COURSE SYLLABUS

LAST REVIEW	Spring 2021
COURSE TITLE	Data Structures C++
COURSE NUMBER	CIST-0240
DIVISION	Career and Technical Education
DEPARTMENT	CIST
CIP CODE	11.0901, 47.0104
CREDIT HOURS	4
CONTACT HOURS/WEEK	Class: 3 Lab: 2
PREREQUISITES	CIST-0180 Programming Algorithms
COREQUISITES	None

COURSE DESCRIPTION

The course starts with introducing the basics of C++ programming and the concepts of object-oriented programming using C++. It will cover classes, inheritance, templates and other OOP related Topics. After that, the course will cover the major data structures such as linked lists, stacks, queues, trees, binary search trees and heaps. Data Structures associated operations are also included: searching, sorting, hashing algorithms. During this course, the fundamental algorithm analysis will be introduced to teach students how to investigate the algorithms efficiency with respect to resources.

PROGRAM ALIGNMENT

This course is part of a program aligned through the Kansas Board of Regents and Technical Education Authority. For more information, please visit: https://kansasregents.org/workforce_development/program-alignment

PROGRAM LEARNING OUTCOMES

1. Develop Applications: Develop a list processing software application.

INSTITUTIONAL LEARNING OUTCOMES

☐ Communication
☒ Computation and Financial Literacy
☐ Critical Reasoning
☒ Technology and Information Literacy
☐ Community and Civic Responsibility
☐ Personal and Interpersonal Skills

TEXTBOOKS

http://kckccbookstore.com/

METHOD OF INSTRUCTION

A variety of instructional methods may be used depending on content area. These include but are not limited to lecture, multimedia, cooperative/collaborative learning, labs and demonstrations, projects and presentations, speeches, debates, panels, conferencing, performances, and learning experiences outside the classroom. Methodology will be selected to best meet student needs.

COURSE OUTLINE

I. C++ Fundamentals
   A. Basic C++ Programming Elements
   B. Expressions
   C. Control Flow
   D. Functions
   E. Pointers and references
   F. Structures
   G. Dynamic memory
II. Object-Oriented Design
   A. Classes
   B. Composition, Inheritance and Polymorphism
   C. Templates
   D. Friend classes and method
   E. Recursion
III. Linear Data Types
   A. Arrays
   B. Vectors
   C. Linked Lists
   D. Stacks
   E. Queues
IV. Trees
   A. General Trees
   B. Binary Trees
   C. Tree Traversal Algorithms
V. Heaps and Priority Queues
   A. Priority Queue Abstract Data Type.
   B. Implementing a Priority Queue with a List.
   C. Heaps
VI. Hash Tables, Maps and Dictionaries
   A. Maps
   B. Hash Tables
   C. Ordered Maps
   D. Dictionaries.

VII. Search Trees
   A. Binary Search Trees
   B. AVL Trees
   C. Splay Trees
   D. (2, 4) Trees
VIII. Sorting Algorithms
   A. Sorting with selection, bubble and insertion sort
   B. Sort data with merge sort, quick sort, radix sort and heap sort

COURSE LEARNING OUTCOMES AND COMPETENCIES

Upon completion of the course, the student will:

A. Write advanced object-oriented code in C++ in the context of data structures.
   1. Implement inheritance.
   2. Implement polymorphism, virtual methods and late binding.
   3. Implement class and function templates.
   4. Write code with classes from the standard template library.
   5. Implement class composition.
   6. Create user-defined exception objects.
   7. Implement friend classes and functions.
   8. Implement overloaded operators.

B. Develop programs that integrate advanced programming topics.
   9. Manage indexing, pointers and multiple levels of indirect addressing.
   10. Manage dynamic memory.
   11. Write recursive programs.
   12. Organize projects into multiple files.
   13. Handle exceptions.
   14. Pass function pointers.
   15. Implement backtracking.
   16. Examine and create recursive grammars and languages.
   17. Implement prefix and postfix expressions.

C. Analyze and create code using fundamental data structures.
   18. Analyze code implementing linked lists, dummy head nodes, circular linked lists and doubly linked lists.
   19. Analyze code implementing stacks.
   20. Analyze code implementing queues.
   21. Analyze code implementing trees and associated traversals.
   22. Analyze code implementing dictionaries.
   23. Analyze code implementing priority queues.
   24. Analyze code implementing heaps.
   25. Analyze code implementing hash tables.
   26. Create applications utilizing fundamental data structures.

D. Examine advanced algorithms and techniques.
   27. Review data sorting with selection, bubble and insertion sort.
   28. Sort data with merge sort, quick sort, radix sort and heap sort.
   29. Analyze algorithmic efficiency.

E. Write code according to commonly accepted programming standards.
   30. Create descriptive identifiers according to language naming conventions.
   31. Write structured and readable code.
   32. Create documentation.

ASSESSMENT OF COURSE LEARNING OUTCOMES AND COMPETENCIES

Student progress is evaluated through both formative and summative assessment methods. Specific details may be found in the instructor's course information document.

COLLEGE POLICIES AND PROCEDURES

Student Handbook
https://www.kckcc.edu/files/docs/student-resources/student-handbook-and-code-of-conduct.pdf

College Catalog
https://www.kckcc.edu/academics/catalog/index.html

College Policies and Statements
https://www.kckcc.edu/about/policies-statements/index.html

Accessibility and Accommodations
https://www.kckcc.edu/academics/resources/student-accessibility-support-services/index.html.