School District of Marshfield Course Syllabus

Course Name: Intro Computer Programming
Length of Course: Semester
Credit: 1/2 Credit

Program Goal:

Empower learners to be college and career ready through standards-based experiences in the classroom and career-based learning experiences with business and industry partners. Design and implement educational experiences for creating a skilled, knowledgeable, and productive workforce. Learners will engage in competencies that enable them to stay up-to-date with evolving skills as they pursue careers directly out of high school, as technical school degree earners, or as university graduates. Our goal is to develop critical thinkers and collaborative problem solvers, providing connections to the issues and challenges facing our local, regional, and global economies.

Course Description:

Learn how to program instead of being programmed. Using hands-on learning experiences, you'll explore the fundamentals of computer programming using a variety of programming languages. A rewarding, collaborative, and creative learning experience, this course is designed for students with little or no prior programming experience but wish to explore one of the most popular STEM/STEAM fields in terms of jobs outlook and salary in our world today. ***Students***

Board Approved May 2024

who take this class play an integral role in selecting the music and developing animation sequences for a significant portion of the Rotary Winter Wonderland light display each year.

Wisconsin Standards for Computer Science (CS)	
Algorithms and Programming (AP)	
AP1: Students will recognize and define computational problems using algorithms and programming.	
Develop algorithms. AP1.a	1.a.8.h: Analyze a problem, and then design and implement an algorithmic solution using sequence, selection and iteration. 1.a.11.h: (+) Decompose a large-scale computational problem by identifying generalizable patterns and applying them in a solution.
AP2: Students will create computational artifacts using algorithms and programming.	
Develop and implement an artifact. AP2.a	2.a.12.h: Design, develop, and implement a computing artifact that responds to an event (e.g., robot that responds to a sensor, mobile app that responds to a text message, sprite that responds to a broadcast). 2.a.16.h: (+) Demonstrate code reuse by creating programming solutions using libraries and Application Program Interfaces (APIs). (e.g., graphics libraries, maps, API).
AP3: Students will communicate about computing ideas.	
Communicate about technical and social issues. AP3.b	3.b.8.h: Evaluate and analyze how algorithms have impacted our society and discuss the benefits and harmful impacts of a variety of technological innovations.
Document code. AP3.c	3.c.5.h: (+) Use application programming interface (APIs) documentation resources.
AP4: Students will develop and use abstractions.	
Create and use abstractions (representations) to solve complex computational problems. AP4.a	4.a.4.h: Demonstrate the value of abstraction for managing problem complexity (e.g., using a list instead of discrete variables). 4.a.6.h: Deconstruct a complex problem into simpler parts using predefined constructs (e.g., functions and parameters and/or classes). 4.a.13.h: (+) Identify abstractions used in a solution (program or software artifact) and reuse those abstractions to solve a different problem.
AP5: Students will collaborate with diverse teams.	
Work together to solve computational problems using a variety of resources. AP5.a	5.a.7.h: Demonstrate how diverse collaborating impacts the design and development of software products (e.g., discussing real-world examples of products which have been improved through having a diverse design team or reflecting on their own team's development experience).

Foster an inclusive computing culture. AP5.b	5.b.3.h: Create design teams taking into account the strengths and perspectives of potential team members.
AP6: Students will test and refine computational solutions.	
Test and debug computational solutions. AP6.a	6.a.4.h: Use a systematic approach and debugging tools to independently debug a program (e.g., setting breakpoints, inspecting variables with a debugger).
Computing Systems (CS)	
CS3: Students will develop and use abstractions in computing systems.	
Generalize in computer systems. CS3.a	3.a.3.h: (+) Describe the steps necessary for a computer to execute high-level source code (e.g., compilation to machine language, interpretation, fetch-decode-execute cycle).
CS4: Students will create and modify computing systems.	
Modify and create computational artifacts. CS4.a	4.a.2.h: Create, extend, or modify existing programs to add new features and behaviors using different forms of inputs and outputs (e.g., inputs such as sensors, mouse clicks, data sets; outputs such as text, graphics, sounds).
Impacts of Computing (IC)	
IC1: Students will understand the impact and effect computing technology has on our everyday lives.	
Understand the impact technology has on our everyday lives, and the effects of computing on the economy and culture. IC1.a	1.a.6.h: Debate the social and economic implications associated with ethical and unethical computing practices (e.g., intellectual property rights, hacktivism, software piracy, new computers shipped with malware).
Understand the effects of computing on communication and relationships. IC1.b	1.b.5.h: Evaluate the negative impacts of electronic communication on personal relationships and evaluate differences between face-to-face and electronic communication.
IC2: Students will experience learning within a collaborative, inclusive computing culture and explain the steps needed to ensure that all people have access to computing.	
Collaborate ethically in the creation of digital artifacts. IC2.c	2.c.5.h: Ethically and safely select, observe, and contribute to global collaboration in the development of a computational artifact (e.g., contribute the resolution of a bug in an open-source project platform, or contribute an online article).
Networking and the Internet (NI)	
NI1: Students will understand the importance of security when using technology.	
Use secure practices for personal computing. NI1.a	1.a.6.h: Provide examples of personal data that should be kept secure and the methods by which individuals keep their private data secure.

Key Vocabulary:			
Debugging	Logical operator	Run-time error	Scope
Graphical User Interface (GUI)	Object-Orientated Programming (OOP)	Assignment Operator	Event Procedure
Method	Assignment Statement	Object	Select Case
Call (procedure call)	String literal	Strongly typed	Case structure
IDE	If/ Then/ Else	Syntax error	Class
Code	Compile	Concatenation	Psuedocode
Text property	Condition	Constant	Relational operator
User interface	Control	Local variable	Remark
Data type	Logic error	Variable	

Topics/Content Outline- Units and Themes:

Quarter 1:

- Introduction Computer Programming (2 weeks)
  - Languages, Development Life Cycle
  - GUI, IDE (Integrated Development Environments)
- Program Design (3 weeks)
  - Windows Applications, Forms & Controls
  - Interface Design, End user vitality
  - Problem solving, Debugging & Testing
- Variables and Arithmetic Operators (2 weeks)
  - Types, Numeric and String
  - Syntax, Concatenation
  - Variables, Declarations and Constants
- Decision Structures (2 weeks)
  - Decisions, If/Then. If/Then/Else
  - Decisions, Nesting

Quarter 2:

- Decisions Structures (1 week)
  - Decisions, Case Statements
- Iteration (3 weeks)
  - Looping Constructs, Determinate, Indeterminate
  - Determinate, For…Next
  - Indeterminate, Do Loops, Until/While
- Debugging (1 week)
  - Errors, compile-time, run-time, logic
- Arrays (3 weeks)
  - Declaration & Initialization, One and Two-dimensional arrays
  - Search and Sort, For/Next, Do/Loop
  - Methods, sort, reverse

- Computing in Context (1 week)
    - Ethics, privacy, legal issues, intellectual property, social and ethical ramifications of computer use