# Florida Department of Education

# Student Performance Standards

**Course Title: Foundations of Programming**

**Course Number: 9007210**

**Course Credit: 1**

**Course Description:**

This course introduces concepts, techniques, and processes associated with computer programming and software development.

| **CTE Standards and Benchmarks** |
| --- |
| 1. Use oral and written communication skills in creating, expressing and interpreting information and ideas. The student will be able to:
 |
| * 1. Select and employ appropriate communication concepts and strategies to enhance oral and written communication in the workplace.
 |
| * 1. Locate, organize and reference written information from various sources.
 |
| * 1. Construct writings and/or communications using developmentally appropriate terminology.
 |
| * 1. Analyze the positive and negative impacts of technology on popular culture and personal life.
 |
| * 1. Discuss how technology has changed the way people build and manage organizations and how technology impacts personal life.
 |
| * 1. Evaluate ways in which adaptive technologies may assist users with special needs.
 |
| * 1. Explain how societal and economic factors are affected by access to critical information.
 |
| * 1. Discuss the challenges (e.g., political, social, and economic) in providing equal access and distribution of technology in a global society.
 |
| 1. Explore the characteristics, tasks, work attributes, options, and tools associated with a career in software development. The student will be able to:
 |
| * 1. Explore a variety of careers to which computing is central.
 |
| * 1. Discuss the impact of computing on business and commerce (e.g., automated inventory processing, financial transactions, e-commerce, virtualization, and cloud computing).
 |
| * 1. Evaluate the impacts of irresponsible use of information (e.g., plagiarism and falsification of data) on collaborative projects.
 |
| * 1. Identify tasks performed by programmers.
 |
| * 1. Describe how businesses use computer programming to solve business problems.
 |
| * 1. Investigate job opportunities in the programming field.
 |
| * 1. Explain different specializations and the related training in the computer programming field.
 |
| * 1. Explain the need for continuing education and training of computer programmers.
 |
| * 1. Understand and identify ways to use technology to support lifelong learning.
 |
| * 1. Explain software as a service (SaaS) and how it impacts business.
 |
| * 1. Describe ethical responsibilities of computer programmers.
 |
| * 1. Identify credentials and certifications that may improve employability for a computer programmer.
 |
| * 1. Identify devices, tools, and other environments for which programmers may develop software.
 |
| 1. Demonstrate an understanding of the characteristics, use, and selection of numerical, non-numerical, and logical data types. The student will be able to:
 |
| * 1. Identify the characteristics (e.g., size, limits) and uses of different numerical and non-numerical data types.
 |
| * 1. Explain the types and uses of variables in programs.
 |
| * 1. Determine the best data type to use for given programming problems.
 |
| * 1. Compare and contrast simple data structures and their uses.
 |
| * 1. Identify the types of operations that can be performed on different data types (e.g., math operations on numerical data types, concatenation, and other string operations).
 |
| * 1. Evaluate arithmetic and logical expressions using appropriate operator precedence.
 |
| * 1. Explain how computers store different data types in memory.
 |
| * 1. Demonstrate the difference between "data" and "information".
 |
| * 1. Use different number systems to represent data.
 |
| * 1. Explain how national and international standards (i.e., ASCII, UNICODE) are used to represent non-numerical data.
 |
| * 1. Use Boolean logic to perform logical operations using Boolean algebra and truth tables.
 |
| 1. Distinguish between iterative and non-iterative program control structures. The student will be able to:
 |
| * 1. Identify the uses of non-iterative and iterative programming structures using pseudocode and flowcharts.
 |
| * 1. Create iterative programming structures and their uses.
 |
| * 1. Explain how sequence, selection, and iteration are building blocks of algorithms.
 |
| 1. Describe the processes, methods, and conventions for software development and maintenance. The student will be able to:
 |
| * 1. Describe a software development process that is used to solve problems at different software development stages.
 |
| * 1. Define alternative methods of program development (e.g., rapid prototyping, waterfall, spiral model, peer coding).
 |
| * 1. List and explain the steps in the program development cycle.
 |
| * 1. Describe different types of documentation used in the program development cycle (*e.g.*, requirements document, program design documents, test plans).
 |
| * 1. Describe different methods used to facilitate version control.
 |
| 1. Explain the types, uses, and limitations of testing for ensuring quality control. The student will be able to:
 |
| * 1. Explain the uses and limits of testing in ensuring program quality.
 |
| * 1. Explain testing performed at different stages of the program development cycle (*e.g.,* unit testing, system testing, user acceptance testing).
 |
| * 1. Describe and identify types of programming errors.
 |
| 1. Create a program design document using common design tool. The student will be able to:
 |
| * 1. Describe different design methodologies and their uses (*e.g.*, object-oriented design, structured design, rapid application development).
 |
| * 1. Describe and use tools for developing a program design (*e.g.*, flowcharts, design documents, pseudocode).
 |
| * 1. Explain the role of existing libraries and packages in facilitating programmer productivity.
 |
| * 1. Participate and contribute to a design review of a program design developed using a common program design tool (*e.g.*, UML, flowcharts, design documents, pseudocode).
 |
| * 1. Develop a software artifact (independently and collaboratively) in phases (or stages) according to a common software development methodology (e.g., Waterfall or Spiral model).
 |
| * 1. Define input and output for a program module using standard design methodology.
 |
| 1. Solve problems using critical thinking skills, creativity and innovation. The student will be able to:
 |
| * 1. Employ critical thinking skills independently and in teams to solve problems and make decisions.
 |
| * 1. Employ critical thinking and collaborative skills to resolve conflicts.
 |
| * 1. Identify and document workplace performance goals and monitor progress toward those goals.
 |
| * 1. Conduct technical research to gather information necessary for decision-making.
 |
| * 1. Discuss digital tools or resources to use for a real-world task based on their efficiency and effectiveness, individually and collaboratively.
 |
| 1. Describe the importance of security and privacy information sharing, ownership, licensure and copyright. The student will be able to:
 |
| * 1. Describe security and privacy issues that relate to computer networks including the permanency of data on the Internet, online identity, and privacy.
 |
| * 1. Discuss the impact of government regulation on privacy and security.
 |
| * 1. Describe how different types of software licenses (e.g., open source and proprietary licenses) can be used to share and protect intellectual property.
 |
| * 1. Explain how access to information may not include the right to distribute the information.
 |
| * 1. Describe differences between open source, freeware, and proprietary software licenses, and how they apply to different types of software.
 |
| * 1. Discuss security and privacy issues that relate to computer networks.
 |
| * 1. Identify computer-related laws and analyze their impact on digital privacy, security, intellectual property, network access, contracts, and harassment.
 |
| 1. Create programs that solve a problem using non-iterative and iterative algorithms. The student will be able to:
 |
| * 1. Apply the developmental cycle methodologies to create a program.
 |
| * 1. Develop a program using string and/or numeric data types.
 |
| * 1. Develop a program using sequential algorithms.
 |
| * 1. Develop a program using selection structures.
 |
| * 1. Develop a program using looping structures.
 |