Bryan Nguyen

Professor Lehr

CIS17A

28 October, 2016

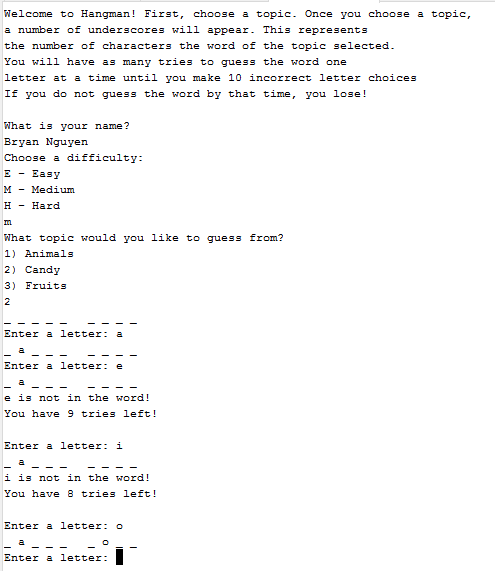
**Introduction–**

This program is a game that many people know by Hangman. It is a game where the user guesses a word, letter by letter. If the user is able to guess all of the letters in the word before they make too many wrong choices, they win. If not, they lose. I wrote this program because I have always enjoyed playing games and want to know how these games are created, so that one day I will be skilled enough to be able to code my own, unique games in the future. Coding this project was important to learn and solidify my knowledge on what I know now. It also shows me how to think deeply about what needs to be done next that would not break the whole program.

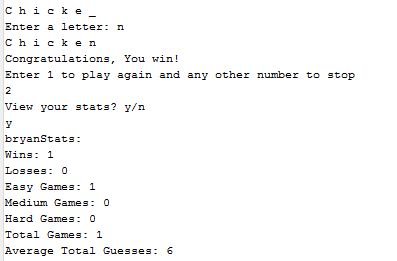
**Summary-**

The code is approximately 450 lines long. The program takes much use on dynamically allocated variables, along with much use of structures in functions. The program works fairy well as a first project but it could use some final touches which could have been done given a little more time. Binary file i/o also works well and saves on exit. Major variables in this program would definitely be the struct variable along with all of the components inside such as the number of wins, losses, averages, and easy, medium, and hard games played. The total number of games could easily be added but it did not seem needed. There was a single structure in the program since any additional structures seemed unnecessary. Coding the program, for the most part, was fairly easy. What was difficult, however, was organizing my functions. Because many of my function arguments included the structure variable, it was difficult to separate them. The reason is that many of these functions intertwined and needed to be inside another function and another. The problem arose when my several functions were not using the structure variable I expected it to use. Another issue is that in very few lines of code, they do nothing. I was expected to use them in later in the project but changed my mind as time passed. The reason I kept them in is because, for some reason, if I take them out, the program does not run properly. Line 46 in my main function and line 20 of my header file are perfect examples of this. I am unable to figure out why the program stops working after the first few lines by variables I have never used. What I do know however, is that my program creates a stack dump file when I remove them. It makes me think that it is due to my file i/o but I see nothing wrong with it. This program took around 6 hours for it to work as expected. Most, if not all of my code has already been discussed in class. A popular function to not is the strlen() which greatly helped out with the success of my project.

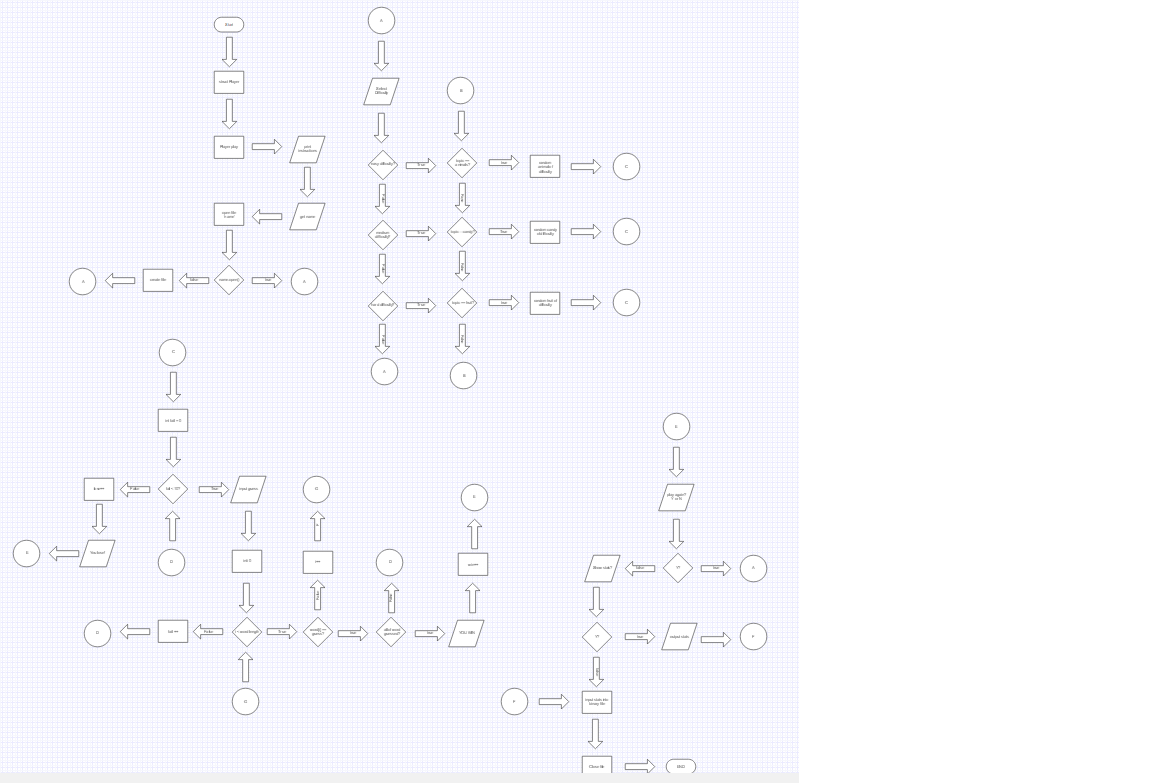
**Description-**



This is the output from the program. The introduction and rules are stated in the beginning and the user is asked for the name to open his/her own personal file. After some thought, opening up separate files for each user sounds inefficient if dealing with larger quantities of people. The name is accepted into a string and .dat is added to it. From there, the file is created or read. For example, from this output, if Bryan Nguyen.dat is not found in a file, Bryan Nguyen.dat will be created.



This is the output when the user beats the game. They enter 1 to play again or enter any other number to stop playing. If they choose to stop playing, all data will be added to the file and the user will be asked if they want to see their stats. By the way, the part where it says “bryanStats” is now fixed. It will now properly output “bryan’s stats.” I am running out of time so I can only use a verbal correction.



This is the flowchart of the program. It is quite large mostly because of how many for loops there are. I also do not know where to find the lines found in other flowcharts so hopefully large arrows would be ok.

**Pseudocode –**

If the start button is pressed

Output introduction and instructions

Get name and open file of name.dat

Display the difficulties of the game

User inputs what difficulty they want

Display multiple topics

User selects what topics they want to guess from

While failures is less than 10

Display “ \_ “ for each character of the random word chosen.

User inputs character

If the inputted character is in the random word

The “ \_ “ turns into the letter

Else

Failures++

If the random word is guessed before failure = 10

User wins

Else

User loses

If user plays again

Start again

Else

Output data into the user’s file

End

**Major Variables:**

Datatype Variable

Structure play - The player structure that stores all information relating to the

Player such as wins, losses, easy, medium, and hard games

Played. Located in **LINE 43**

Char\* name - Inside structure variable. Gets the name of the user. Located in

**LINE 56**

Int easy - Inside struct. Contains how many easy difficulty games

User has played **LINE 167**

Int medium -Inside struct. Contains how many medium difficulty games

User has played **LINE 168**

Int hard - Inside struct. Contains how many hard difficulty games

User has played **LINE 169**

Int win - Inside struct. Total number of wins **LINE 357, 365**

Int lose - Inside struct. Total number of losses **LINE 373**

Int\* guess - Inside struct. Total number of guesses **LINE 324**

Float avgG - Inside struct. Average number of guesses of total games. **88**

Fstream acnt - BINARY FILE IO **LINES 76 – 111**

**All or most of these variables are outputted in LINES 377-387**

**CODE(MAIN FILE)**

/\*

\* File: main.cpp

\* Author: Bryan Nguyen

\*

\* Created on October 27, 2016, 1:58 AM

\*/

#include <cstdlib>

#include <iostream>

#include <fstream>

#include <ctime>

#include <cstring>

using namespace std;

#include "playerData.h"

int SIZE = 10;

Player newPlay(Player);

void account(Player);

void instr(); // instructions

char dfclt(Player); // choose difficulty

int topic(); // choose topic to guess from

void enter(char, Player&); // enter level of difficulty

void eMode(Player&); // easy mode

string eAnim(); // easy animal topic vocabulary

string eCandy(); // easy candy topic vocabulary

string eFruit(); // easy flavor topic vocabulary

void mMode(Player&); // medium mode

string mAnim(); // medium animal topic vocabulary

string mCandy(); // medium candy topic vocabulary

string mFruit(); // medium flavor topic vocabulary

void hMode(Player&); // hard mode

string hAnim(); // hard animal topic vocabulary

string hCandy(); // hard candy topic vocabulary

string hFruit(); // hard flavor topic vocabulary

void play(string, Player&); // enter game of subject

void stats(Player&);

int main(int argc, char\*\* argv) {

srand(static\_cast<unsigned int>(time(0)));

int players = 0;

Player play;

instr();

account(newPlay(play));

fstream file;

return 0;

}

Player newPlay(Player user)

{

fstream acnt;

string fName;

cout << endl << "What is your name?" << endl;

cin.getline(user.name, 50);

fName = user.name;

fName += ".dat";

user.easy = 0;

user.medium = 0;

user.hard = 0;

user.win = 0;

user.lose = 0;

user.avgG = 0;

acnt.open(fName.c\_str(), ios::in | ios::binary);

acnt.read(reinterpret\_cast<char\*>(&user), sizeof(user));

acnt.close();

return user;

}

void account(Player user)

{

char x;

char stat;

fstream acnt;

string fName;

fName = user.name;

fName += ".dat";

do{

acnt.open(fName.c\_str(), ios::out | ios::binary);

if(!acnt)

cout << "Unregistered Account. Creating your Account..." << endl;

acnt.close();

enter(dfclt(user), user);

int games = user.easy+user.medium+user.hard;

user.avgG += user.guess[games];

user.avgG /= static\_cast<float>(games);

cout << "Enter 1 to play again and any other number to stop" << endl;

cin >> x;

while(isalpha(x))

{

cout << "Not a number. Enter a number" << endl;

cin >> x;

}

} while(x=='1');

cout << "View your stats? y/n" << endl;

cin.ignore();

cin >> stat;

if(tolower(stat) == 'y')

stats(user);

acnt.open(fName.c\_str(), ios::out | ios::binary);

acnt.write(reinterpret\_cast<char\*>(&user), sizeof(user));

acnt.close();

acnt.open(fName.c\_str(), ios::in | ios::binary);

acnt.read(reinterpret\_cast<char\*>(&user), sizeof(user));

acnt.close();

delete[] user.guess;

}

void instr()

{

cout << "Welcome to Hangman! First, choose a topic. Once you choose a topic, ";

cout << endl << "a number of underscores will appear. This represents ";

cout << endl << "the number of characters the word of the topic selected. ";

cout << endl << "You will have as many tries to guess the word one ";

cout << endl << "letter at a time until you make 10 incorrect letter choices ";

cout << "\nIf you do not guess the word by that time, you lose!" << endl;

}

char dfclt(Player user)

{

char dfclt;

cout << "Choose a difficulty: " << endl;

cout << "E - Easy" << endl << "M - Medium" << endl << "H - Hard" << endl;

cin >> dfclt;

while(isdigit(dfclt) || ((toupper(dfclt) != 'E') && (toupper(dfclt) != 'M')

&& (toupper(dfclt) != 'H')))

{

cout << "Enter a valid choice" << endl;

cin >> dfclt;

}

dfclt = toupper(dfclt);

if(dfclt == 'E')

user.easy++;

if(dfclt == 'M')

user.medium++;

if(dfclt == 'H')

user.hard++;

return dfclt;

}

int topic()

{

char topic;

cout << "What topic would you like to guess from?" << endl;

cout << "1) Animals" << endl;

cout << "2) Candy" << endl;

cout << "3) Fruits" << endl;

cin >> topic;

while(isalpha(topic) || (topic != '1' && topic != '2' && topic != '3'))

{

cout << "Enter a valid number" << endl;

cin >> topic;

}

return topic;

}

void enter(char dfclt, Player& user)

{

switch(dfclt)

{

case 'E': ++user.easy;eMode(user);break;

case 'M': ++user.medium;mMode(user);break;

case 'H': ++user.hard;hMode(user);break;

default: cout << "Game will now exit" << endl;

}

}

void eMode(Player& user)

{

char choice = topic();

if(choice == '1')

play(eAnim(), user);

if(choice == '2')

play(eCandy(), user);

if(choice == '3')

play(eFruit(), user);

}

string eAnim()

{

string animal[SIZE] = {"Dog", "Cat", "Rabbit", "Bird", "Fish", "Chicken",

"Cow", "Horse", "Lion", "Pig"};

int random = rand()%10;

return animal[random];

}

string eCandy()

{

string candy[SIZE] = {"Twix", "Starburst", "KitKat", "Crunch",

"Butterfinger", "Snickers", "Hersheys",

"Twizzlers", "Reeses", "Skittles"};

int random = rand()%10;

return candy[random];

}

string eFruit()

{

string fruit[SIZE] = {"Apple", "Banana", "Pear", "Kiwi", "Orange",

"Peach", "Lemon", "Cherry", "Grape", "Lime"};

int random = rand()%10;

return fruit[random];

}

void mMode(Player& user)

{

int choice = topic();

if(choice == '1')

play(mAnim(), user);

if(choice == '2')

play(mCandy(), user);

if(choice == '3')

play(mFruit(), user);

}

string mAnim()

{

string animal[SIZE] = {"Elephant", "Kangaroo", "Monkey", "Penguin",

"Cheetah", "Camel", "Squirrel", "Zebra",

"Turtle", "Snail"};

int random = rand()%10;

return animal[random];

}

string mCandy()

{

string candy[SIZE] = {"Candy Corn", "York Peppermint", "Jelly Belly",

"Three Musketeers", "Tootsie Rolls", "Milky Way",

"Swedish Fish", "Junior Mints", "Peeps", "Baby Ruth"};

int random = rand()%10;

return candy[random];

}

string mFruit()

{

string fruit[SIZE] = {"Apricot", "Blackberry", "Coconut", "Cranberry",

"Fig", "Guava", "Mango", "Papaya", "Strawberry",

"Honeydew"};

int random = rand()%10;

return fruit[random];

}

void hMode(Player& user)

{

int choice = topic();

if(choice == '1')

play(hAnim(), user);

if(choice == '2')

play(hCandy(), user);

if(choice == '3')

play(hFruit(), user);

}

string hAnim()

{

string animal[SIZE] = {"Alligator", "Chimpanzee", "Crocodile",

"Giraffe", "Hippopotamus", "Octopus", "Scorpion",

"Anaconda", "Chipmunk"};

int random = rand()%10;

return animal[random];

}

string hCandy()

{

string candy[SIZE] = {"Atomic Fireball", "Almond Joy", "Pixie Stix",

"Smarties", "Toblerone", "Jolly Ranchers",

"Salt water taffy", "Jawbreakers", "Pocky",

"Sugar Daddy"};

int random = rand()%10;

return candy[random];

}

string hFruit()

{

string fruit[SIZE] = {"Gooseberry", "Grapefruit", "Huckleberry",

"Kumquat", "Loquat", "Mulberry", "Clementine",

"Mandarin", "Persimmon", "Lychee"};

int random = rand()%10;

return fruit[random];

}

void play(string subj, Player& user)

{

int games = user.easy+user.medium+user.hard;

user.guess = new int[games];

char guess[30], check[30];

int fail = 0;

char word[30];

strcpy(word, subj.c\_str());

char\* answer = new char[strlen(word)];

for(int i = 0; i < strlen(word); i++)

{

if(isalpha(word[i]))

{

answer[i] = '\_';

}

else

answer[i] = ' ';

cout << answer[i] << " ";

}

int x, count = 0, num = 0, space, y = 0;

user.guess[games] = 0;

while(fail < 10)

{

cout << endl << "Enter a letter: ";

cin >> guess[y];

user.guess[games]++;

for(int i = 0; i < 30; i++)

for(int i = 0; i < 30; i++)

while(toupper(check[i]) == toupper(guess[y]))

{

cout << guess[i] << " has already been entered. "

"Choose a different letter." << endl;

cin >> guess[y];

}

num++;

for(int i = 0; i < strlen(word); i++)

{

if(answer[i] == ' ')

space = 1;

if(tolower(word[i]) == tolower(guess[y]))

{

answer[i] = word[i];

x = i;

count++;

}

cout << answer[i] << " ";

}

if(guess[y] != answer[x])

{

cout << endl << guess[y] << " is not in the word!" << endl;

fail++;

cout << "You have " << 10-fail << " tries left!" << endl;

}

if(space == 1)

if(count+1 == strlen(word))

{

cout << endl << "Congratulations, You win!" << endl;

user.win++;

delete[] answer;

return;

}

if(count == strlen(word))

{

cout << endl << "Congratulations, You win!" << endl;

user.win++;

delete[] answer;

return;

}

check[y] = guess[y];

y++;

}

cout << endl << "You lose! Better luck next time" << endl;

user.lose++;

delete[] answer;

}

void stats(Player& user)

{

cout << user.name << "'s stats: " << endl;

cout << "Wins: " << user.win << endl;

cout << "Losses: " << user.lose << endl;

cout << "Easy Games: " << user.easy << endl;

cout << "Medium Games: " << user.medium << endl;

cout << "Hard Games: " << user.hard << endl;

cout << "Total Games: " << user.easy+user.medium+user.hard << endl;

cout << "Average Total Guesses: " << user.avgG << endl << endl;

}

**HEADERFILE**

/\*

\* File: playerData.h

\* Author: Bryan

\*

\* Created on October 27, 2016, 2:00 AM

\*/

#ifndef PLAYERDATA\_H

#define PLAYERDATA\_H

struct Player

{

char\* name;

int easy; // number of easy games

int medium; // number of medium games

int hard; // number of hard games

int win; // wins

int lose; // losses

char\* word; // word length

float winL; // win lose ratio

int\* guess; // number of guesses in each game

float avgG; // average number of guesses of all games

};

#endif /\* PLAYERDATA\_H \*/