| | |
|---|---|
| **Course Number and Name:** | **CS 04225: Principles of Data Structures** |
| **Credits and contact hours:** | 3 credits/contact hours |
| **Course Coordinator's name:** | Jacob Levy |
| **Text book, title, author, and year:** | C++: Early Objects, 10th Edition. Tony Gaddis, Judy Walters, & Godfrey Muganda |

## Course Information

**Catalog Description:** The course features programs of realistic complexity. The programs utilize data structures (strings, lists, graphs, stacks) and algorithms (searching, sorting, etc.) for manipulating these data structures. The course emphasizes interactive design and includes the use of microcomputer systems and direct access data files.

**Pre-requisites:** Undergraduate level **CS 04103 Minimum Grade of C-** or
Undergraduate level **CS 04113 Minimum Grade of C-**

## Course Goals

1. **Learn the fundamentals of Data Structures and how they are applied in programming solutions**

   *By the end of this course, students will have gained an understanding of a variety of commonly used Data Structures, how they are implemented, how to work with them, and how they are utilized in algorithmic solutions. They will also learn when to use the different structures.*

2. **Design, Analyze, and Implement Efficient Algorithms in C++**

   *By the end of this course, students will have gained the ability to effectively analyze general algorithms in both logical (processing) and physical (memory) complexity. They will also be able to develop and implement efficient algorithms in C++. Students will also gain understanding of more advanced C++ concepts such as pointers, how to use the Standard Template Library, and recursive problem solving.*

3. **Learn the fundamentals of Searching and Sorting**

   *By the end of this course, students will have been exposed to a variety of searching and sorting algorithms. Students will analyze and compare these algorithms, learn when to apply the different algorithms, and why different algorithms are necessary.*

## List of Topics to be covered

Items in **Bold** are required
Items in *Italics* are suggested topics of discussion

***While this outline may be considered a general guideline for the course, topics may be covered in any order, at the Instructor's discretion/preference.***

*0. Review (Strongly Recommended)*
*Loops*
*Control Structures*
*Arrays*
*Functions/Function Calls*
*Classes*

1. **Problem Analysis (Chapter 9)**
    - a. **Problem Specifications**
    - b. **Design**
    - c. **Algorithmic Design & Analysis**
        - i. *Big O Notation*
        - ii. *Asymptotic Complexity Analysis*
        - iii. *Space Complexity (Memory)*
    - d. **Simple Array Searching and Sorting**
        - i. *Complexity Analysis*
            1. *Linear Search vs. Binary Search*
            2. *Selection Sort vs. Bubble Sort*
        - ii. *Hash Tables*
2. **STL (Standard Template Library)**
    - a. **What is it and why we like it**
        - i. *Reusable Templates*
3. **Pointers (Chapter 10)**
    - a. **Pointers vs Variables**
        - i. *Address vs Value*
    - b. **Dynamic Variables**
    - c. **Dynamic Arrays**
    - d. **Pointer Arguments to Functions**
    - e. **Function Pointers**
4. **Basic Data Structures**
    - a. **Abstract Data Types**
    - b. **Unordered Container Classes**
        - i. Bag
        - ii. List
            1. *ArrayList*
            2. *Linked List*
            3. *Doubly Linked List*

- **c. Ordered Containers**
    - **i. Stack**
        - *1. ArrayStack*
        - *2. ListStack*
        - *3. Practical Applications: Depth First Search*
    - **ii. Queue**
        - *1. Array Queue/Circular Queue*
        - *2. List Queue*
        - *3. Double-Ended Queue*
        - *4. Practical Applications : Breadth First Search*
- **d. Intro to Trees (Chapter 20)**
    - **i. Binary Trees**
        - *1. Array Implementation*
        - *2. Node Class Implementation*

**5. Recursion (Chapter 14)**

- **a. Recursion vs Iteration**
- **b. Recursive Functions**
    - i. Base Case vs Typical/Non-base Case
- **c. Solving Problems Recursively**
    - i. Recursive Selection Sort
        - *1. Vs Iterative Selection Sort*
        - *2. Complexity Analysis*
    - ii. Recursive Binary Search
        - *1. vs Iterative Binary Search*
        - *2. Complexity Analysis*
- **d. Recursion & Trees (Chapter 20)**
    - i. Recursive Depth First Search
    - ii. Tree Traversal
        - 1. Pre-order
        - 2. In-order
        - 3. Post-Order
- **e. Algorithm: QuickSort**
    - i. Complexity Analysis
    - *ii. Vs other sorting algorithms*

*6. Advanced Data Structures (Reach Goal)*

- *a. More Sophisticated Trees*
    - *i. B-Trees*
    - *ii. Red/Black Trees*
- *b. Heap*
- *c. Priority Queue*
- *d. Advanced Searching & Sorting*
    - *i. MergeSort*
    - *ii. HeapSort*
    - *iii. Searching HashTables*