# Math 1220: Scientific Programming

**Credit hours:** **3 credit hours**
**Prerequisites:** **MATH 1200 with a minimum grade of C or placement test**

## Course Description

This course offers instruction in scientific programming using a current programming language. Problems, both numerical and non-numerical, are programmed and solved using a personal computer.

## Course Objectives

1. Establish a firm foundation in the principals of scientific programming
2. Solve problems using algorithms
3. Become familiar with a current programming language

## Learning Outcomes

1. Understand and apply the principles of the five-step process for scientific programming (Problem-Solution-Algorithm-Pseudocode-Source Code) by solving and documenting scientific programming projects
2. Employ basic C++ language and syntax to develop source code for scientific programming projects
3. Utilize fundamental principles of mathematical logic to define control structures for complex scientific programming projects
4. Apply C++ structures for functions, input/output files, and arrays to solve large, complex, scientific programming problems
5. Develop the professional skills to work as part of a scientific/engineering team by preparing technical documentation for all parts of the five-step process for scientific programming

## Course Topics

I. INTRODUCTION TO COMPUTING
- A. Overview of computer technology
- B. Introduction to the programming process
    1. Problem definition to pseudocode
    2. Source code
    3. Compile/Link/Run

II. PROBLEM DEFINITION TO PSEUDOCODE
- A. Well-defined problem
- B. Deriving a solution
- C. Algorithm: writing a recipe to implement the solution
- D. Pseudocode: almost a high-level language code

III. WRITING THE SOURCE CODE: PART 1
- A. Declaring variables
    1. Data types and compatibility
- B. Collecting data: input commands
- C. Using commands and syntax to implement the pseudocode
    1. Arithmetic operators
    2. Elementary control loops

D. Displaying results: output commands
    1. Formatting data
E. Documentation: include comments in the code

IV. RUNNING THE CODE
A. Compile/Link/Run
B. Debugging

V. WRITING THE SOURCE CODE: PART II
A. Using predefined functions
B. User-defined functions
C. Local vs. global variables and constants
D. Advanced techniques for using data in functions
E. Input/output via data files

VI. CONTROL LOGIC AND COMMANDS: MORE APPLICATIONS
A. If-else statements
B. Do-while loops
C. For-statements

VII. LIBRARIES OF FUNCTIONS
A. Predefined libraries
B. User defined libraries

VIII. Arrays
A. Introduction to arrays
B. Arrays in functions
C. Multidimensional arrays