**Project 1**

**<Hangman>**

**CIS-17C**

**Name: Andrew Kim**

**Date: 11/11/2014**

**Introduction**

Title: Hangman

This program is coded to play the popular game known has hangman in digital form. The game is coded so it utilizes every concept required for project 1.

The game starts by picking one of the randomized words, which can be changed through the difficulty setting menu. The player then is given a choice of guessing a letter or guessing a word. In either case if the player chooses wrong be it a letter or a word the text picture display will change depending on how many mistakes were made. The player loses if he or she makes six mistakes. The player wins if the word is guessed right or if the player guessed each letter for the given word.

**Summary**

Project size: about 430 lines

The number of variables: about 33 variables

The number of functions: 14 functions

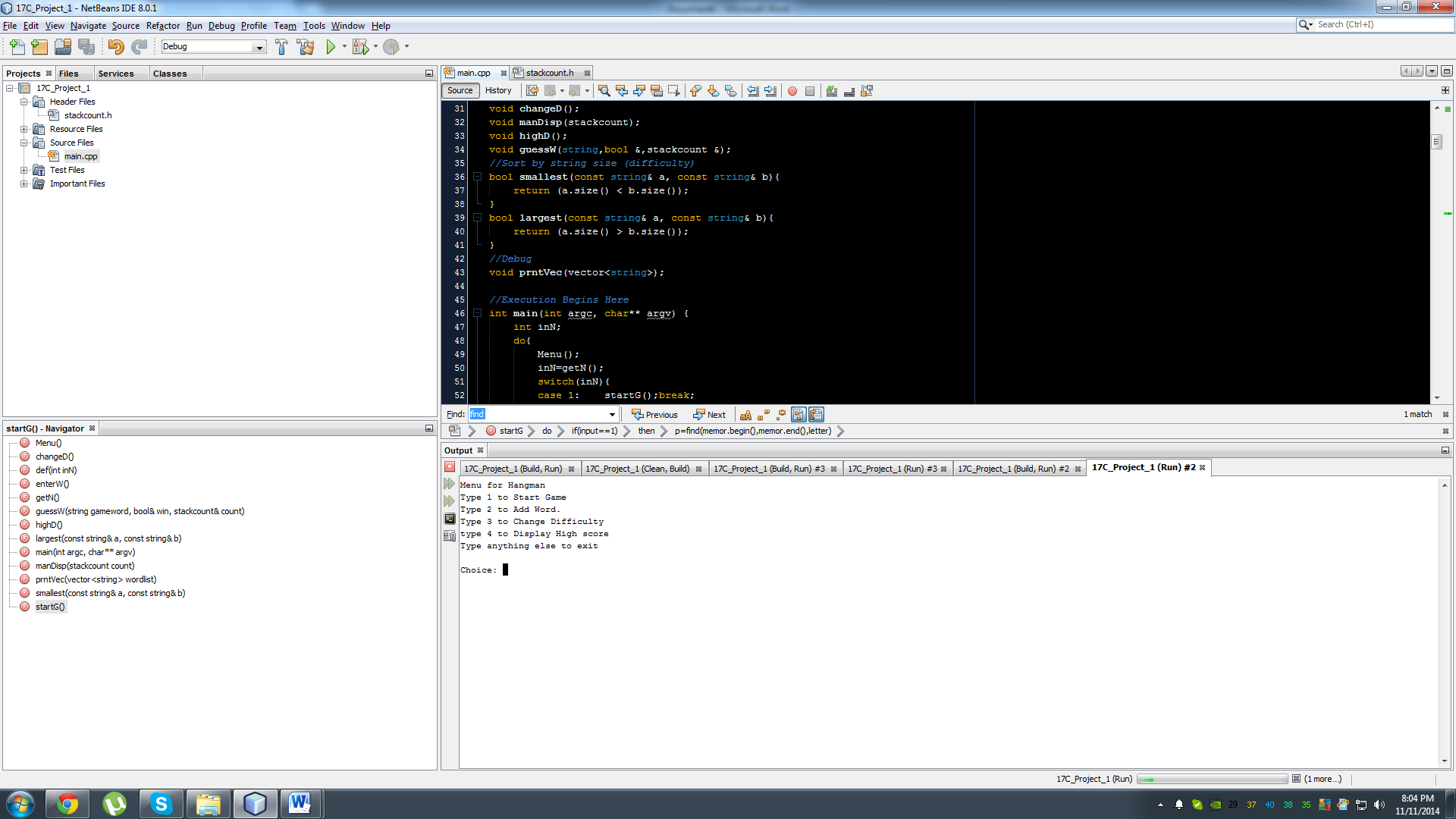
The project includes all the concepts required for the project. Most of the concepts were learned through the documentation of the STL libraries. The game logic was easy to create but implementing the required concepts was tricky. Using the documentation from the online website help lessen the line of 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 creation of the game was to implement all the concepts in class as easily as possible. The game hangman to code requires lots of storage for the word values or the logic for checking, so creating a game for the concepts needed was helpful.

**Input / Output**

The game is navigated and played through a menu system.



For example, the figure above shows a menu and gives you a choice to decide where you would like to go. If by choosing the number 4, it would display the score for people who previously played the game.

**Pseudo Code**

*Initialize*

*Do while loop in the beginning until player exits*

*Display Menu*

*If input 1*

*Start game of hangman*

*If input 2*

*Add word to the game by text file*

*If input 3*

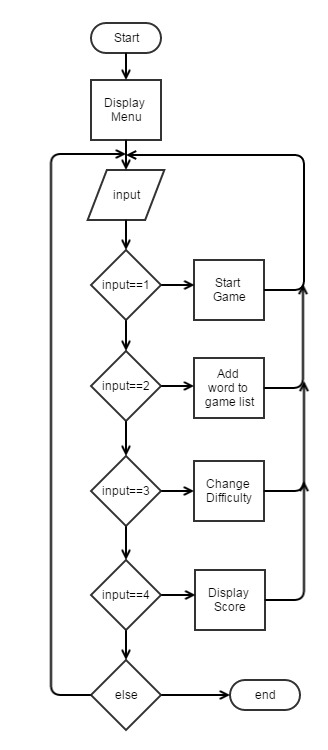
*Change Difficulty*

*If input 4*

*Display Score List*

*Else Exit*

**Flowchart**

****

**Required Concepts**

|  |  |  |  |
| --- | --- | --- | --- |
| **Concept** | **Name** | **Description** | **Location** |
| Maps | found | Utilizes maps to insert letters which where guessed | main.cpp |
| Sets | theset | Used sets to write in highscore | main.cpp |
| Stacks | stackc | Used to count mistakes a player makes | stackcount.h |
| Queues | myq | Reads in highscore and then prints it in a function | main.cpp |
| Iterators | p | 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 guessed | main.cpp |

**Reference**

STL Library Documentation online

**Program**

/\*

\* File: main.cpp

\* Author: Andrew Kim

\* Purpose: Project 1 - Hangman

\* Created on November 9, 2014, 5:31 PM

\*/

//System Libraries

#include <iostream>

#include <cstdlib>

#include <string>

#include <cstring>

#include <fstream>

#include <vector>

#include <algorithm>

#include <map>

#include <set>

#include <stack>

#include <queue>

#include "stackcount.h"

using namespace std;

//Function Prototypes

//Menu

void Menu();

int getN();

void def(int);

//Game

void startG();

void enterW();

void changeD();

void manDisp(stackcount);

void highD();

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

//Sort by string size (difficulty)

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());

}

//Debug

void prntVec(vector<string>);

//Execution Begins Here

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

int inN;

do{

Menu();

inN=getN();

switch(inN){

case 1: startG();break;

case 2: enterW();break;

case 3: changeD();break;

case 4: highD();break;

default: def(inN);}

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

return 0;

}

void Menu(){

cout<<"Menu for Hangman"<<endl;

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

cout<<"Type 2 to Add Word to the game"<<endl;

cout<<"Type 3 to Change Difficulty"<<endl;

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

cout<<"Type anything else to exit \n"<<endl;

}

int getN(){

int inN;

cout<<"Choice: ";

cin>>inN;

return inN;

}

void def(int inN){

cout<<"You typed "<<inN<<" to exit the game."<<endl;

}

void startG(){

//To get word randomly (Easy and Hard = SIZE 247, Random = SIZE 989)

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

int random;

vector<string> wordlist;

//Open game word list

fstream file;

string word;

file.open("game.txt");

while(file>>word){

wordlist.push\_back(word);

}

file.close();

//Check vector size

if(wordlist.size()<=247){

random=rand()%247+1; //Get random from 1~247

}

if(wordlist.size()==989){

random=rand()%989+1; //Get random from 1~989

}

//Word for game

string gameword;

gameword=wordlist[random-1];

//Answer

//cout<<gameword<<endl;

//cout<<"Size: "<<gameword.size()<<endl;

//Game Logic

//Utilize maps to insert found letters from "\_"

map<int,char> found;

bool win=false;

int input;

string guess;

char letter;

//Count mistakes

stackcount count;

//Memorize letter used

vector<char> memor;

memor.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

manDisp(count);

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

//Dont Know

if((float)found[i]!=0){

cout<<found[i]<<" ";

}

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

}

cout<<endl;

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

cout<<"1. Guess 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

memor.push\_back(letter);

//Check if letter is already used

vector<char>::iterator p;

p=find(memor.begin(),memor.end(),letter);

cout<<"Found: "<<\*p<<endl;

if(\*p==letter){

cout<<"Letter already used"<<endl;

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

if(alphabet[i]==letter){

array[i]=true;

}

}

}

//Make character array for string

char cword[gameword.size()];

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

bool checkl=false;

//Check if the word has the letters

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

if(letter==cword[i]){

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

found[i]=letter;

checkl=true;

}

}

//Check if you guessed all letters

int counta=0;

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

if((float)found[i]!=0)counta++;

}

if(counta==gameword.size())win=true;

//If not found count mistake

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

}

//Logic for guessing a word

if(input==2){

guessW(gameword,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;

manDisp(count);

}

if(win==true){

cout<<"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"<<endl;

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

//Display Hangman

manDisp(count);

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

//Dont Know

if((float)found[i]!=0){

cout<<found[i]<<" ";

}

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

}

//Display Answer

cout<<endl;

cout<<"Answer: "<<gameword<<endl;

//Ask for name to put in high score

string name;

string texthigh;

cout<<"Enter your name: ";

cin>>name;

//Utilize Sets

set<string> theset;

//Open high score text file

fstream hfile;

hfile.open("highscore.txt");

while(hfile>>texthigh){

theset.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 File"<<endl;

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

}

writeh.close();

}

cout<<endl;

}

void enterW(){

fstream file;

string word;

cout<<"Type the word to enter be listed in the game: ";

cin>>word;

//Adds to the word list

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

if(file.is\_open()){

file<<word<<endl;

}

file.close();

}

void changeD(){

//Read in word list

fstream file;

string word;

file.open("word.txt");

vector<string> wordlist;

//Read into vector

while(file>>word){

wordlist.push\_back(word);

}

file.close();

//Test if vector has words

//prntVec(wordlist);

int choice;

//Difficulty Setting

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

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

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

cout<<"Choice: ";

cin>>choice;

//Create new text 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(wordlist.begin(),wordlist.end(),smallest);

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

filegame<<wordlist[i]<<endl;

}

}

//Sort word list from largest to smallest

if(choice==2){

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

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

filegame<<wordlist[i]<<endl;

}

}

//No sort

if(choice==3){

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

filegame<<wordlist[i]<<endl;

}

}

filegame.close();

cout<<endl;

}

void manDisp(stackcount count){

if(count.getS()==0){

//Empty

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

cout<<" | |"<<endl;

cout<<" |"<<endl;

cout<<" |"<<endl;

cout<<" |"<<endl;

cout<<" |"<<endl;

cout<<"========="<<endl;

}

if(count.getS()==1){

//Head

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

cout<<" | |"<<endl;

cout<<" 0 |"<<endl;

cout<<" |"<<endl;

cout<<" |"<<endl;

cout<<" |"<<endl;

cout<<"========="<<endl;

}

if(count.getS()==2){

//Torso

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

cout<<" | |"<<endl;

cout<<" 0 |"<<endl;

cout<<" | |"<<endl;

cout<<" |"<<endl;

cout<<" |"<<endl;

cout<<"========="<<endl;

}

if(count.getS()==3){

//One Arm

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

cout<<" | |"<<endl;

cout<<" 0 |"<<endl;

cout<<" |) |"<<endl;

cout<<" |"<<endl;

cout<<" |"<<endl;

cout<<"========="<<endl;

}

if(count.getS()==4){

//Other Arm

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

cout<<" | |"<<endl;

cout<<" 0 |"<<endl;

cout<<" (|) |"<<endl;

cout<<" |"<<endl;

cout<<" |"<<endl;

cout<<"========="<<endl;

}

if(count.getS()==5){

//One Leg

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

cout<<" | |"<<endl;

cout<<" 0 |"<<endl;

cout<<" (|) |"<<endl;

cout<<" d |"<<endl;

cout<<" |"<<endl;

cout<<"========="<<endl;

}

if(count.getS()==6){

//Other Leg

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

cout<<" | |"<<endl;

cout<<" 0 |"<<endl;

cout<<" (|) |"<<endl;

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

cout<<" |"<<endl;

cout<<"========="<<endl;

}

}

void highD(){

cout<<endl;

fstream highs;

queue<string> myq;

string scoring;

highs.open("highscore.txt");

cout<<"Display Score"<<endl;

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

do{

getline(highs,scoring);

myq.push(scoring);

}while(!highs.eof());

//Print Score

cout<<"~~~~~~~~~~~~~~~~~~~~~~"<<endl;

while(!myq.empty()){

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

myq.pop();

}

cout<<"~~~~~~~~~~~~~~~~~~~~~~"<<endl;

highs.close();

}

void guessW(string gameword,bool &win,stackcount &count){

string word;

cout<<"Word: ";

cin>>word;

if(word==gameword){

win=true;

}

else{

count.stackdown();

}

}

void prntVec(vector<string> wordlist){

cout<<endl;

for(int i;i<wordlist.size();i++){

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

}

cout<<endl;

}