software Design	.3				

Oral and Written Communications - Every student is required to submit at least 1 written reports (not including exams, tests, quizzes, or commented programs) of typically 5 pages and to make 0 oral presentations of typically 0 minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

- Object-oriented design document required for object-oriented programming project

Social and Ethical Issues - Please list the topics that address the social and ethical implications of computing covered in all course sections. Estimate the class time spent on each topic. In what ways are the students in this course graded on their understanding of these topics (e.g., test questions, essays, oral presentations, and so forth)?

- Legitimate Code Re-Use, 1 hour, object-oriented programming project

Theoretical Foundations - Please list the types of theoretical material covered, and estimate the time devoted to such coverage in contact (lecture and lab) hours.

- Searching, 3 hours
- Sorting, 3 hours
- Recursion, 2 hours
- Object Oriented (data abstraction, sub-classing, inheritance, templates), 6 hours

Problem Analysis - Please describe the problem analysis experiences common to all course sections.

- Most labs include problem solving with pseudo-code component, or debugging code segments, or determine program output
- Searching & Sorting topics include discussion of big-O analysis

Solution Design - Please describe the design experiences common to all course sections.

- 1 object-oriented project (individual or 2 person teams, 3 weeks, requiring at least 3 interacting classes, design and implementation)

Other Course Information

- Additional Suggested Course Assignments
 - 2 programming quizzes (50 minutes each in lab)
 - 1 midterm exam (100 minutes, around 70% programming)
 - 1 final exam (120 minutes, around 70% programming)
- Planned Course Enhancements

- Change to "objects-first" approach and full coverage of Object-Oriented concepts (including composition, inheritance, and polymorphism). (Fall 2002)
- Potential classification as a "C" course (communications intensive) (Fall 2002)
- Stress problem solving, algorithms, recursion, and design more than programming language. (Fall 2002)
- Change "Prerequisite: CS 105 or consent of instructor." to "Prerequisite: Grade of 'C' or better in CS 105 or consent of instructor."
- Change programming language to Java. (Fall 2003)