Data searching in Visual Basic I

Topics

- Reading from CSV files into parallel arrays
- Match and stop searching including handling "no match" situations
- Answering questions using data
- Boolean datatype

TL;DR

Implement a program to analyze passenger data from the RMS Titanic incident.

Background

On April 15, 1912, the RMS Titanic sank due to a collision with an iceberg. Out of 2224 passengers and crew members, only 722 survived. This incident became one of the most infamous shipwreck stories in history, which led to improved safety standards for ships (adapted from here).

It is believed that "*one of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class passengers.*"

In this lab, we will analyze data on a sample of 887 Titanic passengers to evaluate some of these claims. The data is included in file `Titanic.csv` inside your lab folder; it contains the following six columns, as comma-separated values, in the order shown:

- *Passenger ticket class*: 1 means first class, 2 means second class, and 3 means third class
- *Gender*
- *Passenger name*
- *Age in years*
- *Fare paid* for ticket in British pounds
- *Survival status* where 1 means survived, and 0 means died

We also included a smaller version of the data in a file called `TitanicSmaller.csv` which has ten rows and two columns (Passenger ticket class and Gender); this file will prove useful for testing purposes. Please take some time to study BOTH files.

Instructions

1. Launch the VS Express 2013 software and open the project called `Searching_Part1` inside of your lab folder.

   Complete the action for `btnLoadData` so that it reads the values from the file, `Titanic.csv` into six parallel arrays. Output the information for each of the passengers as well as the total number of passengers' information read from the file (should be 887). The output should have the following format:

```
3 male Mr. Owen Harris Braund 22 7.25 False
1 female Mrs. John Bradley (Florence Briggs Thayer) Cumings 38 71.28 True
3 female Miss. Laina Heikkinen 26 7.93 True
1 female Mrs. Jacques Heath (Lily May Peel) Futrelle 35 53.1 True
...
1 male Mr. Karl Howell Behr 26 30 True
3 male Mr. Patrick Dooley 32 7.75 False
-----
Total passengers in file: 887
```

Checkpoint 1 (30/100)

- ☐ project populates six parallel arrays with values from the file `Titanic.csv`
- ☐ project produces correctly formatted output according to the example above

2. Copy and paste the folder `Searching_Part1`. Rename the copy `Searching_Part2`. Launch the VS Express 2013 software and open the project `Searching_Part2`.

   Complete the action for `btnSeachByName` to search for a passenger, whose name is input by the user, and display their name, ticket class, age, fare paid, and survival status. An error message should be displayed if the input name is not valid. You MUST solve this problem using match and stop searching with a Boolean variable. For example, if the user input "*Miss. Laina Heikkinen*", then the output would be:

   ```
   Miss. Laina Heikkinen 3 26 7.93 1
   ```

   Next, change your program so that it displays `SURVIDED` if the passenger survived (instead of a 1) or `DIED` otherwise (instead of a 0). Test your project for the following input cases and make sure that it produces the expected output formatted EXACTLY as shown below.

Table 1

Input values for passenger name	**Expected program output**
"Miss. Laina Heikkinen"	`Miss. Laina Heikkinen 3 26 7.93 SURVIVED`
"MISS. Laina Heikkinen"	`MISS. Laina Heikkinen PRODUCED NO MATCHES`
"Mr. James Moran"	`Mr. James Moran 3 27 8.46 DIED`
" Mr. James Moran "	`  Mr. James Moran  PRODUCED NO MATCHES`
"Charles"	`Charles PRODUCED NO MATCHES`

Checkpoint 2 (65/100)

- ☐ project uses match and stop searching with a Boolean variable
- ☐ project produces correct search output for any passenger name, even those that do not exist
- ☐ project produces correct output for all test cases in Table 1

3. Copy and paste the folder `Searching_Part2`. Rename the copy `Searching_Part3`. Launch the VS Express 2013 software and open the project `Searching_Part3`.

   Update the action for `btnSeachByName` to make your program handle capitalization issues (such as the second case in Table 1) and additional spaces at the beginning and end of the input name (such as the fourth case in Table 1). The table below shows the new excepted output for your project to produce.

Table 2	
Input values for passenger name	**Expected program output**
"Miss. Laina Heikkinen"	Miss. Laina Heikkinen 3 26 7.93 SURVIVED
"MISS. Laina Heikkinen"	Miss. Laina Heikkinen 3 26 7.93 SURVIVED
"Mr. James Moran"	Mr. James Moran 3 27 8.46 DIED
" Mr. James Moran "	Mr. James Moran 3 27 8.46 DIED
"Charles"	Charles PRODUCED NO MATCHES

Checkpoint 3 (70/100)

- ☐ project uses match and stop searching with a Boolean variable
- ☐ project produces correct search output for any passenger name, even those that do not exist
- ☐ project produces correct output for all test cases in Table 2

4. Copy and paste the folder `Searching_Part3`. Rename the copy `Searching_Part4`. Launch the VS Express 2013 software and open the project `Searching_Part4`.

   Update the action for `btnSeachByName` to allow for partial matches (such as the fifth case in Table 1). The table below shows the new excepted output for your project to produce.

Table 3	
Input values for passenger name	**Expected program output**
"Miss. Laina Heikkinen"	`Miss. Laina Heikkinen 3 26 7.93 SURVIVED`
"MISS. Laina Heikkinen"	`Miss. Laina Heikkinen 3 26 7.93 SURVIVED`
"Mr. James Moran"	`Mr. James Moran 3 27 8.46 DIED`
" Mr. James Moran "	`Mr. James Moran 3 27 8.46 DIED`
"Charles"	`Mr. Charles Eugene Williams 2 23 13 SURVIVED`

Checkpoint 4 (75/100)

- ☐ project uses match and stop searching with a Boolean variable
- ☐ project produces correct search output for any passenger name, including partial names and even those that do not exist
- ☐ project produces correct output for all test cases in Table 3

Submission Instructions

During our next lab meeting, you will be asked to expand your solutions to Part 4. Prior to the meeting, you are expected to

- finish ALL parts of this lab,
- study (again) any material you struggled with in this lab, and
- study new material needed for the next lab

You will submit your work for this lab and the next one together by midnight Thursday.