Department of Education | College, Career and Technical Education

Programming & Logic I

Primary Career Cluster:	Information Technology
Consultant:	Deborah Knoll, (615) 532-2844, Deborah.Knoll@tn.gov
Course Code(s):	6098
Prerequisite(s):	*Algebra I* (0842, 3102), *Information Technology Foundations* (6095)
Credit:	1
Grade Level:	10
Graduation Requirements:	This course satisfies one of three credits required for an elective focus when taken in conjunction with other Information Technology courses.
Programs of Study and Sequence:	This is the second course in the *Programming & Software Development* program of study.
Aligned Student Organization(s):	Skills USA: http://www.tnskillsusa.com Tracy Whitehead, (615) 532-2804, Tracy.Whitehead@tn.gov Technology Student Association (TSA): http://www.tntsa.org Tracy Whitehead, (615) 532-2804, Tracy.Whitehead@tn.gov
Coordinating Work-Based Learning:	Teachers are encouraged to use embedded WBL activities such as informational interviewing, job shadowing, and career mentoring. For information, visit https://tn.gov/education/topic/work-based-learning.
Available Student Industry Certifications:	None
Dual Credit or Dual Enrollment Opportunities:	There are no statewide dual credit/dual enrollment opportunities for this course. If interested in establishing a local opportunity, reach out to a local postsecondary institution.
Teacher Endorsement(s):	037, 041, 055, 056, 057, 152, 153, 203, 204, 311, 434, 435, 436, 474, 475, 476, 477, 582, 595, 740, 742
Required Teacher Certifications/Training:	All endorsements except for 742 will require the equivalent of twelve semester hours of computer course work including at least six hours of programming language.
Teacher Resources:	https://tn.gov/education/article/cte-cluster-information-technology

Course Description

Programming & Logic 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.

Program of Study Application

This is the second course in the *Programming & Software Development* program of study. For more information on the benefits and requirements of implementing this program in full, please visit the Information Technology website at https://tn.gov/education/article/cte-cluster-information-technology.

Course Standards

Computer Programming Overview

1) Using news articles and instructional materials, investigate key milestones in the development of computers and logical devises. Create and present a document and/or illustration depicting the timeline of development that led to modern-day operating systems, programmable controllers, and widespread digital communications via the Internet and wireless networks, citing specific textual evidence.

2) Compare and contrast the benefits, features, and typical applications of common modern programming languages and environments. Craft an argument to defend the choice of a certain language to solve a particular problem, developing claim(s) and counterclaim(s) with specific textual evidence and reasoning.

Ethics

3) Using news articles and text of legislation, analyze ethical programming practices, including but not limited to the issues of confidentiality, privacy, piracy, fraud and misuse, liability, copyright, open source software, trade secrets, and sabotage. For example, research and report on the effects of unethical programming practices on a business.

Programming Skills

4) Differentiate between system-level and application solutions, and identify an appropriate code-based strategy to solve a given problem. For example, given a file management problem, determine when a command-line script will be more efficient than a high-level program solution.

5) Apply the system management tools present in a programming development environment to:
    a. Select the most appropriate programming language for the task at hand
    b. Develop syntactically correct program code using current best practices and emerging classes of development techniques
    c. Use a compiler to interpret the source code and produce executable program code

6) In the process of developing and implementing programming solutions, develop strategies that work within the constraints of major operating system fundamentals, such as:
    a. Security protocols and procedures for accessing files and folders
    b. File management syntax requirements, including but not limited to creating, naming, organizing, copying, moving, and deleting files
    c. File naming 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. For example, code and flowchart a simple process that takes an integer and report whether it is odd or even.

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 at least three of the following structures, the need for which will be dictated by the assigned problem(s) and process(es). The resulting code design can be event-driven, object-oriented, or procedural.
    a. Operations and functions (user-defined and/or library)
    b. Repetition (loops)
    c. Decision (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 operation
    d. Troubleshooting/remedying program problems

Project Planning and Quality Assurance

11) Compile the necessary documentation to understand the nature of a computer programming problem and the customer/client specifications for the request and summarize in an informational text. This will include evidence of the scope of the problem, its attendant input and output information, the required system processing, and the software specifications involved.

12) Analyze a given problem and develop a coherent strategy in the form of a project plan to meet the customer/client's need. The plan will include, but will not be limited to, defining the project scope as addressed by the problem documentation, identifying software

development and implementation issues, timeline and benchmarks for design, and addressing issues associated with software maintenance and life cycle.

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 internal manipulations
    c. The algorithms and guiding formulas used
    d. Constraints on 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.

15) Document the security risks associated with new applications and evaluate the severity of the risk involved in each, including but not limited to:
    a. Identifying threats to information systems facilities, data communications systems, and other applications
    b. Adhering to federal and state legislation pertaining to computer crime, fraud, and abuse
    c. Providing means for preserving confidentiality and encryption of sensitive data
    d. Detailing steps to recover from routine errors or catastrophic failures, such as might be caused by a malicious computer virus

Standards Alignment Notes

*References to other standards include:

- P21: Partnership for 21st Century Skills Framework for 21st Century Learning
    - Note: While not all standards are specifically aligned, teachers will find the framework helpful for setting expectations for student behavior in their classroom and practicing specific career readiness skills.