Workshop: Let’s Talk to Our Rubber Ducks: Scavenger Hunt for Computational Thinking, Analyzing Code, & Debugging

Introduction

A rubber duck can act as a sounding board for programmers to work through difficult concepts or complicated logic sequences. Speaking or explaining code logic out loud is known to be highly beneficial when “stuck” by an error. Andrew Errington created the concept of “rubber duck debugging”. A good programmer needs to develop several essential skills including debugging, computational thinking, and code analysis. How do we instill these concepts into first-year programming students? Introductory programming students are often reluctant to try debugging their code independently. Introductory programming instructors watch their students write lines and lines of code without compiling the code or testing the code.

Workshop Contents

During this workshop attendees will learn how to add fun to their courses by using a code-based scavenger hunt. Each scavenger hunt clue is a small (less than one page) C++ program provided to the students via a hard copy. The students, work in teams of 2-4, are forced to “think like the computer” and analyze the code (further developing their computational thinking skills). By stepping away from the compiler and unable to just run the program, students must work on understanding the specifics of the material. The output statements within the code provide the location of the next clue, with the final clue leading to the students selecting the rubber duck that “quacks” to them. The scavenger hunt gets the students out from behind their computers, introduces the students to an industry practice, and opens the door to future assignments on debugging techniques. The scavenger hunt covers a wide variety of topics, including (1) mathematical expressions, (2) mod operator, (3) integer math, (4) switch statements, (5) if statements, (6) increment/decrement, (7) for loops, (8) while loops, and (9) do-while loops. Typically, the scavenger hunt beings in the classroom. The instructor ensures all teams have started the scavenger hunt and then “disappears” to the final location. The scavenger hunt could take students on a journey of their college/university to become more familiar with important locations (e.g., where office hours are held, department office, computer lab, etc.).

Learning Objectives for the Workshop

By the end of this workshop, attendees should be able to:

1. Explain what rubber duck debugging is and how it is used
2. Understand the importance of computational thinking in programming
3. Explain how the scavenger hunt allows for graceful failure
4. Create their own scavenger hunt for a course they teach

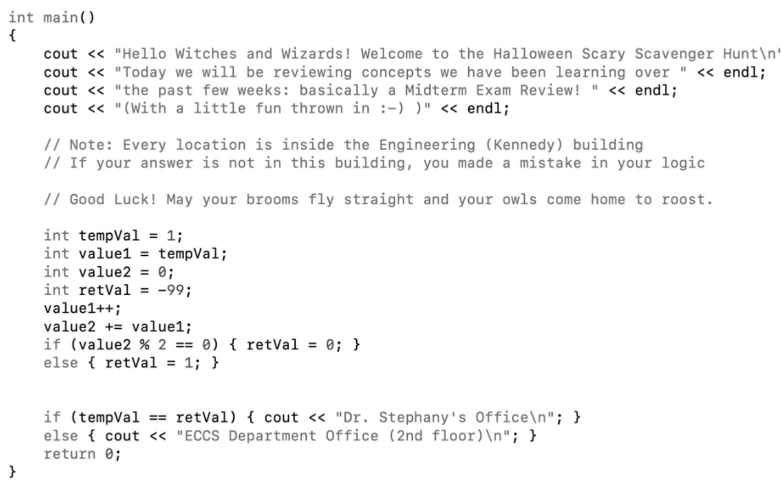
Topics Covered

This workshop will cover background on key concepts discussed (rubber duck debugging, computational thinking, and code analysis), the importance of learning debugging techniques, the specifics of the Computer Science 1 (CS1) scavenger hunt, hints and tips for adapting this for other programming languages, adapting this to courses outside of the computing field, and hints and tips for creating an online version [1] or a version for a course with large enrollment. The workshop session facilitators believe in active learning techniques. Therefore, attendees will have the opportunity to try out a code-based scavenger hunt during the workshop.

Workshop Schedule

1. Introduction, Purpose, and Agenda
2. Talk to your Rubber Duck! Rubber Duck Debugging Explained
3. Let's Go on a Scavenger Hunt!
4. Adaption to other courses or larger class sizes
5. Online Resources, Q & A, Wrap-up

Example Clue from ECCS1611 Programming 1 at Ohio Northern University



```
int main()
{
    cout << "Hello Witches and Wizards! Welcome to the Halloween Scary Scavenger Hunt\n";
    cout << "Today we will be reviewing concepts we have been learning over " << endl;
    cout << "the past few weeks: basically a Midterm Exam Review! " << endl;
    cout << "(With a little fun thrown in :-) )" << endl;

    // Note: Every location is inside the Engineering (Kennedy) building
    // If your answer is not in this building, you made a mistake in your logic

    // Good Luck! May your brooms fly straight and your owls come home to roost.

    int tempVal = 1;
    int value1 = tempVal;
    int value2 = 0;
    int retVal = -99;
    value1++;
    value2 += value1;
    if (value2 % 2 == 0) { retVal = 0; }
    else { retVal = 1; }


    if (tempVal == retVal) { cout << "Dr. Stephany's Office\n"; }
    else { cout << "ECCS Department Office (2nd floor)\n"; }
    return 0;
}
```

Figure 1. Example Starting Scavenger Hunt Clue.

References

[1] S. Coffman-Wolph and K. Gray, "Computer coding scavenger hunt using quick response codes (resource exchange)," in *2020 ASEE Virtual Annual Conference Content Access Proceedings*, 2020.