CSCI 262 Data Structures

Spring 2019

Project 1: Personality Test

(Back to Assignments)

Goals

- Practice with IO
- Practice with vectors
- Review of basics

Overview

The Myers-Briggs Type Indicator (MBTI) implements and categorizes C. G. Jung's theory of psychological types. "The essence of the theory is that much seemingly random variation in the behavior is actually quite orderly and consistent, being due to basic differences in the ways individuals prefer to use their perception and judgment." The psychology theory is then implemented to be easy to understand by Isabel Briggs Myers and Katherine Briggs and broken into 4 categories:

- (E)xtraversion or (I)ntraversion
- (S)ensing or I(N)tuition
- (T)hinking or (F)eeling
- (J)udging or (P)erceiving

Thus, personality types are broken into 16 combinations of the above, such as ESTJ or INFP.

We credit Myers-Briggs Foundation, Human Metrics, and 16 Personalities for the information, questions, and analysis we will use for the project.

This project will guide you through creating a simple personality test program. There are a set of questions to read from a file, each associated with what personality trait it is testing, and will ask the user to answer each question one by one. At the end, the program will report what the user's Myers-Briggs Personality type is! This is an individual project, and you may not share code with others.

If you would like to actually test out your personality type, 16 Personalities has a great intuitive test. Check out this link if you are interested (and credits to them for their resources).

Part 1: Download starter code and begin to understand the project

---

You can download all of the project files as a .zip, or download them individually below.

Task 1.0: Input File Format

Download questions.txt, which contains the 12 questions that the program will ask the user to assess their personality.

The file format is as follows:

1. First line contains the number of questions followed by info about the questions
2. `[Category ID] [QUESTION] [Yes Answer] [No Answer]`
3. Question Credits (unnecessary to read in)

The `Category ID` tells you which category (of the four personality traits) this is asking you about. If it is 1, for example, then it is testing for the Introverted/Extroverted trait. If the user answers yes to the question, then the `Yes Answer` should be used. In the first question, if yes is selected, then points for introverted should be incremented.

NOTE: this specific file has 12 questions, but your program should be able to read more or less than 12 lines as well. We may test your code with files with different number of questions.

Next, download proj1.cpp, personality_test.cpp, and personality_test.h. Go ahead and create a new project in CLion as usual, and upload the starter code and questions.txt to it. The questions.txt file needs to go in your cmake-build-debug folder.

Pro-tip: If you create your project folder first using your operating system's file manager (e.g., File Explorer in Windows) and put all of the files you need into it, then the easiest way to make the project in CLion is to use the "Import Project" option (make sure you select the project folder containing your source code, not anything higher in the folder structure). CLion will then create a correct CMakeLists.txt file for you. (You will still need to move questions.txt to your cmake-build-debug directory.) If you create your project first, then copy the files into the project, CLion will not (unfortunately) update your CMakeLists.txt (plus you will have a spurious main.cpp that you don't need!) So if you go this route, be sure to remove main.cpp and modify CMakeLists.txt (see the CLion FAQ on the course Help page for info on what your CMakeLists.txt file should look like).

Task 1.1: figure out what is going on

Read through the starter code provided. The proj1.cpp does very little work beyond prompting the user for the location of the questions file, and opening the file. Most of the work is done in the personality_test class found in personality_test.h and personality_test.cpp. The personality_test object actually administers the test and informs the user of their personality type. How you decide to store and manage the test questions will be up to you.

Part 2: Load the questions

Task 2.0: IO

First we want the load the questions.txt file. Use the load_file function in proj1.cpp (recommended) to load. Create an ifstream object to read in the file. At the end of this task, we want the following:

```
==================================
Welcome to CSCI262 Personality Test!
==================================


Please enter the input file name, e.g., "questions.txt":
```

Upon entering the file name, the program must check if the ifstream object succesfully opened. If it did not succesfully open, inform the user with the following message:

```
Error opening file "filename"
```

The program must then loop, and keep asking the user for a proper file.

Task 2.1: Load the data

Now that we have an istream object, we should call `personality_test::load` which takes in as parameter an istream object. After calling, we have several options of how to implement the `personality_test::load` function.

One option, is to create a new class, `question` inside the `personality_test.h` file. `question` can then be a class that has attributes `category_id, question, yes_answer, no_answer` and some other helper functions as necessary. Then, your `personality_test` object can have a `vector` of questions for simple management. How you design the program is up to you. You could also create a `vector` of `vector` that contains the data of each question.

Task 2.2: Verify the questions.txt

Any good programmer will ensure that the input file's formatting is correct. For our purposes, we ask that you implement two simple tests to ensure that the input is correct. For example, in our questions.txt file, each question line must begin with an integer (the category ID). Think of another test you should run to check the formatting of the input file (this does not have to be an exhaustive test). If the input file passes all your tests, `personality_test::load` returns true. If they do not pass, return false, and ensure that `load_file` handles this with the following message.

```
Input file "filename" appears to not be a proper file!
```

Return to the loop to ask for another file if the input file has bad formatting. Comment what tests you conducted, and tell us about your tests in the README.

Part 3: (Optional) Printing Out

Implement `personality_test::printout` which prints out each of the questions, along with their category ID, Yes answer and No answer. This is just to make sure that you were successfully able to load the data before moving forwards.

Part 4: Ask questions

Task 4.0: Ask the user

In the starter code, this portion is suggested to be written in `personality_test::analyze_personality`, but feel free to edit as you wish. For this first portion, ask the user questions each of the 12 questions one by one, and read in a yes or no user input. Make sure it can handle Y, y, yes, Yes, YES, N, n, no, No, and NO. If the user provides bad input (like

"yaas"), then display `Sorry, I didn't recognize your input, please type again` and loop until a proper input is given.

Task 4.1: Store Info

As you ask each question, make sure you store responses from the user as well. For example, if the user answers 'y' on the first question, that should increment their (I)ntrovert score. With a 'no' answer, that should increment their (E)xtrovert score. How you decide to keep track of their responses is up to you.

Task 4.2: Analyze

Now that you have the user's responses, it is now time to analyze and categorize the user's personality type into one of the 16 personality types. Take advantage of the category IDs. For example, collect all the results for category ID = 1. If the results are "EEI", then the overall personality trait will be "E". Have `personality_test::analyze_personality` return the string personality type to be used again later (for example, "INTJ").

For checking purposes, if you response yes for all questions, the personality type should be ISFP. With all no you should get ENTJ.

NOTE: The number of questions asked for each category will always be odd; There will be no ties for each category, thus you should always know which trait has majority votes.

Part 5: Provide an analysis

Task 5.1: Download analysis file

Download analysis.txt. The file's first line indicates how many personality types it has written. Then, each personality type is described with the following format:

```
[Personality Type] [Category] [Type] [Description]
```

Task 5.2: IO

Load the analysis.txt file, and similarly to before, ensure that the file has been read in properly (no need to run any tests). Read in the entire .txt file, and store the information in a method of your preference.

Task 5.3: Inform your user

Now that you know your user's personality type, let them know more about what that means! For example, if the user receives the personality type INTJ, then your program should display:

```
Your personality type is: INTJ

The category is: Analysts

You are: The Architect

Description: Imaginative and strategic thinkers, with a plan for everything.
```

Task 5.4: Save results

Ask the user if they would like to save the results. If yes, then ask what they would like to name it. Then, save the results with the provided name. It should save the info that was displayed: for example, if the user specifies the file name "results.txt", then after saving there should exist a file named `results.txt` with the content:

```
Your personality type is: INTJ

The category is: Analysts

You are: The Architect

Description: Imaginative and strategic thinkers, with a plan for everything.
```

Part 6: Play again?

At the end, ask your user if they would like to play again. If the user wishes the run the test again, the program should return to executing `load_file` and continue the test (in case the user wishes to play with a different input file). Before running the test again, ensure that your program deletes memory from a previous run.

Sample Run Through

```
====================================
Welcome to CSCI262 Personality Test!
====================================

Please enter the input file name, e.g., "questions.txt": junk.txt
Error opening file "junk.txt"
Please enter the input file name, e.g., "questions.txt": broken_file.txt
Input file 'broken_file.txt' appears to not be a proper file!
Please enter the input file name, e.g., "questions.txt": questions.txt

After prolonged socializing you feel you need to get away and be alone y/n
y
You often contemplate the complexity of life y/n
y
Your decisions are based more on the feeling of a moment than on thorough planning y/n
y
You are inclined to rely more on improvisation than on prior planning y/n
n
You rapidly get involved in the social life of a new club y/n
```

```
no
You tend to rely on your experience rather than on theory y/n
yes
Strict observance of the established rules is likely to prevent attaining a good outcome y/n
asdadssad
Sorry, I didn't recognize your input, please type again
no
You are always looking for opportunities y/n
no
When with a group of people, you enjoy being directly involved and being at the centre of attention y/n
NO
When solving a problem you would rather follow a familiar approach than seek a new one y/n
YES
Your actions are frequently influenced by your emotions y/n
Yes
As a rule, you proceed only when you have a clear and detailed plan y/n
No

Please enter the analysis file name, e.g., "analysis.txt": junk.txt
Error opening file "junk.txt"
Please enter the analysis file name, e.g., "analysis.txt": analysis.txt

==============
Your personality type is: ISFJ
The category is: Sentinels
You are: The Defender
Description:  Very dedicated and warm protectors, always ready to defend their loved ones
==============

Would you like to save?
y
Please enter a file name
output.txt
Would you like to play again?
n
```

Extra Credit

Make your own test!

For extra credit, make a new categorizing test! These can be something like, which starter pokemon would you be? Feel free to show some creativity. Also feel free to modify the input format, however if you do so, make sure you modify the program (or write a new program) that can handle the differences.

Addendum: Styling

For styling, we are mostly looking for consistency and clarity. Some things to be aware of:

- Have good naming of variables
- Be consistent with how you use braces
- Do not leave `//TODO` comments
- Commenting not required, but if you have comments ensure that they are professional
- Consistent Indenting

Submission

Submit a single .zip file to canvas under the assignment Project 1. If you did the extra credit, be sure to write what you did and how it works in the README. The .zip file must contain:

- proj1.cpp
- personality_test.h
- personality_test.cpp
- questions.txt
- analysis.txt
- README
- supplemental stuff for extra credit

Grading:

README	5 points
Program compiles and executes	5 points
Code conforms to the course style guideline	5 points
Program reads in questions.txt correctly	10 points
Program checks for input file formatting	5 points
Program prompts user each question	5 points
Program will not accept bad input for question response	5 points
Program analyzes and prints results correctly	10 points
Program saves the results correctly	5 points
Program works correctly if test is ran again	5 points
TOTAL:	60 points