# General Computer Science

The Computer Science program provides students with the principles of computer science and programming. Areas of study include methodology, algorithms, data structures and object-oriented programming. Java, Python, javascript and C++ are the primary languages taught.

## Introduction

The competencies in this document are designed to clearly state what the student should know and be able to do upon completion of an advanced high school General Computer Science program. These standards are designed for a two course sequence that prepares the student for technical assessments directly aligned to the standards.

These exit-level competencies are designed for the student to complete all competencies through their completion of a program of study. These competencies are intended to guide curriculum objectives for a program of study.

The standards are organized as follows:

* Competencies are general statements that identify major areas of knowledge, understanding, and the skills students are expected to learn in key subject and career areas by the end of the program.
* Standards follow each content standard. Standards identify the more specific components of each Competency and define the expected abilities of students within each Competency.
* Learning Targets are very specific criteria statements for determining whether a student meets the Standard. Learning Targets may also be used as learning outcomes, which teachers can identify as they plan their program learning objectives.

The crosswalk and alignment section of the document shows where the Learning Targets support the New Hampshire Content Standards. Where correlation with an academic content standard exists, students in the General Computer Science program perform learning activities that support, either directly or indirectly, achievement of the academic Competencies that are listed.

All students are encouraged to participate in the career and technical student organization (CTSO) that relates to the General Computer Science program. CTSOs are co-curricular national organizations that directly reinforce learning in the CTE classroom through curriculum resources, competitive events, and leadership development. CTSOs provide students the ability to apply academic and technical knowledge, develop communication and teamwork skills, and cultivate leadership skills to ensure college and career readiness.

The Employability Skills for Career Readiness identifies the “soft skills” needed to be successful in all careers and must be taught as an integrated component of all CTE course sequences. These Competencies are available in a separate document.

The Standards Reference Code is only used to identify or align Learning Targets listed in the Competency to daily lesson plans, curriculum documents, or national standards. The Standards Reference Code is an abbreviated name for the program, and the Competency, Standard, and Learning Targets are referenced in the program Competency. This abbreviated code for identifying competencies uses each of these items. For example, GCS is the Standards Reference Code for General Computer Science. For Competency 2, Standard 3 and Learning Target 4 the Standards Reference Code would be GCS.2.3.4.

# Algorithms and Programming

Create meaningful and efficient programs including choosing which information to use and how to process and store it, breaking apart large problems into smaller ones, recombining existing solutions, and analyzing different solutions.

## 1.0 Program Design

### 1.1 Develop A Program

1.1.1 Implement the steps in the System Development Life Cycle (SDLC) (e.g., planning, analysis, design, development, testing, implementation, and maintenance)

1.1.2 Develop program requirements/specifications and a testing plan (e.g., user stories, automated testing, and test procedures)

1.1.3 Apply pseudocode or graphical representations to plan the structure of a program or module (e.g., flowcharting, white boarding, and UML)

1.1.4 Use a program editor to enter and modify code

1.1.5 Identify correct input/output statements

1.1.6 Choose the correct method of assigning input to variables including data sanitization

1.1.7 Choose the correct method of outputting data with formatting and escaping

1.1.8 Differentiate between interpreted and compiled code (e.g., steps necessary to run executable code)

1.1.9 Identify the purpose of a build system (e.g., make, rake, ant, maven, SCons, and grunt)

1.1.10 Name identifiers and formatting code by applying recognized conventions

### 1.2 Utilize And Create Community Resources

1.2.1 Identify comments

1.2.2 Utilize comments to describe sections of code

1.2.3 Use standard library functions

1.2.4 Find and use third party libraries (e.g., web-based and package managers)

1.2.5 Explain and interact with an Application Program Interface (API)

1.2.6 Apply industry standards in documentation (e.g., self-documenting code; function-level, program-level, and user-level documentation)

### 1.3 Test And Debug To Verify Program Operation

1.3.1 Identify errors in program modules

1.3.2 Identify boundary cases and generate appropriate test data

1.3.3 Perform integration testing including tests within a program to protect execution from bad input or other run-time errors

1.3.4 Categorize, identify, and correct errors in code, including syntax, semantic, logic, and runtime

1.3.5 Perform different methods of debugging (e.g., hand-trace code and real time debugging tools)

### 1.4 Apply User Design Principles To Include Websites And Applications

1.4.1 Apply W3C standards and style conventions

1.4.2 Construct web pages and applications that are compliant with ADA and sections 504 and 508 standards

1.4.3 Explain the concept of responsive design and applications

1.4.4 Employ graphics methods to create images at specified locations

1.4.5 Choose correct GUI objects for input and output of data to the GUI interface (e.g., text boxes, labels, radio buttons, check boxes, dropdowns, and list boxes)

## 2.0 Data Structures

### 2.1 Utilize Primitive Data Types And Strings In Writing Programs

2.1.1 Compare and contrast fundamental data structures and their uses

2.1.2 Declare numeric, Boolean, character, string variables, and float and double

2.1.3 Choose the appropriate data type for a given situation

2.1.4 Identify the correct syntax and usage for constants and variables in a program

2.1.5 Identify the correct syntax and safe functions for operations on strings, including length, substring, and concatenation

2.1.6 Explain complications of storing and manipulating data (i.e., the Big-O notation for analyzing storage and efficiency concerns, etc.)

2.1.7 Research industry relevant programming languages (i.e., Java, JavaScript, Python, etc.)

### 2.2 Utilize Basic Data Structures In Writing Programs

2.2.1 Demonstrate basic uses of arrays including initialization, storage, and retrieval of values

2.2.2 Distinguish between arrays and hash maps (associative arrays)

2.2.3 Identify techniques for declaring, initializing, and modifying user-defined data types

2.2.4 Create and use two-dimensional arrays

### 2.3 Employ Object-Oriented Programming Techniques

2.3.1 Make a distinction between an object and a class

2.3.2 Differentiate among inheritance, composition, and class relationships

2.3.3 Instantiate objects from existing classes

2.3.4 Read the state of an object by invoking accessor methods

2.3.5 Change the state of an object by invoking a modifier method

2.3.6 Determine the requirements for constructing new objects by reading the documentation

2.3.7 Create a user-defined class

2.3.8 Create a subclass of an existing class

2.3.9 Identify the use of an abstract class as opposed to an interface

2.3.10 Explain the object-oriented concepts of polymorphism, inheritance, and encapsulation

## 3.0 Modularity

### 3.1 Utilize Conditional Structures In Writing Programs

3.1.1 Use the correct syntax for decision statements (e.g., if/else, if, and switch case)

3.1.2 Compare values using relational operators (e.g., =, >, <, >=, <=, and not equal)

3.1.3 Evaluate Boolean expressions (e.g., AND, OR, NOT, NOR, and XOR)

3.1.4 Use the correct nesting for decision structures

### 3.2 Breakdown Programs In Smaller, More Manageable Parts

3.2.1 Demonstrate refactoring techniques to reduce repetitive code and improve maintainability

3.2.2 Demonstrate the use of parameters to pass data into program modules

3.2.3 Demonstrate the use of return values from modules

3.2.4 Write code that creates and calls functions.

3.2.5 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.

### 3.3 Utilize Iterative Structures In Writing Programs

3.3.1 Identify various types of iteration structure (e.g., while, for, for-each, and recursion)

3.3.2 Identify how loops are controlled (variable conditions and exits)

3.3.3 Use the correct syntax for nested loops

3.3.4 Compute the values of variables involved with nested loops

## 4.0 Algorithms

### 4.1 Perform Basic Computer Mathematics In Information Technology

4.1.1 Identify and correctly use arithmetic operations applying the order of operations (precedence) with respect to programming

4.1.2 Interpret and construct mathematical formulas

4.1.3 Create and implement basic algorithms

### 4.2 Perform Search Algorithms

4.2.1 Search data in an array  
4.2.2 Describe the efficiency of linear vs. binary searches [e.g., O(n) and O(log n)]

### 4.3 Perform Sort Algorithms

4.3.1 Sort data in an array

4.3.2 Describe the efficiency of different sorting algorithms (e.g., bubble, insertion, and merge)

# Data and Analysis

Synthesize concepts, practices and processes of data collection, resource management, and techniques to different types of data in order to discover useful information that can communicate storytelling and to inform decision-making.

## 5.0 Data and Analysis

### 5.1 Utilize Storage And Understand How It Functions In A Computer System

5.1.1 Translate between different bit representations of real-world phenomena, such as characters, numbers, and images, e.g., convert hexadecimal colors to decimal percentages, ASCII/Unicode representation

5.1.2 Evaluate the tradeoffs in how data elements are organized and where data is stored

5.1.3 Demonstrate the ability to store bit representation of real-world phenomena, characters, numbers, and images

### 5.2 Collect, Visualize, And Transform Data

5.2.1 Create interactive data visualizations or alternative representations using software tools to help others better understand real-world phenomena

5.2.2 Use data analysis tools and techniques to identify patterns in data representing complex systems

5.2.3 Select data collection tools and techniques to generate data sets that support a claim or communicate information

### 5.3 Apply Inference And Models To A Data Set

5.3.1 Create computational models that represent the relationships among different elements of data collected from a phenomenon, process, or model

5.3.2 Evaluate the ability of models and simulations to test and support the refinement of hypotheses

# Networks and the Internet

Apply networking concepts, using various models to implement protocols and standards when moving data. Design systems with working switching and routing "packets" to ensure data flows to the correct destination. Ensure data traffic flows through the internet effectively.

## 6.0 Networks and Internet

### 6.1 Use Safety Procedures and Proper Tools

6.1.1 Demonstrate the proper use of safety devices

6.1.2 Research the environmental impact of production

6.1.3 Research local, state, and federal regulations related to material handling

6.1.4 Demonstrate secure disposal of technology materials

6.1.5 Explain the relationship between organization and safety

6.1.6 Demonstrate an organized work environment

6.1.7 Demonstrate electrical safety (e.g., grounding, ESD (static), etc.)

6.1.8 Apply installation safety (e.g., lifting, overhead movements, etc.)

### 6.2 Explain Network System Hardware

6.2.1 Define industry standard vocabulary

6.2.2 Identify internetworking equipment

6.2.3 Identify various networking topologies

6.2.4 Differentiate between various network transmission media

6.2.5 Describe the use of each of the classifications of hardware components

6.2.6 Compare and configure network devices

### 6.3 Explain System Network Protocols

6.3.1 Describe the OSI model and relate to hardware in a network

6.3.2 Implement the appropriate industry policy and procedures

6.3.3 Compare and contrast the ports and protocols (HTTP, NetBIOS, SMTP, TCP, UDP, etc.)

6.3.4 Configure and apply appropriate ports and protocols (FTP, SSH, Telnet, DHCP, TFTP, etc.)

6.3.5 Utilize appropriate wired connections

6.3.6 Utilize appropriate wireless connections

6.3.7 Describe encapsulation/de-encapsulation

6.3.8 Apply numbering systems (e.g., binary, octal, hexadecimal)

6.3.9 Demonstrate addressing and subnetting techniques

# Computing Systems

Apply concepts of physical components and software that make up a computing system which communicate and process information in digital form, along with practices and methodology for troubleshooting issues in those systems.

## 7.0 Computing Systems

### 7.1 Explain Computing System Hardware

7.1.1 Categorize system unit components (e.g., power supply connectors, motherboard characteristics, form factors, CPU features, memory module attributes, and expansion business types)

7.1.2 Use industry standard vocabulary to identify components

7.1.3 Configure the settings of basic hardware components

7.1.4 Categorize audio and video device components, connectors, and cables

7.1.5 Configure display and video settings

7.1.6 Identify external device components, connectors, and cables

7.1.7 Connect external devices using the appropriate connectors and cables

7.1.8 Describe various levels and types of memory and storage devices

### 7.2 Explain Computing System Software

7.2.1 Use industry standard vocabulary in relation to operating systems (OS)

7.2.2 Explain various features and tools of operating systems

7.2.3 Select appropriate operating system features and tools based on customer requirements

7.2.4 Configure Windows Update Settings

7.2.5 Use Windows Control Panel utilities

7.2.6 Perform common preventive maintenance procedures using the appropriate Windows OS tools

### 7.3 Apply Computing System Troubleshooting Techniques

7.3.1 Troubleshoot basic hardware components and resolve issues

7.3.2 Troubleshoot common PC security issues using best practices

7.3.3 Explain key terms and acronyms used in diagnostic testing and troubleshooting

7.3.4 Identify common symptoms for a given discrepancy

7.3.5 Develop a solution for a given discrepancy

7.3.6 Document the solution

# Cybersecurity

Prove how to detect, prevent and mitigate threats in order to secure a computing system or network in an ethical manner, and in accordance with international, federal,state, local and cyber laws and regulations.

## 8.0 Cybersecurity

### 8.1 Evaluate Network, Communication and Organization

8.1.1 Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing

8.1.2 Describe the issues that impact network functionality, e.g., bandwidth, load, delay, topology

### 8.2 Recognize Security Issues

8.2.1 Illustrate how sensitive data can be affected by malware and other attacks

8.2.2 Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts

8.2.3 Compare various security measures, considering tradeoffs between the usability and security of a computing system

8.2.4 Explain trade offs when selecting and implementing cybersecurity recommendations

8.2.5 Compare ways software developers protect devices and information from unauthorized access

### 8.3 Examine Legal And Ethical Issues Related To Information Technology

8.3.1 Analyze legal issues in technology

8.3.2 Evaluate intellectual property laws

8.3.3 Explain differences between licensing, copyright, and infringement

8.3.4 Explain the differences between restricted content, prohibited or illegal content

8.3.5 Examine state, federal, and international regulations related to technology (e.g.; legal holds, disposal methods, data retention, discoverability, data protection, etc.)

8.3.6 Analyze acceptable use policies

8.3.7 Explain the difference between technology policies, privacy standards, and best practices

8.3.7 Explain data and privacy encryption issues related to using technology

# Computer Science Curriculum Framework

## Program of Study

The program of study illustrates the sequence of academic and career and technical education coursework that is necessary for the student to successfully transition into postsecondary educational opportunities and employment in their chosen career path.

## Program Structure

The core course sequencing provided in the following table serves as a guide to schools for their programs of study. Each course is listed in the order in which it should be taught and has a designated level. Complete program sequences are essential for the successful delivery of all state standards in each program area.

**Computer Science I**

**Computer Science II**

**Computer Science III (optional)**

The core course sequencing with the complementary courses provided in the following table serves as a guide to schools for their programs of study. Each course is listed in the order in which it should be taught and has a designated level. A program does not have to utilize all of the complementary courses in order for their students to complete their program of study. Complete program sequences are essential for the successful delivery of all state standards in each program area.

## Computer Science I

This course will introduce students to the essential ideas of computer science and show how computing and technology can influence the world. This course focuses on technology and programming as a means to solve computational problems and find creative solutions. The appropriate use of technology and industry-standard equipment is an integral part of this course.

### Technical Competencies

**1.0 Program Design**

1.1.1 - 1.1.10 Develop A Program

1.3.1 - 1.3.5 Test And Debug To Verify Program Operation

1.4.1 - 1.4.5 Apply User Design Principles To Include Websites And Applications

**2.0 Data Structures**

2.1.1 - 2.1.7 Utilize Primitive Data Types And Strings In Writing Programs

2.2.1 - 2.1.4 Utilize Basic Data Structures In Writing Programs

**3.0 Modularity**

3.1.1 - 3.1.4 Utilize Conditional Structures In Writing Programs

3.2.1 - 3.2.5 Breakdown Programs In Smaller, More Manageable Parts

3.3.1 - 3.3.4 Utilize Iterative Structures In Writing Programs

**4.0 Algorithms**

4.1.1 - 4.1.3 Perform Basic Computer Mathematics In Information Technology

**5.0 Data and Analysis**

5.1.1 - 5.1.3 Utilize Storage And Understand How It Functions In A Computer System

**6.0 Networks and Internet**

6.1.1 - 6.1.8 Use Safety Procedures and Proper Tools

6.2.1 - 6.2.6 Explain Network System Hardware

6.3.1 - 6.3.9 Explain System Network Protocols

**7.0 Computing Systems**

7.1.1 - 7.1.8 Explain Computing System Hardware

7.3.1 - 7.3.6 Apply Computing System Troubleshooting Techniques

**8.0 Cybersecurity**

8.2.1 - 8.2.5 Recognize Security Issues

8.3.1 - 8.3.7 Examine Legal And Ethical Issues Related To Information Technology

### CTE Professionalism and IT Essentials Competencies

Terminology and Communications

Tools and Equipment

Project Management

Applied Mathematics

Safety

## Computer Science II

This course follows The College Board Advanced Placement curriculum and prepares students for the AP Computer Science Principles exam. This course will introduce students to the essential ideas of computer science and show how computing and technology can influence the world. This course focuses on technology and programming as a means to solve computational problems and find creative solutions. Students will creatively address real-world issues and concerns while using the same processes and tools as artists, writers, computer scientists, and engineers to bring ideas to life. The appropriate use of technology and industry standard equipment is an integral part of this course.

### Technical Competencies

**1.0 Program Design**

1.2.1 - 1.2.6 Utilize And Create Community Resources

**2.0 Data Structures**

2.3.1 - 2.3.10 Employ Object-Oriented Programming Techniques

**3.0 Modularity**

3.1.1 - 3.1.4 Utilize Conditional Structures In Writing Programs

3.2.1 - 3.2.5 Breakdown Programs In Smaller, More Manageable Parts

3.3.1 - 3.3.4 Utilize Iterative Structures In Writing Programs

**4.0 Algorithms**

4.2.1 - 4.2.2 Perform Search Algorithms

4.3.1 - 4.3.2 Perform Search Algorithms

**5.0 Data and Analysis**

5.2.1 - 5.2.3 Collect, Visualize, And Transform Data

5.3.1 - 5.3.2 Apply Inference And Models To A Data Set

**7.0 Computing Systems**

7.2.1 - 7.2.6 Explain Computing System Software

7.3.1 - 7.3.6 Apply Computing System Troubleshooting Techniques

**8.0 Cybersecurity**

8.1.1 - 8.1.2 Evaluate Network, Communication and Organization

### CTE Professionalism and IT Essentials Competencies

Terminology and Communications

Tools and Equipment

Project Management

Applied Mathematics

Safety

## Computer Science III

This course is a continuation of Computer Science I or AP Computer Science Principles. This course provides intermediate computer science students with instruction in advanced techniques and processes, particularly as it relates to the language of Java. The areas of major emphasis in the course will be on object-oriented programming methodology, algorithms, data structures and ethics. Topics will include program design, program implementation, standard data structures, and standard algorithms. The appropriate use of technology and industry-standard equipment is an integral part of this course.

### Technical Competencies

Students have achieved all program content standards and will pursue advanced study through investigation and indepth research

### CTE Professionalism and IT Essentials Competencies

Students have achieved all program content standards and will pursue advanced study through investigation and indepth research.

### Sample Topics

* Internship
* Capstone Project
* Portfolio
* Class Project Manager
* Teaching Assistant
* CTSO Leadership