

Computer Programming I & II*

Career Cluster	Information Technology
Course Code	10152
Prerequisite(s)	Computer Applications, Introduction to Information Technology Careers (recommended), Computer Hardware & Software (recommended)
Credit	.5-1
Program of Study and Sequence	Computer Programming or a dual credit equivalent is required for the Programming Pathway and recommended for the Networking & Hardware Pathway
Student Organization	SkillsUSA, Future Business Leaders of America (FBLA), CyberPatriots
Coordinating Work-Based Learning	Job Shadowing, Tours, Informational Interviews, Internships
Industry Certifications	None
Dual Credit or Dual Enrollment	TBD
Teacher Certification	K-12 Educational Technology Endorsement, K-12 Classroom Technology
Resources	

Course Description:

Computer Programming I introduces students to the fundamentals of computer programming. Students will learn to design, code, and test their own programs while applying mathematical concepts. Teachers introduce concepts and problem solving skills through a programming language such as C, C++, C#, Java, Python, or Visual Basic. Computer Programming II reviews and builds on the concepts introduced in Computer Programming I and introduces students to more complex data structures. Topics include sequential files, arrays, and classes.

(*Computer Programming II)

Program of Study Application

Computer Programming is required for the Programming Pathway and recommended for the Networking & Hardware Pathway.

Notes:

All computer programming standards integrate aspects of language arts and mathematics.

Course Standards**Indicator # CP 1 Identify and use a programming environment.**

<i>Webb Level</i>	<i>Sub-indicator</i>	<i>Integrated Content</i>
1	CP 1.1 Demonstrate knowledge of external and internal computer hardware. Examples: <ul style="list-style-type: none"> • Describe the functions of basic external computer hardware devices (monitor, printer, keyboard, mouse, adapters, other devices) • Describe the functions of the internal components of computers (CPU, RAM, ROM, motherboard, graphics card, hard drive, optical drive) 	
1	CP 1.2 Demonstrate knowledge of software concepts. Examples: <ul style="list-style-type: none"> • Define the distinction between computer software and hardware • Identify software categories such as application software, web-based software, or operating system • Describe the difference between an interpreted language vs a compiled language 	
2	CP 1.3 Demonstrate the ability to compile, debug, and execute programs. Examples: <ul style="list-style-type: none"> • Demonstrate how to use an editor/integrated development environment (IDE) to compile and run programs • Understand the difference between syntax, run-time, and logic errors • Demonstrate how to debug programs 	

Notes:

Indicator # CP 2 Employ standard conventions for creation and design of a software program.

<i>Webb Level</i>	<i>Sub-indicator</i>	<i>Integrated Content</i>
2	CP 2.1 Demonstrate the ability to use a standard programming style. Examples: <ul style="list-style-type: none"> • Demonstrate how to use white space properly • Employ a syntax specific naming convention • Construct identifiers with meaningful format (e.g.: camelCase, under_scores, PascalCase, and ALLCAPS) 	
2	CP 2.2 Recognize software development processes. Examples: <ul style="list-style-type: none"> • Identify specifications and requirements • Decompose a problem into appropriate components • Design solutions using algorithms and other problem solving techniques 	
1	CP 2.3 Identify the syntactical components of a program. Examples: <ul style="list-style-type: none"> • Identify keywords, identifiers, operators, operands, and literals • Identify the entry-point of a program 	

Notes:

Indicator # CP 3 Properly use language-fundamental commands and operations.

<i>Webb Level</i>	<i>Sub-indicator</i>	<i>Integrated Content</i>
2	CP 3.1 Demonstrate the ability to use basic elements of a specific language. Examples: <ul style="list-style-type: none"> • Declare, initialize, and assign values to constants and variables • Demonstrate the ability to use input and output commands • Communicate clearly with output values stored in identifiers • Demonstrate the ability to use strings in programs 	
2	CP 3.2 Employ basic arithmetic expressions in programs. Examples: <ul style="list-style-type: none"> • Use basic arithmetic operators (modulus, multiplication, division, addition, subtraction) • Understand order of operation of expressions 	Algebra
3	CP 3.3 Demonstrate the ability to use data types in programs. Examples: <ul style="list-style-type: none"> • Declare and use variables and constants • Differentiate between data types and their application (Boolean, integer, floating point, strings) • Declare and use enumerators as a list of constants 	Algebra
2	CP 3.4 Incorporate functions/methods. Examples: <ul style="list-style-type: none"> • Write functions for repeated procedures • Identify return values 	Algebra

Notes:

Indicator # CP 4 Apply control structures.

<i>Webb Level</i>	<i>Sub-indicator</i>	<i>Integrated Content</i>
2	CP 4.1 Demonstrate the ability to use relational and logical operators in programs. Examples: <ul style="list-style-type: none">• Compare values using relational operators• Form complex expressions using logical operators	
3	CP 4.2 Investigate conditional statements. Examples: <ul style="list-style-type: none">• Incorporate IF-ELSE structures• Make multiple-way selections (switch, case)	
3	CP 4.3 Implement loops in programs. Examples: <ul style="list-style-type: none">• Use initial, terminal, and incremental values in loops• Construct while, do-while, and for loops• Identify nested and infinite loops	

Notes:

Indicator # CP 5 Explore career opportunities in programming.

<i>Webb Level</i>	<i>Sub-indicator</i>	<i>Integrated Content</i>
1	<p>CP 5.1 Identify personal interests and abilities related to Computer Programming/Software Engineering careers.</p> <p>Examples:</p> <ul style="list-style-type: none"> • Identify personal creative talents • Identify technical/programming talents 	Portfolio, SDMyLife
3	<p>CP 5.2 Investigate career opportunities, trends, and requirements related to computer programming/software engineering careers.</p> <p>Examples:</p> <ul style="list-style-type: none"> • Research job opportunities • Investigate trends associated with computer programming/software engineering careers • Discuss related career pathways 	
2	<p>CP 5.3 Demonstrate job skills for programming industries.</p> <p>Examples:</p> <ul style="list-style-type: none"> • Attendance and punctuality • Positive attitude • Positive work ethic • Use of proper social skills • Display ability to work as part of team and take direction from others 	

Notes:

Indicator # CP 6: Integrate arrays.*

<i>Webb Level</i>	<i>Sub-indicator</i>	<i>Integrated Content</i>
2	CP 6.1 Demonstrate the ability to use arrays in programs. Examples: <ul style="list-style-type: none">• Declare arrays• Initialize arrays• Add and remove items from array	Placement of topic varies based on different computer languages
3	CP 6.2 Demonstrate the ability to use strings in programs. Examples: <ul style="list-style-type: none">• Compare string identifiers• Concatenate string identifiers• Locate substring positions	

Notes:

Indicator # CP 7: Implement object-oriented programming techniques.*

<i>Webb Level</i>	<i>Sub-indicator</i>	<i>Integrated Content</i>
3	CP 7.1 Demonstrate the ability to use existing classes. Examples: <ul style="list-style-type: none">• Instantiate objects• Use object data members• Incorporate functions	
4	CP 7.2 Demonstrate the ability to create user-defined classes. Examples: <ul style="list-style-type: none">• Create and use data members• Create a constructor to initialize data members• Create and use instance functions	
4	CP 7.3 Demonstrate proper design principles with classes. Examples: <ul style="list-style-type: none">• Create classes that are well encapsulated (data members private)• Properly use modifiers and accessors (getters and setters)• Apply private and public modifiers according to program design	

Notes: