Course Outline for Computer Science 14
INTRODUCTION TO STRUCTURED PROGRAMMING IN C++

Catalog Description:

14 – Introduction to Structured Programming In C++ 4 Units

Introduction to structured programming and problem solving using the C++ language. Problem solving techniques, algorithm design, testing and debugging techniques, and documentation standards. C++ syntax: elementary operators, data types, control structures, user-defined and library functions, basic input/output, sequential files, arrays and structs. Appropriate for students with little or no programming experience, but comfortable using computers with modern GUI operating systems. Prerequisite: Mathematics 55, 55B, 55L, 54 or 54L (Completed with a grade of "C" or higher) or an appropriate skill level demonstrated through the Mathematics Assessment process, or Computer Science 7 (Completed with a grade of "C" or higher). 3 hours lecture, 3 hours laboratory.
[Typical contact hours: lecture 52.5, laboratory 52.5]

Prerequisite Skills:

Before entering the course the student should be able to:

1. perform basic operations on complex numbers;
2. solve quadratic equations by factoring, completing the square, and quadratic formula;
3. find complex roots of a quadratic equation;
4. sketch the graphs of functions and relations:
   a. algebraic, including polynomial and rational
   b. logarithmic
   c. exponential
   d. circles;
5. find and sketch inverse functions;
6. perform function composition;
7. solve exponential and logarithmic equations;
8. apply the concepts of logarithmic and exponential functions;
9. solve systems of linear equations in three unknowns using elimination and substitution;
10. apply the properties of and perform operations with radicals;
11. apply the properties of and perform operations with rational exponents;
12. solve equations and inequalities involving absolute values;
13. solve equations involving radicals;
14. graph linear inequalities in two variables;
15. find the distance between two points;
16. find the midpoint of a line segment;
17. define and/or illustrate:
   a. segment
   b. ray
   c. angle
   d. distance between points on a line
   e. perpendicular and parallel lines
   f. midpoint of a segment;
18. demonstrate:
   a. the elements of a formal proof
   b. the ability to utilize (1) above in the solution of problem material
   c. geometric inequalities
   d. the relation of arcs and angles formed by chords, secants and tangents to circles through proofs and problems
   e. the relation of circles and regular polygons, both inscribing and circumscribing;
19. form a conclusion based on mathematical logic;
20. compute areas and volumes of geometric figures;
21. do constructions with straight edge and compass.

Expected Outcome of Students:

Upon completion of the course the student should be able to:

1. when using the college computer laboratory, follow the procedures to sign in and out;
2. write, edit, compile, run and debug programs;
3. demonstrate steps involved in program development;
4. write simple C++ data types in programs and apply how they are represented in the machine;
5. write C++ expressions using selected operators, and apply the rules of precedence used in their
6. evaluation;
7. apply the structured programming constructs: sequence, selection and iteration;
8. perform elementary interactive input and output operations;
9. code void and value-returning functions with value and reference parameters and use them in a program;
10. define and use the structured C++ data types: array, string, struct in applications drawn from
11. mathematics, the sciences, and other areas;
12. use text files to record and retrieve information in elementary applications;
13. produce well-documented, user-friendly programs of short to medium length.

Course Content (Lecture):

1. Review of program development
   a. Top-Down design
   b. Methods of specifying algorithms: structure charts, flow charts, pseudocode
   c. Formulating algorithms to solve problems on a computer
   d. Program documentation standards
   e. Program testing issues and construction of test data
2. Elementary Data types in C++
   a. Simple data types: int, unsigned, long, char, float, double
   b. Machine representation of the simple data types: int, long and char (Optional)
3. Expressions and assignment statements in C++
   a. Selected C++ operators: Arithmetic, Logical, Assignment, Relational, size of
   b. Expressions and rules of precedence for their evaluation
   c. Assignment statements – lvalues and rvalues
4. Selection and Iteration structures
   a. IF and IF-ELSE statements
   b. SWITCH statements
   c. Syntax of FOR, WHILE and DO WHILE loops
   d. Types of loop control: counter, sentinel, user response, flag and EOF controlled loops
   e. Nested loops
5. Elementary interactive input and output in C++
   a. cin and cout
   b. I/O of numeric, char, string
   c. Using sentinel controlled loop to read from the keyboard
   d. Using an EOF controlled loop to read from a file
6. Use of functions in modular programming
   a. Role of program modules in well designed programs
   b. Using built-in library functions
   c. Function declarations and calls
   d. Parameter passing mechanisms: value vs. reference
   e. Value returning vs. void functions
7. One dimensional arrays

   a. Definition of array and motivation for use
   b. Input fixed or varying sized data set into an array from keyboard or file
   c. How to pass arrays as parameters
   d. Other array manipulation such as summing values
   e. C-String, an array of characters
   f. Binary search and elementary sorts
   g. Mention array names in relation to pointer variables (Optional)
8. Structs (Required), Array of Structs (Optional)
   a. Syntax and Declarations
   b. How to define and access Structs
   c. Compares and contracts with arrays
   d. Struct as function parameters
9. Sequential files
   a. Definition of text file
   b. How to define, open, and close text files
   c. Elementary reading/writing text files (loop/EOF)
   d. How to pass a file variable as a function parameter
10. User-friendly programs
   a. Input Validation techniques
   b. Output prompts
   c. Output design

Course Content (Laboratory):

1. Rules and procedures when using College Computer Laboratory
   a. Signing in and out procedures
   b. Rules for use of computers, printers and laboratory etiquette

2. Use the C++ integrated development environment
   a. Create, edit, compile and run programs
   b. Provide program comments and documentation
   c. Test and debug programs
   d. Evaluate program efficiency

Methods of Presentation:

1. Student use of appropriate computer laboratory
2. Lecture, discussion and classroom demonstrations

Assignments and Methods of Evaluating Student Progress:

1. Typical Assignments
   a. Submit a structure chart that shows function parameters as is done on p. 315: Fig. 6.9 with your source code and listing of your output file.
   b. Revise your previous angle assignment so that the main program does little but perform tile initialization, loop control and call functions.
   c. This assignment uses material from chapter 8 on files. In this assignment folder you will find a slightly revised version of copyfile.cpp. This is figure 8.4 on p. 397 – p. 399. Revise this program so that each line contains a line number followed by a colon and a couple of spaces. The original text should still line up exactly as it did before.

2. Methods of Evaluating Student Progress
   a. Quizzes, midterms, and a final examination
   b. Assigned programs
   c. Homework assignments

Textbook(s) (Typical):

Starting Out with C++: From Control Structures through Objects – sixth edition, Pearson Publisher, 2009. ISBN-13: 9780321545886.

Special Student Materials:

Portable storage device such as a USB flash drive