Math Tutor Program

Scott Nevin

CSC-17B

12/10/14

Introduction:

This program is a simple math tutor for young, elementary grade, children to practice the four basic math skills. The GUI is simple, the main window presents the user with five options: Addition, Subtraction, Multiplication, Division, or Quit. When the user selects one of the operations, a new dialog appears with a simple equation with two randomly generated numbers of 2 digits in length, and a space for the user to enter their answer. A “Check” button is below the text entry point, which a user will click once they have entered their answer. The user will then receive immediate feedback in the form of an information dialog, telling them if they answered the question correctly or not. The user may then advance to the next question, or quit the window to select a different operation or quit.

Method:

The method of the program is simple. The main window is built from QDialog, and contains 5 push buttons along with a few labels. The button presses launch the appropriate quiz dialog using basic signals and slots. Classes were used to build all dialogs and all were written directly in code, no designer was used. The way the program checks the user’s answer is by pulling the correct answer as a public variable from the class and converting it to a string. It then compares this string with what the user has entered in the line edit. These strings matching triggers a correct answer. When the user clicks “Next” the program terminates the current quiz dialog and creates a new one with newly generated numbers.

Intended features:

Many features were intended for this program, but issues with the basic program prevented further development. Intended features included: Score keeping, saving statistics to a file, a slider changing the size of numbers generated, and forcing the user to correctly answer a question before advancing to another.

Known Issues:

The biggest known problem about the program deals with the division section. It was last priority and I was unable to get the program to work well with floats. If an answer isn’t a whole number in the division section, it will say that the user’s answer is incorrect every time. Also, when running out of time, I was unable to keep the different operations all in one file. I couldn’t make the program conditionally execute a certain operation based on which button was pressed. So, each operation has its own header and implementation file that creates an identical window to the others. This project contains a significantly lower unique line count than something I would normally turn in, but my shaky understanding of implementing classes along with a struggle to understand signals and slots made for far more problems and setbacks than I had expected.

Statistics:

Unique lines of code: Roughly 300

Functions: Less than 10 unique functions

Push buttons: Roughly 10 unique push buttons

Different dialogs: 5 different dialogs

Constructs used:

-Primitive data types

-Classes

-Qt libraries

-Basic dialogs

-Push buttons

-QMessagebox (QuizDialog.cpp line 92)

-Coded layouts (MainWindow.cpp lines 51-65)

-Labels

-Line edits

-Signals and slots

-Functions

Screenshots:



