Course: OBrien AP Computer Science A (Nitro) 2021

Lesson 4.5: Informal Code Analysis

CodeHS

https://codehs.com/lms/assignment/55325680/lesson_plan

Description	In this lesson, students will examine the concept of informal code analysis. This includes an algorithm's correctness, efficiency and the ability to be understood. This lesson corresponds with AP Computer Science A topic 4.5.
Objective	Students will be able to: • Compute statement execution counts of iterative statements • Compute informal run-time comparison of iterative statements
Activities	4.5.1 Video: Informal Code Analysis 4.5.2 Check for Understanding: Informal Code Analysis 4.5.3 Example: Loop Execution Count 4.5.4 Example: While Loop Time 4.5.5 Example: For Loop Time 4.5.6 Free Response: Time Comparisons 4.5.7 Exercise: Improving findChar Speed 4.5.8 Example: Improving findChar Speed Check 4.5.9 Free Response: findChar Speed Reflection
Prior Knowledge	• Create and call objects and methods • Boolean expressions and relational operators • If/else and else if statements • While loops, for loops and nested loops • Traverse through a String
	• There are a few practice questions found in the video/slides. Consider pausing and taking a class poll, or having a student volunteer the answer before revealing the slide with the solution. • There is a handout associated with this lesson. Consider using it in class, or assigning it to students as homework.

Planning Notes

- This is the last lesson of the unit. It is strongly recommended that teachers assign the personal progress check for Unit 4 from the AP Classroom as review to students after this lesson is complete (https://myap.collegeboard.org/login). It's important to note that the personal progress checks cannot be used for a grade - to give students a summative assessment for credit, use the CodeHS *Iteration* quiz located in the supplemental *Unit Quizzes* module.

Standards Addressed

NY 9-12 Standards

Name	Description
9-12.CT.1	Create a simple digital model that makes predictions of outcomes.
9-12.CT.5	Modify a function or procedure in a program to perform its computation in a different way over the same inputs, while preserving the result of the overall program.

Teaching and Learning Strategies

Lesson Opener:

- Have students brainstorm and write down answers to the discussion questions listed below. Students can work individually or in groups/pairs. Have them share their responses. [5 mins]

Activities:

- Watch the lesson video and take the corresponding quiz. This quiz is a quick check for understanding. [5-6 mins]
- Explore the *Loop Execution Count* example. [5 mins]
- Explore the *While Loop Time* example. [5 mins]
- Explore the *For Loop Time* example. [5 mins]
- Complete the *Time Comparisons* free response activity. [10 mins]
- Complete the *Improving isChar Speed* exercise. [15 mins]
- Explore the *Improving isChar Speed Check* example. [5 mins]
- Complete the *isChar Speed Reflection* free response activity. [10 mins]

Lesson Closer:

- Have students reflect and discuss their responses to the end of class discussion questions. [5 mins]
- If there is time, consider having students complete the associated handout. [10 mins]

Discussion Questions

Beginning of Class:

- If you and another student were tasked with creating a solution to a CS problem, is it possible for you to create two different solutions but get to the same result?
  - *Yes, the beauty of programming is that there are different approaches to solving problems! (Whether it's the most efficient is another thing)*
- What are some potential reasons one person's code might be "better" than another? Is there a scenario where that is the case?
  - *Using less code is generally a better approach because it is more readable and scalable.*

End of Class:

- What is an algorithm?
  - *An algorithm is simply a step by step set of instructions to complete a task.*
- Is it possible that some algorithms are better than others?
  - *Some are more efficient. Often that coincides with programmer's definition of better.*
- Consider the following code segment:

```
// for loop
for (int count = 0; count < 5; count++)
{
    System.out.print("*");
}

// while loop
int number = 0;
while (number <= 5)
{
  number++;
}
```

- How many times does the for loop iterate? How many times does the while loop iterate? Why are they different?
  - *The for loop will iterate 5 times. The while loop will iterate 6 times. The while loop will iterate one more time because the inequality <= is used instead of <. The while loop will run*

	when the variable = 5 whereas the for loop will not.
Resources/Handouts	Exploring Algorithm Efficiency (Teacher) Exploring Algorithm Efficiency

Vocabulary

Term	Definition
Algorithm	Step-by-step process that solves a problem.
Statement execution count	The number of times a statement is executed by the program.
Big-O Notation	A way to represent how long an algorithm will take to execute. It helps to determine how efficient different approaches to solving a problem are.

Modification: Advanced	Modification: Special Education	Modification: English Language Learners
• Pair programming with another student • Print out slides for students to reference	• Pair programming with another student • Print out slides for students to reference	