Prepared by the Department of Mathematics

Date of Departmental Approval: April 10, 2017
Date Approved by Curriculum and Programs: April 12, 2017 Effective: Fall 2017

1. **Course Number: CSC120**
   Course Title: Computer Programming I: C++

2. **Description:** Students use projects and teamwork to design, implement, and test programs in C++. Programming style, expression, and documentation are emphasized. Object-oriented programming methodology, graphical user interfaces, debugging techniques, pointers, simple recursion, and string processing are covered.

3. **Student Learning Outcomes (instructional objectives, intellectual skills):** Upon successful completion of the course, the student shall be able to:
   - explain the object-oriented design process and the concept of software engineering
   - develop algorithms and solve problems with C++
   - design, code, debug, test, and document programs using predefined and programmer-defined methods and classes, formatting, casting, constructors, overloaded methods, get/set methods, primitive and reference variables, pointers, logical and relational operators, random number generators, decision statements, loops, and arrays
   - Apply consistent documentation and program style standards that contribute to the readability and maintainability of software.
   - identify and code with legal C++ identifiers, variable names, arithmetic, and Boolean expressions, comments, data types, methods (including constructors), objects, and classes
   - evaluate arithmetic and Boolean expressions
   - determine scope of variables and effect of access modifiers
   - convert between conditional and unconditional loops and between various decision statements
   - Explain programming fundamentals, including statement, control flow and recursion.
   - Apply the concepts of class, method, constructor, instance, data abstraction, function abstraction, inheritance, overriding, overloading, and polymorphism.
   - Explain the concept of pointers, trace code with pointers, and implement a solution to a problem using pointers.
   - Program with basic data structures using array, list, and linked structures.
   - choose appropriate loops, decision statements, and class hierarchies given program specifications
   - utilize predefined string methods to examine string literals and user-input strings
   - process data in one-dimensional array, use sequential search with parallel arrays, and sort one-dimensional array on different keys
   - Apply consistent documentation and program style standards that contribute to the readability and maintainability of software.

4. **Credits:** 4 credits

5. **Satisfies General Education Requirement:** No

6. **Prerequisite:** MAT035 (Algebra for Non-STEM) or satisfactory basic skills assessment scores and working knowledge of any programming language

7. **Semesters Offered:** Fall

8. **Suggested General Guidelines for Evaluation:** Comprehensive final examination, hour tests, programs, quizzes, and homework papers

9. General Topical Outline:

   1. **Computers and C++**
      a. C++ Development Environment

b. Software Development Terminology

2. **Input/Output and Operators**

3. **Classes, Objects, Member Functions and Strings**
    a. Headers and Source-Code Files
    b. UML Class Diagrams

4. **Algorithm Development and Control Statements**
    a. Algorithms and Pseudocode
    b. Control Structures
    c. Iteration Statements
    d. Arithmetic Overflow
    e. Input Validation
    f. Top-Down, Stepwise Refinement
    g. Logical Operators

5. **Functions and Recursion**
    a. Random-Number Generator
    b. Function-Call Stack and Activation Records
    c. Inline Functions
    d. References and Reference Parameters
    e. Function Overloading
    f. Recursion vs. Iteration

6. **Class Templates array and vector**
    a. Sorting and Searching
    b. Multidimensional arrays

7. **Catching Exceptions**

8. **Pointers**
    a. Pass-by-Reference with Pointers
    b. Built-In Arrays

9. **Operator Overloading**
    a. Fundamentals of Operator Overloading
    b. Dynamic Memory Management

10. **Object-Oriented Programming: Inheritance**
    a. Base Classes and Derived Classes
    b. Constructors and Destructors in Derived Classes

11. **Object-Oriented Programming: Polymorphism**
    a. Relationships Among Objects in an Inheritance Hierarchy
    b. Abstract Classes and Pure Virtual Functions
    c. Polymorphism, Virtual Functions and Dynamic Binding

12. **Stream Input/Output**
13. **File Processing**
14. **Standard Library Containers, Iterators, and Algorithms**
15. **Exception Handling**