IT-2670: C/C++ PROGRAMMING LANGUAGE

Cuyahoga Community College

Viewing: IT-2670 : C/C++ Programming Language

Board of Trustees:
November 2019

Academic Term:
Fall 2020

Subject Code
IT - Information Technology

Course Number:
2670

Title:
C/C++ Programming Language

Catalog Description:
Introduction to programming using the C and C++ programming languages, emphasizing program development and design, debugging techniques, and common basics of the C/C++ languages. Topics include Object-Oriented concepts (including classes, objects, attributes, methods and object communication), Structured Programming concepts (including control statements, conditions, loops) and Data Structures (including collections), data types, functions, argument passing, arrays, strings, structures, data files, and classes.

Credit Hour(s):
4

Lecture Hour(s):
3
Lab Hour(s):
2

Requisites

Prerequisite and Corequisite
IT-1050 Programming Logic.

Outcomes

Course Outcome(s):
Demonstrate the ability to analyze, design and implement solutions to programming problems using C/C++.

Objective(s):
1. Design and express program logic using structured programming techniques.
2. Demonstrate the ability to test, debug, and correct program code.
3. Create executable C/C++ programs.

Course Outcome(s):
Differentiate C/C++ and the current standard practices utilized in software engineering.

Objective(s):
1. Demonstrate an understanding of accepted language style conventions and documentation.
2. Correctly implement C/C++ syntax, including program control, function calls, data types, and file handling.
3. Explain the relationship of C and C++ languages.

Methods of Evaluation:

1. Class participation and discussion
2. Oral and/or written reports
3. Homework assignments
4. Hands-on computer lab projects
5. Comprehensive projects
6. Quizzes
7. Objective examinations
8. Hands-on computer lab examinations
9. Other methods deemed appropriate by the department

Course Content Outline:

1. C/C++ language fundamentals
   a. Syntax and Semantics
   b. Syntax Templates
   c. Naming program elements: identifiers
   d. Naming Elements: declarations
   e. Executable statements
   f. Comments - practice and conventions
2. Program anatomy
   a. Blocks
   b. Preprocessor
   c. Compiling and running a program
   d. Testing and debugging
3. Data types
   a. Numeric
      i. Integral
      ii. Floating-point
      iii. Arithmetic expressions
      iv. Compound arithmetic expressions
      v. Function calls and library functions
      vi. Formatting output
      vii. String operations
      viii. Vectors
   b. Software design methodologies
      i. Structured Programming
      ii. Object-Oriented Programming
4. Structured Programming
   a. Conditional and logical expressions
   b. Nested and complex expressions
   c. logical operators
   d. looping structures
5. Functions (structured)
   a. Functional decomposition with Void functions
   b. User-defined functions
   c. Syntax and semantics of Void functions
   d. Parameters
   e. Scope and lifetime of functions and variables
   f. Interface design
   g. Value-returning functions
   h. Type coercion in assignments, argument passing, and return of a function
6. User-defined data types
   a. Simple vs. structured data types
   b. Structs
   c. Unions
   d. Pointers
   e. Reference types
   f. Enumerations

7. Arrays and strings
    a. One and two-dimensional arrays
    b. Arrays of records
    c. Passing two-dimensional arrays as arguments
    d. Subarray processing
8. Object-Oriented Programming
    a. OOP concepts
        i. Inheritance
        ii. Polymorphism
        iii. Encapsulation
        iv. Composition
    b. Methods
    c. Classes, objects, and members
        i. Concrete Types
        ii. Abstract Types
        iii. Virtual Functions
        iv. Class hierarchies
        v. Constructors
        vi. Overloading/Overriding
    d. Unified Modeling Language (UML) diagrams for Object Models
9. Exceptions
    a. Throw
    b. Try-catch
    c. Exception handlers
    d. Standard exceptions
    e. Exception class hierarchies
10. Data Structures/Collections
    a. Abstract data structures vs implementations
    b. Stacks, Queues and Priority Queues
    c. Vectors
    d. Standard Template Library (STL)
    e. Binary trees
    f. Hash Tables
    g. Recursion
11. File handling
    a. I/O state
    b. I/O of User-Defines Types
    c. File Streams
    d. String Streams
12. History and Compatibility
    a. History
    b. C++ extensions
    c. C/C++ compatibility

Resources

Deitel, Harvey M., and Paul J. Deitel. *C++, How to Program.* 9th ed. Upper Saddler River, NJ: Prentice Hall, 2013.

Thareja, R. *Intorduction to C Programming.* Pap/Cdr ed. Oxford: University Press, 2013.

Stroustrup, B. *The C++ Programming Language.* 4th ed. Boston: Addison-Wesley, 2013.

Liang, Y. *Introduction to Programming with C++.* 3rd.ed. Upper Saddle River, NJ: Prentice Hall, 2013.

Meyers, S. *Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14.* 1st ed. O'Reilly, 2014.

Polukhin, A. *Boost C++ Application Development Cookbook.* Packt Publishing, 2013.

Top of page
Key: 2511