Maryland CTE Programs of Study

Computer Programming II - Swift (2025)

GENERAL REQUIREMENTS. This course is recommended for students in Grades 10-12. 1

1 GENERAL REQUIREMENTS. This course is recommended for students in Grades 10-12. 1

INTRODUCTION 2

A Career and Technical Education (CTE) instruction provides content aligned with challenging academic standards and relevant technical knowledge and skills for students to further their education and succeed in current or emerging professions. 2.A

B The Digital Technology Career Cluster focuses on building linkages in IT occupations for entry level, technical, and professional careers related to the design, development, support, and management of hardware, software, multimedia, and systems integration services. 2.B

C The Computer Programming Program of Study provides high school students with comprehensive software development skills through mastery of Python and Swift programming languages. Students' progress from fundamental coding concepts to complex development projects while earning industry certifications and building professional portfolios. Graduates emerge prepared for both college computer science programs and immediate entry into software development careers, equipped with practical experience in current programming practices and software engineering principles. 2.C

D Computer Programming II – Swift introduces students to iOS app development using the Swift programming language. Students will navigate Xcode, master Swift fundamentals, and build fully functional iOS applications through hands-on projects. They'll learn essential skills including variables, functions, data structures, interface design with Interface Builder and Swift, multi-screen navigation, and debugging techniques. Following Apple's official curriculum, the course prepares students for the App Development with Swift Certified User certification, demonstrating their ability to create iOS apps that adhere to standard practices with proper UI elements, layouts, and navigation interfaces. 2.D

Skip to main content

tudents will participate in at least two Career-Connected Education and Work-ased Learning experiences in this course, which might include informational

interviews or job shadowing relevant to the program of study. 2.E

F **Students are encouraged to participate in extended learning experiences through aligned Career and Technical Student Organizations (CTSOs). CTSOs are a cocurricular requirement in the Carl D. Perkins Act, and alignment to CTSO activities is an expectation for CTE programs in the state of Maryland.** 2.F

KNOWLEDGE AND SKILLS 3

A **The student demonstrates the necessary skills for career development, maintenance of employability, and successful completion of course outcomes. The student is expected to:** 3.A

1. Identify and demonstrate positive work behaviors that enhance employability and job advancement, such as regular attendance, promptness, proper attire, maintenance of a clean and safe work environment, and pride in work. 3.A.1
2. Demonstrate positive personal qualities such as flexibility, open-mindedness, initiative, active listening, and a willingness to learn. 3.A.2
3. Employ effective reading, writing, and technical documentation skills. 3.A.3
4. Solve problems using critical thinking techniques and structured troubleshooting methodologies. 3.A.4
5. Demonstrate leadership skills and collaborate effectively as a team member. 3.A.5
6. Implement safety procedures, including proper handling of hardware and following cybersecurity guidelines. 3.A.6
7. Exhibit an understanding of legal and ethical responsibilities in the IT field, following data privacy laws and best practices for security. 3.A.7
8. Demonstrate time-management skills and the ability to prioritize tasks in a technical setting. 3.A.8

B **The student identifies various career pathways in the computer programming field. The student is expected to:** 3.B

1. Develop a career plan that includes the necessary education, certifications, job skills, and experience for specific roles in computer programming. 3.B.1
2. Create a professional resume and portfolio that reflect skills, projects, certifications, and recommendations. 3.B.2
3. Demonstrate effective interview skills for roles in IT and computer programming. 3.B.3

C **The student develops technology and digital literacy skills. The student is expected to:** 3.C

1. Use technology as a tool for research, organization, communication, and problem-solving. 3.C.1
2. Use digital tools, including computers, mobile devices, collaboration platforms, and cloud services, to access, manage, and create information. 3.C.2

Skip to main content

3 Demonstrate proficiency in using emerging and industry-standard technologies, including virtualization tools, network management software, and cybersecurity applications. 3.C.3

4 Understand ethical and legal considerations for technology use, including the principles of data protection, copyright, and responsible technology use. 3.C.4

D The student integrates core academic skills into computer programming practices. The student is expected to: 3.D

1 Demonstrate the use of clear communication techniques, both written and verbal, that are consistent with industry standards. 3.D.1

2 Evaluate algebraic functions and equations, implement Boolean algebra and logic, apply basic statistics for data analysis, utilize different number systems (binary, decimal), calculate percentages and proportions, and employ basic geometry for graphics problems. 3.D.2

3 Apply systematic investigation and controlled testing methodologies. Analyze data collection methods and pattern recognition. Implement documentation processes for replicable results. 3.D.3

E The student can describe the iOS app development ecosystem. The student is expected to: 3.E

1 Explain what Swift is and how it's used in iOS development. 3.E.1

2 Identify the components of the Apple development platform. 3.E.2

3 Describe the process of creating, testing, and deploying iOS apps. 3.E.3

4 Understand the purpose of the App Store and app distribution methods. 3.E.4

5 Recognize how Swift relates to Apple's hardware and software ecosystem. 3.E.5

F The student can set up and navigate the development environment. The student is expected to: 3.F

1 Install and configure Xcode on a Mac computer. 3.F.1

2 Navigate the Xcode workspace including the navigator, editor, and inspector areas. 3.F.2

3 Create, open, and manage Swift projects and playgrounds. 3.F.3

4 Use the iOS simulator to test applications. 3.F.4

5 Access documentation and resources within Xcode. 3.F.5

G The student can understand and apply Swift programming fundamentals. The student is expected to: 3.G

1 Declare and use basic Swift types (Int, Double, String, Bool). 3.G.1

2 Demonstrate when to use constants (let) versus variables (var). 3.G.2

3 Apply arithmetic, comparison, and logical operators to create expressions. 3.G.3

4 Interpret and predict the results of Swift expressions. 3.G.4

Skip to main content

5 Use Swift playgrounds to experiment with code and see immediate results. 3.G.5

H The student can implement control flow in Swift programs. The student is expected to: 3.H

1 Create decision structures using if, else if, and else statements. 3.H.1

2 Apply logical operators (&&, ||, !) to form compound conditions. 3.H.2

3 Use switch statements to handle multiple potential states. 3.H.3

4 Implement the guard statement for early exits from functions. 3.H.4

5 Develop programs that respond to different conditions. 3.H.5

I The student can use loops for repetitive tasks in Swift. The student is expected to: 3.I

1 Create for-in loops to iterate through ranges and collections. 3.I.1

2 Implement while loops for condition-based execution. 3.I.2

3 Apply loop control statements (break, continue) to control execution flow. 3.I.3

4 Iterate through characters in strings and elements in collections. 3.I.4

5 Choose appropriate loop structures for specific programming tasks. 3.I.5

J The student can manage data using Swift collection types. The student is expected to: 3.J

1 Create and initialize arrays to store ordered collections of data. 3.J.1

2 Access and modify array elements using index notation. 3.J.2

3 Perform common array operations (append, insert, remove, count). 3.J.3

4 Create and use dictionaries to store key-value pairs. 3.J.4

5 Access and modify dictionary values using key subscripting. 3.J.5

K The student can define and use functions in Swift. The student is expected to: 3.K

1 Create functions with parameters and return values. 3.K.1

2 Implement functions with default parameter values. 3.K.2

3 Customize internal and external parameter names in functions. 3.K.3

4 Understand function scope and variable lifetime. 3.K.4

5 Organize code using functions for reusability and maintainability. 3.K.5

L The student can work with optional types in Swift. The student is expected to: 3.L

1 Explain the purpose of optionals in Swift programming. 3.L.1

2 Unwrap optionals safely using if let and guard let statements. 3.L.2

3 Implement optional binding to safely access optional values. 3.L.3

Apply optional chaining to access properties and methods of optional values. 3.L.4

Use nil coalescing operators to provide default values for optionals. 3.L.5

Skip to main content

M **The student can create and use Swift structures and classes. The student is expected to:** 3.M

1. Define structures with properties and methods. 3.M.1
2. Initialize structures using memberwise and custom initializers. 3.M.2
3. Define classes with properties and methods. 3.M.3
4. Differentiate between structures and classes in Swift. 3.M.4
5. Implement property observers (willSet, didSet). 3.M.5

N **The student can debug Swift code effectively. The student is expected to:** 3.N

1. Set breakpoints to pause execution at specific points in code. 3.N.1
2. Use the debugger to step through code line by line. 3.N.2
3. Inspect variable values during debugging sessions. 3.N.3
4. Implement logging to track program execution. 3.N.4
5. Diagnose and fix common errors in Swift programs. 3.N.5

O **The student can create user interfaces with Interface Builder. The student is expected to:** 3.O

1. Add and configure UI elements from the Object Library. 3.O.1
2. Set properties for UI components using the Attributes Inspector. 3.O.2
3. Create connections between Interface Builder and Swift code using outlets and actions. 3.O.3
4. Design interfaces using storyboards and view controllers. 3.O.4
5. Implement Auto Layout for responsive interfaces. 3.O.5

P **The student can create simple Swift views. The student is expected to:** 3.P

1. Understand the declarative syntax of Swift. 3.P.1
2. Create and customize basic Swift views. 3.P.2
3. Apply modifiers to change the appearance and behavior of views. 3.P.3
4. Implement stacks (VStack, HStack, ZStack) to arrange views. 3.P.4
5. Create simple layouts that adapt to different screen sizes. 3.P.5

Skip to main content

Q **The student can build navigation and multi-screen apps. The student is expected to:** 3.Q

1. Implement navigation between screens using NavigationStack and NavigationLink. 3.Q.1
2. Present modal views using sheets. 3.Q.2
3. Create tab-based interfaces for organizing app content. 3.Q.3
4. Pass data between different screens in an app. 3.Q.4
5. Design user flows that follow iOS interface guidelines. 3.Q.5

R **The student can complete guided app projects to build practical skills. The student is expected to:** 3.R

1. Build a simple flashlight app (Light) that demonstrates basic UI interaction. 3.R.1
2. Create a word-guessing game app (Apple Pie) using UIKit components and Auto Layout. 3.R.2
3. Develop a personality quiz app that showcases navigation between screens. 3.R.3
4. Plan, prototype, and implement a personal app project. 3.R.4
5. Document the development process and design decisions. 3.R.5