Programming Languages (CS424/CS524)

Programming Assignment II

Mar 15, 2020

Prepared By : Pooja Khanal , Anushka Bhattacharjee

Includes :

- **Software Description Document**
- **Software Test Plan Document**

Table of contents

Software Description Document

1.System Overview

A program is developed in GO programming language to read Baseball Player objects from an input file into a list/collection. The players are stored in a collection in such a way that when they are displayed to the screen, they are in sorted order by Lastname(then the first name to break name ordering tiles). Each line of the data file will contain one player's name and stats.

{ firstname lastname plateappearances atbats singles doubles triples homeruns walks hitbypitch}

Also, It is possible there are errors in the data lines. The errors are checked for that each numeric value is valid, and that all 10 values are on the line. The program reads until the end of the file to read for players.

2.Referenced Documents

[1] Programming Assignment II, CS-424/524 Spring 2020

[2] https://doc.rust-lang.org/

3. Concepts of Execution

3.1. Operations/Computations

batting average: this is the sum of the hits (singles, doubles, triples, home runs) divided by the number of at-bats.

slugging percentage The slugging percentage is a weighted average. It is computed by:

slugging = (*singles*+2∗*doubles*+3∗*triples*+4∗h*omeruns*)/*number of at bats*

on base percentage OBP is the sum of all hits, walks and hit-by-pitch divided by the number of plate appearances.

Team's overall batting average is also computed.

3.2. User input and Prompts

The user is asked for **inputfilename, Y/N** to declare whether to write the output to file or not, and **outputfilename** if to be written.

4.Abstract Data type (ADT)

4.1. Player structure

The player structure has the following attributes.
-{firstname lastname plateappearances atbats singles doubles triples homeruns walks hitbypitch}
Each player is added to the list of players after reading from file, one by one

5.Code Outline

List of Functions used

1. **pub fn get_batting_average(&self) -> f32**
   a. **Purpose:** Calculate the batting average of each player.Calculation mentioned above
   b. **Argument:** None
   c. **Return Type:** f32

2. **pub fn get_slugging_percentage(&self) -> f32**
   a. **Purpose:** Calculate the slugging percentage of each player. Calculation mentioned above
   b. **Argument:** None
   c. **Return Type:** f32

3. **pub fn get_on_base_percentage(&self) -> f32**
   a. **Purpose:** Calculate the Base percentage of each player. Calculation mentioned above
   b. **Argument:** None
   c. **Return Type:** f32

4. **pub fn get_player(s: &Vec<&str>, line_count: i32) -> Result<Player, String>**
   a. **Purpose:** get the details of the player and create an instance of each Player Structure
   b. **Argument:** Each string of line, number of lines
   c. **Return Type:** Result Type which either contains player or error as a string

5. **fn check_errors(s: &Vec<&str>, line_count: i32) -> Result<(), String>**
   a. **Purpose:** check for errors in the line
   b. **Argument:** Each string of line, number of lines
   c. **Return type:**Result Type which contains error as a string

6. **fn get_all_players() -> Result<Vec<Result<Player, String>>, String>**
   a. **Purpose:** Read from the file and append each player to the Player Structure
   b. **Argument:** None
   c. **Return Type:** Result Type of multiple players and their errors in each line

7. **fn print_error_report(errors: Vec<&String>, out: &mut BufWriter<Box<dyn Write>>)**
   a. **Purpose:** print the overall error report
   b. **Argument:** vector of errors , output writer
   c. **Return type:** void

8. **fn print_player_report(players: Vec<&Player>, out: &mut BufWriter<Box<dyn Write>>)**
   a. **Purpose:** print the overall player report with the tabular structure
   b. **Argument:** vector of players, output writer
   c. **Return type:** void

9. **pub fn handle_print_to_file(results: &Vec<Result<Player, String>>)**
   a. **Purpose:** Output stream handler to write to file
   b. **Argument:** vector of players and errors combined
   c. **Return Type:** void

10. **fn main()**
    a. **Purpose:** The main function to run the program. This is implemented in main.rs
    b. **Arguments:** Not Applicable
    c. **Return type:** void

d. It contains the test abstraction and calls the required test function for running the application.

Software Test Plan Document

1. Test Procedures

a. **Purpose:** To test the validity of software and to test if the software is acceptable to the market or not. For this program, various test cases are created

b. **Procedures followed for Testing and Testing Strategies**

i. **Black box testing**

1. The input file was processed and seen if the output was written correctly to the output file or not.
2. Below mentioned test cases were run.

ii. **White box testing**

1. Modular approach was taken and each function was tested to see if it works or not. For such, the return type and return values were also compared.
2. After all the modules (functions) were tested, all the modules were integrated into one program and tested.

2. Test cases

1. File without errors
2. File with all type of errors
3. File with one type of error
4. File with the line having both types of error
5. Integers with long int (instead of string)
6. Alphabetically sorted or not

3. Tested Environments:

1. cargo 1.41.0 (626f0f40e 2019-12-03) / rustc 1.41.1 (f3e1a954d 2020-02-24)
2.

4. Sample Runs

Command to compile and build:

- **Cargo run** in the cargo directory (baseball directory)
- The input file must be in the cargo directory (baseball directory)

a. Sample Input I

```
Hank Aaron 13941 12364 2294 624 98 755 1402 32

Chipper Jones 10614 8984 1671 2020 38 468 567 897

Ty Cobb 111 11434 3053 724 295 117 1249 94 889

Jonny Bench 8674 7658 1254 212 24 389 891 19

Tony Gwynn 2147483647 9288 2378 543 85 135 434 24

John Smoltz 1167 948 118 26 2 5 79 67 89
```

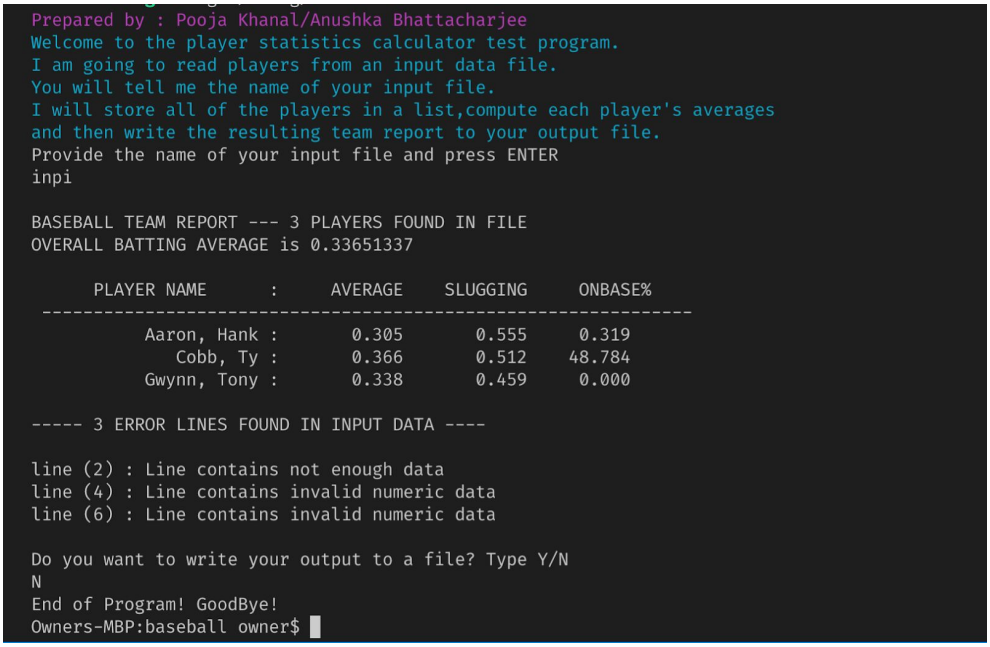
Output :

```
Prepared by : Pooja Khanal/Anushka Bhattacharjee
Welcome to the player statistics calculator test program.
I am going to read players from an input data file.
You will tell me the name of your input file.
I will store all of the players in a list,compute each player's averages
and then write the resulting team report to your output file.
Provide the name of your input file and press ENTER
inpu_sam

BASEBALL TEAM REPORT --- 6 PLAYERS FOUND IN FILE
OVERALL BATTING AVERAGE is 0.3135585

      PLAYER NAME      :     AVERAGE    SLUGGING     ONBASE%
 ---------------------------------------------------------------
           Aaron, Hank :       0.305       0.555     0.319
          Bench, Jonny :       0.245       0.432     0.277
              Cobb, Ty :       0.366       0.512    48.784
           Gwynn, Tony :       0.338       0.459     0.000
         Jones, Chipper :       0.467       0.857     0.489
          Smoltz, John :       0.159       0.207     0.250

----- 0 ERROR LINES FOUND IN INPUT DATA ----


Do you want to write your output to a file? Type Y/N
N
End of Program! GoodBye!
```

b. Sample Input II

```
Hank Aaron 13941 12364 2294 624 98 755 1402 32
Chipper Jones 10614 8984 1671 -2.345 38 468 1512
Ty Cobb 111 11434 3053 724 295 117 1249 94 889
Jonny Bench 8674 7658 1254 string 24 389 891 19
Tony Gwynn 2147483647 9288 2378 543 85 135 434 24
John Smoltz 11677777777777777777 948 118 26 2 5 79 67 89
```

Output:



5. Assumptions

1. The file format (input) is always consistent.
2. There is no newline at the end of the input file.
3. Round of to 3 decimal places for all float values.