COURSE NUMBER: CT2300

COURSE TITLE: Applied Programming

COURSE DESCRIPTION:

This is a course designed to introduce the technology learner to the concepts of problem solving using computer programming. The course will be taught using a high level language such as C or C++. Learners will write programs to solve problems within their related disciplines and will learn the concepts of troubleshooting and problem solving. The course covers the following areas: structured programming concepts, data types, decision statements, loop and iteration procedures, Input/Output procedures, and files.

PREREQUISITES: MA1101 - Mathematics
Or
CE1140 – Network Computer Essentials
ET1151 – Circuit Analysis II

CO-REQUISITES: None

CREDIT VALUE: Four (4)

COURSE HOURS PER WEEK: Three (3)

LAB HOURS PER WEEK: Two (2)

SUGGESTED TEXT:

One of:

Larsen, R. (2011). *LabVIEW for engineers*. Prentice Hall. ISBN-13: 978-0136094296

Zak, D. (2012). *An introduction to programming with C++* (7th ed.). Boston: Course Technology. ISBN-13: 9781285061474

LEARNING RESOURCES: To be determined by instructor

MAJOR TOPICS:

1.0 Programming Fundamentals
2.0 Computer Fundamentals
3.0 Number Systems and Variable Types
4.0 Input/Output Functions
5.0 Decision Statements and Loops

6.0 Structured Programming
7.0 Strings and Arrays

LEARNING OBJECTIVES:

The expected learning outcomes are that the learner will be able to:

1.0 Programming Fundamentals

1.1 Introduction to Programming Languages
- 1.1.1 Discuss the history of computers
- 1.1.2 Describe low level and high level programming
- 1.1.3 Describe the history of programming languages

1.2 Using the Editor to Write Programs
- 1.2.1 Describe the programming environment
- 1.2.2 Demonstrate how to store and retrieve files

1.3 Compiling Programs
- 1.3.1 Describe the process of compiling a program
- 1.3.2 Explain the conversion from source code to machine code

1.4 Linking and Executable Files
- 1.4.1 Explain the function of the object file
- 1.4.2 Explain the function of an executable file
- 1.4.3 Demonstrate the function of a linker

1.5 Using the main () Function
- 1.5.1 Explain the main () function
- 1.5.2 Illustrate the purpose of the main () function in programs

1.6 Header Files
- 1.6.1 Explain header files
- 1.6.2 Explain the use of header files of repetitive operations
- 1.6.3 Demonstrate how to use the include command with header files

1.7 C Programming Syntax
- 1.7.1 Illustrate the proper format for a function
- 1.7.2 Illustrate the proper order for header files and functions
- 1.7.3 Demonstrate how to add comments to a program

2.0 Computer Fundamentals

2.1 Introduction to the Central Processing Unit
- 2.1.1 Describe the purpose of the CPU
- 2.1.2 Illustrate the use of the ALU, stack and registers

2.2 Storage Systems
- 2.2.1 Explain how programs are stored
- 2.2.2 Demonstrate how to store and retrieve files
- 2.2.3 Describe logical structures of hard drives

2.3 Memory Systems
- 2.3.1 Explain memory systems
- 2.3.2 Describe Random Access Memory
- 2.3.3 Describe Read Only Memory
- 2.3.4 Illustrate how the program is stored and executed in memory

2.4 Input and Output Devices
- 2.4.1 Introduce the concepts of Input and Output
- 2.4.2 Demonstrate input using the keyboard as an example
- 2.4.3 Demonstrate output using the monitor as an example

3.0 Number Systems and Variable Types

3.1 Introduction to Base Number Systems
- 3.1.1 Describe the number systems (decimal, binary, octal and hexadecimal)
- 3.1.2 Describe the importance of binary and hexadecimal in computers

3.2 Decimal and Nondecimal Numbers
- 3.2.1 Explain the integer number type
- 3.2.2 Explain the floating point number type
- 3.2.3 Describe how integers are stored and calculated
- 3.2.4 Describe how floating point numbers are stored and calculated

3.3 Variables and Constants
- 3.3.1 Explain the concept of a variable
- 3.3.2 Explain the need for variables in programming
- 3.3.3 Illustrate how to use constants in a formula
- 3.3.4 Introduce the concept of a constant
- 3.3.5 Illustrate constants using a constant in a formula
- 3.3.6 Declare global constants in programs

3.4 Declaring Variable Types
- 3.4.1 Demonstrate the proper way to declare variables
- 3.4.2 Declare variables of float, character and integer types

4.0 Input/Output Functions

4.1 Introduction to User Interaction
- 4.1.1 Demonstrate user input and output
- 4.1.2 Illustrate the use of input and output devices

4.2 Using Built-In I/O Functions
- 4.2.1 Use built-in input functions to get input from users
- 4.2.2 Use built-in output functions to generate program outputs

5.0 Decision Statements and Loops

5.1 Relational Operators
- 5.1.1 Define relational operators
- 5.1.2 Define the proper syntax relational operators
- 5.1.3 Illustrate the use of relational operators

5.2 If... Else Statements
- 5.2.1 Explain if... else decision making statements
- 5.2.2 Illustrate the use of relational operators in if... else statements
- 5.2.3 Demonstrate the proper syntax for if... else statements

5.3 Case Statements
- 5.3.1 Define case statements decision making statements
- 5.3.2 Explain the switch () function
- 5.3.3 Illustrate the proper syntax for case statements

5.4 While Loops
- 5.4.1 Explain loops
- 5.4.2 Illustrate the use of relational operators with while loops
- 5.4.3 Define the proper syntax for while loops

5.5 Do... While Loops
- 5.5.1 Define a do... while loop
- 5.5.2 Explain the increment and decrement commands
- 5.5.3 Illustrate the proper syntax for do... while loops

5.6 Nested Loops
- 5.6.1 Define nested loops
- 5.6.2 Illustrate how to put one loop inside another
- 5.6.3 Describe the problem of endless loops

6.0 Structured Programming

6.1 Program Design Using Pseudo Code
- 6.1.1 Define pseudo code
- 6.1.2 Demonstrate pseudo code as a means to define the programming solution

6.2 Program Design Using Flow Charts
- 6.2.1 Define flow chart
- 6.2.2 Illustrate the use of flow charts in program problem solving

6.2.3 Draw the symbols for proper flow charting

6.3 Introduction to Functions
- 6.3.1 Define a function
- 6.3.2 Outline how programs can be broken down into smaller functions
- 6.3.3 Demonstrate how functions simplify programming

6.4 Calling Functions
- 6.4.1 Demonstrate the proper way to call a function
- 6.4.2 Demonstrate what happens when a function is called

6.5 Using Function Prototypes
- 6.5.1 Define function prototypes
- 6.5.2 Demonstrate the purpose of declaring functions

7.0 Strings and Arrays

7.1 Introduction to Strings
- 7.1.1 Define a string
- 7.1.2 Illustrate how to declare and use strings

7.2 Storing Strings
- 7.2.1 Demonstrate how strings are stored in memory

7.3 Introduction to Arrays
- 7.3.1 Define array
- 7.3.2 Describe how arrays are used in programming
- 7.3.3 Implement a single dimensional array

7.4 Multidimensional Arrays
- 7.4.1 Define multidimensional arrays
- 7.4.2 Define rows and columns
- 7.4.3 Demonstrate multidimensional arrays
- 7.4.4 Define array applications using spreadsheets and database examples

7.5 Array Applications
- 7.5.1 Implement algorithms to compute the average of values stored in an array
- 7.5.2 Implement algorithms to perform a linear search and a binary search on an array
- 7.5.3 Implement algorithms to sort array elements, such as insertion sort, quick sort, merge sort

EVALUATION:

Assignments:	25%
Projects:	15%

Quizzes:	60%		
DATE DEVELOPED:	March 25, 1996	**DATE REVIEWED:**	
REVISION NUMBER:	4	**DATE REVISED:**	March 2014

Note to instructor: *Check PIRS to ensure this outline is the most current version.*