Programming Project: Quiz

CPSC 1020 – Spring 2024

School of COMPUTING
Clemson University

Introduction

Write a program that allows a player to answer quiz questions. The game allows adding questions and answers to a database in a simple text file. The program asks the questions in the database in random order. The player types an answer and hits enter. If the answer is correct, the player is told so and the player earns 1 point. If the answer is incorrect, the player is told so and the players loses 1 point. Once all questions have been asked, the program displays how many questions have been asked, the number of correct answers, the number of wrong answers, and the player's final score.

The example database given to you is called database.txt and has four questions with their answers. Your program needs to work independently of the number of question/answer pairs in the database.

Implementation and Requirements

Develop your program following these steps:

1) Build Question class

Each question with its corresponding answer is represented as an object of type QA. Write a QA class based on the following UML class diagram:

QA
- question: string - answer: string - score: int, static
+ QA() + QA(q: string, a: string) + getQuestion(): string + getAnswer(): string + getScore(): int, static + letterCount(): int + updateScore(val: int): void, static

Additional specifications:

- Use in-class initialization to initialize `question` and `answer` to "" (an empty string).
- Use the constructor `QA(q: string, a: string)` with member list initialization to allow creating an instance of QA with strings for `question` and `answer` passed as arguments.
- `score` is a static variable.
- `getScore` is a static member function that returns the value of score
- `letterCount` returns the number of letters contained in the question and answer. NOTE: whitespace and punctuation does NOT count for this. Only the letters A – Z regardless of the case. For example: How many 1's are in this question? Has a letterCount value of 25.
- `updateScore` is a static member function that changes the value of score by the value of the number passed as `val`, which can be a positive or negative integer. Use this function in your main program as follows:
  - If a player gives a correct answer, score is increased by 1 (i.e., `updateScore(1)`)
  - If a player gives an incorrect answer, score is decreased by 1 (i.e., `updateScore(-1)`). However, a player's score cannot be less than zero, which must be implemented in the `updateScore` function.
- The QA class should be declared and implemented in files QA.h and QA.cpp, respectively.
- **Do NOT use "using namespace std;" in your .h or .cpp file.**

2) Parse the text file and create QA objects

1. The database where you save your questions and answers is the text file database.txt given to you. You can assume that it is formatted correctly with questions starting with "Question:" and answers starting with "Answer:".
2. In your main program, write code to read in the content from database.txt.
3. Store every question-answer pair as a QA object.
4. Store all QA objects in a single vector named `quizVector`.

Requirements:

- The name of the input file, database.txt, should be provided as a command line argument. That is, it must be possible to run your final program as follows:

```
./main.out [database name]
```

where [database name] is the name of the input file, e.g. file database.txt provided to you.

- The file database.txt contains 4 question-answer pairs. Do NOT hard code this number in your program. Instead, your program must accept input files with any number of question- answer pairs.

3) Ask questions and collect player input

In your main program, randomly select a question-answer pair. Show the question to the player, collect their input, and display the results as follows:

- If the player has given the correct answer, print
  ```
  Correct!
  Current score: [score]
  ```
  where [score] is the player's current score that has been updated by 1 point for correctly answering the question.

- If the player has given a wrong answer, print
  ```
  Wrong! Correct answer: [correct
  answer] Current score: [score]
  ```
  where [correct answer] is the correct answer from the database and [score] is the player's current score that has been updated by -1 point for not correctly answering the question.

Requirements:

- Questions must be randomly selected. To select a random question-answer pair, use the `random_shuffle` function from the algorithm library to randomly shuffle the objects in `quizVector`. You can read more about this function here: https://www.cplusplus.com/reference/algorithm/random_shuffle/

- Add the following line to your program *before* you call `random_shuffle`: `srand(unsigned(time(0)));`

  Read the srand manual to understand the purpose of this line: https://www.cplusplus.com/reference/cstdlib/srand/. Here is a short explanation:

  - `srand` is a function from the `<cstdlib>` library that is used to initialize the random number generator;
  - `time` is a function from the `<ctime>` library that is used to return the current time on a computer;
  - by passing `time` to `srand`, we initialize the random number generator to the current runtime value, which for our purpose is distinctive enough to randomize the order returned by `random_shuffle`.

4) Print results

At the end of the quiz, your program must print the results as follows:

```
Number of questions: [#questions]
Number correct answers: [#correct]
Number wrong answers: [#wrong]
Final score: [score]
```

which must be handled by a function `printResult(int questions, int correct, int wrong)`

Requirements:

- `printResult` must use a stringstream to build and return a string.
- The number of questions, correct answers, and wrong answers are passed as arguments to the functions.
- Make `printResult` a friend of class QA so that the function can access the static variable score.
- The function should be declared and implemented in files printResult.h and printResult.cpp, respectively.
- **Do NOT use "using namespace std;" in your .h or .cpp file.**

Sample Run

```
./main.out database.txt
True or false (T/F)? Constructors have the return type void.
Type in your answer: F
Correct!
Current score: 1

What is a member function called that is not a member of a class, but has
access to the private members of the class?
Type in your answer: friend
Correct!
Current score: 2

What do you declare a member variable if all objects of a class should
share that variable?
Type in your answer: const
Wrong! Correct answer: static
Current score: 1

What keyword is used to access the built-in pointer that is automatically
passed as a hidden argument to all non-static member functions?
Type in your answer: this Correct!
Current score: 2

Number of questions: 4
Number correct answers: 3
Number wrong answers: 1
Final score: 2
```

Submission Instructions

- Submit the following 6 files via Gradescope:
  - main.cpp
  - QA.h
  - QA.cpp
  - printResult.h
  - printResult.cpp
  - Makefile
- Make sure your program compiles and runs on the SoC Linux machines; the autograder automatically assigns 0 points to programs that do not compile.
- Code style (proper indentation, consistency, readability, use of comments, etc.) will be considered when grading your submission.

Collaboration Policy: This project is an individual programming assignment. All work you submit must be your own individual work. You may identify errors or ask about ambiguous specifications in a programming assignment on MS Teams, but your are not allowed to complete an assignment with the help from peers or outside tutors, and you are not allowed to share code with others or copy someone else's code or code from the internet. Please make sure to carefully read the Academic Integrity Statement in the syllabus, which explains what is considered cheating in this class.

Late Policy: Work submitted late will be subject to a 10 percentage point deduction on the 0-100 scale (i.e., one letter grade) per day.

Gradescope Submission: Code must be submitted via Gradescope. Please note that the Gradescope "autograder" is designed to give you feedback on the correctness on your submission. This includes checking if all necessary files are submitted, if the unit tests used for autograding compile, and if the core functionality of your submission is correct (e.g., default constructor, argument constructor, some of the member functions). The points displayed by the autograder when you submit your program do NOT correspond to your final grade for this assignment. Additional tests might be added to the autograder after the submission deadline, and parts of the program will be manually graded.