Unit 2: Basic Programming Analysis

Content Area: **Computer Science**
Course(s):
Time Period: **Marking Period 1**
Length: **3-4 Weeks**
Status: **Published**

Summary

Students will learn to design a piece of software, or program, and to solve a given problem. They will learn about current programming software functionality. Students will learn about different levels of the debugging process, and how to test their code for all possibilities of input.

Revised October 2020

CS.9-12.8.1.12.AP.3	Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
CS.9-12.8.1.12.AP.7	Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
CS.9-12.8.1.12.AP.8	Evaluate and refine computational artifacts to make them more usable and accessible.
CS.9-12.8.1.12.DA.3	Translate between decimal numbers and binary numbers.
CS.9-12.8.1.12.DA.4	Explain the relationship between binary numbers and the storage and use of data in a computing device.
LA.K-12.NJSLSA.L4	Determine or clarify the meaning of unknown and multiple-meaning words and phrases by using context clues, analyzing meaningful word parts, and consulting general and specialized reference materials, as appropriate.
LA.K-12.NJSLSA.L5	Demonstrate understanding of word relationships and nuances in word meanings.
MA.N-Q.A.3	Choose a level of accuracy appropriate to limitations on measurement when reporting quantities.
MA.9-12.1.2.12prof.Cr	Creating
MA.9-12.1.2.12prof.Cr2	Organizing and developing ideas.
MA.9-12.1.2.12prof.Cr3	Refining and completing products.
MA.A-CED.A.1	Create equations and inequalities in one variable and use them to solve problems.
TECH.8.2.12.E.2	Analyze the relationships between internal and external computer components.
TECH.8.2.12.E.3	Use a programming language to solve problems or accomplish a task (e.g., robotic functions, website designs, applications, and games).

Essential Questions / Enduring Understandings

Essential Questions:

- How is object-oriented desgin implemented into programs?
- Given a problem, what operations need to be perfroemed in order to solve the problem?

- How can you analyze a program's correctness?
- How is an algorithm used in a program?
- How do we write programs in our current programming environment?
- How do we manipulate objects in our current programming environment?
- What are objects?
- How is information encapsulated?
- Given a program (or module of a program), what are the boundry cases?
- Given a program (or module of a program), how would you generate appropraite test data?
- What are good programming practices?

Enduring Understandings:

- Commong coding practices are in place to keep standards within the code.
- Reusing code, either in whole or in part, is a postive coding practice.
- Objects can be manipulated in the current programming environment.

Objectives

Students will know:

- ways of adapting a program to changing circumstances
- how to write specifications for software
- how to make adaptable software
- object-oriented development
- encapsulation
- how to choose test data
- how to debug a program

Students will be skilled at:

- evaluating efficiency of a program
- developing programs to scale
- flexibility in programming

Learning Plan

- Preview the essential questions and connect learning throughout the unit.
- Using algorithms, have students develop programs that meet certain specifications.
- Discuss good techniques in designing a user interface.
- Have students develop programs that include a user interface that is appropriate for its given specifications
- Discuss with the students the purpose of testing modules of their programs.
- Go over the different methods of testing modules depending upon the situation.
- Using programs, discuss which method of testing would be best.
- Have students write programs to a given set of specifications, and then implement a variety of tests upon the program.
- Identify boundry cases for programs and how best to test those boundary cases.
- Expand on existing programs to enable programs to take on larger inputs.

Assessments

- Assessments
    - Formative: Daily assessments using examples from class notes and CodeHS.com, AP Classroom/Albert Checks for Understanding
    - Summative: Teacher-created assessments/projects and CodeHS Computer Science Projects, AP Classroom/Albert Unit Assessments
    - Benchmark: Check for understanding benchmark assessments on CodeHS, AP Classroom/Albert/Khan Academy Diagnostics
    - Alternative Assessments: Student-centered activities such as a doorbell coding project, game design projects, and other activities involving real world applications
- Complete performance tasks that require the students to plan, structure and develop their own code to solve given problems.
- Be observed by the teacher during individual work on the performance tasks.
- Complete quizzes/test: Object Oriented Programming, Program Developement, Test Values
- Conduct self-assessments and reflections
- Conduct peer evaluations
- Answer the essential questions
- Participate in class discussions.

Materials

- Core instructional materials: Core Book List

Supplemental materials: CodeHS

- Computer
- Reference books

Integrated Accommodations and Modifications

See Linked Document.