**Project 1**

**The**

**Hangman**

**Theory**

**By: Philip Calderon**

**November 11, 2014**

**CIS-17C**

**Introduction:**

The Hangman Theory

This coded program utilizes the information we have used in class along with the information we have researched separately to form the creating we have come up with for project one. The game is initialized whenever the player/user chooses that option or they can insert words or other look at the current highscore. When the player does choose the option to start the game the game finds a random word from the difficulty settings. From there the user gets a choice of guessing the current word or imputing a letter to guess. If the player chooses the right letter imputs it into the code but if it’s wrong the hangman’s noose adds parts till the game ends. The player only gets 6 tries before the game is over or until they get the correct word in which case they win.

**Summary:**

The size of the Project: ~ 585 lines

Variables: about 33 variables

Functions: 13 functions

Spaces: 40

The project includes all the concepts required for the project. Most of the concepts were learned through the documentation of the STL libraries. The game logic was easy to create but implementing the required concepts was tricky. Using the documentation from the online website help lessen the line of code needed for the project. Functions such as find, delete, sort which each has more than 40 lines of code is lessen to a single line. I have chosen to create the game hangman because the concepts required are mostly on storing values into different type of storage organization.

**Description**

The game was interesting to implement together and I just couldn’t decide what game I wanted to do so I decided on the easiest game and that would be hangman. Although at first I had no idea what I wanted to do but hangman seems like a fun game to program in the end.

**Input / Output**

The input and output of this game is to utilize a menu system and options system to get the users input on what they want to do so they have ALL the control on what new word to add to the game, what difficulty they want it at in other words the range of how many words the game randomly chooses from to make you guess for the game, the decision of choosing a letter or guessing the word.

**Pseudo Code:**

*Initialize our variables, functions, etc.*

*Do {}while; loop at start // so that the player can keep playing and exit at any time*

*Display Menu // Shows the current options of the game*

*//Wait for information from the User/ Player*

*If input == 1*

*Start the game*

*If input == 2*

*Add a word to the game that is sent to the word.txt file*

*If input == 3*

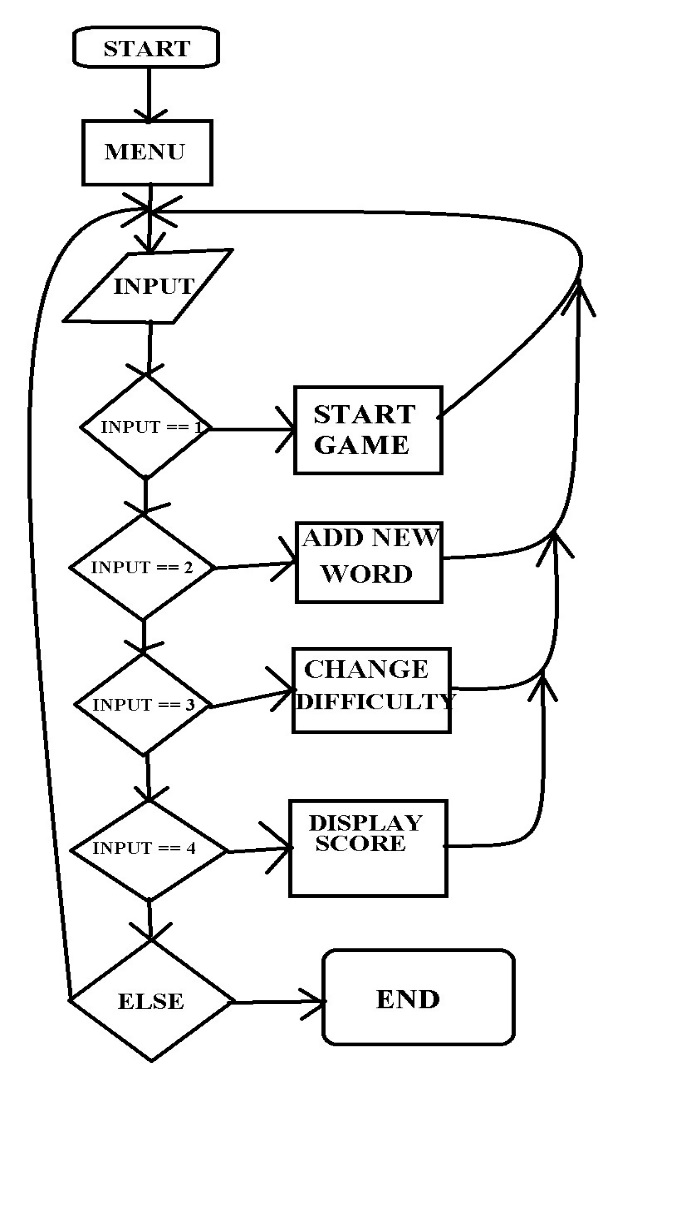
*Changes the current difficulty of the game*

*If input == 4*

*Displays the score from the highscore.txt file*

*Else Exit // Exit stage THAT A WAY!!! -->>[ ]*

**Flowchart:**

****

**Concepts:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Concept:** | **Name:** | **Description:** | **Location:** |
| Maps | fLetters | Utilizes maps to insert letters which where guessed | main.cpp |
| Sets | infoB | Used sets to write in highscore | main.cpp |
| Stacks | sImp | Implemented to keep track of the mistakes the user makes | sImp.h |
| Queues | muffinsQ | Reads in highscore and then prints it in a function | main.cpp |
| Iterators/Vectors | b | Iterator p is used for find function in stl | main.cpp |
| Algorithms | sort, find | sort is used to sort by string size, find uses to check if letter is guessed | main.cpp |

**Reference:**

STL Library Documentation

Cplusplus.com 🡨- Helped a lot with figuring out how to code some parts with less complexity.

**Program:**

**Main:**

// File: main.cpp

// Philip Calderon

// November 11, 2014

// Project 1

//System Libraries

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <string>

#include <cstring>

#include <fstream>

#include <vector>

#include <map>

#include <set>

#include <stack>

#include <queue>

#include <algorithm>

//Our Created Library

#include "sImp.h"

using namespace std;

//Function Prototypes

//For the Menu

void Menu();

int getN(); //Nodes

void def(int);

// For the Game

void startG();

void difficultyC();

void enterW();

void guessW(string, bool &, sImp &);

void hDisplay(sImp);

void scoreBrd();

//Debug

void prntVec(vector<string>);

//Execution

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

{

int inN;

do{

Menu();

inN = getN();

switch(inN){

case 1: startG();break;

case 2: enterW();break;

case 3: difficultyC();break;

case 4: scoreBrd();break;

default: def(inN);}

}while(inN >= 1 && inN <= 4);

return 0;

}

//////////////////////////////////

// Menu

//////////////////////////////////

void Menu()

{

cout << "HANGEMAN!!!" << endl;

cout << "Type 1 to Start A New Game"<<endl;

cout << "Type 2 to Add Word to The Game" << endl;

cout << "Type 3 to Change The Difficulty of The Game" << endl;

cout << "type 4 to Display The Score" << endl;

cout << "Press Another Key to Exit! \n" << endl;

}

int getN()

{

int inN;

cout << "Choice: ";

cin >> inN;

return inN;

}

void def(int inN)

{

cout << "Your exit key was " << inN << "." << endl;

}

//////////////////////////////////

// End of Menu

//////////////////////////////////

void startG()

{

//To get word randomly (Easy and Hard = SIZE 300, Random = SIZE 900)

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

int random;

vector<string> hWords;

//Open game word list

fstream file;

string word;

file.open("game.txt");

while(file >> word)

{

hWords.push\_back(word);

}

file.close();

//Check vector size

if(hWords.size()<=300)

{

// Get a random word from the list between 1 to 300

random = rand()%300+1;

}

if(hWords.size()==900)

{

// Get a random word from the list between 1 to 900

random = rand()%900+1;

}

//Word for the current game

string gWord;

gWord = hWords[random - 1];

//Utilizing maps to insert the found letters from the spacing underneath the noose "\_"

map<int,char> fLetters;

bool win = false;

int input;

string guess;

char letter;

//Count the mistakes

sImp count;

//Memorize letter used

vector<char> memW;

memW.push\_back('0');

//Array for all the Alphabet and check

char alphabet[26] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};

bool array[26] = {false};

do{

//Display Hangman

hDisplay(count);

for(int i = 0; i < gWord.size(); i++)

{

//Dont Know

if((float)fLetters[i]!=0)

{

cout << fLetters[i] << " ";

}

else if((float)fLetters[i]==0)cout << "\_";

}

cout << endl;

cout << "Options: " << endl;

cout << "1. Choose a Letter" << endl;

cout << "2. Guess the Word" << endl;

cout << "Choice: ";

cin >> input;

//Logic for guessing a letter

if(input == 1)

{

cout << "Letter: ";

cin >> letter;

//Store to vector

memW.push\_back(letter);

//Check if letter is already used

vector<char>::iterator b;

b = find(memW.begin(),memW.end(),letter);

cout << "The letter you used is: " << \*b << endl;

if(\*b == letter)

{

cout << "Sorry Letter has already been used!" << endl;

for(int i=0; i < 26; i++)

{

if(alphabet[i] == letter)

{

array[i] = true;

}

}

}

//Make character array for string

char cword[gWord.size()];

strcpy(cword,gWord.c\_str());

bool checkl = false;

//Check if the word has the letters

for(int i = 0; i < gWord.size(); i++)

{

if(letter == cword[i])

{

//In first element is the position, second element is the letter

fLetters[i] = letter;

checkl = true;

}

}

//Check if you guessed all letters in the current game

int countA = 0;

for(int i = 0; i < gWord.size(); i++)

{

if((float)fLetters[i]!=0)countA++;

}

if(countA == gWord.size())win = true;

//If the letters aren't found count the number of mistake

if(checkl == false)count.sLow();

}

//Logic for guessing a word

if(input == 2)

{

guessW(gWord, win, count);

}

cout<<"------------------------------------------------------"<<endl;

//Display Letter Used

cout<<"Letter Used: ";

for(int i = 0; i < 26; i++)

{

if(array[i] == true)

{

cout << alphabet[i] << " ";

}

}

cout << endl;

}while(win == false&&count.getS() < 6);

if(count.getS() == 6)

{

cout<<"YOU LOSE!"<<endl;

cout<<"------------------------------------------------------"<<endl;

hDisplay(count);

}

if(win == true)

{

cout<<"------------------------------------------------------"<<endl;

cout<<"You Win!"<<endl;

//Display Hangman

hDisplay(count);

for(int i=0; i < gWord.size(); i++)

{

//Dont Know

if((float)fLetters[i] != 0)

{

//Finds the letter and puts it into the allocated space if found

cout << fLetters[i] << " ";

}

else if((float)fLetters[i] == 0)cout << "\_ ";

}

//Display's The Answer

cout << endl;

cout << "Answer: " << gWord << endl;

//Ask for Your name to put in the highscore.txt

string name;

string texthigh;

cout << "Please Enter your name for Score: ";

cin >> name;

//Utilize Sets

set<string> infoB;

//Open high score text file

fstream hfile;

hfile.open("highscore.txt");

while(hfile >> texthigh)

{

infoB.insert(texthigh);

}

hfile.close();

//Entering the score for you

//Open to write high score

fstream writeh;

writeh.open("highscore.txt",fstream::in|fstream::out|fstream::app);

if(writeh.is\_open())

{

cout << "Writing to the File" << endl;

writeh << name << " " << count.getS() << endl;

}

writeh.close();

}

cout<<endl;

}

void enterW()

{

fstream file;

string word;

cout << "Type the word to be entered into the game: ";

cin >> word;

//Adds to the input word into the word.txt file

// Pulls the file, checks if it's in and if it's not applies the new word

file.open("word.txt",fstream::in|fstream::out|fstream::app);

if(file.is\_open())

{

file << word << endl;

}

file.close();

}

void guessW(string gWord,bool &win,sImp &count)

{

string word;

cout << "Word: ";

cin >> word;

if(word == gWord)

{

win = true;

}

else{

count.sLow();

}

}

//Sorting

bool smallest(const string& a, const string& b)

{

return (a.size() < b.size());

}

bool largest(const string& a, const string& b)

{

return (a.size() > b.size());

}

void difficultyC()

{

//Read inside the word.txt

fstream file;

string word;

file.open("word.txt");

vector<string> hWords;

//Read into vector

while(file >> word)

{

hWords.push\_back(word);

}

file.close();

//Test if the vector has words

//prntVec(hWords);

int choice;

//Difficulty Setting

cout << "Type 1 for Easy" << endl;

cout << "Type 2 for Hard" << endl;

cout << "Type 3 for Random Difficulty" << endl;

cout << "Choice: ";

cin >> choice;

//Creating a new file for the game

fstream filegame;

//Clear text file

filegame.open("game.txt",fstream::out|fstream::trunc);

//Sort word list from smallest to largest

if(choice == 1)

{

sort(hWords.begin(),hWords.end(),smallest);

for(int i = 0; i < hWords.size()/4; i++)

{

filegame << hWords[i] << endl;

}

}

//Sort word list from largest to smallest

if(choice == 2)

{

sort(hWords.begin(),hWords.end(),largest);

for(int i = 0; i < hWords.size()/4; i++)

{

filegame << hWords[i] << endl;

}

}

//No sort

if(choice == 3)

{

for(int i = 0; i < hWords.size(); i++)

{

filegame << hWords[i] << endl;

}

}

filegame.close();

cout << endl;

}

void hDisplay(sImp count)

{

if(count.getS() == 0)

{

//Empty Noose

cout << " @\_\_\_\_\_#" << endl;

cout << " | |" << endl;

cout << " | |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << "------------------" << endl;

}

if(count.getS() == 1)

{

//Addition of The Head

cout << " @\_\_\_\_\_#" << endl;

cout << " | |" << endl;

cout << " | |" << endl;

cout << " | O" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << "------------------" << endl;

}

if(count.getS() == 2)

{

//Addition of The Torso

cout << " @\_\_\_\_\_#" << endl;

cout << " | |" << endl;

cout << " | |" << endl;

cout << " | O" << endl;

cout << " | |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << "------------------" << endl;

}

if(count.getS() == 3)

{

//Addition of The Right Arm

cout << " @\_\_\_\_\_#" << endl;

cout << " | |" << endl;

cout << " | |" << endl;

cout << " | O" << endl;

cout << " | | ]" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << "------------------" << endl;

}

if(count.getS() == 4)

{

//Addition of The Left Arm

cout << " @\_\_\_\_\_#" << endl;

cout << " | |" << endl;

cout << " | |" << endl;

cout << " | O" << endl;

cout << " | [ | ]" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << "------------------" << endl;

}

if(count.getS() == 5)

{

//Addition of The Right Leg

cout << " @\_\_\_\_\_#" << endl;

cout << " | |" << endl;

cout << " | |" << endl;

cout << " | O" << endl;

cout << " | [ | ]" << endl;

cout << " | b" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << "------------------" << endl;

}

if(count.getS() == 6)

{

//Addition of The Left Leg

cout << " @\_\_\_\_\_#" << endl;

cout << " | |" << endl;

cout << " | |" << endl;

cout << " | O" << endl;

cout << " | [ | ]" << endl;

cout << " | d b" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << " |" << endl;

cout << "------------------" << endl;

}

}

void scoreBrd()

{

cout << endl;

fstream highScore;

queue<string> muffinsQ;

string scoring;

highScore.open("highscore.txt");

cout << "Display Score" << endl;

cout << "(# = mistakes)" << endl;

do{

getline(highScore,scoring);

muffinsQ.push(scoring);

}while(!highScore.eof());

//Print Score

cout << "------------------------------------" << endl;

while(!muffinsQ.empty())

{

cout << muffinsQ.front() << endl;

muffinsQ.pop();

}

cout << "------------------------------------" << endl;

highScore.close();

}

void prntVec(vector<string> hWords)

{

cout << endl << endl;

for(int i; i < hWords.size(); i++)

{

cout << hWords.at(i) << endl;

}

cout << endl;

}

**Header:**

#ifndef SIMP\_H\_INCLUDED

#define SIMP\_H\_INCLUDED

#include <stack>

#include <vector>

using namespace std;

class sImp

{

public:

sImp()

{

stkI.push(6); // Ends Here (If reversed it just ends first turn)

stkI.push(5);

stkI.push(4);

stkI.push(3);

stkI.push(2);

stkI.push(1);

stkI.push(0); // Starts here

}

void sLow()

{

stkI.pop();

}

int getS()

{

return stkI.top();

}

private:

stack<int> stkI;

};

#endif // STACKIMP\_H\_INCLUDED