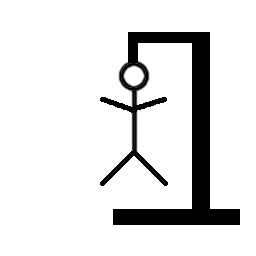
Hangman Game



Project 2

CSC – 5 #43952 Intro C++

Student Name: Michael Masli

Instructor: Professor M. Lehr

Content

1. Introduction……………………………………………………………….3

2. Useful Major Tools Information…………………………………6

3. List of Variables……………………….………….…………………..11

4. Covered Topics (Checklist)……………………………………….13

6. Libraries included…………………………………………………….17

7. Flowchart…………………………………………………………………17

8. Programming Code………………………………………………….25

Lines of code: 324

1. Introduction

**Rules and Gameplay**

Hangman is a guessing game for two or more players. One player thinks of a word, phrase or sentence and the other tries to guess it by suggesting letters or numbers. The word to guess is represented by a row of dashes, representing each letter of the word. Words you cannot use include proper nouns such as names, places, and brands. If the guessing player suggests a letter which occurs in the word, the other player writes it in all its correct positions. If the suggested letter or number does not occur in the word, the other player draws one element of a hanged man stick figure as a tally mark. The game is over when:

* The guessing player completes the word, or guesses the whole word correctly
* The other player completes the diagram of “Hangman”

**Strategy for the Game**

A common strategy is to guess vowels first, as English only has six vowels (a, e, i, o, u and y), and almost every word has at least one.

**Run through Example: Word = HANGMAN**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | [Hangman-0.png](http://en.wikipedia.org/wiki/File:Hangman-0.png)  Guess #1   |  |  | | --- | --- | | Word: | **\_ \_ \_ \_ \_ \_ \_** | | Guess: | e | | Misses: |  | |
|  | [Hangman-1.png](http://en.wikipedia.org/wiki/File:Hangman-1.png)  Guess #2   |  |  | | --- | --- | | Word: | **\_ \_ \_ \_ \_ \_ \_** | | Guess: | t | | Misses: | E | |
|  | [Hangman-2.png](http://en.wikipedia.org/wiki/File:Hangman-2.png)  Guess #3   |  |  | | --- | --- | | Word: | **\_ \_ \_ \_ \_ \_ \_** | | Guess: | a | | Misses: | e,t | |
|  | [Hangman-2.png](http://en.wikipedia.org/wiki/File:Hangman-2.png)  Guess #4 4dddd   |  |  | | --- | --- | | Word: | **\_ a \_ \_ \_ a \_** | | Guess: | o | | Misses: | e,t | |
|  | [Hangman-3.png](http://en.wikipedia.org/wiki/File:Hangman-3.png)  Guess #5   |  |  | | --- | --- | | Word: | **\_ a \_ \_ \_ a \_** | | Guess: | i | | Misses: | e,o,t | |
|  | [Hangman-4.png](http://en.wikipedia.org/wiki/File:Hangman-4.png)  Guess #6   |  |  | | --- | --- | | Word: | **\_ a \_ \_ \_ a\_** | | Guess: | s | | Misses: | e,i,o,t | |
|  | [Hangman-5.png](http://en.wikipedia.org/wiki/File:Hangman-5.png)  Guess #7   |  |  | | --- | --- | | Word: | **\_ a \_ \_ \_ a \_** | | Guess: | n | | Misses: | e,i,o,s,t | |
|  | [Hangman-5.png](http://en.wikipedia.org/wiki/File:Hangman-5.png)  Guess #8   |  |  | | --- | --- | | Word: | **\_ a n \_ \_ a n** | | Guess: | h | | Misses: | e,i,o,s,t | |
|  | [Hangman-5.png](http://en.wikipedia.org/wiki/File:Hangman-5.png)   |  |  | | --- | --- | | Word:  Guess #9 | **h a n \_ \_ a n** | | Guess: | r | | Misses: | e,i,o,s,t | |
|  | [Hangman-6.png](http://en.wikipedia.org/wiki/File:Hangman-6.png)   |  |  | | --- | --- | | Word:  Guess #10  (Number of guess has reached limit) | **h a n \_ \_ a n** | | Guess: | z | | Misses: | e,i,o,r,s,t | |
| Player lost the game - the correct word was “HANGMAN”. | | |

**Personal thoughts on Game**

I think the game is pretty straight forward and simple. The main ability the player need is a strategy of guessing because the next guess is based on the result of the previous guesses. The player needs to use the previous results and cross comparison to inference the right letter. Therefore, the main goal is to guess the correct word within a category in the game.

2. Useful Major Tools Information

1. **Arrays**  
   An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier. For example, five values of type “int” can be declared as an array without having to declare 5 different variables (each with its own identifier). Instead, using an array, the five “int” values are stored in contiguous memory locations, and all five can be accessed using the same identifier, with the proper index. Example of an array from my project:

string music[SIZE] = {

"rock", "hiphop", "pop", "folk", "classical", "jazz", "alternative", "blues","punk", "country" };

1. **Parallel Array**

A group of parallel arrays is a data structure for representing arrays of records. It keeps a separate, homogeneous array for each field of the record, each having the same number of elements. Then, objects located at the same index in each array are implicitly the fields of a single record. Pointers from one object to another are replaced by array indices. This contrasts with the normal approach of storing all fields of each record together in memory.

This is an example of a **parallel array** that I used in my project:

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

if (guess == **word[i]**){

**unknown[i]** = guess;

match=true;

}

}

1. **2 Dimensional Array**

A two-dimensional array is, in essence, a list of one-dimensional arrays. A two-dimensional array can be think as a table, which will have “x” number of rows and “y” number of columns. Think of it like a matrix in math. These are the examples that I used in my program:

char record[COL][COL];

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

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

record[i][j]='\_';

}

}

1. **Sorting an Array**

The Goal of sorting is to compare each array element and swap them if they are in the wrong position. Example of code:

//Show the Array (new)

**cout<<"Table of input: "<<endl;**

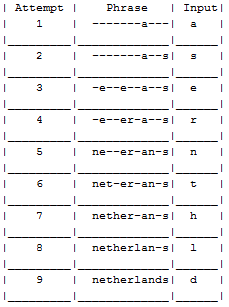
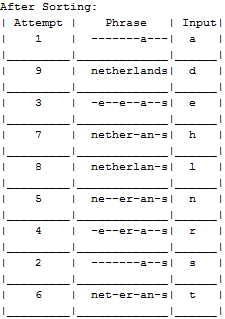
**showAry(tryArr, inputAr, record, ttlTry, unknown.length());**

//Sort the Array

**sortAry(tryArr, inputAr, record, ttlTry, unknown.length());**

**cout<<"After Sorting:\n";**

**showAry(tryArr, inputAr, record, ttlTry, unknown.length());**

 Example of the output:

As we can see in this example, the first table represents table of the player’s input. The other table represents the alphabetized order of input after sorting.

In addition, this is the sort array code that I did in my program:

**void sortAry**(int tryArr[], char inputAr[], char record[][COL], int ttlTry, int length){

bool swap;

char temp;

int itemp;

do{

swap=false;

for(int i=0;i<ttlTry-2;i++){

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

swap=true;

itemp=tryArr[i];

tryArr[i]=tryArr[i+1];

tryArr[i+1]=itemp;

temp=inputAr[i];

inputAr[i]=inputAr[i+1];

inputAr[i+1]=temp;

for(int j=0;j<length;j++){

temp = record[i][j];

record[i][j]= record[i+1][j];

record[i+1][j] = temp;

}

}

}

}while(swap);

}

1. **String Length**

String length returns the length of the string, in terms of bytes. This is the number of actual bytes that conforms the contents of the string. One example that I used in my project is (see it in red):

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

unknown+="-";

}

1. **Loops**  
   A loop is a way of repeating a statement a number of times until some way of ending the loop occurs. It might be run for a preset number of times, typically in a “for loop”, repeated as long as an expression is true (a “while loop”) or repeated until an expression becomes false in a “do while loop”. In this project, I utilized a “for” loop and a “do while” loop. Here is one of example of loops that I utilized in my program:

For Loop: for(int i=0;i<word.length();i++){

unknown+="-";

}

1. **Function Prototypes**  
   A function prototype is a declaration of a function that specifies the function's name and type signature (parameter types, return type, etc), but omits the function body. Elsewhere in the program, a function definition must be provided if one wishes to use this function. There is another function that acts differently as a function prototype, which is called the **void function**. A function with void result type ends either by reaching the end of the function or by executing a return statement with no returned value. In other words, the function takes no arguments. It's important to be aware that a declaration of a function does not need to include any arguments. In this program, my function prototypes are:

* **bool** letFill (char, string, string&) 🡪Letter Fill function
* **void** getWord (string [], string &, string &) 🡪 Unknown word function
* **float** percent (int, int) 🡪 guessing accuracy percentage in decimal
* **void** display(int) 🡪 to display the “HANGMAN”
* **bool** valid(char , string) 🡪 Input validation
* **void** showAry(int [], char [], char[][COL], int total\_try, int unknown\_length) 🡪 To print the array
* **void** sortAry(int [], char [], char[][COL], int, int) 🡪 Sorting the array

1. **String**

String class is a standard representation for a text string. In this project, I utilized string combined with an array, which stores the unknown words that the guesser has to guess in the game. One of the examples from my project:

string sports[SIZE] = {

"football", "swimming", "soccer", "basketball", "cricket", "baseball", "running", "tennis", "badminton", "racing" };

3. List of Variables

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Type** | **Variable Name** | **Description** | **Line** |
| int | count | To count the number of guess, then display hangman | 323 |
|  | nWrong=0 | Initializing the number of wrong guesses | 35 |
|  | choice | Type 1, 2, or 3 to pick a category | 104 |
|  | atmpt | Number of guess attempted | 391 |
|  | max | Number of max. tries | 391 |
|  | itemp | Temporary integer | 417 |
|  | length | The length of unknown word in sort array | 414 |
|  | tryArr[COL] | number of tries in array | 40 |
|  | ttlTry=1 | Initialize the number of tries | 41 |
| const int | MaxTRY=8 | Initialize the number of maximum tries | 33 |
|  | SIZE=10 | Size of an array is 10 | 37 |
|  | COL=26 | size of 2D array | 20 |
| char | letter | Input a letter to guess the word | 34 |
|  | guess | Our number of guess | 293 |
|  | inputAr[COL] | For the input | 38 |
|  | record[COL][COL] | Record of unknown and number of tries | 39 |
|  | temp | Temporary variable | 416 |
| string | word | The unknown word that we are trying to guess | 36 |
|  | words[SIZE] | Country names category stored in an array | 51 |
|  | sports[SIZE] | Sports category stored in an array | 64 |
|  | music[SIZE] | Music genre category stored in an array | 77 |
|  | unknown | The unknown word | 90 |
|  | temp | Temporary variable is declared in order to help input the file | 94 |
|  | output="" | Starts as a blank string. Then, when we do the for loop, it’ll just add the 2D array for ‘record’ | 405 |
| float | percent() | Gamer’s guessing accuracy in decimal | 391 |
| bool | match=false | Initialize if the answer matches as false | 295 |
|  | swap | To swap | 415 |
| fstream | input | Input stream (open file) | 92 |
|  | output | Output stream (close file) | 311 |

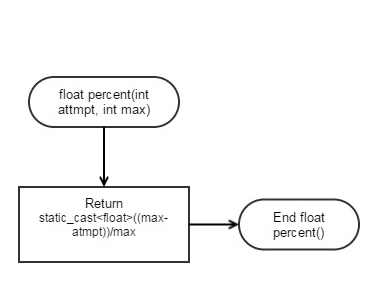
4. Covered Topics (Checklist)

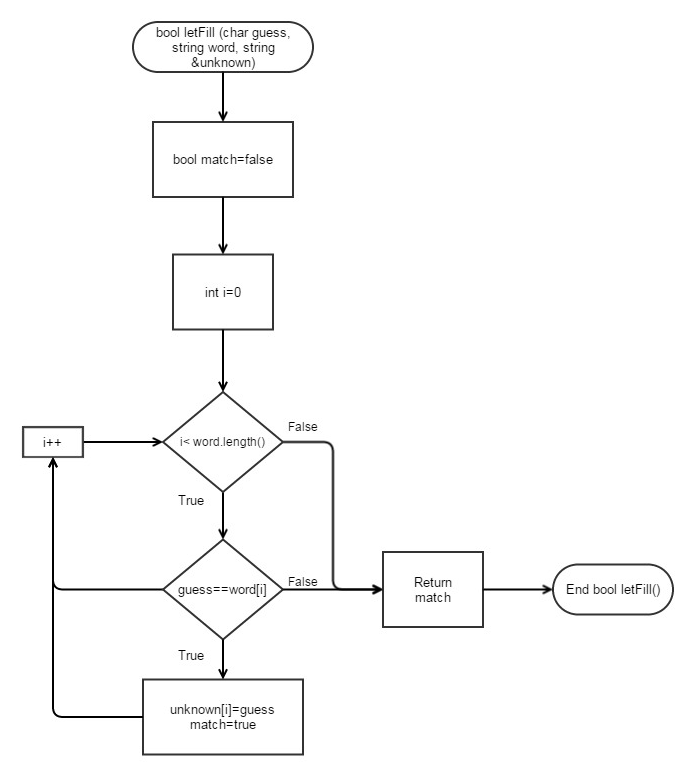
|  |  |  |  |
| --- | --- | --- | --- |
| **Chapter** | **Type** | **Code** | **Line** |
| 2.1 Variables | int | int nWrng=0; | 32 |
| 2.2 Input Output | cin | cin>>letter; | 115 |
|  | cout | cout<<unknown<<endl; | 126 |
| 2.3 Data Types | char | char letter; | 31 |
|  | bool | bool match=false; | 231 |
|  | string | string word; | 33 |
| 2.4 Condition | = | Int nWrng=0; | 32 |
|  | == | if (count==1) | 255 |
|  | ++ | i++; | 249 |
| 2.5 Style | comment | //choose and copy a word from array of words randomly | 241 |
| 3.1 Expression | >, &&, != | while(nWrng<MaxTRY && word!=unknown){ | 134, 176, 218 |
| 3.2 Multiway branches | switch | switch(choice){ | 106 |
|  | if | (letFill(letter, word, unknown)==false){ | 118 |
|  | else | else{ cout<<endl<< "Yes! You found a letter, keep going!" <<endl; } | 123-125 |
|  | nested | for (int i = 0; i< word.length(); i++){ | 232 |
|  |  | do{ | 95 |
|  | break | break; | 219 |
| 3.3 Type of Loop | for | for(int i=0;i<word.length();i++){ | 249 |
|  | do-while | do{} while(choice>=1 && choice<=3); | 225 |
| 4.2 Predefined Function | srand, time | srand (time(0)); | 39 |
|  | rand | word=arr[rand()%10]; | 242 |
| 4.3 Function Prototypes | float | float percent (); | 322 |
| 5.1 Void Function | void | void display(); | 254 |
| 5.2 Call-by-reference | & | void getWord(string arr[], string &word, string &unknown); | 240 |
| 6.1 Streams and Basic | fstream declare | fstream output; | 243 |
|  | output | output.open("word.txt", ios::out); | 244 |
|  | close | output.close(); | 246 |
| 7.1 Array | string array | string words[SIZE] | 40 |
| 7.4 Array Initialization | const int | const int SIZE=10 | 37 |
|  | string | string sports[SIZE] = {  "football",  "swimming",  "soccer",  "basketball",  "cricket",  "baseball",  "running",  "tennis",  "badminton",  "racing"  }; | 64 |
| 7.5 Processing Array Contents | string | output+=record[i][j] | 403 |
| 7.6 Using Parallel Arrays | char | if (guess == word[i]){  unknown[i] = guess;  match=true;  } | 297 |
| 7.7 Arrays as Function Arguments | string array | void getWord(string arr[], string &word, string &unknown) | 304 |
| 7.8 Two Dimensional Arrays | char | char record[COL][COL] | 39 |
| 7.9 Array of Strings | string | string music[SIZE] = {  "rock",  "hiphop",  "pop",  "folk",  "classical",  "jazz",  "alternative",  "blues",  "punk",  "country"  }; | 77 |
| 8.5 Selection Sort | void | //Show the Array  cout<<"Table of input: "<<endl;  showAry(tryArr, inputAr, record, ttlTry, unknown.length());  //Sort the Array  sortAry(tryArr, inputAr, record, ttlTry, unknown.length());  cout<<"After Sorting:\n";  showAry(tryArr, inputAr, record, ttlTry, unknown.length()); | 166-170 |

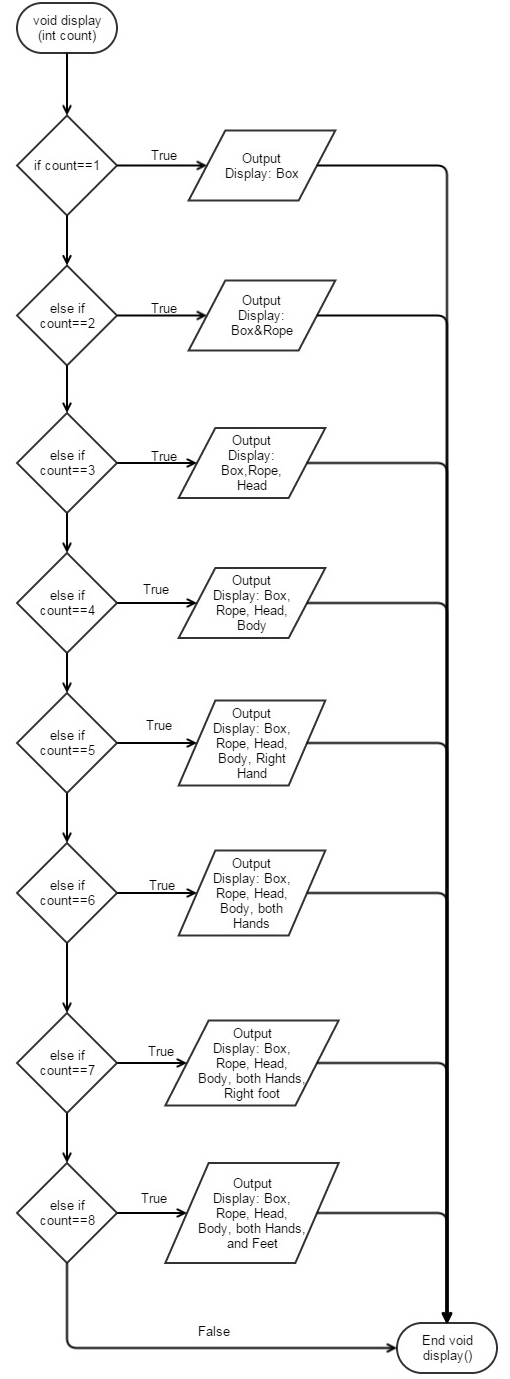
5. Libraries included

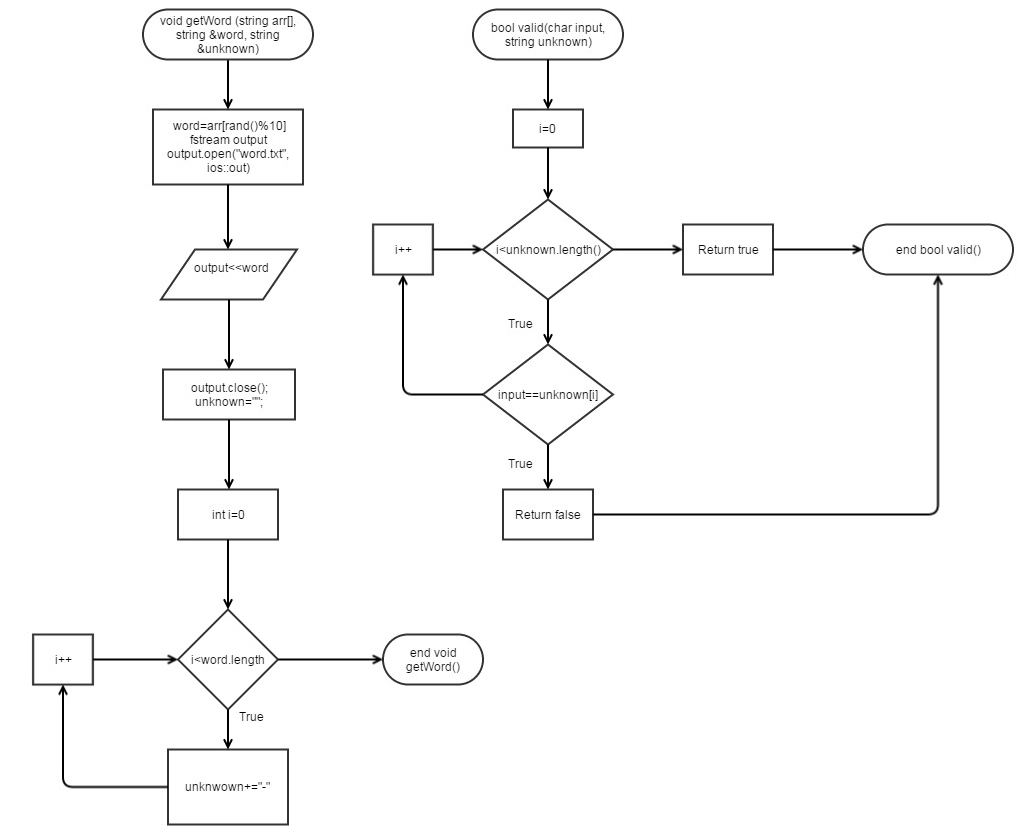
* #include <string>
* #include <iostream>
* #include <cstdlib>
* #include <ctime>
* #include <iomanip>
* #include <fstream>

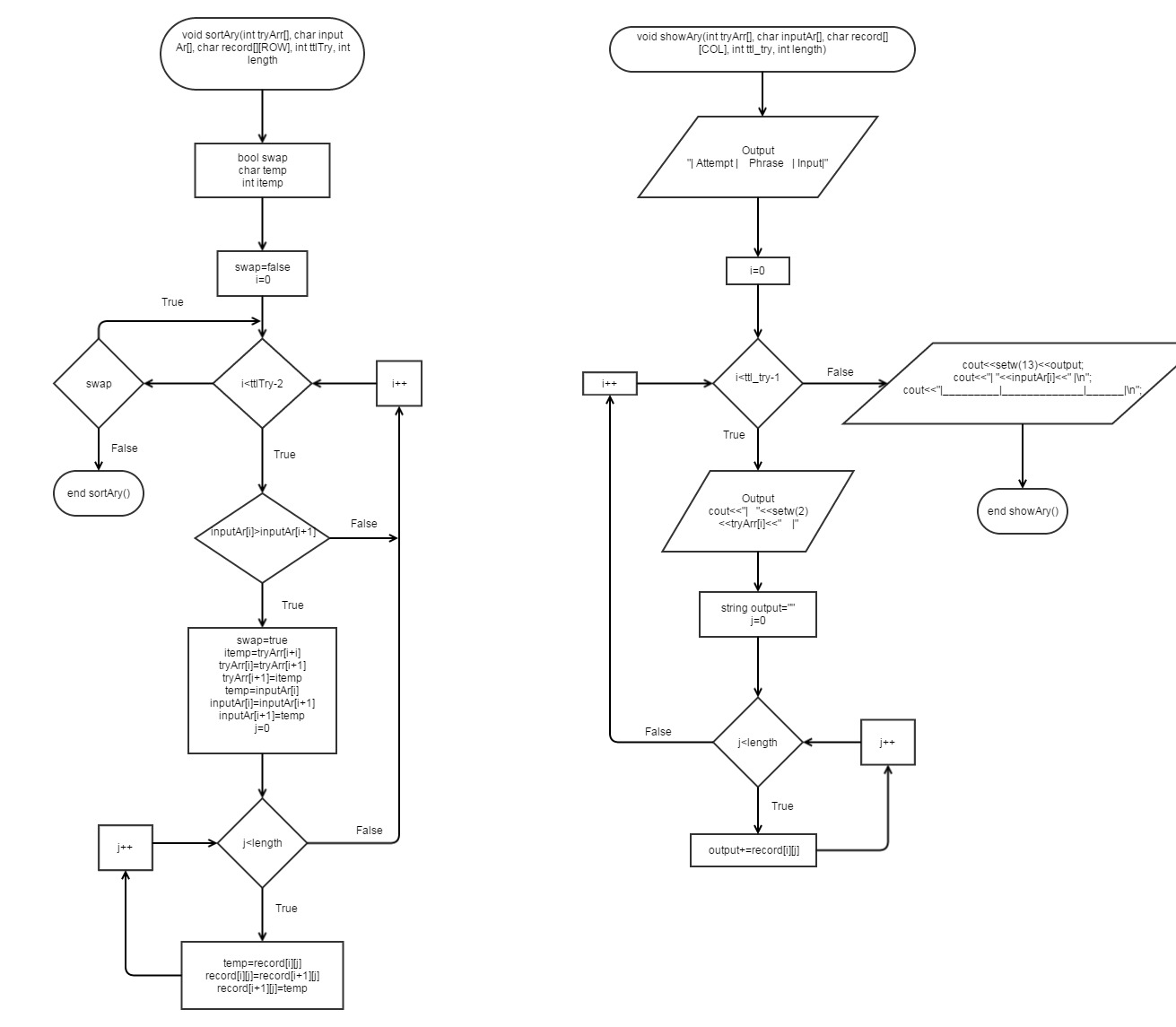
6. Flowchart

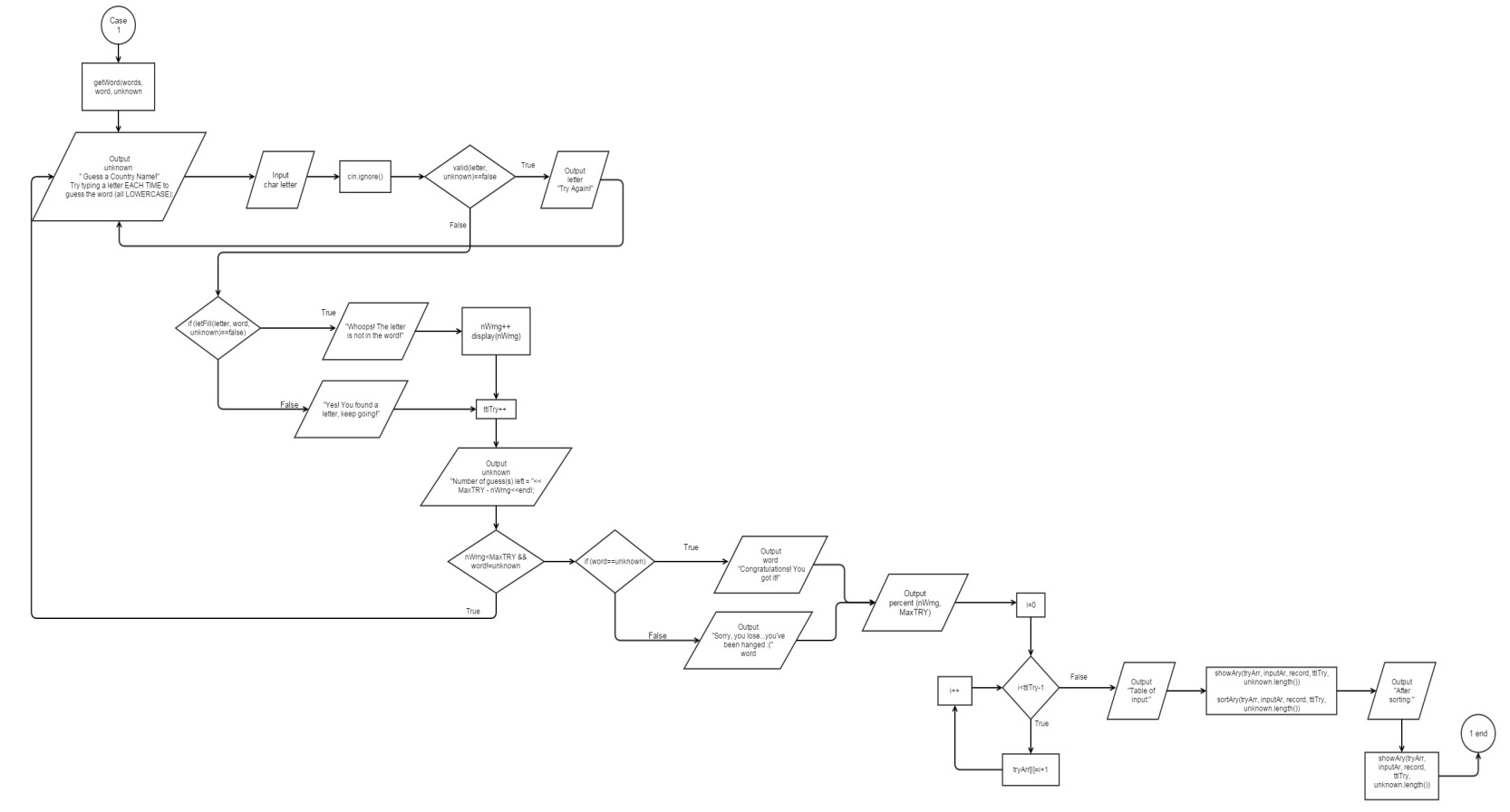
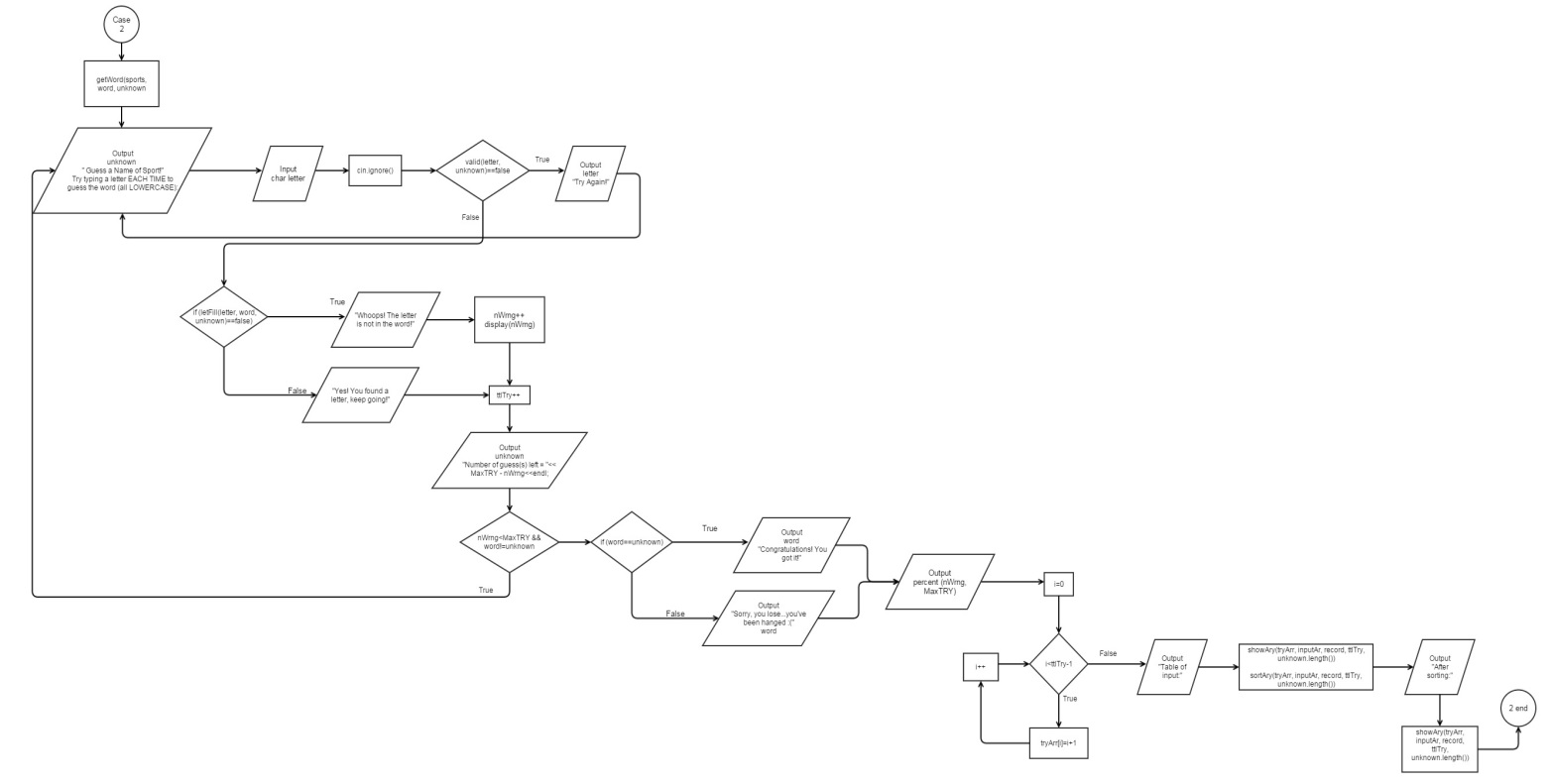
* Function Prototypes

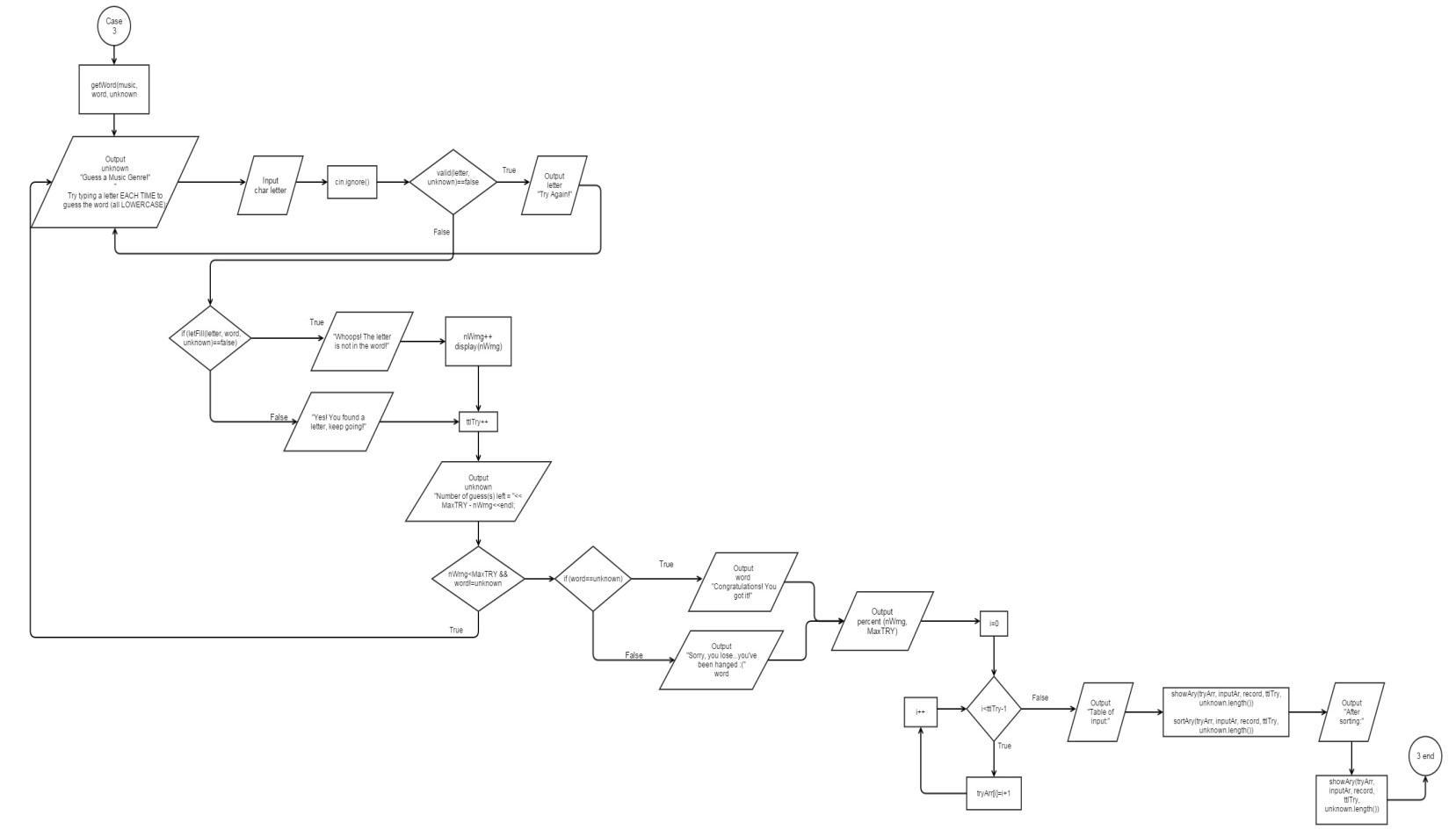




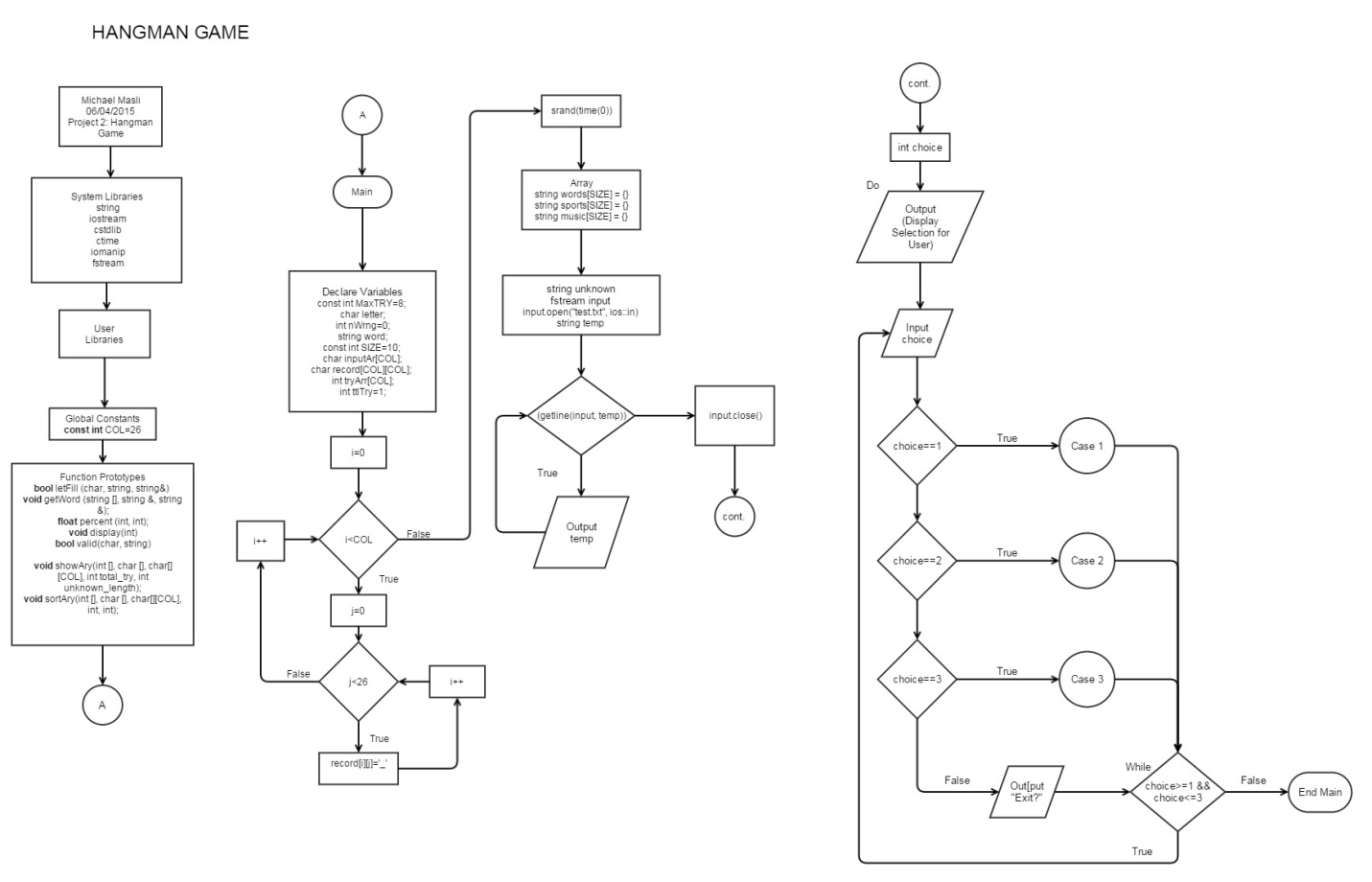




* Case 1
* Case 2
* Case 3



* Main



**7. Programming Code**

/\*

\* File: main.cpp

\* Author: Michael Masli

\*

\* Created on May 25, 2015, 14:32 AM

\* Purpose: Hangman Game

\*/

//User Libraries

//System Libraries

#include <string>

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <iomanip>

#include <fstream>

using namespace std;

//Global Constants

const int COL=26; //size of 2d array (alphabet characters = 26 letters)

//Function Prototypes

bool letFill (char, string, string&, int, char[][COL]); //Letter Fill function

void getWord (string [], string &, string &); //'&' call by referrence

float percent (int, int); // guessing accuracy percentage in decimal

void display(int); //display hangman

bool valid(char, string); //input validation

//Display input array and 2d record array

void showAry(int [], char [], char[][COL], int total\_try, int unknown\_length);

void sortAry(int [], char [], char[][COL], int, int);

//Execution Begins Here

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

//Declare Variables

const int MaxTRY=8; //number of maximum tries

char letter; // input the letter to guess the word

int nWrng=0; //num. of wrong guesses

string word;

const int SIZE=10; //size of array

char inputAr[COL]; //1D array (new))

char record[COL][COL]; //record of unknown and try 2D (new)

int tryArr[COL]; //new

int ttlTry=1; //initialize number of try (new)

cout<<setprecision(2)<<fixed<<showpoint; //decimal format

//New Loop (for the "unknown" unrevealed word)

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

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

record[i][j]='\_'; //2D array print out unrevealed unknown word

}

}

//set the random time seed

srand(time(0));

string words[SIZE] = {

"china",

"germany", //7

"england", //

"netherlands", //9

"philippines",

"australia",

"turkey",

"greece",

"uganda",

"indonesia"

};

string sports[SIZE] = {

"football",

"swimming",

"soccer",

"basketball",

"cricket",

"baseball",

"running",

"tennis",

"badminton",

"racing"

};

string music[SIZE] = {

"rock",

"hiphop",

"pop",

"folk",

"classical",

"jazz",

"alternative",

"blues",

"punk",

"country"

};

string unknown; //for the unknown word

//Input file

fstream input;

input.open("test.txt", ios::in);

string temp;

while(getline(input, temp))cout<<temp<<endl;

input.close();

//Prompt the User

// cout<<"Hello there! Welcome to HANGMAN...Guess the correct word."<<endl;

// cout<<"Each letter is represented by the character '-' "<<endl;

// cout<<"You get to type a letter in each try."<<endl;

// cout<<"You have 8 tries to guess the correct word in the selected category."<<endl;

//Menu format

int choice;

//Repeat the menu

do{

nWrng=0;

//General Menu Format

//Display the selection

cout<<"\nType 1 to guess a country name."<<endl;

cout<<"Type 2 to guess a name of sport."<<endl;

cout<<"Type 3 to guess a music genre."<<endl;

cout<<"Type anything else to quit playing this game."<<endl;

//Read the choice

cin>>choice;

cin.ignore();

//Solve a problem that has been chosen.

switch(choice){

case 1:{

getWord(words, word, unknown);

//Loop until the guesses are used up

do{

//cout<<"\ntotal try = "<<ttlTry<<endl;

//Input Validation

do{

cout<<"\n"<< unknown;

cout<<"\nGuess a Country Name!\nTry typing a letter EACH TIME to guess the word (all LOWERCASE): ";

cin>>letter;

cin.ignore();

if(valid(letter, unknown)==false)cout<<"Letter '"<<letter<<"' was input before, try again!\n";

}while(valid(letter, unknown)==false);

inputAr[ttlTry-1]=letter;

//Conditions

if (letFill(inputAr[ttlTry-1], word, unknown, ttlTry, record)==false){

cout<<endl<< "Whoops! The letter is not in the word!"<<endl;

nWrng++;

display(nWrng);

}

else{

cout<<endl<< "Yes! You found a letter, keep going!" <<endl;

}

ttlTry++; //increment ttl try (new)

//cout<<unknown<<endl;

//Inform the user for how many guess the user has

cout<<"Number of guess(s) left = "<< MaxTRY - nWrng;

// Check if user guessed the word.

}while(nWrng<MaxTRY && word!=unknown);

if (word==unknown){

cout <<"\nThe word is "<<word<<endl;

cout << "Congratulations! You got it!"<<endl;

}

else{

cout << "\nSorry, you lose...you've been hanged :(" << endl;

cout << "The correct word was : " << word << endl;

}

//cout<<nWrng<<" "<<MaxTRY<<endl;

cout<<"Your guessing accuracy in decimal point is "<<percent(nWrng, MaxTRY)<<endl;

for(int i=0;i<ttlTry-1;i++){ //new loop

tryArr[i]=i+1;

}

//Show the Array (new)

cout<<"Table of input: "<<endl;

showAry(tryArr, inputAr, record, ttlTry, unknown.length());

//Sort the Array

sortAry(tryArr, inputAr, record, ttlTry, unknown.length());

cout<<"After Sorting:\n";

showAry(tryArr, inputAr, record, ttlTry, unknown.length()); //Show the array after sorting

break;

}

case 2:{

getWord(sports, word, unknown);

//Loop until the guesses are used up

do{

//Input Validation

do{

cout<<"\n"<< unknown;

cout<<"\nGuess a Name of Sport\nTry typing a letter EACH TIME to guess the word (all LOWERCASE): ";

cin>>letter;

cin.ignore();

if(valid(letter, unknown)==false)cout<<"Letter '"<<letter<<"' was input before, try again!\n";

}while(valid(letter, unknown)==false);

inputAr[ttlTry-1]=letter;

//Conditions

if (letFill(inputAr[ttlTry-1], word, unknown, ttlTry, record)==false){

cout<<endl<< "Whoops! The letter is not in the word!"<<endl;

nWrng++;

display(nWrng);

}

else{

cout<<endl<< "Yes! You found a letter, keep going!" <<endl;

}

ttlTry++;

//cout<<unknown<<endl;

//Inform the user for how many guess the user has

cout<<"Number of guess(s) left = "<< MaxTRY - nWrng;

// Check if user guessed the word.

}while(nWrng<MaxTRY && word!=unknown);

if (word==unknown){

cout <<"\nThe word is "<<word<<endl;

cout << "Congratulations! You got it!"<<endl;

}

else{

cout << "\nSorry, you lose...you've been hanged :(" << endl;

cout << "The correct word was : " << word << endl;

}

//cout<<nWrng<<" "<<MaxTRY<<endl;

cout<<"Your guessing accuracy in decimal point is "<<percent(nWrng, MaxTRY)<<endl;

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

tryArr[i]=i+1;

}

//Show the Array

cout<<"Table of input: "<<endl;

showAry(tryArr, inputAr, record, ttlTry, unknown.length());

//Sort the Array

sortAry(tryArr, inputAr, record, ttlTry, unknown.length());

cout<<"After Sorting:\n";

showAry(tryArr, inputAr, record, ttlTry, unknown.length()); //Show the array after sorting

break;

}

case 3:{

getWord(music, word, unknown); //getWord--> randomly pick word from music array

//Loop until the guesses are used up

do{

//Input Validation

do{

cout<<"\n"<< unknown;

cout<<"\nGuess a name of Music Genre!\nTry typing a letter EACH TIME to guess the word (all LOWERCASE): ";

cin>>letter;

cin.ignore();

if(valid(letter, unknown)==false)cout<<"Letter '"<<letter<<"' was input before, try again!\n";

}while(valid(letter, unknown)==false);

//Conditions

inputAr[ttlTry-1]=letter;

if (letFill(inputAr[ttlTry-1], word, unknown, ttlTry, record)==false){

cout<<endl<< "Whoops! The letter is not in the word!"<<endl;

nWrng++;

display(nWrng);

}

else{

cout<<endl<< "Yes! You found a letter, keep going!" <<endl;

}

ttlTry++; //increment ttl try

//cout<<unknown<<endl;

//Inform the user for how many guess the user has

cout<<"Number of guess(s) left = "<< MaxTRY - nWrng;

// Check if user guessed the word.

}while(nWrng<MaxTRY && word!=unknown);

if (word==unknown){

cout <<"\nThe word is "<<word<<endl;

cout << "Congratulations! You got it!"<<endl;

}

else{

cout << "\nSorry, you lose...you've been hanged :(" << endl;

cout << "The correct word was : " << word << endl;

}

//cout<<nWrng<<" "<<MaxTRY<<endl;

cout<<"Your guessing accuracy in decimal point is "<<percent(nWrng, MaxTRY)<<endl;

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

tryArr[i]=i+1;

}

//Show the Array

cout<<"Table of input: "<<endl;

showAry(tryArr, inputAr, record, ttlTry, unknown.length());

//Sort the Array

sortAry(tryArr, inputAr, record, ttlTry, unknown.length());

cout<<"After Sorting:\n";

showAry(tryArr, inputAr, record, ttlTry, unknown.length()); //Show the array after sorting

break;

}

default:{

cout<<"Exit?"<<endl;

}

}

} while(choice>=1 && choice<=3);

//Exit Stage Right

return 0;

}

//The function of changing unknown

bool letFill (char guess, string word, string &unknown, int ttlTry, char record[][COL]){

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

bool match=false;

for (int i = 0; i< word.length(); i++){ //word.length = how many characters are in side

if (guess == word[i]){ //parallel array same index for word and unknown

unknown[i] = guess;

match=true;

}

}

//add unknown to record

for(int j=0;j<unknown.length();j++){

record[ttlTry-1][j]=unknown[j]; //num of input variable is what? change it

}

return match;

}

void getWord(string arr[], string &word, string &unknown){

//choose and copy a word from array of words randomly

word=arr[rand()%10];

fstream output;

output.open("word.txt", ios::out);

output<<word;

output.close();

unknown="";

//Initialize the unkn word with the "-" character.

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

unknown+="-";

}

//cout<<word;

}

//Function that displays hangman

void display(int count){

if(count==1){

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

cout<<"| |\n";

cout<<"| |\n";

cout<<"| |\n";

cout<<"| |\n";

cout<<"|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

}

else if(count==2){

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

cout<<"| | |\n";

cout<<"| |\n";

cout<<"| |\n";

cout<<"| |\n";

cout<<"|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

}

else if(count==3){

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

cout<<"| | |\n";

cout<<"| O |\n";

cout<<"| |\n";

cout<<"| |\n";

cout<<"|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

}

else if(count==4){

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

cout<<"| | |\n";

cout<<"| O |\n";

cout<<"| | |\n";

cout<<"| |\n";

cout<<"|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

}

else if(count==5){

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

cout<<"| | |\n";

cout<<"| O |\n";

cout<<"| /| |\n";

cout<<"| |\n";

cout<<"|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

}

else if(count==6){

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

cout<<"| | |\n";

cout<<"| O |\n";

cout<<"| /|\\ |\n";

cout<<"| |\n";

cout<<"|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

}

else if(count==7){

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

cout<<"| | |\n";

cout<<"| O |\n";

cout<<"| /|\\ |\n";

cout<<"| / |\n";

cout<<"|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

}

else if(count==8){

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

cout<<"| | |\n";

cout<<"| O |\n";

cout<<"| /|\\ |\n";

cout<<"| / \\ |\n";

cout<<"|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n";

cout<<"YOU JUST LOST!\n";

}

}

float percent (int atmpt, int max){

return static\_cast<float>((max-atmpt))/max;

}

bool valid(char input, string unknown){

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

if(input==unknown[i]) return false;

}

return true;

}

void showAry(int tryArr[], char inputAr[], char record[][COL], int ttl\_try, int length){

cout<<"| Attempt | Phrase | Input|\n";

for(int i=0;i<ttl\_try-1;i++){

cout<<"| "<<setw(2)<<tryArr[i]<<" |";

string output=""; //starts as a blank string

for(int j=0;j<length;j++){

output+=record[i][j];

}

cout<<setw(13)<<output;

cout<<"| "<<inputAr[i]<<" |\n";

cout<<"|\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_|\n";

}

}

void sortAry(int tryArr[], char inputAr[], char record[][COL], int ttlTry, int length){

bool swap;

char temp;

int itemp;

do{

swap=false;

for(int i=0;i<ttlTry-2;i++){

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

swap=true;

itemp=tryArr[i];

tryArr[i]=tryArr[i+1];

tryArr[i+1]=itemp;

temp=inputAr[i];

inputAr[i]=inputAr[i+1];

inputAr[i+1]=temp;

for(int j=0;j<length;j++){

temp = record[i][j];

record[i][j]= record[i+1][j];

record[i+1][j] = temp;

}

}

}

}while(swap);

}