Dublin City Schools
STEAM
Graded Course of Study
2022

DCS STEAM Vision

Dublin City Schools is committed to providing purposeful STEAM learning experiences to students throughout their K-12 journey. These experiences will be in the form of integrated, interdisciplinary experiences as well as focused pathways in the areas of STEAM.

We commit to transforming STEAM into more than the integration of Science, Technology, Engineering, and Mathematics with vision to expand ownership to all disciplines and grade levels. By creating a culture of thinking, curiosity and creativity across content, students will engage in interest based learning that will help them develop the attitudes and skills that will support them in a variety of career and life pathways. These learning experiences will support students as lifelong, adaptable learners who can thrive in a quickly changing world.

We believe in STEAM learning for all students and commit to creating equitable access so that our STEAM classrooms are representative of our school populations and communities.

Instructional Agreements:

- We recognize the importance of early access and exposure to STEAM learning.
- We prioritize learning where students will identify and solve open-ended problems and engage in experiential learning.
- We will engage students through a lens of design thinking and promote opportunities for PBL.
- We will provide students with industry connections and experiences.
- We prioritize educating the whole child, in addition to our content. This includes a commitment to employability skills and emotional intelligence.
- We value students seeing themselves in STEAM fields.

Introduction to Computer Programming

Introduction to Computer Programming Course Goals: Introduction to Computer Programming provides an introductory study of techniques in programming utilizing Java, C++ and other languages. Topics include structure of programming, input and output, data types and structures, logical operations and loops. Students will apply computing resources in a variety of curriculum areas. The class is designed as a programming/lecture/laboratory class with emphasis on programming/debugging. Upon completion of this course the student will have a solid background in program methodology.

Systems		
Strand	**Topic**	**Content Statement**
Computing Systems	**Hardware and Software**	CS.HS.9-12.F.a Compare and contrast interactions between application software, system software and hardware
	Troubleshooting	CS.T.9-12.F.a Apply a systemic process to identify problems and take steps to correct them within an integrated computing system.

Data and Storage		
Strand	**Topic**	**Content Statement**
Networks and the Internet	**Data Collection and Storage**	DA.DCS.9-12.F.a Analyze patterns in a real-world data store through hypothesis, testing and use of data tools to gain insight and knowledge. DA.DCS.9-12.F.b Investigate data storage systems to compare and contrast how data is stored and accessed.

Algorithm Analysis and Programming		
Strand	**Topic**	**Content Statement**
Algorithmic Thinking and Programming	**Algorithms**	ATP.A.9-12.F.a Define and use appropriate problem solving strategies and visual artifacts to create and refine a solution to a real world problem. ATP.A.9-12.F.b Define and implement an algorithm by decomposing problem requirements from a problem statement to solve a problem. ATP.A.9-12.A.a Define and explain recursive algorithms to understand how and when to apply them. ATP.A.9-12.A.b Use recursion to effectively solve problems. ATP.A.9-12.A.c Define and explain sorting and searching algorithms to understand how and when to apply them. ATP.A.9-12.A.d Use sorting and searching to analyze and organize data.
	Variables and Data Representation	ATP.VDR.9-12.F.a Identify types of variables and data and utilize them to create a computer program that stores data in appropriate ways. ATP.VDR.9-12.A.a Utilize different data storage structures to store larger and more complex data than variables can contain. ATP.VDR.9-12.A.b Identify the appropriate data structures or variables to use to design a solution to a complex problem.
	Control Structures	ATP.CS.9-12.F.a Define control structures and Boolean logic and use them to solve real-world scenarios. ATP.CS.9-12.F.b Use appropriate syntax to create and use a method.

		ATP.CS.9-12.F.c Use data scoping to isolate data. ATP.CS.9-12.A.a Write programs that use library methods and control structures and methods to solve a problem. ATP.CS.9-12.A.b Refactor a program to be smaller and more efficient.
	Modularity	ATP.M.9-12.F.a Break down a solution into procedures using systematic analysis and design. ATP.M.9-12.F.b Create computational artifacts by systematically organizing, manipulating and/or processing data. ATP.M.9-12.A.a Construct solutions to problems using student created components (e.g., procedures, modules, objects). ATP.M.9-12.A.b Design or redesign a solution to a large-scale computational problem by identifying generalizable patterns. ATP.M.9-12.A.c Create programming solutions by reusing existing code (e.g., libraries, Application Programming Interface (APIs), code repositories).
	Program Development	ATP.PD.9-12.F.a Investigate software development methodologies to select the appropriate one for a project to complete as a team. ATP.PD.9-12.F.b Compare test methodologies to evaluate why each is used and to determine their benefits and costs. ATP.PD.9-12.F.c Correctly use consistent naming conventions, version control and comments to demonstrate why these are important for future use, maintenance and reuse of code. ATP.PD.9-12.A.a Fully implement the most appropriate software methodology to complete a team programming project.

		ATP.CS.9-12.A.a Write programs that use library functions, methods and control structures to solve a problem.