Final Project

Wheel of Fortune

**CSC 17C - 48948**

**Name: Javier Borja**

**Date: 12/13/2017**

Table of Contents

[Introduction 2](#_Toc497072324)

[Tutorial 3](#_Toc497072326)

[Project Summary 5](#_Toc497072327)

[Project Requirements: 7](#_Toc497072328)

[Class/UML Diagrams 9](#_Toc497072329)

[Pseudocode and Flowcharts 10](#_Toc497072330)

[Major Variables: 17](#_Toc497072340)

[References: 18](#_Toc497072341)

[Source Code: 18](#_Toc497072342)

# Introduction

First half of semester: For this project I have chosen to revisit a game I am familiar with and enjoy. I previously wrote a Wheel of Fortune for CSC 17A, but this effort is much better. This time around I aimed to utilize the STL Library to improve performance and decrease line counts while increasing the number of features. This project was not overly difficult, but it took a good effort to learn and utilize a library that I had no familiarity with. As I was learning these new constructs, I had to consult several texts to properly implement the STL library. The end result is a game that I am proud of.

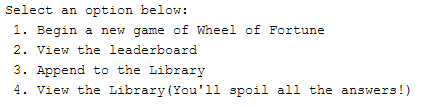
Second half of semester: For the second half of the semester I extended my project with some of the concepts we have covered in class. In terms of gameplay, there are not many changes, but there are changes behind the scenes and in the code. I will fully explain my project extension in the Project Summary and Requirements sections. All in all, I feel like this project is really complete and it is difficult to think of significant improvements to the actual game.

# Input Validation

The menus in this game are self-explanatory and easy to navigate. All inputs are validated and are not case sensitive. Options that have a “(default)” label are selected if the user types an invalid input.

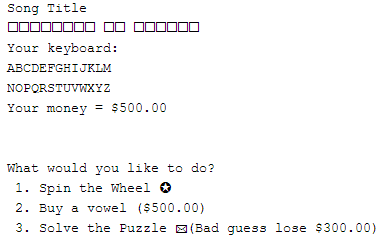
# Tutorial

If a save file is detected, you will have to option to continue. If do you not continue a save file, you will input your name and start with $500.00 and 0 Points. You will then be taken to a menu with four options.



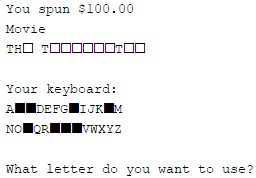
**Playing a Game:**

To win, you must guess the phrase; if you run out of money, you lose. Once you begin playing, you are given a category and phrase to guess. Displayed will be your hidden phrase with spaces, used/unused letters, and your money. Select an appropriate option to continue.



**Spin the Wheel:**

After spinning, you will be displayed a monetary value. If you correctly guess a letter, you will be awarded that amount of money and gain 10 points for each letter in the phrase that matched, else if you guess incorrectly, you will lose that amount. You can keep guessing if you have not used all the letters. Every letter you used will be blacked out and each letter you correctly guessed will be displayed.



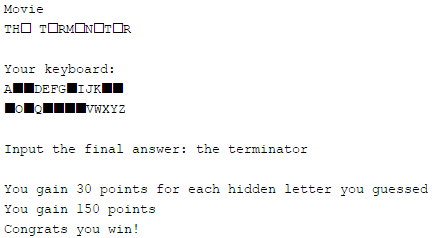
**Buy a Vowel:**

You will be displayed the same graphics as above, except you must buy a vowel. You will lose $500.00 for buying a vowel.



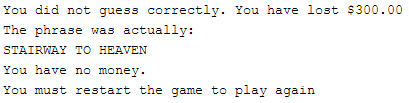
**Solve the Puzzle:**

Input the phrase you think is the answer. It is not case sensitive, but you do need to correctly match all the letters and spaces. If you incorrectly guess, you will lose $300.00; if you correctly guess, you will gain 30 points for each hidden letter revealed. You will then be displayed the amount of money left in your account and current amount of points you have earned.



**Losing and Leaderboard:**

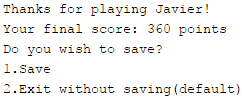
If you run out of money, the correct phrase will be displayed. You lose the game and have to exit the program to play again. You will have the option to put your score in the leaderboard.



If you lost, you can exit the program through the menu and input your score to the leaderboard.



If you still have money you can save your game and continue later.



Project Summary

|  |  |
| --- | --- |
| Project size | 1553 Lines |
| Lines of code | 1126 Lines |
| Comment/Blank lines | 427 Lines |

First half of semester:

I slowly implemented the STL library because I was not familiar with it. In each version I tried to utilize more and more concepts from it. Like most programming efforts, there were many bugs for each version that I had to fix. A difficulty I had was taking chunks of code and replacing it with a required construct. It was hard to test out since many class functions depended on others.

Second half of semester:

This development process was a bit of trouble because I had spent so much effort in the first version, that making significant changes to accommodate the concepts we covered in class was not easy because the gameplay is pretty much hard coded. I did, however, attempt to add the new constructs we learned. I created two new templated functions for recursive sorting and developed two new classes for trees and hashes. I honestly could not find a way to incorporate graphs in my game and I hope that my final exam shows enough proof of my understanding of this concept.

You can see previous versions of this program on the github link.

<https://github.com/javierborja95/CSC17C>

**Unsolved issue**: Very rarely the clues don’t read in properly. Attempting to play will show an extremely long array of blocks. I cannot pinpoint the source to the problem.

Version 1

In this working version, the only STL library constructs implemented were queues, a vector iterator, and a single algorithm.

Version 2

In this version I implemented maps for the clue categories while cleaning up some of the code.

Version 3

This version was mostly touch up by getting rid of unnecessary dynamic memory allocation. I also edited code here and there.

Version 4

I successfully implemented lists for letters and phrases. Began to implement extra sorting options.

Version 5

Here I added load/saving and implemented queues again while replacing one queue with a stack. I also added several sorting algorithms for different types of sorting.

Version 6

In this version I created two new templated search functions. I took two search functions I was familiar with, bubble and insertion sorts, and made them recursive and templated so that I could use my classes with them.

Version 7

This final version I implemented a binary search tree and made that templated as well. I had to overload stream operators in my classes to use the binary search tree. I accidentally did not copy version 7 and create a version 8, but updated version 7.In this update I developed a hash class with my own hashing algorithm. I created a bucket array of linked lists in case of collisions. I used it to check for answer inputs.

# Project Requirements:

First half of semester:

|  |  |
| --- | --- |
| Maps | **🗸** |
| Sets | **🗸** |
| Lists | **🗸** |
| Stacks | **🗸** |
| Queues | **🗸** |
| Iterators | **🗸** |
| Algorithms | **🗸** |

While requirements were to utilize the STL Library, I have included many more concepts that I learned in CSC-5 and CSC-17A, including arrays, pointers, strings, structures, classes and inheritance, and exceptions. I have included every required construct and I will go in detail to show where in code the required constructs are used plus some comments that I had about the STL library.

Second half of semester:

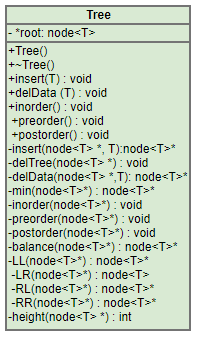
|  |  |
| --- | --- |
| Recursive Sorts | **🗸** |
| Hashing | **🗸** |
| Trees | **🗸** |
| Graphs | **🗴** |

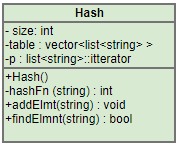
Only graphs were not implemented because I could not find a use for them. It is not a valid excuse but I hope my implementation in the Final Exam shows my understanding of the concept.

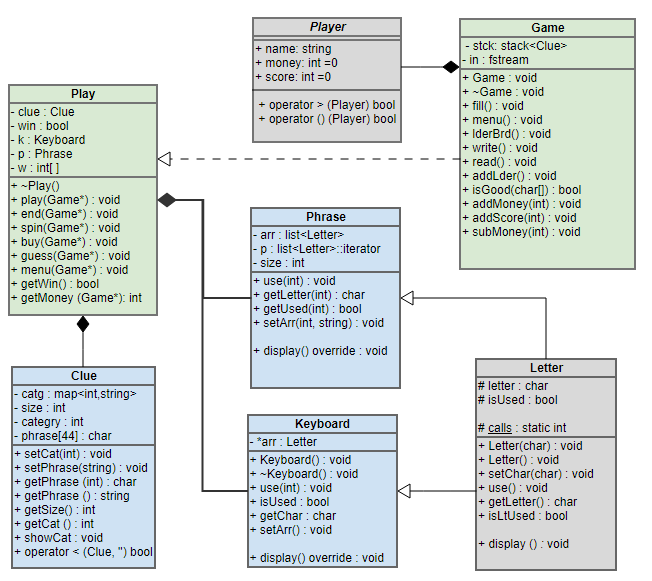
You can find the examples by searching the final pages with ctrl-f.

|  |  |  |
| --- | --- | --- |
| Concept | Examples | Comments |
| Maps | map<int, string> catg;  catg[1]="TV Show";  cout<<catg[categry]<<endl; | These were useful because I did not have to do any if else statements or switch statements to access categories, since they are already numbered. |
| Sets | set<Player,greater<Player>> arr;  set<Clue> tset;  tset.insert(clue); | These were utilized for organizing players on the leaderboard and clues since they automatically sort themselves. I had to overload the < operator to get these to work correctly. |
| Lists | list<Letter> arr;  arr.push\_back(s[i]);  p->use(); | I used it for Keyboard/Phrase classes since they are made up of Letter classes. I didn’t need to random access these, I just iterated through them. |
| Stacks | stack <Clue> stck;  this->stck.push(\*p);  temp=a->stck.top();  a->stck.pop(); | These were highly useful because after importing the clues and phrases, I had to access them and then pop them off so they don’t repeat. |
| Queues | queue<Clue> que;  while(!que.empty())  {clue=que.front(); que.pop(); | I could have used stacks as well but I utilized it to read clues through a file and I utilized these clues in the order that I read them in. |
| Iterators | list<Letter>::iterator p;  p=arr.begin(); advance(p,i);  for(p=arr.begin();p!=arr.end();p++) | I used a lot of iterators to go through the many containers in my program. I learned the hard way that iterating through a queue is not possible, but that concept makes sense. |
| Algorithms | sort(a.begin(),a.end(),name\_sort());  random\_shuffle(arr.begin(),arr.end()); | These were great at cutting down blocks of code into single lines. I felt like I did not use a lot of Algorithms, but their uses were very helpful. |
| Vector | vector<Clue> arr;  arr.push\_back(temp);  vector<Player>::iterator p; | Not really a requirement, but these are also STL containers. I used them a lot because they are highly versatile, especially when working with arrays with variable memory because they delete themselves without problems. |
| Recursive Sorts | void insertRec(vector<T> &array,int size){ … insertRec(array,size-1);  void bubbleRec(vector<T> &array,int size){ …bubbleRec(array,size-1); | I used recursive sorts to sort arrays of classes to display. I overloaded the less than operator to perform these sorts. |
| Hashing | Hash table;  int bucket=hashFn(s)%size;  p=table[bucket].begin(); | Implemented a hash table with arrays of linked lists. I put clues into arrays and a player submits an answer string; if no collision, turn lost, else iterate through bucket linked list to see if its there. |
| Trees | Tree<Clue> tree;  void Tree<T>::inorder(node<T> \*leaf){  tree.insert(clue); | I used a binary tree to display phrases inorder. While I implemented the tree pretty normally, for this game I could have made some branches have a category, so the overall tree would have several roots. I thought this could work because the STL sort was the best because I was able to make it sorted with categories in mind. |

# Class/UML Diagrams







# Pseudocode and Flowcharts

## Main:

Create a Game object

Show Game.menu

Input menu choice

Do{

Switch(choice)

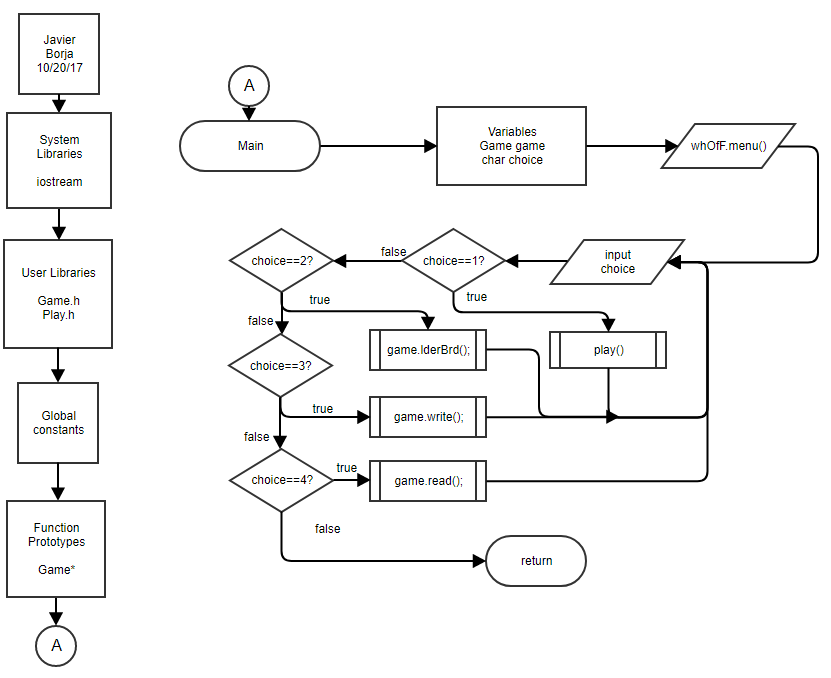
Case 1: Play()

Case 2: Game.leaderBrd()

Case 3: Game.write()

Case 4: Game.read()

}While (choice is 1-4)



## Play:

Create Play object

object.play(this)

Do{

Output display()

Input option

Switch(option){

Case 1: object.spin()

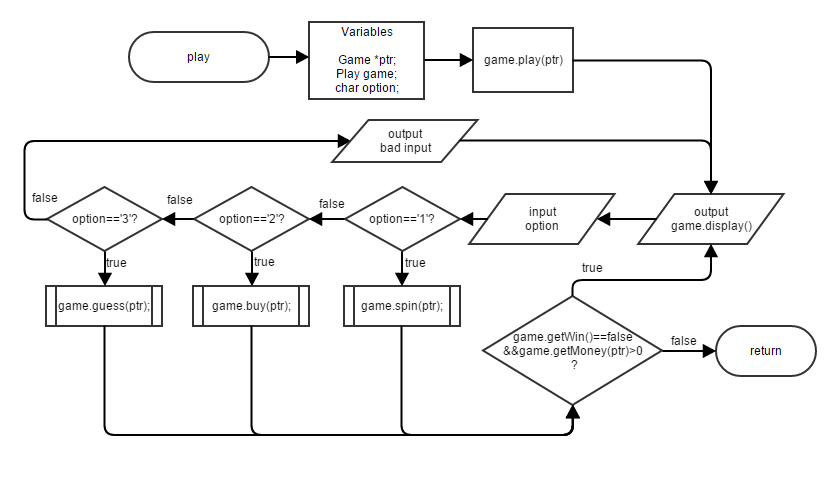
Case 2: object.buy()

Case 3: object.guess()

Default: Output error message

}

}While(object.getWin()==false and object.getMoney(this) is greater than 0)



## Fill: [Game::fill()]

Open phrase file

While(reading from file)

Set temporary clues

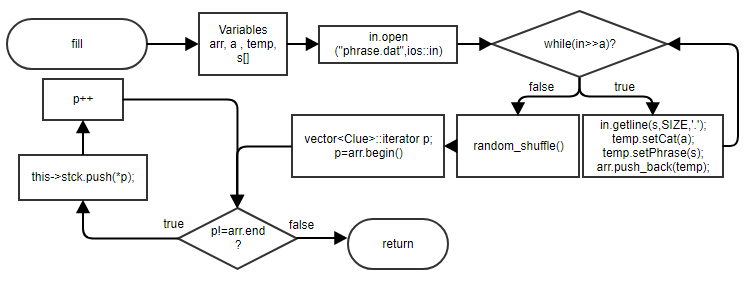
Push temp into array

Shuffle array

For(beginning of array, not end of array, iterator++){

Push into stack

} return array



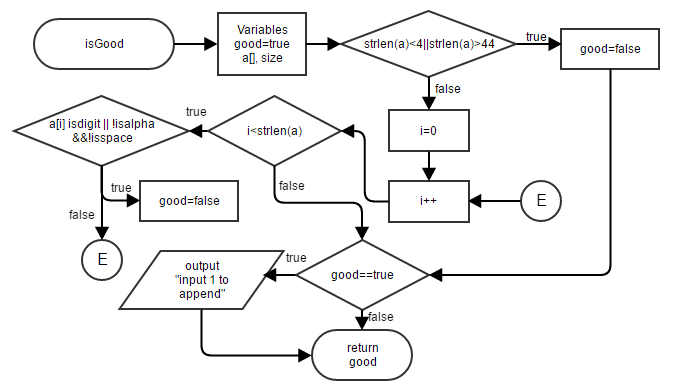
## isGood:[Game::isGood()]

(Is length of string <4||>44?)

T: Good=false

F: For(i=0;i<length of string;i++){

(If character is not letter or space) good=false

If(good==true)

T: Ask to append word

Return (good)

## Spin:[Play::spin()]

Spin wheel

Show board and keyboard

Do{

Error=false

Input letter to use

(is letter==vowel||non alphabet||or already used?)

Error=true

While(error==true)

If(letter input==hidden letter)

T: {add points

Match=true}

If(match==true)

T:{Add money

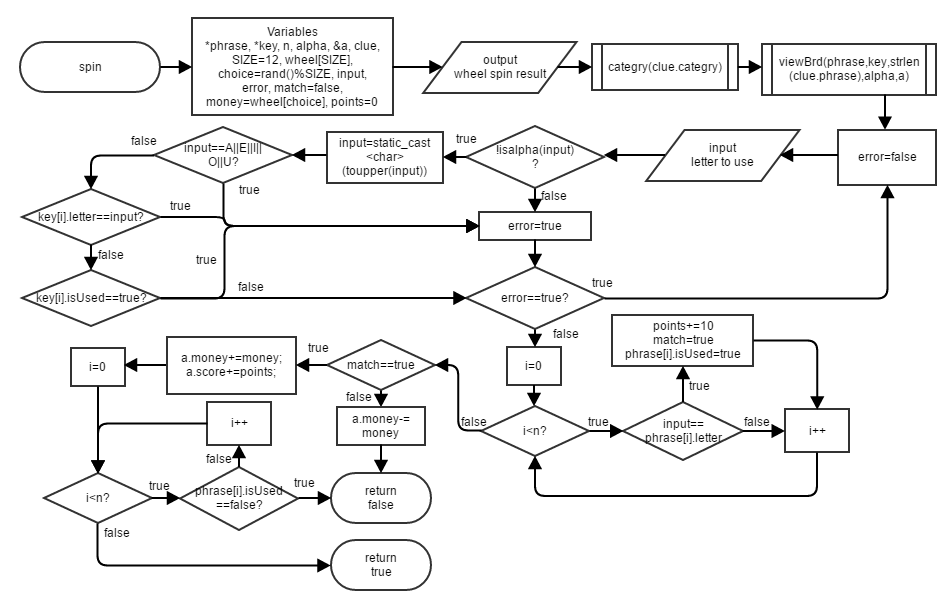
Add points to score

Make hidden letters shown

(If all letters revealed) Return win

}

F: lose money, return loss



## Vowel: [Play::buy()]

(if all vowels are used) return

Do{

Error =false

Show board and keyboard

Input letter

(if input is not vowel)

Error =true

(if vowel is used)

T: error =false

F: make key used

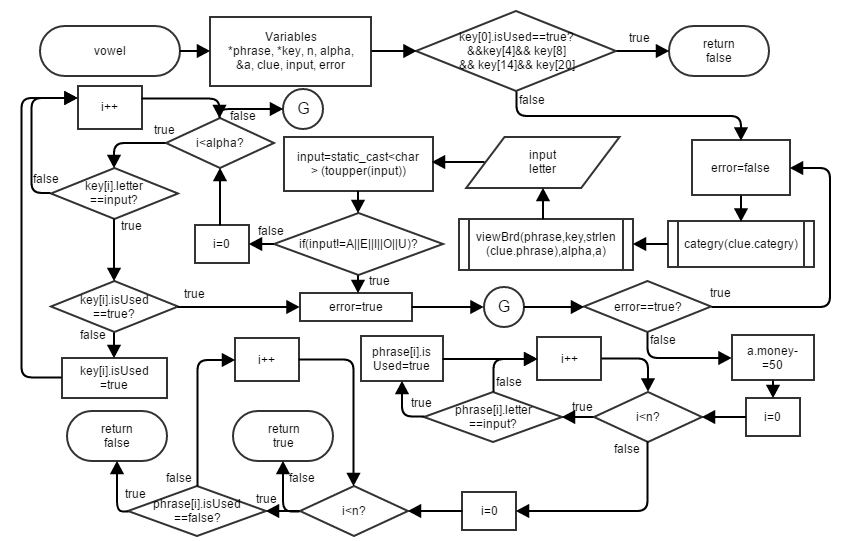
While(error==true)

Subtract money

Reveal vowels from phrase

(If all letters are revealed) return win

Else return loss



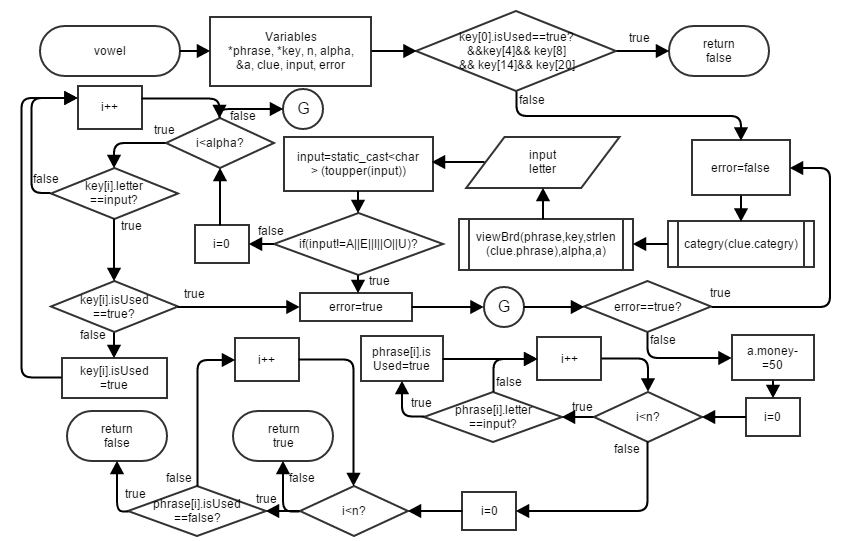
## 

## Guess:[Play::guess()]

Show board and keyboard

Input phrase

(if input matches board phrase)

 T: Return win

F: Subtract money, return loss

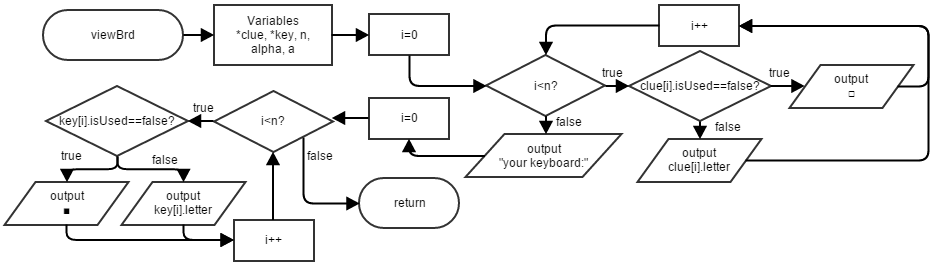
## viewBrd:[Play::display][Keyboard::display()][Phrase::display()]

(If phrase letter is hidden)

T: Show square

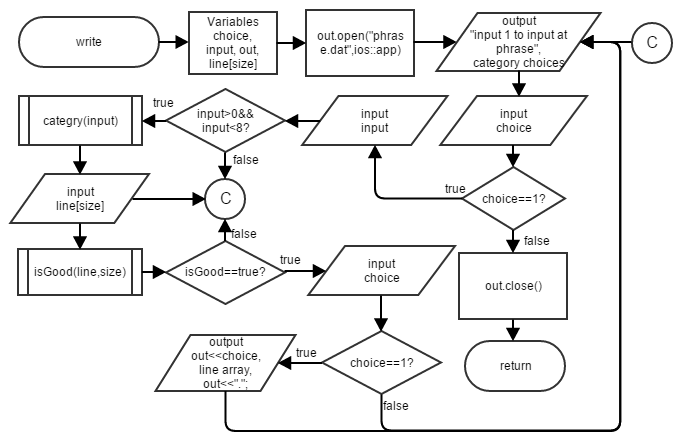
F: Show letter

(If keyboard letter is used)

 T: Show square

F: Show letter

## Write: [Game::write()]

Open file to append

Input category

Input phrase

isGood(phrase)

(if good)

T:Ask to input

Input choice

(If choice is true)

write to file

Close file

# Major Variables:

Local variables of different member functions are not included.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **Location** |
| Player | user | Contains user name, money and score | Game.h, Play.h |
| stack <Clue> | stck | A stack of clues | Game.h |
| map <int, string> | catg | Map of categories |  |
| int | size | Generic variable that shows size | Clue.h, Letter.h |
| Game | game | Wheel of fortune game object | main |
| int | category | Number to represent a category | Clue.h |
| char [ ] | phrase | Phrase | Clue.h |
| string | name | Name of player | Player.h |
| int | money | Money | Player.h |
| int | score | Score | player.h |
| Clue | clue | Holds category and clue phrase | Play.h |
| bool | win | Represents whether one lost | Play.h |
| Keyboard | k | A keyboard object | Play.h |
| Phrase | p | A phrase object | Play.h |
| int [ ] | w | Wheel spin options | Play.h |
| list <Letter> | arr | List of letters | Keyboard.h, Phrase.h |
| list<Letter>::iterator | p | Iterator to travers list | Keyboard.h, Phrase.h |
| char | letter | A single character that letter represents | Letter.h |
| bool | isUsed | Shows whether to reveal or hide letter | Letter.h |
| fstream | in | General input from files | Game..h |
| Hash | table | A hash table full of clues and phrases | Game.h Play.h |
| Tree<Clue> | tree | A tree of clues for inorder outputting | Game.cpp |

# References:

CSC-17C class

--For concepts learned

https:://opendsa-server.cs.vt.edu

--Online book for trees and hash introduction

Savitch, Walter. (2014) *Problem Solving with C++*. Pearson

--For STL introduction

<https://www.sgi.com/tech/stl/>

--Extra information on STL

<http://wheeloffortuneanswer.com/>

--Copied phrases to fill dictionary

# Source Code:

﻿/\*

\* File: main.cpp

\* Author: Javier Borja

\* Created on October 20, 12:00 PM

\* Purpose: Wheel of fortune. Player guesses a phrase with category as a clue.

\*/

//System Libraries

#include <iostream> //Input/Output

using namespace std;

//User Libraries

#include "Game.h"

#include "Play.h"

//Global Constants

//Function Prototypes

//Execution

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

//Variables

Game game; //Wheel of fortune Game object

char choice; //Menu choice

//Input Data

do{

game.menu();

cin>>choice;

cin.ignore();

cout<<"\n\n\n\n\n";

//Process Data

switch(choice){

case'1':{

game.play();

break;

}

case'2':{

game.lderBrd();

break;

}

case'3':{

game.write();

break;

}

case'4':{

game.read();

break;

}

}

}while((choice=='1'||choice=='2'||

choice=='3'||choice=='4'));

//Process Data

return 0;

}

/\* File: Play.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Implementation File for play class

\*/

//User Libraries

#include "Play.h"

void Play::play(Game \*a){

//Variables

win=false;

unsigned int c; //Temp char

char s[SIZE]; //Temp string

//Input Data

Clue temp;

temp=a->stck.top();

a->stck.pop();

this->clue.setCat(temp.getCat());

this->clue.setPhrase(temp.getPhrase());

//Create a new Phrase

Phrase p;

p.setArr(temp.getPhrase());

//Copy Phrase to pointer

this->p=p;

}

void Play::end(Game \*a){

//Output Data

if(a->getMoney()<=0){

cout<<"The phrase was actually: "<<endl;

cout<<clue.getPhrase()<<endl;

cout<<"You have no money.\n"

"You must restart the game to play again"<<endl;

}else cout<<"Congrats you win!\n"

"You have $"<<a->getMoney()\*10<<".00 left in your account"<<endl;

}

void Play::spin(Game \*a){

//Variables

int choice=rand()%WHEEL;//Random wheel choice

char input; //Letter input

bool error; //Incorrect letter input

bool match=false; //Did letter match?

int money=w[choice]; //Money to add or subtract from user's money

int points=0; //Counter for points

bool win=true;

//Input Data

cout<<"Spinning...\nPress Enter to continue";

cin.get();

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

if(money==0) cout<<"You spun a free guess"<<endl;

else cout<<endl<<"You spun $"<<money\*10<<".00"<<endl;

display();

do{

try{

error=false;

cout<<"What letter do you want to use? ";

cin>>input;

cin.ignore();

//Process Data

if(!isalpha(input)){

throw "Input must be part of the alphabet";

}

input=static\_cast<char>(toupper(input)); //Make uppercase

if(input=='A'||input=='E'||input=='I'||input=='O'||input=='U'){

throw "You have to buy vowels";

}

for(int i=0;i<ALPHA;i++){

if(k.getChar(i)==input){

if(k.isUsed(i)==true){

cout<<"You already used that letter"<<endl;

return;

}else k.use(i);

}

}

}

catch(char const\* s){

cout<<s<<endl;

error=true;

cout<<"Press enter to continue"<<endl;

cin.get();

}

}while(error); //Keep looping until valid input

for(int i=0;i<p.getSize();i++){

if(input==p.getLetter(i)){ //If letter matches

points+=10; //Add ten points

match=true; //Match is true

p.use(i); //Don't hide letter anymore

}

}

//Output Data

if(match){ //If match is true

cout<<"You have been awarded $"<<money\*10<<".00"<<endl;

a->addMoney(money);

cout<<"You gain 10 points for each letter guessed"<<endl;

cout<<"You gained "<<points<<" points"<<endl<<endl;

a->addScore(points);

for(int i=0;i<p.getSize();i++){

if(p.getUsed(i)==false){

win=false; //Not all letters are revealed, win=false;

}

}

this->win=win; //All letters of phrase are revealed, win=true

}else{ //Match is not true

a->subMoney(money);

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"You have lost $"<<money\*10<<".00."<<endl<<endl;

}if(a->getMoney()<=0){

end(a);

}

if(this->win==true){

end(a);

}

}

void Play::buy(Game \*a){

//Conditions to return

if((k.isUsed(0))&&(k.isUsed(4))&&(k.isUsed(8))&&(k.isUsed(14))&&(k.isUsed(20))){

cout<<"You have already bought all the vowels"<<endl;

return; //Exit

}

if(a->getMoney()<=50){

cout<<"You don't have enough money!"<<endl;

cout<<"Spin the wheel or guess the puzzle"<<endl;

cout<<"Input a key to continue: ";

cin.get();

return; //Exit

}

//Variables

char input; //Input for vowel

bool error; //Error

bool win=true; //Win

//Input Data

do{

try{

error=false;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

display();

cout<<"Which vowel do you want to buy? ";

cin>>input;

cin.ignore();

input=static\_cast<char>(toupper(input));

if(input=='A'||input=='E'||input=='I'||input=='O'||input=='U'){

}

else{

throw "You did not choose a vowel";

}

for(int i=0;i<ALPHA;i++){

if(k.getChar(i)==input){

if(k.isUsed(i)==true){

throw "You already used that letter";

}else k.use(i);

}

}

}

catch(char const\* s){

cout<<s<<endl;

error=true;

cout<<"Press enter to continue"<<endl;

cin.get();

}

}while(error==true); //Loop until valid input

//Process Data

cout<<"You have bought a vowel for $500.00"<<endl;

a->subMoney(50); //Subtract money from user

for(int i=0;i<p.getSize();i++){

if(p.getLetter(i)==input) //Reveal vowels from clue phrase

p.use(i);

}

for(int i=0;i<p.getSize();i++){

if(p.getUsed(i)==false){

win=false;//Not all letters are revealed, win=false;

}

}

if(win==true){ //All letters of phrase are revealed,

this->win=win;//win=true

end(a);

}

}

void Play::guess(Game \*a){

//Variables

string answer; //Player answer

int counter=0; //Amount of empty letters in keyboard array

int score=30; //Points=score\*counter

bool win; //true=win---false=loss

//Input Data

display();

cout<<"Input the final answer: ";

getline(cin,answer);

//Process Data

for(int i=0;i<p.getSize();i++){ //Convert to uppercase

answer[i]=static\_cast<char>(toupper(answer[i]));

}

win=a->table.findElmnt(answer);

cout<<endl;

//998133622

//Output Data

if(win==true){

for(int i=0;i<p.getSize();i++){ //Go through phrase array to add

if((p.getUsed(i))==false){ //points for each letter that is not used

counter++;

}

}

score\*=counter;

cout<<"You gain 30 points for each hidden letter you guessed"<<endl;

cout<<"You gain "<<score<<" points"<<endl;

a->addScore(score);

this->win=win; //Make private member win=local win;

end(a); //Go to end

}else{

cout<<"You did not guess correctly. You have lost $300.00\n";

a->subMoney(30);

}

if(a->getMoney()<=0){

end(a);

}

}

void Play::display(){

//Output Data

clue.showCat();

p.display();

k.display();

cout<<endl;

}

void Play::menu(Game \*a){

//Output Data

cout<<"Your money = $"<<a->getMoney()\*10<<".00"<<endl;

cout<<endl<<endl<<"What would you like to do?"<<endl;

cout<< " 1. Spin the Wheel ✪\n"

" 2. Buy a vowel ($500.00)\n"

" 3. Solve the Puzzle ✉(Bad guess lose $300.00)\n"<<endl;

}

/\* File: Phrase.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Implementation File for Phrase class

\*/

//User Libraries

#include "Phrase.h"

#include "Letter.h"

void Phrase::use(int i){

p=arr.begin();

advance(p,i);

p->use();

}

void Phrase::setArr(string s){

//Input Data

size=s.length();

p=arr.begin();

for(int i=0;i<size;i++,p++){ //Initialize phrase array with clue

arr.push\_back(s[i]);

if(isspace(p->getLetter())){//If letter is space

p->use(); //Don't hide it

}

}

}

void Phrase::display(){

//Output Data

for(p=arr.begin();p!=arr.end();p++){ //Go through clue array

if(p->isLtUsed()==false){ //If letter has not been used, hide letter

cout<<"□";

}else{

p->display();

}

}

cout<<endl;

}

char Phrase::getLetter(int i){

p=arr.begin();

advance(p,i);

return p->getLetter();

}

bool Phrase::getUsed(int i){

p=arr.begin();

advance(p,i);

return p->isLtUsed();

}

/\* File: Letter.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Specification File for Letter class

\*/

//User Libraries

#include "Letter.h"

Letter::Letter(char a){

//Process Data

letter=a;

isUsed=false;

}

Letter::Letter(){

//Process Data

letter=' ';

isUsed=false;

}

/\* File: Keyboard.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Specification File for Keyboard class

\*/

//User Libraries

#include "Keyboard.h"

Keyboard::Keyboard(){

//Initializing the keyboard

for(int i=0;i<ALPHA;i++){ //Initialize with alphabet

arr.push\_back('A'+i);

}

}

void Keyboard::display(){

//Output Data

cout<<endl<<"Your keyboard:"<<endl;

p=arr.begin();

for(p=arr.begin();p!=arr.end();p++){//Go through keyboard list

if(p->isLtUsed()==false){ //If letter has not been used, hide letter

p->display();

}else cout<<"■";

if((distance(arr.begin(),p)+1)%13==0) cout<<endl;

}

}

void Keyboard::use(int i){

p=arr.begin();

advance(p,i);

p->use();

}

bool Keyboard::isUsed(int i){

p=arr.begin();

advance(p,i);

p->isLtUsed();

}

char Keyboard::getChar(int i){

p=arr.begin();

advance(p,i);

p->getLetter();

}

/\* File: Hash.h

\* Author: Javier B

\* Created on December 10, 2017, 8:56 PM

\* Purpose: Class Implementation File for a Hash Class

\*/

//User Libraries

#include "hash.h"

int Hash::hashFn(const string s){

//Hash function

int seed = 7567;

for(int i=0;i<s.length();i++){

seed^=s[i]\*seed<<3;

seed+=s[i]+7793;

}

if(seed<0) return seed\*-1;

else return seed;

}

//list<string> list; list.begin();

void Hash::addElmnt(string s){

int bucket=hashFn(s)%size; //Create a hash and bucket

table[bucket].push\_back(s); //Insert element into bucket

//Linked list takes care of collisions

}

bool Hash::findElmnt(string s){

//Look for bucket based on string

int bucket=hashFn(s)%size;

if(table[bucket].empty()){cout<<"empty"<<endl; return false;} //If empty, string is not in hash table

else{ //Look through list to see if elements match string

p=table[bucket].begin();

while(p!=table[bucket].end()){

if(s==\*p){cout<<"Found"<<endl; return true; }//Return if found

p++;

}

}

cout<<"NOT Found"<<endl;

return false; //Return not found

}

/\* File: Game.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Implementation File for Game class

\*/

//User Libraries

#include "Game.h"

#include "Play.h"

#include "Tree.h"

#include "RecursiveSorts.h"

Game::Game(){

//Variables

int n; //To hold score and money

char s[SIZE]; //String to hold phrase

//Initialize random seed.

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

//Fill Library

fill();

//Try to load

in.open("save.dat",ios::in);

try{

if(in.fail()){

cout<<"No save detected"<<endl;

throw 0;

}else{

if(in>>n){

cout<<"Save file detected: would you like to continue?"<<endl

<<"(Both options will delete the profile, you can save later)\n"

<<"1.Continue\n2.New Game(default)"<<endl;

cin>>s[0];

cin.ignore();

if(s[0]=='1'){

in.getline(s,SIZE,'.');

user.name=s;

user.money=n;

in>>user.score;

}else throw 0;

}else{

cout<<"No save detected"<<endl;

throw 0;

}

}

}

catch(const int &){

cout<<"Input your name: ";

getline(cin,user.name);

}

cout<<"Welcome to Wheel of Fortune "<<user.name<<"!\n";

//Close istream

in.close();

in.clear();

//Clear save

ofstream out;

out.open("save.dat",ios::out | ios::trunc);

out.close();

}

Game::~Game(){

//Variables

ofstream out; //Output

//Output Data

char choice;

cout<<"Thanks for playing "<<user.name<<"!"<<endl;

cout<<"Your final score: "<<user.score<<" points"<<endl;

if(user.money>0){

cout<<"Do you wish to save?\n1.Save\n2.Exit without saving(default)\n";

cin>>choice;

cin.ignore();

}

if(choice!='1'||user.money<=0){

cout<<"\nDo you wish to add your score to the leaderboard?\n"

"Input 1 to add\n"

"Input 2 to exit(default): ";

cin>>choice;

if(choice=='1') addLder(); //Add to leaderboard

}else{

if(user.name.size()==0){

cout<<"Error: No name"<<endl;

return;

}

out.open("save.dat",ios::out);

out<<user.money<<user.name<<"."<<user.score;

out.close();

cout<<"Your file has been saved"<<endl;

}

}

void Game::addLder(){

if(user.name.size()==0){

cout<<"Error: No name"<<endl;

return;

}

//Variables

ofstream out; //Output

int n; //Size of string

//Output Data

out.open("users.dat",ios::out|ios::app);

out<<user.name.size()<<user.name<<user.score<<endl;

cout<<"Your score has been added"<<endl;

//Close files

out.close();

}

void Game::fill(){

//Variables

vector<Clue> arr; //Vector that contains clues and category

unsigned int a; //Temp char

Clue temp; //Temp clue

char s[SIZE]; //Temp char array

//Open File

in.open("phrase.dat",ios::in);

if(in.fail()){

cout<<"CRITICAL ERROR: File opening failed"<<endl;

exit(1);

}

//Input Data

while(in>>a){

in.getline(s,SIZE,'\r');

temp.setCat(a);

temp.setPhrase(s);

arr.push\_back(temp);

table.addElmnt(s); //Add string into hash table

}

//Process Data

random\_shuffle(arr.begin(),arr.end());

vector<Clue>::iterator p;

for(p=arr.begin();p!=arr.end();p++){

this->stck.push(\*p);

}

//Close Files

in.close();

}

void Game::lderBrd(){

//Variables

fstream in; //Input from file

int n; //Size of string that is read from file

set<Player,greater<Player> > arr;//Set of Player structures

Player temp; //Temp Player for input

string a; //Player inputs to continue

in.clear();

try{

//Open files

in.open("users.dat",ios::in);

if(in.fail()){

throw "users.dat not found";

}

//Input Data

while(in>>n){ //Get size of string

temp.name.resize(n); //Resize string size to n

in.read(&temp.name[0],n);//In name string of size n

in>>temp.score;

arr.insert(temp);

}

cout<<"Input sort method:\n1.By score(default)\n2.By name\n";

cin>>n;

cin.ignore();

//Output Data

if(n==2){

vector<Player> a(arr.begin(),arr.end());

sort(a.begin(),a.end(),name\_sort());

cout<<"Sorted Leaderboard by name:"<<endl;

vector<Player>::iterator p;

for(p=a.begin();p!=a.end();p++){

cout<<p->name;

cout<<setw(5)<<right<<p->score<<" points"<<endl<<endl;

}

}else{

Tree<Player> tree;

cout<<"Sorted Leaderboard by score:"<<endl;

set<Player>::iterator p;

for(p=arr.begin();p!=arr.end();p++){

tree.insert(\*p);

}

tree.inorder();

}

cout<<"Press enter to continue"<<endl;

getline(cin,a);

}

catch(char\* const s){

in.close();

cout<<s<<endl;

}

//Close files

in.close();

}

void Game::read(){

//Variables

Clue clue; //Temporary Clue to fill

unsigned int n; //Categories are numbered

char s[SIZE]; //String to hold phrase

char choice; //Choice variable

queue<Clue> que;//Queue of clues

//Open File

in.clear();

in.open("phrase.dat",ios::in);

if(in.fail()){

cout<<"CRITICAL ERROR: File opening failed"<<endl;

return;

}

cout<<"Choose display option:\n1.Unsorted(default)\n2.Sorted\n";

cin>>choice;

cin.ignore();

//Output Data

if(choice=='2'){

cout<<"Choose sort option:\n1.Binary tree inorder output(default)\n"

"2.Recursive Sort\n"

"3.STL sort\n";

cin>>choice;

cin.ignore();

if(choice=='3'){

set<Clue>::iterator p;

for(int i=0;i<7;i++){

set<Clue> tset;

in.clear();

in.seekg(0,ios::beg); //Go back to beginning

while(in>>n){ //Repeat until in can't extract a char

in.getline(s,SIZE,'\r');

if(n==i+1){

clue.setCat(n);

clue.setPhrase(s);

tset.insert(clue);

}

}

for(p=tset.begin();p!=tset.end();p++){

que.push(\*p);

}

}

int a=1,b=1; //Variables for algorthm

while(!que.empty()){

clue=que.front();

que.pop();

if(clue.getCat()==a){

b=clue.getCat();

}

if(a==b){ //Algorithm for showing category once

cout<<endl;

clue.showCat();

a++;

}

cout<<clue.getPhrase()<<endl;

}

}else if(choice=='2'){

vector<Clue> array;

in.clear();

in.seekg(0,ios::beg); //Go back to beginning

while(in>>n){

in.getline(s,SIZE,'\r');

clue.setCat(n);

clue.setPhrase(s);

array.push\_back(clue);

}

cout<<"\nChoose type of recursive sort\n"

"1. Recursive Bubble(default):\n2. Recursive Insertion:"<<endl;

cin>>choice;

cin.ignore();

if(choice=='2') insertRec(array,array.size());

else bubbleRec(array,array.size());

vector<Clue>::iterator p;

for(p=array.begin();p!=array.end();p++){

cout<<\*p<<endl;

}

}else{

cout<<"Choice 1"<<endl;

//Variables

Tree<Clue> tree;

in.clear();

in.seekg(0,ios::beg); //Go back to beginning

while(in>>n){ //Repeat until in can't extract a char

in.getline(s,SIZE,'\r');

clue.setCat(n);

clue.setPhrase(s);

tree.insert(clue);

}

tree.inorder();

}

}

else{

while(in>>n){ //Repeat until in can't extract a char

in.getline(s,SIZE,'\r');

clue.setCat(n); //Set category

clue.showCat(); //View category

cout<<s<<endl; //Output string

}

}

//Input Data

in.close();

cout<<endl<<"Input anything to continue: ";

cin.get();

}

void Game::write(){

//Variables

char choice; //Menu choice

char input; //Input for sub-menu

fstream out; //Output to file

char line[SIZE];//Character array of size=44

Clue clue;

//Open File

out.open("phrase.dat",ios::app);

//Input Data

cout<<endl<<"Input 1 to input a phrase\n"

"Input 0 to exit: ";

cin>>choice;

cin.ignore();

if(choice=='1'){

cout<<endl<<"Input a category:\n";

for(int i=1;i<=7;i++){

cout<<i<<" ";

clue.setCat(i);

clue.showCat();

}

cout<<endl<<"0 Exit(default)"<<endl;

cin>>input;

cin.ignore();

//Output Data

if(input>48&&input<56){ //If input is '1'-'7'

clue.setCat(input-48);

cout<<"Input your phrase(max 44 characters): "<<endl;

cin.getline(line,SIZE);

if(isGood(line)){ //If input is good ask if wish to append

cin>>choice;

cin.ignore();

if(choice=='1'){

out<<input;

for(int i=0;i<strlen(line);i++){

out<<static\_cast<char>(toupper(line[i])); //Make uppercase

}

out<<'\r';

cout<<"You must restart the game for effects to take effect"<<endl;

}

}

}

}

//Close File

out.close();

}

bool Game::isGood(char a[]){

//Process Data

try{

if(strlen(a)<4||strlen(a)>44){ //If char array doesn't fit size limit

throw "ERROR: Phrase must be greater than 3 characters and less than 44";

}

for(int i=0;i<strlen(a);i++){

if(isdigit(a[i])||(!isalpha(a[i])&&!isspace(a[i]))){//If not space or letter

throw "ERROR: Input must be characters only\n";

}

}

//Output Data

cout<<"Do you really wish to add the following phrase?"<<endl;

for(int i=0;i<strlen(a);i++){

cout<<static\_cast<char>(toupper(a[i]));

}

cout<<endl<<endl<<"Input 1 to append\n"

"Or anything else to cancel: ";

return true;

}

//Catch errors

catch(char const\* s){

cout<<s<<endl;

return false;

}

}

void Game::menu(){

//Output Data

cout<<"\n\nYour money: $"<<user.money\*10<<".00\n"

"Your score: "<<user.score<<" points\n\n"

"Select an option below:\n"

" 1. Begin a new game of Wheel of Fortune\n"

" 2. View the leaderboard\n"

" 3. Append to the Library\n"

" 4. View the Library(You'll spoil all the answers!)\n\n";

if(user.money>0) cout<<"Any other input to exit and save your progress: ";

else cout<<"Any other input to exit: ";

}

void Game::play(){

if(user.money<0){

cout<<"You have no money. Restart to play again."<<endl;

return;

}

//Variables

Play obj; //Play object

char option; //Menu option

obj.play(this); //Start playing

//Input Data

do{

obj.display(); //Display hidden phrase and available keyboard letters

do{

obj.menu(this); //Display menu

cin>>option;

cin.ignore();

switch(option){

case'1':

obj.spin(this);

break;

case'2':

obj.buy(this);

break;

case'3':

obj.guess(this);

break;

default: cout<<"ERROR: Bad Input"<<endl;

}

}while(option<49||option>51);

//Loop until win or lose

}while((obj.getWin()==false)&&(obj.getMoney(this)>0));

}

/\* File: Clue.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Implementation File for Clue class

\*/

//User Libraries

#include "Clue.h"

Clue::Clue(){

catg[1]="TV Show";

catg[2]="Event";

catg[3]="Movie";

catg[4]="Landmark";

catg[5]="Famous Person";

catg[6]="Thing";

catg[7]="Song Title";

}

void Clue::setCat(unsigned int n){

categry=n;

}

void Clue::setPhrase(string s){

size=s.length();

//Input Data

for(int i=0;i<s.length();i++){

phrase[i]=s[i];

}

phrase[size]='\0';

}

/\* File: Hash.h

\* Author: Javier B

\* Created on December 10, 2017, 8:56 PM

\* Purpose: Class Specification File for a Hash Class

\*/

#ifndef HASH\_H

#define HASH\_H

//System Libraries

#include <iostream> //Input/ Output Stream Library

#include <vector> //Vectors

#include <list> //Lists

using namespace std; //Namespace of the System Libraries

//User Libraries

class Hash{

private:

int size=500; //Max hash table size to minimize collisions. 500 buckets.

vector<list<string> > table;

list<string>::iterator p; //Iterator to traverse linked list

int hashFn(const string); //Hash Function

public:

Hash()

{vector<list<string> > preTable(size);

table=preTable;} //Pre allocating memory

void addElmnt(string); //Add element into hash table

bool findElmnt(string);//Find element in the hash table. True if found.

};

#endif

/\*

\* File: Tree.cpp

\* Author: Javier Borja

\* Created on December 4, 2017, 10:00 AM

\* Reference: http://www.geeksforgeeks.org/binary-tree-data-structure/

\* https://en.wikipedia.org/wiki/AVL\_tree

\* Purpose: Tree class Specification and Implementation file

\*/

#ifndef TREE\_H

#define TREE\_H

#include<iostream>

using namespace std;

template<class T>

struct node{

T data;

node\* left=NULL;

node\* right=NULL;

int height=0;

};

template<class T>

class Tree{

private:

node<T> \*root;

//Setter

node<T>\* insert(node<T> \*,T); //Insert data as a new leaf

//Delete

void delTree(node<T> \*); //Deletes itself

node<T>\* delData(node<T> \*,T); //Delete a single node, and replace it

node<T>\* min(node<T> \*); //Helper for deleting, no need for max

//Display

void inorder(node<T> \*); //View left-root-right

void preorder(node<T> \*); //View root-left-right

void postorder(node<T> \*); //View left-right-root

//Balacing Functions

node<T>\* balance(node<T> \*); //Performs balancing based on bottom conditions

node<T>\* LL(node<T> \*); //1 Right rotation

node<T>\* LR(node<T> \*); //Left rotate, then Right rotate

node<T>\* RL(node<T> \*); //Right rotate, then left rotate

node<T>\* RR(node<T> \*); //1 Left rotation

int height(node<T> \*leaf) //Helper to keep tree balanced

{if(leaf==NULL) return -1; return leaf->height;}

public:

//Constructor

Tree()

{root=NULL;}

Tree(T data)

{root=NULL; root=insert(root,data);}

//Destructor

~Tree()

{delTree(root);}

//Set

void insert(T data) //Insert data as a new leaf

{root=insert(root,data);}

//Mutate

void delData(T data)

{root=delData(root,data);}

//Display

void inorder() //View left-root-right

{inorder(root); cout<<endl;}

void preorder() //View root-left-right

{preorder(root); cout<<endl;}

void postorder() //View left-right-root

{postorder(root); cout<<endl;}

};

template<class T>

void Tree<T>::delTree(node<T> \*leaf){

if(leaf!=NULL){

delTree(leaf->left);

delTree(leaf->right);

delete leaf;

}

}

template<class T>

node<T>\* Tree<T>::insert(node<T> \*leaf,T data){

if(leaf==NULL){

leaf=new node<T>;

leaf->data=data;

}else if(data<leaf->data){

leaf->left=insert(leaf->left,data);

if(height(leaf->left)-height(leaf->right)==2){

if(data<leaf->left->data)

leaf=LL(leaf);

else

leaf=RL(leaf);

}

}else if(data>leaf->data){

leaf->right=insert(leaf->right,data);

if(height(leaf->right)-height(leaf->left)==2){

if(data>leaf->right->data) leaf=RR(leaf);

else leaf=LR(leaf);

}

}

int hLeft=height(leaf->left), hRight=height(leaf->right);

if(hLeft>hRight) leaf->height=hLeft+1;

else leaf->height=hRight+1;

return leaf;

}

template<class T>

node<T>\* Tree<T>::LL(node<T> \*leaf){

node<T> \*child=leaf->left;

leaf->left=child->right;

child->right=leaf;

{

int hLeft=height(leaf->left), hRight=height(leaf->right);

if(hLeft>hRight) leaf->height=hLeft+1;

else leaf->height=hRight+1;

}

{

int hLeft=height(child->left), hRight=leaf->height;

if(hLeft>hRight) child->height=hLeft+1;

else child->height=hRight+1;

}

return child;

}

template<class T>

node<T>\* Tree<T>::LR(node<T> \*leaf){

leaf->right=LL(leaf->right);

return RR(leaf);

}

template<class T>

node<T>\* Tree<T>::RL(node<T> \*leaf){

leaf->left=RR(leaf->left);

return LL(leaf);

}

template<class T>

node<T>\* Tree<T>::RR(node<T> \*leaf){

node<T>\* child=leaf->right;

leaf->right=child->left;

child->left=leaf;

{

int hLeft=height(leaf->left), hRight=height(leaf->right);

if(hLeft>hRight) leaf->height=hLeft+1;

else leaf->height=hRight+1;

}

{

int hLeft=height(leaf->right), hRight=leaf->height;

if(hLeft>hRight) child->height=hLeft+1;

else child->height=hRight+1;

}

return child;

}

template<class T>

node<T>\* Tree<T>::min(node<T> \*leaf){

if(leaf==NULL) return NULL;

else if(leaf->left==NULL) return leaf;

else return min(leaf->left);

}

template<class T>

node<T>\* Tree<T>::delData(node<T> \*leaf,T data){

if(leaf == NULL) return NULL;

else if(data<leaf->data) //Go left if data is less than

leaf->left=delData(leaf->left, data);

else if(data>leaf->data) //Go right if data is greater than

leaf->right=delData(leaf->left, data);

else if(leaf->left!=NULL&&leaf->right!=NULL){

node<T>\* temp=min(leaf->right);

leaf->data=temp->data;

leaf->right=delData(leaf->right,leaf->data);

}else{

node<T>\* temp=leaf;

if(leaf->left==NULL)

leaf=leaf->right;

else if(leaf->right==NULL)

leaf=leaf->left;

delete temp;

}if(leaf==NULL)

return leaf;

int hLeft=height(leaf->left),hRight=height(leaf->right);

if(hLeft>hRight) leaf->height=hLeft+1;

else leaf->height=hRight+1;

leaf=balance(leaf);

return leaf;

}

template<class T>

node<T>\* Tree<T>::balance(node<T> \*leaf){

if(height(leaf->left)-height(leaf->right)==2){

if(height(leaf->left->left)-height(leaf->left->right)==1)

return RR(leaf); //Rotate left once

else return LR(leaf); //Rotate left, then right

}

else if(height(leaf->right)-height(leaf->left)==2){

if(height(leaf->right->right)-height(leaf->right->left)==1)

return LL(leaf); //Rotate right once

else return RL(leaf); //Rotate right, then left

}

return leaf;

}

template<class T>

void Tree<T>::inorder(node<T> \*leaf){

if(leaf!=NULL){

inorder(leaf->left);

cout<<leaf->data<<endl;

inorder(leaf->right);

}

}

template<class T>

void Tree<T>::preorder(node<T> \*leaf){

if(leaf!=NULL){

cout<<leaf->data<<endl;

preorder(leaf->left);

preorder(leaf->right);

}

}

template<class T>

void Tree<T>::postorder(node<T> \*leaf){

if(leaf!=NULL){

postorder(leaf->left);

postorder(leaf->right);

cout<<leaf->data<<endl;

}

}

#endif /\* TREE\_H \*/

/\* File: RecursiveSorts.h

\* Author: Javier B

\* Created on December 7, 2017, 11:49 AM

\* Purpose:Specification File for Recursive Sorts

\*/

#ifndef RECURSIVESORTS\_H

#define RECURSIVESORTS\_H

//System Libraries

#include <vector>

using namespace std; //Namespace of the System Libraries

//User Libraries

//Functions

//Recursive bubble Sort (Parameters:Vector,Sizeof() vector)

template<class T>

void bubbleRec(vector<T> &array,int size){

if(size==1) return; //Finished recursion

//Process Data

for(int i=0;i<size-1;i++){

if(array[i]>array[i+1]){

T temp=array[i];

array[i]=array[i+1];

array[i+1]=temp;

}

}

//End is sorted, sort previous

bubbleRec(array,size-1);

}

//Recursive insertion Sort (Parameters:Vector,Sizeof() vector)

template<class T>

void insertRec(vector<T> &array,int size){

if(size<=1) return; //Keep recursion going until you reach first index

//Process Data

insertRec(array,size-1); //From beg()array to end()array

T key=array[size-1];

int flag=size-2;

while(flag>=0&&array[flag]>key){

//swap()

array[flag+1]=array[flag];

flag-=1;

}

array[flag+1]=key;

}

#endif /\* RECURSIVESORTS\_H \*/

/\* File: Player.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Struct Specification File for Player

\*/

#ifndef PLAYER\_H

#define PLAYER\_H

//System Libraries

#include <iostream>

#include <string>

using namespace std; //Namespace of the System Libraries

//User Libraries

struct Player{

string name;

int money;

unsigned int score;

Player(){

money=50; //Player starts with $500.00

score=0; //Player starts with 0 points

}

friend bool operator> (const Player &left, const Player &right)

{return left.score>right.score;}

friend bool operator< (const Player &left, const Player &right)

{return left.score<right.score;}

friend ostream& operator<<(ostream &out,const Player p){

out<<p.name<<setw(5)<<right<<p.score<<" points"<<endl;

}

};

struct name\_sort{

bool operator()(const Player &left, const Player &right)

{return left.name<right.name;}

};

#endif /\* PLAYER\_H \*/

/\* File: Play.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Specification File for play class

\*/

#ifndef PLAY\_H

#define PLAY\_H

//System Libraries

#include <string> //Strings

using namespace std; //Namespace of the System Libraries

//User Libraries

#include "Game.h"

#include "Keyboard.h"

#include "Phrase.h"

//Variables

const int WHEEL=12; //Size of wheel

class Play{

private:

Clue clue; //Category and clue phrase

bool win; //Win or lose

Keyboard k; //Keyboard

Phrase p; //Phrase

int w[WHEEL]={0,5,5,10,10,15,15,20,25,30,35,40}; //Wheel spin options

public:

//Member Functions

void play(Game \*); //The actual game

void end(Game \*); //Ending screen, win or lose

void spin(Game \*); //Spin the wheel

void buy(Game \*); //Buy a vowel

void guess(Game \*);//Guess the phrase

void display(); //Display the keyboard and hidden phrase

void menu(Game \*); //Outputs the game menu

//Accessors

bool getWin() //Returns win boolean

{return win;}

int getMoney(Game \*a) //Returns player's money

{return a->getMoney();}

};

#endif /\* PLAY\_H \*/

/\* File: Phrase.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Specification File for Phrase class

\*/

#ifndef PHRASE\_H

#define PHRASE\_H

//System Libraries

#include <iostream> //Input/Output

#include <string> //String Library

using namespace std; //Namespace of the System Libraries

//User Libraries

#include "Letter.h"

class Phrase: public Letter{

private:

list<Letter> arr; //List

list<Letter>::iterator p;//Iterator to traverse

int size; //Size of array

public:

//Mutators

void use(int i);

//Accessors

int getSize(){return size;}

char getLetter(int);

bool getUsed(int);

//Member Functions

void setArr(string);

void display() override;

};

#endif /\* PHRASE\_H \*/

/\* File: Letter.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Specification File for Letter class

\*/

#ifndef LETTER\_H

#define LETTER\_H

//System Libraries

#include <iostream> //Input/Output

#include <list> //list

using namespace std; //Namespace of the System Libraries

//User Libraries

class Letter{

protected:

char letter;

bool isUsed;

public:

//Constructors

Letter(char);

Letter();

//Mutators

void setChar(char a)

{letter=a;}

void use()

{isUsed=true;}

//Accessors

char getLetter(){return letter;}

bool isLtUsed(){return isUsed;}

//Member functions

virtual void display(){cout<<letter;}

};

#endif /\* LETTER\_H \*/

/\* File: Keyboard.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Specification File for Keyboard class

\*/

#ifndef KEYBOARD\_H

#define KEYBOARD\_H

//System Libraries

#include <iostream> //Input/Output

using namespace std; //Namespace of the System Libraries

//User Libraries

#include "Letter.h"

//Constants

const int ALPHA=26; //Size of the alphabet

class Keyboard: public Letter{

public:

list<Letter> arr; //List

list<Letter>::iterator p;//Iterator to traverse

public:

//Constructor

Keyboard();

//Mutators

void use(int);

//Accessors

bool isUsed(int);

char getChar(int);

//Member functions

void display() override;

};

#endif /\* KEYBOARD\_H \*/

/\* File: Game.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Specification File for Game class

\*/

#ifndef GAME\_H

#define GAME\_H

//System Libraries

#include <iomanip> //Output manipulation

#include <vector> //Vectors

#include <algorithm> //For performing some algorithm functions

#include <stack> //Stacks

#include <iterator> //Iterator

#include <set> //Sets

#include <queue> //Queues

using namespace std; //Namespace of the System Libraries

//User Libraries

#include "Player.h"

#include "Clue.h"

#include "Hash.h"

//Variables

const int SIZE=44; //Max Size of Char array

class Game{

private:

Hash table; //A hash table full of phrases

Player user; //The player

stack <Clue> stck; //A stack of clues

fstream in; //Input

public:

//Constructor

Game(); //Load game, Introduction, sets random seed, creates library

//Destructor

~Game(); //closes file streams, appends to leaderboard, save game

//Mutators

void setName(string s) //Sets a player's name

{user.name=s;}

void setScore(unsigned int n) //Sets a player's score

{user.score=n;}

void setMoney(int n) //Sets a player's money

{user.money=n;}

//Accessors

string getName(){return user.name;}

unsigned int getScore(){return user.score;}

int getMoney(){return user.money;}

//Member Functions

void fill(); //Creates an index to the library

void menu(); //Displays the menu

void lderBrd(); //Displays a leaderboard

void write(); //Appends to the library

void read(); //Displays the entire library

void addLder(); //Adds profile to leaderboard

bool isGood(char[]); //Input verification

void play(); //Play the game

//Add Functions

void addMoney(int n)

{user.money+=n;}

void addScore(unsigned int n)

{user.score+=n;}

//Subtract Functions

void subMoney(int n)

{user.money-=n;}

//Play class can access private members of Game class

friend class Play;

};

#endif /\* GAME\_H \*/

/\* File: Clue.h

\* Author: Javier B

\* Created on October 20, 12:00 PM

\* Purpose: Class Specification File for Clue class

\*/

#ifndef CLUE\_H

#define CLUE\_H

//System Libraries

#include <string> //Strings

#include <iostream> //Input/Output

#include <fstream> //File input/Output

#include <cstring> //Cstrings for strlen() function

#include <map> //Maps

using namespace std; //Namespace of the System Libraries

//User Libraries

class Clue{

private:

map<int, string> catg;//Map of categories

int size;

unsigned int categry; //Number to represent a category

char phrase[44]; //Max Phrase length

public:

Clue(); //Initialize the map

//Mutators

void setCat(unsigned int);

void setPhrase(string);

//Accessors

char getPhrase(int i)

{return phrase[i];}

string getPhrase()

{return phrase;}

int getSize()

{return strlen(phrase);}

unsigned int getCat()

{return categry;}

//Output

void showCat()

{cout<<catg[categry]<<endl;}

//Operator overload

friend bool operator< (const Clue &left,const Clue &right){

string a=left.phrase,b=right.phrase;

return a<b;

}

friend bool operator> (const Clue &left,const Clue &right){

string a=left.phrase,b=right.phrase;

return a>b;

}

friend ostream& operator<<(ostream& out, Clue &clue){

clue.showCat();

out<<clue.phrase<<endl;

}

};

#endif /\* CLUE\_H \*/