TN Department of Education | College, Career and Technical Education

Programming & Logic II

Primary Career Cluster:	Information Technology (IT)
Consultant:	Deborah Knoll, (615) 532-2844, Deborah.Knoll@tn.gov
Course Code(s):	6099
Prerequisite(s):	*Programming & Logic I* (6098)
Credit:	1
Grade Level:	11
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 third 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:	CompTIA A+
Dual Credit or Dual Enrollment Opportunities:	There are no known dual credit/dual enrollment opportunities for this course. If interested in developing, reach out to a local postsecondary institution to establish an articulation agreement.
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 II challenges students to develop advanced skills in problem analysis, construction of algorithms, and computer implementation of algorithms as they work on programming projects of increased complexity. In so doing, they develop key skills of discernment and judgment as they must choose from among many languages, development environments, and strategies for the program life cycle. Course content is reinforced through numerous short- and long-term programming projects, accomplished both individually and in small groups. These projects are meant to hone the discipline and logical thinking skills necessary to craft error-free syntax for the writing and testing of programs. Upon completion of this course, proficient students will demonstrate an understanding of object-oriented programming language using high-level languages such as FOCUS, Python, or SAS.

Program of Study Application

This is the third course in *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

Software Development Environments

1) Evaluate at least two software development environments (SDEs) that are tailored to different programming languages on the basis of their suitability for a range of programming tasks, ease of use, and how ubiquitous they are within the IT community. Document in an oral presentation the similarities and differences between the two, and the features that lend themselves to the chosen programming languages. For example, students assigned to code a basic database interface can compare the benefits and features of a freeware SDE such as *JDeveloper* and a commercial SDE like *Microsoft Visual Studio*.

2) Investigate the typical process around creating new software within a software development environment. Describe and furnish examples of the steps taken within the SDE to guarantee reliable output, from prototyping and authoring to deployment and debugging.

3) Administer the process of creating new software within a software development environment to manage the prototyping, authoring, revising, compiling, testing, deploying, and debugging of student-developed software. For example, for an object-oriented payroll program assignment (retrieving file data to produce a run of paychecks and paystubs for a small business), perform and document the steps taken within the SDE to ensure the reliable and accurate output of paychecks.

Software Development Life Cycle

4) Synthesize information from a range of sources (including original tests and simulations) to critique the features of different software development life cycles (agile, iterative, and sequential types). Using domain-specific terminology, explain to a technical audience the distinguishing features of each that make one more appropriate for certain types of applications.

5) For a selected assignment or project involving the development of original software, choose and defend a strategy to follow for the program's development life cycle. At the completion of the assignment, offer recommendations for other environments and alternative strategies that could improve the development process.

6) Research common and best-practice techniques in programming analysis, design, and implementation. Drawing on model practices used by businesses and industry, employ analysis, design, and implementation techniques to satisfy a programming need, using an appropriate software lifecycle model.

7) Employ a requirement management tool during a program's development life cycle, documenting the evolving versions, storage attributes, system elements, status tracking, and access permissions afforded by the tool, as well as the successful attainment of the project vision.

Designing Computer Applications

8) For a given programming assignment, choose and defend a programming language with regard to the language's capabilities and suitability to task, availability, portability, maintainability, and cost.

9) For the assignment outlined in standard 8, identify the method of data processing most appropriate for the task (e.g., batch, interactive, or event-driven). For example, a weekly payroll application would handle its data differently (i.e., batch processing) than a web-based search engine (i.e., interactive processing), and still differently than a microprocessor control program for a washing machine (i.e., event driven).

10) Define the specifications of the data management plan, including variables (naming, scope, and types), validation measures (to protect the data from corruption), and data handling (storing, input/output, and back-up). For example, programs handling historical temperature data would be best suited to floating point values stored in multidimensional arrays, written to permanent storage, and displayed with limited precision.

11) For a selected programming assignment involving an object-oriented language, design and define the classes, objects, properties, methods, and inheritance structures prior to the start of the development cycle. Revise the plan (modifications, additions, and subtractions) as needed throughout the development cycle.

Coding Computer Applications

12) For selected programming assignments, create, edit, and improve documentation for technical support intended for fellow programmers, including within the program code itself as well as within supplemental documents. For example, for a lawn sprinkler system microcontroller, the technical documentation would define the variables, functions and subroutines, and the critical events.

13) For selected programming assignments, create, edit, and improve end-user documentation. End-user documentation would include how to interact with the user interface, the capabilities and limitations of the system, and the required conditions for successful operation.

14) Incorporate structured, object-oriented, and event-driven programming techniques that employ sequence, selection, and/or repetition (loops) to solve programming projects.

15) For each programming task, consider and defend the choice of various programming approaches (such as data-driven or event-driven, top-down or bottom-up), citing examples from the syntax illustrating the chosen approach.

16) Design and develop an app for a mobile computing device, using an online programming interface, such as AppMakr, BuzzTouch, Appsbar, PhoneGap, or AppYet.

Software Testing Procedures & Quality Assurance

17) During the development, testing, and deployment of a new program, implement checks for data and procedure accuracy, correctness, currency, and relevance, making and documenting revisions where justified.

18) Analyze the code written by another programmer to create a flowchart, suggesting points of confusion or generality in the program that could become problematic in future revisions. Cite specific examples in the code to support recommendations.

19) Conduct quality testing of program code, striving for satisfactory results at four levels or perspectives:
    a) Unit (component/module level verifications)
    b) Integration (verifying the interfaces between components, adding one at a time)
    c) System (verifying that the whole package meets the requirements and specifications without corrupting other systems)
    d) Acceptance (customer satisfaction)

Project Management

20) Design, manage, and develop a course-long programming project pre-approved by the instructor. The project will embody a variety of strategies and resources taught in this course, and require periodic reviews, status reports, and final project presentation. Use a software development environment to manage, document, test, deploy, and maintain the resources and assets of the finished project.

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.