TN Department of Education | College, Career and Technical Education

May 2023

Coding II

Primary Career Cluster:	Information Technology (IT)
Course Contact:	CTE.Standards@tn.gov
Course Code(s):	C10H15
Prerequisite(s):	*Coding I* (C10H14)
Credit:	1
Grade Level:	11
Focus Elective Graduation Requirements:	This course satisfies one of three credits required for an elective focus when taken in conjunction with other *Information Technology* courses.
Program of Study (POS) Concentrator:	This course satisfies one out of two required courses that meet the Perkins V concentrator definition, when taken in sequence in the approved program of study.
Programs of Study and Sequence:	This is the third course in the *Coding* program of study.
Aligned Student Organization(s)	SkillsUSA: http://www.skillsusatn.org/ Technology Student Association (TSA): http://www.tntsa.org
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://www.tn.gov/education/educators/career-and-technical-education/work-based-learning.html.
Available Student Industry Credentials:	Credentials are aligned with post-secondary and employment opportunities and with the competencies and skills that students acquire through their selected program of study. For a listing of promoted student industry credentials, visit https://www.tn.gov/content/tn/education/educators/career-and-technical-education/student-industry-certification.html.
Teacher Endorsement(s):	037, 041, 055, 056, 057, 152, 153, 173, 203, 204, 311, 413, 434, 435, 436, 470, 474, 475, 476, 477, 582, 595, 596, 740, 742, 952, 953
Required Teacher Certifications/Training:	All endorsements except for 173 and 742 will require either the NOCTI test code 5906: Computer Programming certification **or** the equivalent of twelve semester hours of computer course work including at least six hours of programming language.
Teacher Resources:	https://www.tn.gov/education/educators/career-and-technical-education/career-clusters/cte-cluster-information-technology.html Best For All Central: https://bestforall.tnedu.gov/

Course at a Glance

CTE courses provide students with an opportunity to develop specific academic, technical, and 21st century skills necessary to be successful in career and in life. In pursuit of ensuring every student in Tennessee achieves this level of success, we begin with rigorous course standards which feed into intentionally designed programs of study.

Students engage in industry relevant content through general education integration and experiences such as career and technical student organizations (CTSO) and work-based learning (WBL). Through these experiences, students are immersed with industry standard content and technology, solve industry-based problems, meaningfully interact with industry professionals, and use/produce industry specific, informational texts.

Using a Career and Technical Student Organization (CTSO) in Your Classroom

CTSOs are a great resource to put classroom learning into real-life experiences for your students through classroom, regional, state, and national competitions, and leadership opportunities. Below are CTSO connections for this course, note this is not an exhaustive list.

- Participate in CTSO Fall Leadership Conference to engage with peers by demonstrating logical thought processes and developing industry specific skills that involve teamwork and project management.
- Participate in contests that highlight job skill demonstration, interviewing skills, community service activities, extemporaneous speaking, and job interview.
- Participate in leadership activities such as Student2Student Mentoring, National Week of Service, Officer Training, and Community Action Project.

For more ideas and information, visit Tennessee SkillsUSA at http://www.skillsusatn.org/.

Using Work-Based Learning (WBL) in Your Classroom

Sustained and coordinated activities that relate to the course content are the key to successful work-based learning. Possible activities for this course include the following. This is not an exhaustive list.

- **Standards 1.1-1.3** | Have a guest software developer come and speak about their development process.
- **Standards 2.1-2.4** | Have students job shadow a software developer.
- **Standards 3.1-3.4** | Have students do a project on computer applications partnered with industry professionals.
- **Standards 4.1-4.5** | Have students observe and/or assist an industry partner with a real coding project.
- **Standards 5.1-5.3 |** Invite an industry professional to discuss quality insurance.
- **Standards 6.1** | Have students job shadow a project manager.

Course Description

Coding 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.

Course Standards

1. Software Development Environments

1.1 Software Development Environments: 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*.

1.2 Software Creation Process: 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.

1.3 Software Creation Management: 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.

2. Software Development Life Cycle

2.1 Software Development Life Cycles: 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.

2.2 Development of Original Software: 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.

2.3 Best Practices Techniques: 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.

2.4 Management Tool: 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.

3. Designing Computer Applications

3.1 Programming Language Defense: 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.

3.2 Method of Data Processing: 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).

3.3 Specifications of Data Management: 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.

3.4 Development Cycle Preparation: 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.

4. Coding Computer Applications

4.1 Documentation: 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.

4.2 End-User Documentation: 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.

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

4.4 Programming Approaches: 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.

4.5 App Development: Design and **develop an app for a mobile computing device, using an online programming interface**, such as AppMakr, BuzzTouch, Appsbar, PhoneGap, or AppYet.

5. Software Testing Procedures & Quality Assurance

5.1 Quality Assurance: 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.

5.2 Code Optimization: **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.

5.3 Quality Testing: **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); and
d. acceptance (customer satisfaction).

6. Project Management

6.1 Programming Project: **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.