Programming and Software Development CTAG Alignments

This document contains information about four Career-Technical Articulation Numbers (CTANs) for Programming and Software Development Career-Technical Assurance Guide (CTAG). The CTANs are:

1. **Computer Logic**
2. **Java Programming**
3. **C++ Programming**
4. **Microsoft .NET Fundamentals (CTIT012)**

1. Computer Logic: CTAN alignment with the Tech Prep Programming and Software Development Pathway in the Career Field Technical Content Standards of the Ohio Department of Education

Course Description: This course introduces students to the concepts of logic in computer programming design. Students will use tools such as flowcharts and pseudocode to model problem solutions. The course will cover logic structures such as sequencing, selection and looping. Students will also learn about data types, arrays, and using variables for input/output operations. Data validation and program debugging techniques will also be covered.

Advising Notes:

- Student passage of Ohio Career-Technical Competency Assessment end of program assessment Semester Credit Hours: 3

Alignment:

Learning Outcomes **The student will be able to:**	**Outcomes and/or Competencies in ODE's Revised Career Field Technical Content Standards**

1. Describe the Process of Program Development.	5.1.1. Describe how computer programs and script can be used to solve problems (e.g., desktop, mobile, enterprise). 5.1.2. Explain how algorithms and data structures are used in information processing.

	5.1.4. Describe, compare, and contrast the basics procedural, structured, object-oriented (OO), and event-driven programming. 5.1.5. Describe the concepts of data management through programming languages.
2. Identify programming languages and their applications.	5.1.6. Analyze the strengths and weaknesses of different languages for solving a specific problem. 5.1.7. Compare and contrast the functions and operations of compilers and interpreters.
3. Use modeling tools such as using pseudocode and/or flowchart to solve programming problems.	5.1.3. Model the solution using both graphic tools (e.g., flowcharts) and pseudocode techniques. 5.6.7 Document a design using the appropriate tools (e.g., program flowchart, dataflow diagrams, Unified Modeling Language [UML]).

4. Identify data types and use variables for input and output operations and demonstrate the ability to create logical expressions and mathematical calculations.	**5.2 Computational and String Operations: Develop code that performs computational and string operations.** 5.2.1. Compare and contrast primitive types of numeric and nonnumeric data (e.g., integers, floats, Boolean, strings). 5.2.2. Identify the scope of data (e.g., global versus local, variables, constants, arrays). 5.2.3. Write code that uses arithmetic operations. 5.2.4. Write code that uses subtotals and final totals. 5.2.5. Write code that applies string operations (e.g., concatenation, pattern matching, substring).
5. Utilize data structures, such as an array, to store and manipulate a collection of related elements.	5.2.2 Identify the scope of data (e.g., global vs. local, variables, constants, arrays).
6. Identify and use conditional logic structures such as decision structures and loops.	**5.3 Logical Operations and Control Structures: Develop code that uses logical operations and control structures.** 5.3.1. Explain Boolean logic.

	5.3.2. Solve a truth table. 5.3.3. Write code that uses logical operators (e.g., and, or, not). 5.3.4. Write code that uses relational operators and compound conditions. 5.3.5. Write code that uses conditional control structures (e.g., if, if-then-else). 5.3.6. Write code that uses repetition control structures (e.g., while, for). 5.3.7. Write code that uses selection control structures (e.g., case, switch). 5.3.8. Write code that uses nested structures and recursion. 5.3.9. Write code that creates and calls functions. 5.3.10. Code error-handling techniques 5.3.11. Write code to access data repositories. 5.3.12. Write code to create classes, objects, and methods.
7. Describe and use errorchecking and data validation.	5.3.10 Code error-handling techniques 5.4.5 Test the program using defined test cases. 5.4.6 Correct syntax and runtime errors. 5.4.7 Debug logic errors. 5.5.1 Develop programs using data validation techniques. 5.6.14 Ensure code quality by testing and debugging the application (e.g. system testing, user acceptance testing). 5.6.17 Collect application feedback and maintain the application

8. Create program documentation.	5.1.8 Describe version control and the relevance of documentation. 5.6.8 Create documentation (e.g., implementation plan, contingency plan, data dictionary, user help).
9. Create and use functions and modules	5.3.9. Write code that creates and calls functions.

2. Java Programming: CTAN alignment with the Tech Prep Programming and Software Development Pathway in the Career Field Technical Content Standards of the Ohio Department of Education

Course Description: This course introduces object-oriented concepts such as instantiation, polymorphism, inheritance, and encapsulation. Students will learn how to create classes, objects and methods. Java data types, data structures, and events will be covered. Students will use Java to create console, desktop, and mobile applications.

Advising Notes:

- Student passage of Ohio Career-Technical Competency Assessment end of program assessment
- Recommended Prerequisite - Computer Logic
- Regarding Learning Outcome 8, please demonstrate that the course uses data structures and algorithms and program development.

Semester Credit Hours: 3
Alignment:

Learning Outcomes **The student will be able to:**	**Outcomes and/or Competencies in ODE's Revised Career Field Technical Content Standards**
1. Apply object oriented concepts to develop programs, including encapsulation, abstraction, inheritance, polymorphism, and interfaces.	5.1.4. Describe, compare, and contrast the basics of procedural, structured, object-oriented (OO), and event-driven programming.

2. Use development tools to develop programs.	**5.4 Integrated Development Environment: Build and test a program using an integrated development environment (IDE).** 5.4.1. Configure options, preferences, and tools. 5.4.2. Write and edit code in the IDE. 5.4.3. Compile or interpret a working program. 5.4.4. Define test cases. 5.4.5. Test the program using defined test cases. 5.4.6. Correct syntax and runtime errors. 5.4.7. Debug logic errors. 5.6.4 Identify a programming language, framework, and an integrated development environment (IDE). 5.6.7 Document a design using the appropriate tools (e.g., program flowchart, dataflow diagrams, Unified Modeling Language [UML]). 5.6.12 Compare and contrast software methodologies (e.g. agile, waterfall)

3. Create classes, objects and methods using an object oriented language	5.5.4 Develop programs that call other programs. 5.6.11 Develop the application. 5.3.12. Write code to create classes, objects, and methods.

4. Use primitive and reference data types in computational and string operations.	**5.2 Computational and String Operations: Develop code that performs computational and string operations.** 5.2.1. Compare and contrast primitive types of numeric and nonnumeric data (e.g., integers, floats, Boolean, strings). 5.2.2. Identify the scope of data (e.g., global versus local, variables, constants, arrays). 5.2.3. Write code that uses arithmetic operations. 5.2.4. Write code that uses subtotals and final totals. 5.2.5. Write code that applies string operations (e.g., concatenation, pattern matching, substring).
5. Use error checking and exception handling in program development.	5.4.5 Test the program using defined test cases.
6. Debug and test program code.	5.4.6 Correct syntax and runtime errors. 5.4.7 Debug logic errors. 5.6.14 Ensure code quality by testing and debugging the application (e.g. system testing, user acceptance testing).
7. Test and validate program output.	5.5.1 Develop programs using data validation techniques. 5.6.13 Perform code reviews (e.g. peer walkthrough, static analysis) 5.6.14 Ensure code quality by testing and debugging the application (e.g. system testing, user acceptance testing). 5.6.17 Collect application feedback and maintain the application.

8. Use data structures in program development	5.1.2 Explain how algorithms and data structures are used in information processing. 5.2.2 Identify the scope of data (e.g., global vs. local, variables, constants, arrays). 5.3.8 Write code that uses nested structures and recursion.
9. Use I/O methods to develop programs	5.6.5. Identify input and output (I/O) requirements. 5.6.6. Design system inputs, outputs, and processes.
10. Write executable object oriented source code.	5.4.1. Configure options, preferences, and tools. 5.4.2. Write and edit code in the IDE. 5.4.3. Compile or interpret a working program. 5.6.11 Develop the application. 5.6.16 Deploy the application.

3. C++ Programming: CTAN alignment with the Tech Prep Programming and Software Development Pathway in the Career Field Technical Content Standards of the Ohio Department of Education

Course Description: This course introduces object-oriented concepts such as instantiation, polymorphism, inheritance, and encapsulation. Students will learn how to create classes, objects, and member functions. C++ data types, pointers, structures, and arrays will be covered. Students will use C++ to create object oriented console programs.

Advising Notes:

- Student passage of Ohio Career-Technical Competency Assessment end of program assessment
- Recommended prerequisite: Computer Logic

Semester Credit Hours: 3
Alignment:

Proposed Learning Outcomes The student will be able to:	Outcomes and/or Competencies in ODE's Revised Career Field Technical Content Standards
1. Apply object-oriented concepts to develop programs.	5.1.4. Describe, compare, and contrast the basics of procedural, structured, object-oriented (OO), and event-driven programming.
2. Use development tools to develop programs.	**5.4 Integrated Development Environment: Build and test a program using an integrated development environment (IDE).** 5.4.1. Configure options, preferences, and tools. 5.4.2. Write and edit code in the IDE. 5.4.3. Compile or interpret a working program. 5.4.4. Define test cases. 5.4.5. Test the program using defined test cases. 5.4.6. Correct syntax and runtime errors. 5.4.7. Debug logic errors. 5.6.4 Identify a programming language, framework, and an integrated development environment (IDE). 5.6.7 Document a design using the appropriate tools (e.g., program flowchart, dataflow diagrams, Unified Modeling Language [UML]). 5.6.12 Compare and contrast software methodologies (e.g. agile, waterfall)
3. Create classes, objects and methods using an object oriented language	5.5.4 Develop programs that call other programs. 5.6.11 Develop the application.

	5.3.12. Write code to create classes, objects, and methods.
4. Use primitive and reference data types such as pointers in computational and string operations.	**5.2 Computational and String Operations: Develop code that performs computational and string operations.** 5.2.1. Compare and contrast primitive types of numeric and nonnumeric data (e.g., integers, floats, Boolean, strings). 5.2.2. Identify the scope of data (e.g., global versus local, variables, constants, arrays). 5.2.3. Write code that uses arithmetic operations. 5.2.4. Write code that uses subtotals and final totals. 5.2.5. Write code that applies string operations (e.g., concatenation, pattern matching, substring).
5. Use error checking and exception handling in program development.	5.4.5 Test the program using defined test cases.
6. Debug and test program code.	5.4.6 Correct syntax and runtime errors. 5.4.7 Debug logic errors. 5.6.14 Ensure code quality by testing and debugging the application (e.g. system testing, user acceptance testing).
7. Test and validate program output.	5.5.1 Develop programs using data validation techniques. 5.6.13 Perform code reviews (e.g. peer walkthrough, static analysis) 5.6.14 Ensure code quality by testing and debugging the application (e.g. system testing, user acceptance testing). 5.6.17 Collect application feedback and maintain the application.

8. Use data structures in program development	5.1.2 Explain how algorithms and data structures are used in information processing. 5.2.2 Identify the scope of data (e.g., global vs. local, variables, constants, arrays).

	5.3.8 Write code that uses nested structures and recursion.
9. Use logic structures to develop programs	**5.3 Logical Operations and Control Structures: Develop code that uses logical operations and control structures.** 5.3.1. Explain Boolean logic. 5.3.2. Solve a truth table. 5.3.3. Write code that uses logical operators (e.g., and, or, not). 5.3.4. Write code that uses relational operators and compound conditions. 5.3.5. Write code that uses conditional control structures (e.g., if, if-then-else). 5.3.6. Write code that uses repetition control structures (e.g., while, for). 5.3.7. Write code that uses selection control structures (e.g., case, switch). 5.3.8. Write code that uses nested structures and recursion. 5.3.9. Write code that creates and calls functions. 5.3.10. Code error-handling techniques 5.3.11. Write code to access data repositories. 5.3.12. Write code to create classes, objects, and methods.
10. Use I/O methods to develop programs	5.6.5. Identify input and output (I/O) requirements. 5.6.6. Design system inputs, outputs, and processes.

11. Produce object oriented source code	5.4.1. Configure options, preferences, and tools. 5.4.2. Write and edit code in the IDE.
	5.4.3. Compile or interpret a working program. 5.6.11 Develop the application. 5.6.16 Deploy the application.

4. Microsoft .NET Fundamentals (CTIT012): CTAN alignment with the Tech Prep Programming and Software Development Pathway in the Career Field Technical Content Standards of the Ohio Department of Education.

Course Description: This course uses Visual Basic .NET, as an object-oriented/event-driven environment in which to teach programming concepts. The student will use .NET applications to create and test windows based business programs.

Advising Notes:

- Course prepares students to take MTA Exam 98-372 (Microsoft .NET Fundamentals) or current equivalent.
- Student passage of Ohio Career-Technical Competency Assessment end of program assessment required for postsecondary credit.
- Recommended Prerequisite – Computer Logic
- The content of Learning Outcome #4 will be addressed in the recommended prerequisite—Computer Logic Credit Hours:

  3

Alignment:

Learning Outcomes The student will be able to:	**Outcomes and/or Competencies in ODE's Revised Career Field Technical Content Standards**

1. Use .NET framework concepts (i.e. basic application settings, variables and constants, basic control structures such as sequence, selection and iteration, event handling and error/exception handling)	5.2.2. Identify the scope of data (e.g., global vs. local, variables, constants, arrays). **5.3 Logical Operations and Control Structures: Develop code that uses logical operations and control structures.** 5.3.1. Explain Boolean logic. 5.3.2. Solve a truth table.
	5.3.3. Write code that uses logical operators (e.g. and, or, not). 5.3.4. Write code that uses relational operators and compound conditions. 5.3.5. Write code that uses conditional control structures (e.g., if, if-then-else). 5.3.6. Write code that uses repetition control structures (e.g., while, for). 5.3.7. Write code that uses selection control structures (e.g., case, switch). 5.3.8. Write code that uses nested structures and recursion. 5.3.9. Write code that creates and calls functions. 5.3.10. Code error-handling techniques 5.3.11.Write code to access data repositories. 5.3.12. Write code to create classes, objects, and methods.
2. Use namespaces, classes, methods and attributes in the .NET framework	5.3.8. Write code that uses nested structures and recursion. 5.3.9. Write code that creates and calls functions.
3. Compile .NET code	5.4.3. Compile or interpret a working program.

4. Use I/O classes in the .NET framework	4.6.5. Identify input and output (I/O) requirements. 4.6.6. Design system inputs, outputs, and processes.
5. Describe .NET security	3.2.1. Identify and implement data and application security.