CS 10C Course Outline as of Fall 2024

CATALOG INFORMATION

Dept and Nbr: CS 10C Title: PROGRAMMING CONCEPTS 2
Full Title: Programming Concepts and Methodologies 2
Last Reviewed: 3/27/2023

Units		**Course Hours per Week**		**Nbr of Weeks**	**Course Hours Total**	
Maximum	4.00	Lecture Scheduled	3.00	17.5	Lecture Scheduled	52.50
Minimum	4.00	Lab Scheduled	3.00	3	Lab Scheduled	52.50
		Contact DHR	0		Contact DHR	0
		Contact Total	6.00		Contact Total	105.00
		Non-contact DHR	0		Non-contact DHR	0

Total Out of Class Hours: 105.00 Total Student Learning Hours: 210.00

Title 5 Category:	AA Degree Applicable
Grading:	Grade or P/NP
Repeatability:	00 - Two Repeats if Grade was D, F, NC, or NP
Also Listed As:	
Formerly:	CS 11

Catalog Description:
Students in this course will apply knowledge of software engineering techniques to the design and development of large programs, including data abstraction, structures, and associated algorithms.

Prerequisites/Corequisites:
Course Completion of CS 10B

Recommended Preparation:
Eligibility for ENGL 1A or equivalent

Limits on Enrollment:

Schedule of Classes Information:
Description: Students in this course will apply knowledge of software engineering techniques to the design and development of large programs, including data abstraction, structures, and associated algorithms. (Grade or P/NP)
Prerequisites/Corequisites: Course Completion of CS 10B
Recommended: Eligibility for ENGL 1A or equivalent
Limits on Enrollment:

Transfer Credit: CSU;UC.
Repeatability: Two Repeats if Grade was D, F, NC, or NP

ARTICULATION, MAJOR, and CERTIFICATION INFORMATION:

AS Degree:	**Area**			Effective:	Inactive:
CSU GE:	**Transfer Area**			Effective:	Inactive:
IGETC:	**Transfer Area**			Effective:	Inactive:
CSU Transfer:	Transferable	Effective:	Spring 1991	Inactive:	
UC Transfer:	Transferable	Effective:	Spring 1991	Inactive:	

CID:

CID Descriptor:	COMP 132	Programming Concepts and Methodology II
SRJC Equivalent Course(s):		CS10C

Certificate/Major Applicable:
Major Applicable Course

COURSE CONTENT

Student Learning Outcomes:
At the conclusion of this course, the student should be able to:
1. Write programs in C++ that use arrays, linked lists, stacks, queues, hash tables, and recursion.
2. Explain how object-oriented programming uses abstraction to increase reusability of software.
3. Summarize the differences between programming paradigms.

Objectives:
At the conclusion of this course, the student should be able to:
1. Write programs that use each of the following data structures: arrays, records, strings, linked lists, stacks, queues, and hash tables.
2. Implement, test, and debug simple recursive functions and procedures.
3. Evaluate tradeoffs in lifetime management (reference counting vs. garbage collection).
4. Explain how abstraction mechanisms support the creation of reusable software components.
5. Design, implement, test, and debug simple programs in an object-oriented programming language.
6. Compare and contrast object-oriented analysis and design with structured analysis and design.

Topics and Scope:

I. Programming Fundamentals
- A. Primitive types
- B. Arrays
- C. Records
- D. Strings and string processing
- E. Data representation in memory
- F. Static, stack, and heap allocation
- G. Runtime storage management
- H. Pointers and references

- I. Linked structures
- J. Implementation strategies for stacks, queues, and hash tables
- K. Implementation strategies for trees
- L. Strategies for choosing the right data structure

II. Recursion
- A. The concept of recursion
- B. Recursive mathematical functions
- C. Simple recursive procedures
- D. Divide-and-conquer strategies
- E. Recursive backtracking
- F. Implementation of recursion

III. Declarations and Types
- A. The conception of types as a set of values together with a set of operations
- B. Declaration models (binding, visibility, scope, and lifetime)
- C. Overview of type-checking
- D. Garbage collection

IV. Abstraction Mechanisms
- A. Procedures, functions, and iterators as abstraction mechanisms
- B. Parameterization mechanisms (reference vs. value)
- C. Activation records and storage management
- D. Type parameters and parameterized types - templates or generics
- E. Modules in programming languages

V. Object-Oriented Programming
- A. Object-oriented design
- B. Encapsulation and information-hiding
- C. Separation of behavior and implementation
- D. Classes and subclasses
- E. Inheritance (overriding, dynamic dispatch)
- F. Polymorphism (subtype polymorphism vs. inheritance)
- G. Class hierarchies
- H. Collection classes and iteration protocols
- I. Internal representations of objects and method tables

VI. Software Design
- A. Fundamental design concepts and principles
- B. Design strategy

All topics are covered in both the lecture and lab parts of the course.

Assignment:

Lecture-Related Assignments:
1. Reading (approximately 30 pages/week)
2. Examinations including final exam (2-8)

Lab-Related Assignments:
1. Programming assignments, with written program documentation, using the C++ programming language (10-15)
2. Participation and attendance (optional)

Methods of Evaluation/Basis of Grade:

Writing: Assessment tools that demonstrate writing skills and/or require students to select, organize and explain ideas in writing.

Written program documentation	Writing 10 - 20%

Problem Solving: Assessment tools, other than exams, that demonstrate competence in computational or non-computational problem solving skills.

Programming assignments	Problem solving 20 - 60%

Skill Demonstrations: All skill-based and physical demonstrations used for assessment purposes including skill performance exams.

None	Skill Demonstrations 0 - 0%

Exams: All forms of formal testing, other than skill performance exams.

Examinations including final exam	Exams 20 - 60%

Other: Includes any assessment tools that do not logically fit into the above categories.

Participation and attendance	Other Category 0 - 10%

Representative Textbooks and Materials:
Data Abstraction and Problem Solving with C++: Walls and Mirrors. 7th ed. Carrano, Frank M. and Henry, Timothy M. Pearson. 2016 (classic).