Joshua Liu

105136031

October 26, 2018

CS 31

Description of Obstacles I Overcame

While working on the program, it was hard for me to determine a way to process the string. I was stumped at first about how to begin checking the string, but I soon developed a way for the program to read each section of the string and delete it from a temporary string to keep everything organized as much as possible. Another obstacle I encountered was making sure that every possible case of inputs was considered, especially for the function that checks the syntax. To overcome this, I needed to think of various cases to test the program with and ensure that every possibility was correctly taken care of. In general, creating the framework of the program was the hardest part as well as determining which sections of the string-checking code should be looped and when to break out of the loops, but taking time to think about every possible case helped through the process.

Description of the design of my program

The general concept of my program is that it duplicates the poll string data into another temporary string, processes the temporary string by reading the next sections of the code(state code, digits, party code, or a comma), interpreting it, and then subtracting each consecutive individual sections from the temporary string and begin processing another.

Include all the needed libraries(<iostream>, <cctype>, <cassert>, <string>, as well as “using namespace std;”.

isValidUppercaseStateCode(string stateCode) was provided for us, determines if state code is valid.

bool hasProperSyntax(string pollData)

In this function, I utilized two do-while loops, one nested in the other, to test for the syntax in order for the program to run at least once before looping again. Before the loops, the string is checked to see if it should enter the loop. The first loop tests for the state code, and the second loop checks everything proceeding the state code.

Pseudocode:

*make all letters in poll data uppercase*

*if poll data has spaces*

*false*

*if poll data is empty string*

*true*

*duplicate poll data onto a temporary string*

*repeatedly (if string length less than two):*

*check state code*

*if not valid*

*break*

*subtract state code from temporary string*

*repeatedly:*

*check for empty string and comma*

*if next char is not digit*

*return false*

*find next digit or two digits (predicted seats)*

*if not, return false*

*subtract digits from temporary string*

*if next char is not a letter (party code)*

*return false*

*subtract letter*

*if temporary string empty*

*return true*

*remove ',' for next section of poll data*

*return false if string less than two*

int tallySeats(string pollData, char party, int& seatTally)

This function processes the string like bool hasProperSyntax(string pollData), but instead of checking for syntax as its main job, it determines what to return and the number of seats for a given party code.

Pseudocode:

int tallySeats(string pollData, char party, int& seatTally)

*if not proper syntax(calls hasProperSyntax function)*

*return 1*

*if party code is not alphabet*

*return 2*

*make all letters in poll data uppercase*

*if party character is valid and poll data is empty*

*seat perdition is zero and return 0*

*repeatedly:*

*remove state code from string*

*repeatedly:*

*if string is empty or a comma*

*break*

*find the digits of prediction and convert to int*

*assign value to corresponding variable*

*remove digits and party code from string*

*if string is empty*

*break*

*if string is empty*

*break*

*make party code uppercase*

*give the seat tally the correct value*

*return 0*

The main function consists of testing data for the functions.

Data to Test Program

hasProperSyntax 🡪 return true

Just the state code, 2 uppercase (“AZ”)

Just the state code, 1 uppercase (“aZ”)

Just the state code, 0 uppercase (“az”)

State code with number of polls, single and double digits, and party codes (“tX12r6m”)

State code with number of polls and 1 party code (“ut10u”)

State code with two states and multiple tallies/party codes (“IA3r7y9q,Wv6q2s9k”)

One state and mixed uppercase/lowercase letters (“Ca4r6d9i”)

Empty string poll data (“”)

Number of seats begins with 0 (“ga07t,ne0d02k”)

!hasProperSyntax 🡪 return true

Invalid state code, 2 uppercase (“ZA”)

Invalid state code, 1 uppercase (“Za”)

Invalid state code, 0 uppercase (“za”)

Invalid party code character (“gA5%i$”)

Invalid party code character in second section of string (“gA5k1l,HIe3\*i”)

Spaces within poll data (“f l2k”)

Spaces within second section of string (“Cor4,WI4 w”)

Space before string (“ OH”)

Space after string (“OH ”)

String with space(“ “)

Poll data with empty section(“La4j,,ky1e”)

tallySeats

Valid poll data that counts 22 z’s, seats == 2, return 0, multiple digits in seats, and has section with only state code "CT7d,NY1t18z1I,vT,Ne07r00z04z", 'z', seats)

Valid poll data that returns 2 since party character is not a letter, seats unchanged ("de8i4f,nc5y8v", '%', seats)

Valid poll data with empty string and E, seats == 0 and return 0 ("", 'E', seats)

Invalid poll data that returns 1, leave seats unchanged ("pa4#", 'E', seats”)

Invalid poll data with invalid party code, returns 1 and leaves seats unchanged ("ne4^", '(', seats)

Valid poll data, return 0, 23 for seats ("ca9R16D1I,IA,ks3d7r4D,NJ7r", 'r', seats)

Same valid poll data as previous but different party code, return 0, 23 for seats ("ca9R16D1I,IA,ks3d7r4D,NJ7r", 'd', seats)

Same valid poll data as previous but different party code, return 0, 1 for seats ("ca9R16D1I,IA,ks3d7r4D,NJ7r", 'i', seats)

Valid poll data, return 0, 0 for seats since party code is not in any of the sections (“ca4l,md8f,nc7y”, ‘z’, seats)

Valid poll data, return 0, 1 for seats, every alphabet for party code ("mn1a1b1c1d1e1f1g1h,ia1i1j1k1l1m1n1o,pa1p1q1r1s,wy1t1u1v1w1x1y1z", 't', seats)