Coding I

Level 2: Student may have explored previously; first pathway specific course
Pathway(s): Coding

Description

Coding I is a course intended to teach students the basics of computer programming. The course places emphasis on practicing standard programming techniques and learning the logic tools and methods typically used by programmers to create simple computer applications. Upon completion of this course, proficient students will be able to solve problems by planning multistep procedures; write, analyze, review, and revise programs, converting detailed information from workflow charts and diagrams into coded instructions in a computer language; and will be able to troubleshoot/debug programs and software applications to correct malfunctions and ensure their proper execution.

Student Learning Outcomes

Computer Programming Overview

1) Identify key milestones in the development of computers and logical devices
    a. Evaluate how computing impacts practices within the following contexts
        i. Personal
        ii. Ethical
        iii. Social
        iv. Economic
        v. cultural
2) Predict how computational innovations that have revolutionized aspects of our culture might evolve

Ethics

3) Analyze ethical programming practices, including but not limited to the issues of:
    a. Confidentiality
    b. Privacy
    c. Piracy
    d. Fraud and misuse
    e. Liability
    f. Copyright
    g. Open source software
    h. Trade secrets
    i. Sabotage

Programming Skills

4) Differentiate between system-level and application solutions and identify an appropriate code-based strategy to solve a given problem.
5) Apply the system management tools present in a programming development environment to:

    a. Develop syntactically correct program code using current best practices and emerging classes of development techniques
    b. Use a compiler/interpreter to produce executable code
6) Develop strategies that work within the constraints of major operating system fundamentals, such as:
    a. Opening a file for reading and writing
    b. File management syntax requirements, including but not limited to creating, naming, organizing, copying, moving, and deleting files
    c. File name conventions, as they apply across multiple software applications and file types
7) Write pseudocode and construct a flowchart for a process before starting to develop the program code.
8) Organize and develop a plan to acquire and manage the data values for a process, including the following:
    a. Data types, such as string, numeric, character, integer, and date
    b. Program variable names
    c. Variables and constants
    d. Arrays (at least one-and two-dimensional), subscripts
    e. Input from files and user responses
    f. Output to files and reports
9) Using a programming language specified by the instructor, convert the pseudocode for a selected process to program code, incorporating some of the following:
    a. Operations and functions (user-defined and/or library)
    b. Repetition (loops)
    c. Decisions (if…else, case)
    d. Recursion
10) Verify the correct operation of the resulting program code with several test cases:
    a. All valid values
    b. Error trapping of invalid values
    c. Error trapping of invalid program operations
    d. Troubleshooting/remedying program problems
11) Convert numbers from base-10 to binary and hexadecimal

Project Planning & Quality Assurance

12) Compile the necessary documentation to understand the nature of a computer programming problem and the customer/client specifications for the request and summarize. This will include:
    a. Evidence of the scope of problem
    b. Its intended input and output information
    c. The required system processing
    d. Software specifications involved
13) In the software development process, articulate the nature of the program designs by creating documentation that addresses topics including, but not limited to:
    a. The procedural, object-oriented, event-driven, or other nature of the various portions of the resulting application
    b. The data structures used for inputs, outputs and the internal manipulations
    c. The algorithms and guiding formulas used

d. Constraints on the accurate operation and results
e. Modular designs that enable portability
f. Interface details that permit ready maintenance and upkeep

14) Apply principles of quality assurance during application development to certify bug tracking, audit trails, testing results and other quality considerations. Annotate each quality assurance task with evidence from best practices endorsed by industry or research