COSC118-Computer Programming Fundamentals with C++ 3 Credits
SYLLABUS

CATALOG DESCRIPTION

An Introduction to programming and computing integrating problem solving and algorithmic design using the object-oriented programming language C++.

Prerequisites: MATH 096 or APAL 061, ENGL 095/APEN 070, RDNG 095/APRD 058

Semester Offered: Fall & Spring

Common Student Learning Outcomes

Upon successful completion of San Juan College programs and degrees, the student will demonstrate competency in…

BROAD AND SPECIALIZED LEARNING

Students will actively and independently acquire, apply, and adapt skills and knowledge with an awareness of global contexts.

CRITICAL THINKING

Students will think analytically and creatively to explore ideas, make connections, draw conclusions and solve problems.

CULTURAL AND CIVIC ENGAGEMENT

Students will act purposefully, reflectively, and ethically in diverse and complex environments.

EFFECTIVE COMMUNICATION

Students will exchange ideas and information with clarity in multiple contexts.

INFORMATION LITERACY

Students will be able to recognize when information is needed and have the ability to locate, evaluate, and use it effectively.

INTEGRATING TECHNOLOGIES

Students will demonstrate fluency in the application and use of technologies in multiple contexts.

Student work from this class may be randomly selected and used anonymously for assessment of course, program, and/or institutional learning outcomes. For more information, please refer to the Dean of the appropriate School.

GENERAL LEARNING OBJECTIVES

I. To instill in students the basics of problem solving.

II. To equip students with a basic understanding of the C++ language and syntax.

III. To provide students with hands-on programming experience in both procedural and object-oriented programming.

SPECIFIC LEARNING OUTCOMES

Upon successful completion of the course, the student will be able to …

1. Analyze, design, implement and test a computing problem (I) (T, A).
2. Describe the Input/Process/Output Algorithmic Pattern (I) (T).
3. Understand the nuts of bolts of C++ programs: variable declaration, initialization, assignment, Input/Output, and arithmetic expressions (II) (L).
4. Design and implement the Prompt then Input pattern (I) (T, I, A).
5. Evaluate and write arithmetic expressions (I, II, III) (T, A).
6. Use existing functions (II) (I, A).
7. Read and write function heading, preconditions and postconditions (II, III) (T, I, C, A).
8. Do int quotient/remainder division with / and % (I, II) (T, I, C, A).
9. Comprehend the relationship between objects and classes (III) (L, T, C, I, A).
10. Send messages to member functions (II, III) (C, A).
11. Appreciate why software is divided up into classes and functions (II, III) (L, I).
12. Write free (non-member) functions (II) (I, A).
13. Discern why software is structured into functions communicating via argument/parameter associations and returns (II) (L, T, I, C, A).
14. Appreciate the importance of testing (I, III) (L, T, I, C, A).
15. Use test drivers to help implement and test free functions (III) (T, A).
16. Properly define and understand scope rules (II, III) (I, A).
17. Read and write class definitions (III) (L, T, I, C, A).
18. Discuss class design issues (public versus private; choosing members) (III) (L, T, C).
19. Implement classes including data members and member functions given the class definition (III) (T, I, A).
20. Explain the role of constructors, accessors, modifiers and how to implement them (III) (L, T, I, C, A).
21. Implement algorithms using if, if…else, and nested if…else (I, II) (L, A).
22. Develop programs using repetition (while, for and do while loops) (I, II) (L, A).
23. Distinguish between determinate and indeterminate loops (I, II) (L, A).
24. Recognize nested repetition (I, II) (L, T, I, C, A).
25. Use file input and output (II) (L, A).
26. Read until the end of file using the indeterminate loop pattern (II) (L, T, I, C, A).
27. Utilize vectors and arrays as object that stores a collection of objects (I, II, III) (L, A).
28. Present sequential search and a simple O(n2) sorting algorithm (selection sort) (I, II, III) (L, T, I, C, A).
29. Outline binary search and indicate that it is more efficient than sequential search by comparing O(log n) and O(n) (I, II, III) (L, T, I, C, A).

COURSE CHALLENGE PROCEDURES

This course may be challenged by taking and passing all the semester tests and the final exam. A grade of 70% is required to successfully challenge this course.