Yarmouk University
Hijjawi Faculty for Engineering Technology
Department of Computer Engineering

CpE 150: Introduction to Programming
Summer 2017

http://elearning.yu.edu.jo

Section	Time and Location	Instructor(s)	Email	Teaching Assistants
1	SMTWTh 08 : 00 – 09 : 00	Ahmed Tamrawi	atamrawi@yu.edu.jo	TBD
2	SMTWTh 09 : 15 – 10 : 15	Yazan M. Alissa	alissay@yu.edu.jo	
3	SMTWTh 11 : 45 – 12 : 45	Ahmed Tamrawi	atamrawi@yu.edu.jo	
4	SMTWTh 10 : 30 – 11 : 30	Ola A. Taani	olataani@yu.edu.jo	
5	SMTWTh 13 : 00 – 14 : 00	Osameh Al-Kofahi	osameh@yu.edu.jo	

Course Description: Computer Engineering 150 is a four credit course designed to introduce programming techniques and problem solving methods to engineering students using the C++ programming language. Emphasis is placed on the fundamentals of computer programming instead of the peculiarities of C++. Therefore, the knowledge that you acquire in this class should be easily generalizable to other programming languages. Students learn how to write programs in an object-oriented high-level programming language. Topics covered include fundamentals of algorithms, flowcharts, problem solving, programming concepts, classes and methods, control structures, arrays, and strings. Weekly laboratory experiments will provide hands-on experience on the topics covered in this course.

Course Objectives: The following are the main course objectives sought to be achieved:

1. *engage* students in problem solving activities using programming as the tool; a process known as computational thinking.
2. *understand* the fundamentals of problem solving, debugging, and the process of moving from a problem statement to a computational formulation of a method for solving the problem
3. *work* to identify a problem and create a solution using computer programs.
4. *learn* a language for expressing computations

Course Learning Outcomes: Upon successful completion of the course, the students will be able to:

1. *write* programs to solve problems, including strategies for structuring code, dividing problems up into pieces that can be solved independently, then integrating the pieces into a whole to solve a large problem.
2. *use* data structures such as arrays and lists as a framework for solving a problem and understanding when to select the different types of data structures.
3. *debug* a program to identify logical, synatx and runtime errors.
4. *read* and *comprehend* written programs to answer questions such as: What does the program do? and How does the program compute X?

Main Textbook: C++ How to Program (4th Edition) by Harvey M. Deitel, Paul J. Deitel
Other Reference(s):

- A First Book of C++ (Introduction to Programming) 4th Edition by Gary J. Bronson
- Big C++, 2nd Edition by Cay Horstmann, Timothy Budd
- C++ Programming: From Problem Analysis to Program Design 7th Edition by D. S. Malik
- Starting Out with C++ from Control Structures to Objects (8th Edition) 8th Edition by Tony Gaddis

Lecture and Lab Attendance: It is compulsory for students to attend at least 85% of the whole time allocated to the course. More than four (4) unjustified absences from the lectures OR more than two (2) unjustified absences from the lab will make you ineligible to attend the final exam. Lab attendance will be taken in the last 15 minutes of the Lab time.

Theoretical and Lab Exams: There will be three theoretical paper-based exams during the semester and two practical "Lab" exams. Exams are closed books and notes. Students who did not attend the first/second exam due to exceptional situations (e.g., medical emergency) will be given a makeup exam.

Lab Tasks: Assigned labs are **only** to be completed during lab time and will be graded according to the following guidelines:

- Program Compilation: 30%
- Program Correctness: 40%
- Code readability/adherence to programming guidelines: 30%

Any program that does not attempt to solve the assigned problem will be awarded a grade of zero (0). Any program that does not work on the campus Windows computers will not be graded. Late assignments will not be graded.

Lab Quizzes: There will be a lab quiz every two lab sessions assessing things learnt in the last two labs.

Seeking Help in the Lab Feel free to ask questions to the friendly TAs. However, you will be expected to put in a fair amount of time struggling on your own as well. We want to encourage development and debugging skills, so try not to get frustrated when we won't tell you exactly how to fix something or what to do next. As long as you make steady progress during the lab, the TAs will try to help you stay on track. Also, please do not email source code to your TA. If you cannot fix something during the normal lab hours, arrange a time with your TA to review your code.

Plagiarism: Student caught cheating will be awarded a 0 in the exam and will be subject to university disciplinary actions.

Disability: Students with disability need to approach the course instructor so that commodities will be put in place to assist them.

Lecture and Lab Grading Policy: The grading breakup will be tentaively as follows:

Bi-Weekly Labs	**35%**
Lab Tasks (due in lab)	7%
Quizzes (every two labs)	10.5%
Midterm Exam	7%
Final Exam	10.5%
Exam #1	**15%**
Program Comprehension and Debugging	7.5%
Coding Skills	7.5%
Exam #2	**15%**
Program Comprehension and Debugging	7.5%
Coding Skills	7.5%
Final Exam	**35%**
Program Comprehension and Debugging	17.5%
Coding Skills	17.5%
Professionalism Penalty	up to **−10%**

Excessive missed classes, rude behavior toward instructor or classmates, unauthorized homework assistance, etc can be held against a student when final grades are calculated.

Tentative Course Outline:

Welcome and Course Overview .... ≈ 1 day

Chapter 1: Introduction to Computers and C++ Programming .... ≈ 4 days

- 1.1 Introduction
- 1.2 What is a Computer?
- 1.6 Machine Languages, Assembly Languages, and High-Level Languages
- 1.14 Basics of a Typical C++ Environment
- 1.21 A Simple Program: Printing a Line of Text
- 1.22 Another Simple Program: Adding Two Integers
- 1.23 Memory Concepts
- 1.24 Arithmetic

Chapter 2: Control Structures .... ≈ 5 days

- 2.1 Introduction
- 2.2 Algorithms
- 2.3 Pseudocode
- 2.4 Control Structures
- 2.5 if Selection Structure
- 2.6 if/else Selection Structure
- 2.7 while Repetition Structure
- 2.8 Counter-Controlled Repetition
- 2.9 Sentinel-Controlled Repetition
- 2.10 Nested Control Structures
- 2.11 Assignment Operators
- 2.12 Increment and Decrement Operators
- 2.13 Essentials of Counter-Controlled Repetition
- 2.14 for Repetition Structure
- 2.15 Examples Using the for Structure
- 2.16 switch Multiple-Selection Structure
- 2.17 do/while Repetition Structure
- 2.18 break and continue Statements
- 2.19 Logical Operators
- 2.20 Confusing Equality (==) and Assignment (=) Operators
- 2.21 Structured-Programming Summary

Chapter 3: Functions .... ≈ 7 days

- 3.1 Introduction
- 3.2 Program Components in C++
- 3.3 Math Library Functions
- 18.9 Character-Handling Library
- 3.4 Functions
- 3.5 Function Definitions
- 3.6 Function Prototypes
- 3.7 Header Files
- 3.8 Random Number Generation
- 3.9 Example: A Game of Chance and Introducing enum
- 3.10 Storage Classes
- 3.11 Scope Rules
- 3.12 Recursion
- 3.13 Example Using Recursion: The Fibonacci Series
- 3.14 Recursion vs. Iteration
- 3.15 Functions with Empty Parameter Lists
- 3.16 Inline Functions
- 3.17 References and Reference Parameters
- 3.18 Default Arguments
- 3.19 Unary Scope Resolution Operator
- 3.20 Function Overloading
- 3.21 Function Templates

Chapter 4: Arrays .... ≈ 7 days

- 4.1 Introduction
- 4.2 Arrays
- 4.3 Declaring Arrays
- 4.4 Examples Using Arrays

- 4.5 Passing Arrays to Functions
- 4.6 Sorting Arrays
- 4.7 Case Study: Computing Mean, Median and Mode Using Arrays
- 4.8 Searching Arrays: Linear Search and Binary Search
- 4.9 Multiple-Subscripted Arrays

Chapter 5: Pointers and Strings .... $\approx$ 8 days

- 5.1 Introduction
- 5.2 Pointer Variable Declarations and Initialization
- 5.3 Pointer Operators
- 5.4 Calling Functions by Reference
- 5.5 Using const with Pointers
- 5.6 Bubble Sort Using Pass-by-Reference
- 5.7 Pointer Expressions and Pointer Arithmetic
- 5.8 Relationship Between Pointers and Arrays
- 5.9 Arrays of Pointers
- 5.10 Case Study: Card Shuffling and Dealing Simulation
- 5.11 Function Pointers
- 5.12 Introduction to Character and String Processing
- 5.12.1 Fundamentals of Characters and Strings
- 5.12.2 String Manipulation Functions of the String-Handling Library
- 5.X Dynamic Memory Allocation

Chapter 6: Classes and Data Abstraction .... $\approx$ 3 days

- 6.1 Introduction
- 6.2 Structure Definitions
- 6.3 Accessing Structure Members
- 6.4 Implementing a User-Defined Type Time with a struct
- 6.5 Implementing a Time Abstract Data Type with a class
- 6.6 Class Scope and Accessing Class Members
- 6.7 Separating Interface from Implementation
- 6.8 Controlling Access to Members

Chapter 14: File Processing .... $\approx$ 3 days

- 14.1 Introduction
- 14.2 The Data Hierarchy
- 14.3 Files and Streams
- 14.4 Creating a Sequential-Access File
- 14.5 Reading Data from a Sequential-Access File

Chapter 18: Bits, Characters, Strings and Structures .... $\approx$ 2 days

- 18.7 Bitwise Operators