323.35 Project 2: Password Checker C++

Student: Matthew F Flammia
Project Due Date: 9/7/2020

****
IV. main ()
****

```
Step 0: outFile  open argv[1] to write
Step 1: displayRules (outFile)

Step 2: askPasswd (outFile)
 password  screen input from the user
 passwordLength <-- the length of password (use strlen)

step 3: repeat step 2 if the length of password is NOT within the range of 8 - 32
 print error message
step 4: i  0

step 5: index  checkCharType(password[i]) // make sure the index is within 0 - 4
 charCount[index] ++

step 6: i++
step 7: repeat step 5 to step 6 until the last password character is checked.
step 8: validYesNo  checkRules ()
 if validYesNo is not good (!= 1)
 call displayFail(outFile)
step 9: repeat step 1 to step 8 if validYesNo is 0
step 10: display and ask user to re-type his/her password; also write to outFile

step 11: secondPassword  from the user
step 12: matchYesNo  matching (password, secondPassword)

step 13: if matchYesNo is no good (== 0)
 displayMatchFail(outFile)
step 14: repeat step 1 to step 13 until matchYesNo == 1
step 15: call displaySucess (outFile)
```

Source Code

```
#include <iostream>
#include <string>
#include <fstream>

using namespace std;

char* outfile;

/**
* Matthew F Flammia, 23661371
* 323.35 Designs and Analysis of Algorithms
* To compile: g++ -o project2 FlammiaM_Project2_CPP.cpp
* To run: project2.exe outfile.txt
**/

class passWordChecker{
      public:
            string password;
            string secondPassword;
            int passwordLength;
            int charCount[5];

            passWordChecker(){
                  password = "";
                  secondPassword ="";
                  passwordLength = 0;
                  for(int i=0;i<5;i++)
                        charCount[i] = 0;
            }

            //displays the rules, requires ofstream input
            void displayRules(ofstream &output){
                  output.open(outfile,ios::out | ios::app);
                  cout<<"Please create a password using the following:\n";
                  cout<<"1) The password length: 8-32 Characters\n";
                  cout<<"2) Must use at least one number\n";
                  cout<<"3) Must use at least one upper case character\n";
                  cout<<"4) Must use at least one lower case character\n";
                  cout<<"5) Must use at least one of the specified special
characters:\n   # $ * ( ) % & ^\n";

                  output<<"Please create a password using the following:\n";
                  output<<"1) The password length: 8-32 Characters\n";
                  output<<"2) Must use at least one number\n";
                  output<<"3) Must use at least one upper case character\n";
                  output<<"4) Must use at least one lower case character\n";
                  output<<"5) Must use at least one of the specified special
characters:\n   # $ * ( ) % & ^\n";

                  output.close();
            }
            //asks for password, stores password and password length into
object variables
            string askPasswd(ofstream &output){
                  output.open(outfile,ios::app);
                  cout<<"Please enter your password\n";
                  output<<"Please enter your password\n";
                  cin>>this->password;
                  this->passwordLength = this->password.length();
                  output<<password<<endl;
                  output.close();
                  return this->password;
            }

            //final output message
            void displaySuccess(ofstream &output){
                  output.open(outfile,ios::app);
                  cout<<"Your password will be updated shortly...\n";
                  output<<"Your password will be updated shortly...\n";
                  output.close();
            }

            //fail message when one of the rules for a password are not met
            void displayFail(ofstream &output){
                  output.open(outfile,ios::app);
                  cout<<"Your password failed one or more password rules\n";
                  output<<"Your password failed one or more password rules\n";
                  output.close();
            }

            //fail message when passwords dont match during retype phase
            void displayMatchFail(ofstream &output){
                  output.open(outfile,ios::app);
                  cout<<"Match fail....\n";
                  output<<"Match fail....\n";
                  output.close();
            }

            //passwords character checker, which goes character by character
and modifies the objects charCount array
            void checkCharType(char ch){
                  //0 stores illegal special chars
                  //1 stores numerics
                  //2 stores lowercase
                  //3 stores uppercase
                  //4 stores legal special chars # $ % & ( ) * ^

                  /**
                  * I used checking based on the ASCII value of the character
                  * This method makes the most sense since we are checking for
                  * specific characters that are in sequence with each other
                  * specifically 0-9, a-z, A-Z. The only multicase is the
                  * special characters, but this is easily achieved through OR
statements.
```

```
                  * This also avoids any characters outside of the legal
range.
                  **/

                  //checks numerics
                  if(ch >=48 && ch <=57)
                        this->charCount[1]++;
                  //checks lowercase
                  else if(ch >=97 && ch <=122)
                        this->charCount[2]++;
                  //checks uppercase
                  else if(ch >=65 && ch <=90)
                        this->charCount[3]++;
                  //checks legal special chars
                  else if((ch >=35 && ch <=38) || (ch >=40 && ch <=42) ||
(ch==94))
                        this->charCount[4]++;
                  //any illegal characters
                  else
                        this->charCount[0]++;
            }

            //checks objects charCount array that 0 index is 0 and all other
catagories
            bool checkRules(){
                  if(this->charCount[0] != 0){
                        return false;
                  }
                  for(int i=1;i<5;i++){
                        if(this->charCount[i]==0)
                              return false;
                  }
                  return true;
            }

            //checks if 2 strings are the same, by going character by
character
            bool matching(string s1, string s2){
                  int strlen1 = s1.length();
                  int strlen2 = s2.length();
                  if(strlen1 != strlen2)
                        return false;
                  for(int i=0;i<strlen1;i++){
                        if(s1[i]!=s2[i])
                              return false;
                  }
                  return true;
            }

            //takes user input and stores it to objects secondPassword
variable
            void retypePassword(ofstream &output){
                  output.open(outfile,ios::app);
                  cout<<"Please retype your password\n";
                  output<<"Please retype your password\n";
                  cin>>this->secondPassword;
                  output<<this->secondPassword;
                  output.close();
            }
};

int main(int argc, char* argv[]){
      if(argc <= 1 || argc >=3){
            cout<<"Must include only 1 filename in command line arg\n";
            return 0;
      }
      //step 0
      outfile = argv[1];
      ofstream output;

step1:
      passWordChecker usersPassword;
      //step 1
      usersPassword.displayRules(output);
      //step 2
      usersPassword.askPasswd(output);
      //step 3
      while(usersPassword.passwordLength < 8 || usersPassword.passwordLength >
32){
            usersPassword.displayFail(output);
            usersPassword.displayRules(output);
            usersPassword.askPasswd(output);
      }
      //step 4, 5, 6, 7
      for(int i=0;i<usersPassword.passwordLength;i++){
            usersPassword.checkCharType(usersPassword.password[i]);
      }
      //step 8, 9
      if(!usersPassword.checkRules()){
            usersPassword.displayFail(output);
            goto step1;
      }
      //step 10
      int attempts = 0;
      while(attempts != 3){
            usersPassword.retypePassword(output);
            if(usersPassword.matching(usersPassword.password,
usersPassword.secondPassword)){
                  usersPassword.displaySuccess(output);
                  return 0;
            }
            usersPassword.displayMatchFail(output);
            attempts++;
      }
      //if they fail to retype 3 times the process resets
      goto step1;
      return 0;
}
```

Program Output

!COMMENTS HAVE BEEN ENTERED TO SHOW FAIL CASE NUMBER!

```
Please create a password using the following:
1) The password length: 8-32 Characters
2) Must use at least one number
3) Must use at least one upper case character
4) Must use at least one lower case character
5) Must use at least one of the specified special characters:
  # $ * ( ) % & ^
Please enter your password
aA$3                                    //FAIL CASE 1: LESS THAN 8 CHARACTERS
Your password failed one or more password rules
Please create a password using the following:
1) The password length: 8-32 Characters
2) Must use at least one number
3) Must use at least one upper case character
4) Must use at least one lower case character
5) Must use at least one of the specified special characters:
  # $ * ( ) % & ^
Please enter your password
aA$3butthistimeitsreallylongimeanlongerthanyouwouldevermakeapasswordbufferoverflowcheckbasically
//FAIL CASE 2: MORE THAN 32 CHARACTERS
Your password failed one or more password rules
Please create a password using the following:
1) The password length: 8-32 Characters
2) Must use at least one number
3) Must use at least one upper case character
4) Must use at least one lower case character
5) Must use at least one of the specified special characters:
  # $ * ( ) % & ^
Please enter your password
abcdABCD123$!                           //FAIL CASE 3: ILLEGAL SPECIAL CHARACTERS
Your password failed one or more password rules
Please create a password using the following:
1) The password length: 8-32 Characters
2) Must use at least one number
3) Must use at least one upper case character
4) Must use at least one lower case character
5) Must use at least one of the specified special characters:
  # $ * ( ) % & ^
Please enter your password
abcdABCD$                                      //FAIL CASE 4: MISSING NUMERIC
Your password failed one or more password rules
Please create a password using the following:
1) The password length: 8-32 Characters
2) Must use at least one number
3) Must use at least one upper case character
4) Must use at least one lower case character
5) Must use at least one of the specified special characters:
  # $ * ( ) % & ^
Please enter your password
abcdabcd5&                                     //FAIL CASE 5: MISSING UPPER CASE
```

```
Your password failed one or more password rules
Please create a password using the following:
1) The password length: 8-32 Characters
2) Must use at least one number
3) Must use at least one upper case character
4) Must use at least one lower case character
5) Must use at least one of the specified special characters:
   # $ * ( ) % & ^
Please enter your password
ABCDABCD5%                              //FAIL CASE 6: MISSING LOWERCASE
Your password failed one or more password rules
Please create a password using the following:
1) The password length: 8-32 Characters
2) Must use at least one number
3) Must use at least one upper case character
4) Must use at least one lower case character
5) Must use at least one of the specified special characters:
   # $ * ( ) % & ^
Please enter your password
abcdABCD5                          //FAIL CASE 7: MISSING SPECIAL CHARACTER
Your password failed one or more password rules
Please create a password using the following:
1) The password length: 8-32 Characters
2) Must use at least one number
3) Must use at least one upper case character
4) Must use at least one lower case character
5) Must use at least one of the specified special characters:
   # $ * ( ) % & ^
Please enter your password
thePerfectPassword1999&()*
Please retype your password
thePerfectPassword1999&()*
Your password will be updated shortly…     //FINAL OUTPUT ON SUCCESS
```