

## Introduction to Programming Concepts TCSU IS 110

### A. Description

This course may be taught in the Java programming language. Its purpose is to expose students to the fundamental concepts of programming.

### B. Minimum Unit Requirement

3 semester units

### C. Course Topics

1. Software life-cycle including design, development, styles, documentation, testing and maintenance
2. Procedural versus objected oriented programming
  - a. Survey of current languages
3. Object oriented programming
  - a. Class members
  - b. Fields, methods, and constructors
4. Program design tools and programming environments
5. Documentation
6. Coding conventions
7. Data types
8. Arrays
  - a. Declaring and allocating arrays
  - b. Multiple-subscripted arrays
9. Control structure
  - a. Selective structures: if and switch
  - b. Repetitive structures: loops
10. Algorithms including simple sorting and searching
11. File I/O
  - a. Files and streams
  - b. Sequential access files
12. Error handling
13. Passing parameters by value and by reference
14. Principles of testing and designing test data

### D. Student Learning Outcomes

1. To gain a general understanding of the software development life-cycle.
  - a. Describe and understand the importance of each step in the complete life-cycle.
2. To learn the distinctive characteristics and typical uses of widely-used programming languages.
  - a. Describe some of the distinctive characteristics of programming languages widely used in contemporary software development; and

- b. Understand the appropriate use of different languages for different applications.
3. To understand the principles of structured programming, and to be able to state and define basic concepts of object-oriented programming.
  - a. Distinguish between structured and object oriented programs; and
  - b. Define the distinguishing characteristic of object-oriented concepts such as class, object, instantiation, property, method, constructor, and destructor, and of encapsulation, inheritance, and polymorphism.
4. To understand and to be able to use standard program design tools and documentation.
  - a. Explain the importance of system requirements, design documents, and user documentation; and
  - b. Produce a sketch of a program in pseudocode from a short sample program specification.
5. To understand and explain how software stores and represents data.
  - a. Differentiate between complex and primitive data types; and
  - b. distinguish between individual variables and single-dimensional and multi-dimensional arrays.
6. To understand and use the three fundamental control structures of sequence, selection, and repetition.
  - a. Describe the three fundamental control structures and use each one in writing a short program in an appropriate language;
  - b. Describe and explain the differences between the different types of selection including case/switching; and
  - c. Explain the differences and uses of various repetition structures such as: “while”-test-at-top, “until”-test-at-top, “while”-test-at-bottom, and “until”-test-at-bottom.
7. To understand the concept of an algorithm and to become familiar with some examples.
  - a. Explain what an algorithm is and it’s importance in computer programming;
  - b. Using pseudocode, sketch algorithms for a linear search and for a binary search and a sort; and
  - c. Understand the difference between a queue and a stack.
8. To become familiar with standard terminology for file handling and to use some simple file handling techniques.
  - a. Explain the difference between sequential and random access and be able to use each in a program to open, read, write and close a file;
  - b. Explain the concept of a “stream” and the connection between a stream and file operations; and
  - c. Explain the concept of a “buffer” and the connection between a buffer and file operations.
9. To understand the importance of error handling and to be able to implement simple error handling techniques.
  - a. Explain the importance of error handling in a program and be familiar with common situations requiring error handling; and
  - b. Use error handling in a short program using an appropriate language.
10. To understand and implement the use of procedures or subroutines.
  - a. Describe the concept of program subroutines or procedures, and explain the general reason for their use;

- b. Understand the difference between a global variable and a local variable; and
  - c. Use subroutines in a short program using an appropriate language.
- 11.** To understand the importance of testing and to be able to implement a process of testing.
- a. Explain the importance of testing in the process of software development.