Project 1

Hangman

CSC-5 Summer 46024

Taught by Dr. Mark Lehr

Guthrie Price

13 July 2014

Introduction

Game Title: Hangman

Description: A simple word guessing game.

Rules

- **Objectives**
  - The user attempts to figure out a hidden word of a given length.
- **Constraints**
  - The user may only guess one letter at a time.
  - There are only a limited number of wrong guesses until a loss is declared. (A wrong guess is a guess of a letter that is not in the hidden word)
- **Game Ending Conditions**
  - The user wins if he/she successfully guesses all the letters in the hidden word.
  - The user loses if he/she makes too many wrong guesses.

Summary

Outline of implementation goals

I wanted to create a fully functional hangman game with some extra features. The extra features include: complete input validation and reading, writing, and outputting the user's win/loss and correct/incorrect guesses statistics.

For the list of words to be used in my hangman game, I wanted to be able to load a large selection of words from a file and choose a new word randomly for each play.

Basic project size information

Total lines: 310

Lines of code: 192

Comment lines: 85

Blank lines: 33

Major variables: 29

Major Variables

Type	Identifier	Description	Line # for main usage (* initialized at this line)
int	a_len	The length of the answer word	135*, 137, 154, 198
	gs_num	The number of guesses for the current game	140*, 158, 177, 241
	gs_misd	The number of missed guesses for the current game	140*, 152, 230
	w_ld	The number of words loaded from the words.dat file	65*, 95, 134
	gms_tot	The total number of games played	(74 or 81)*, 258, 261, 275, 303
	gms_won	The number of games won	(74 or 81)*, 258, 261, 276, 303
	gms_lst	The number of games lost	258*, 277
	gs_tot	The total number of guesses	(74 or 81)*, 259, 267, 280, 303
	gs_cor	The number of correct guesses	(74 or 81)*, 259, 267, 281, 303
	gs_miss	The number of missed guesses	259*, 282
short	DEC_PCT_CNV	Decimal/Percent conversion (const unsigned)	24*, 261, 267
	MGS_MAX	Maximum number of missed guesses (const unsigned)	33*, 152, 230
	GS_MAX	Maximum possible number of guesses (const unsigned)	34*, 40, 142, 144
	W_MAX	Maximum number of words to load (const unsigned)	35*, 37, 95
float	pgms_won	Percentage of games won	(261 or 265)*, 262, 278

	pgms_lst	Percentage of games lost	(262 or 265)*, 279
	pgs_cor	Percentage of guesses correct	(267 or 271)*, 268, 283
	pgs_misd	Percentage of guesses missed	(268 or 271)*, 284
char	m_chse	The users menu choice	123*, 131, 297
	guess	The users guess for a letter	167*, 178, 190, 193, 202
char[]	guesses	Keeps track of what the user has guessed so far	143*, 159, 178, 193
bool	running	Used to determine if the current hangman game is over	146*, 149, 225, 235
	is_match	Used to determine if the user has guessed a letter that is in the answer	197*, 204, 210
	was_gsd	Used to determine if the user has already guessed the same letter	175*, 179, 185, 190
string	answer	The answer to the current hangman game	134*, 202, 218
	a_so_far	The portion of the answer the user has guessed correctly	134*, 155, 218
string[]	words	All the loaded words from the file	95*, 134
fstream	stats	Stream for the statistics file	(70 or 75)*, 81, 303
ifstream	w_file	Stream for the words file	88*, 95

Concept checklist

Data Types

- **Primitive data types:** int, short, float, char, bool, char[]
- **Other data types:** string, fstream, ifstream
- **Modifiers used:** const, unsigned

System Level Libraries

- **iostream:** used for I/O
- **iomanip:** used for setprecision() function
- **fstream:** used for fstream and ifstream data types
- **string:** used for string data type
- **cstdlib:** used for random numbers
- **ctime:** used to seed the random number generator
- **cstring:** used for string comparison
- **cctype:** used for checking if characters are alphabetic
- **limits:** used for bad input handling

Operators

Operators used (character, line #)	Operators unused (character)
Subtraction (-, 251)	Addition (+)
Division (/, 242)	Logical Xor (^)
Multiplication (*, 242)	Less than or Equal (<=)
Modulo (%, 131)	Inequality (!=)
Assignment (=, 180)	Decrement and Assignment (--)
Logical Not (!, 89)	Subtraction and Assignment (-=)
Logical And (&&, 92)	Multiplication and Assignment (*=)
Logical Or (||, 190)	Division and Assignment (/=)
Greater than or Equal (>=, 206)	Modulo and Assignment (%=)
Equality (==, 179)	
Increment and Assignment (++, 187)	
Addition and Assignment (+=, 223)	

Conditionals and Loop Constructs

Conditionals and Loops used (line #)	Conditionals unused
if (68)	else if
else (76)	ternary operator (?:)
switch (128)	
while (92)	
do-while (110)	
for (134)	

Reading/Writing from Files

In order to save statistics I had to both read and write the statistics to the file. The loading takes place at the beginning, on lines 67-82. The writing takes place once the user decides to quit, on lines 272-278.

I also wanted to read words from a file, which takes place on lines 85-96.

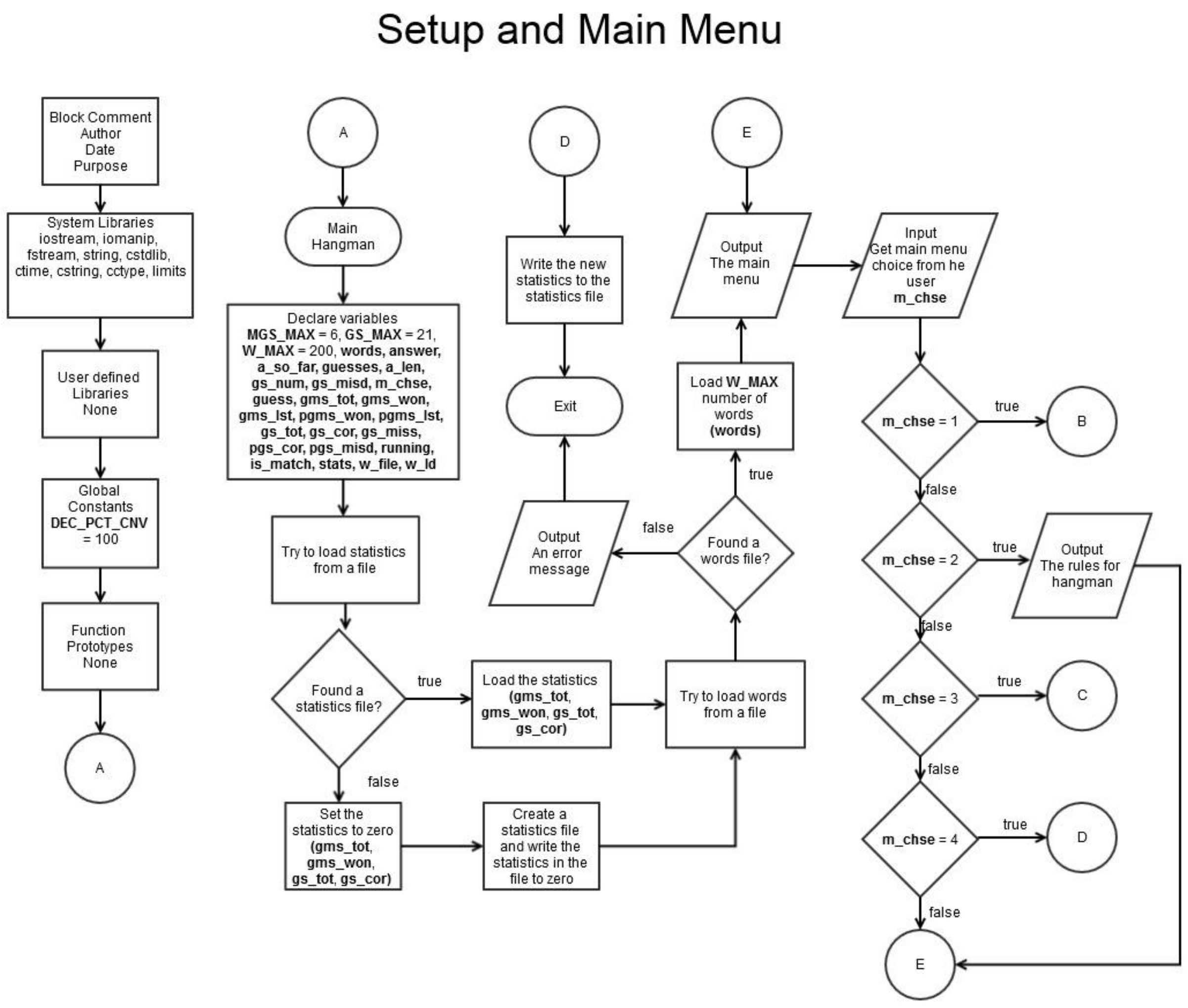
Other Comments

In order to accomplish random word generation for the main game I had to research a little bit into arrays and how to access them by index. Additionally, there didn't seem to be a good way to load an unknown quantity of elements into an array (maybe use vectors?) so I ended up just loading a set number of words into an array (defaulting at 1000).

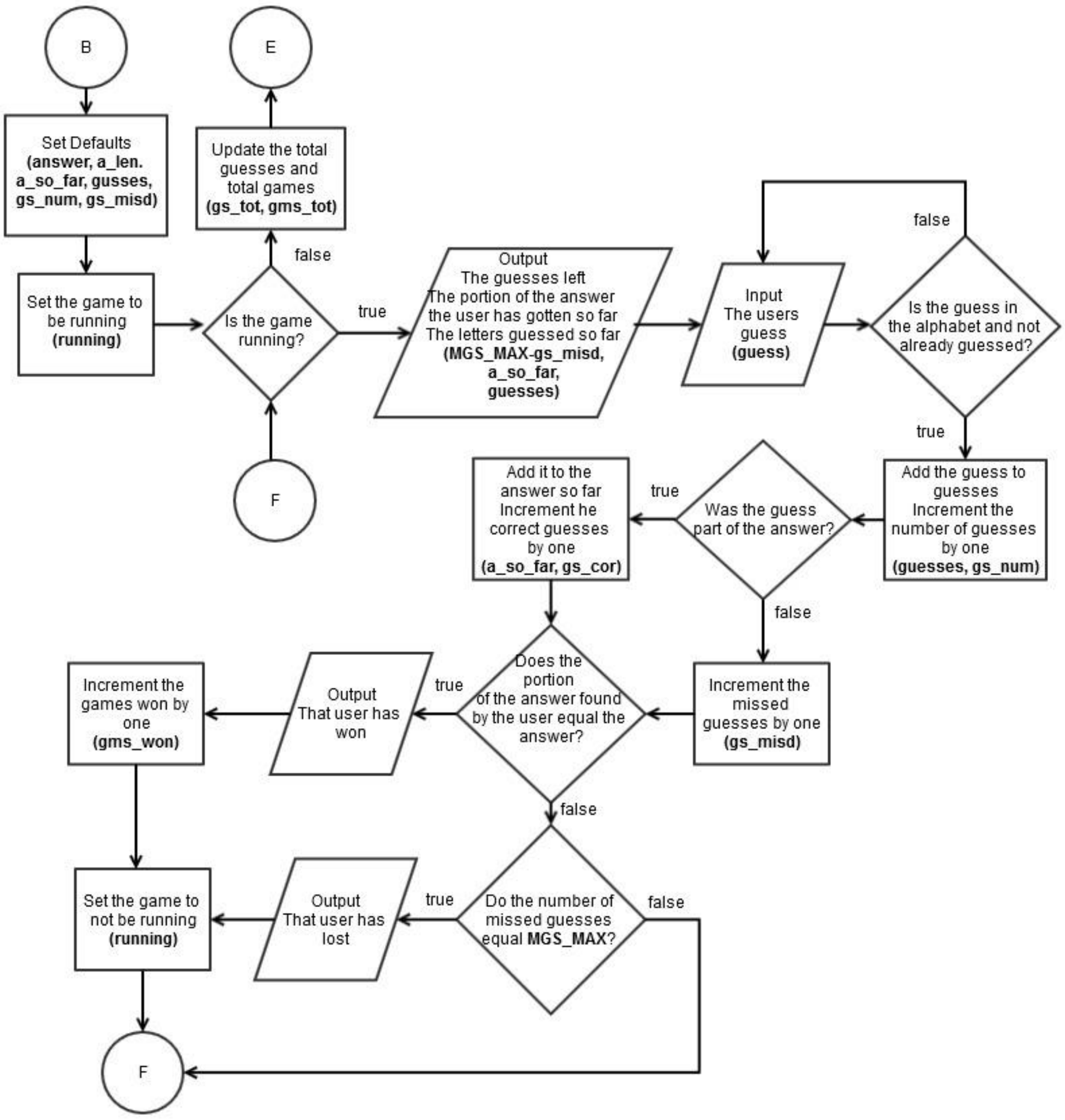
References

http://www.cplusplus.com/doc/tutorial/arrays/

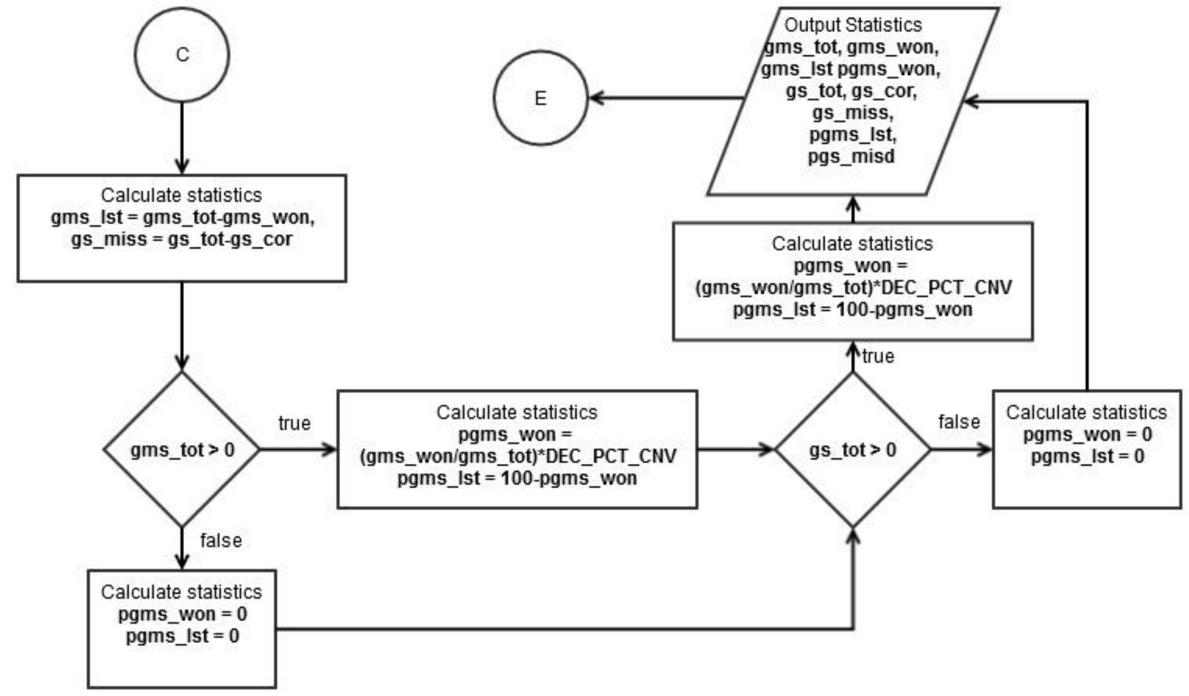
Flowchart


Setup and Main Menu
Block Comment Author Date Purpose
System Libraries iostream, iomanip, fstream, string, cstdlib, ctime, cstring, cctype, limits
User defined Libraries None
Global Constants DEC_PCT_CNV = 100
Function Prototypes None
A
Main Hangman
Declare variables MGS_MAX = 6, GS_MAX = 21, W_MAX = 200, words, answer, a_so_far, guesses, a_len, gs_num, gs_misd, m_chse, guess, gms_tot, gms_won, gms_lst, pgms_won, pgms_lst, gs_tot, gs_cor, gs_miss, pgs_cor, pgs_misd, running, is_match, stats, w_file, w_ld
Try to load statistics from a file
Found a statistics file?
true
false
Load the statistics (gms_tot, gms_won, gs_tot, gs_cor)
Set the statistics to zero (gms_tot, gms_won, gs_tot, gs_cor)
Create a statistics file and write the statistics in the file to zero
Try to load words from a file
Found a words file?
false
Output An error message
true
Load W_MAX number of words (words)
D
Write the new statistics to the statistics file
Exit
E
Output The main menu
Input Get main menu choice from he user m_chse
m_chse = 1
true
B
false
m_chse = 2
true
Output The rules for hangman
false
m_chse = 3
true
C
false
m_chse = 4
true
D
false
E

Main Game


B
Set Defaults (answer, a_len. a_so_far, gusses, gs_num, gs_misd)
Set the game to be running (running)
Is the game running?
false
Update the total guesses and total games (gs_tot, gms_tot)
E
F
true
Output The guesses left The portion of the answer the user has gotten so far The letters guessed so far (MGS_MAX-gs_misd, a_so_far, guesses)
Input The users guess (guess)
Is the guess in the alphabet and not already guessed?
false
true
Add the guess to guesses Increment the number of guesses by one (guesses, gs_num)
Was the guess part of the answer?
true
Add it to the answer so far Increment he correct guesses by one (a_so_far, gs_cor)
false
Increment the missed guesses by one (gs_misd)
Does the portion of the answer found by the user equal the answer?
true
Output That user has won
Increment the games won by one (gms_won)
false
Do the number of missed guesses equal MGS_MAX?
true
Output That user has lost
false
Set the game to not be running (running)
F

Statistics



C
Calculate statistics gms_lst = gms_tot-gms_won, gs_miss = gs_tot-gs_cor
gms_tot > 0
true
false
Calculate statistics pgms_won = (gms_won/gms_tot)*DEC_PCT_CNV pgms_lst = 100-pgms_won
Calculate statistics pgms_won = 0 pgms_lst = 0
gs_tot > 0
true
false
Calculate statistics pgms_won = (gms_won/gms_tot)*DEC_PCT_CNV pgms_lst = 100-pgms_won
Calculate statistics pgms_won = 0 pgms_lst = 0
Output Statistics gms_tot, gms_won, gms_lst pgms_won, gs_tot, gs_cor, gs_miss, pgms_lst, pgs_misd
E

Pseudocode

```
Initialize
Load statistics and words
While the user doesn’t choose to quit
    Output menu
    Ask user to choose a menu option
    If the user chooses 1
        Choose a random word for the game
        Set the game status to running
        While the game is running
            Output the guesses the user has left
            Output the portion of the answer the user has gotten so far
            Output what the user has guessed
            While the input isn’t valid or the input has already been guessed
                Ask user to guess a letter
            If the guess is in the answer
                Add the guess to guesses
                Update the portion of the answer the user has gotten so far
                Add one to the correct guesses statistic
            Else
                Add one to the missed guesses
```

```
                        If the portion of the answer so far is equal to the actual answer
                            Output that the user wins
                            Add one to the won games statistic
                            Set the game status to not running
                        If the missed guesses is equal to the maximum missed guesses
                            Output that the user loses
                            Set the game status to not running
                    Update the total games and total guesses statistics
                If the user chooses 2
                    Output hangman rules
                If the user chooses 3
                    Calculate current statistics
                    Output statistics
                If the user chooses 4
                    Quit the game
            Write the current statistics to a file
            Exit the program
```

Actual Code

```
//System Level Libraries
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include <cstdlib>
#include <ctime>
#include <cstring>
#include <cctype>
#include <limits>
using namespace std;

//User Defined Libraries

//Global Constants
const unsigned short DEC_PCT_CNV = 100;//Conversion from decimal to percent

//Function Prototypes

//Begin Execution

int main(int argc, char** argv) {
    //Declare variables
    //Constants
    const unsigned short MGS_MAX = 6;//Maximum number of missed guesses
    const unsigned short GS_MAX = 21;//Maximum possible number of guessed letters
    const unsigned short W_MAX = 1000;//Maximum number of words to load
    //Main game variables
    string words[W_MAX];//Array of loaded words
    string answer;//The answer to the current hangman game
    string a_so_far;//Keep track of how the user is doing on the answer to the game
    char guesses[GS_MAX+1];//Keep track of the current guesses plus the delimiter
    int a_len;//Length of the answer word
    int gs_num;//The number of guesses so far
    int gs_misd;//The number of missed guesses (current game only)
    //Inputs
    char m_chse;//The menu choice
    char guess;//The current guess
    //Used for statistics
    int gms_tot;//Total number of games played
    int gms_won;//Number of games won
    int gms_lst;//Number of games lost
    float pgms_won;//Percentage of games won
    float pgms_lst;//Percentage of games lost
    int gs_tot;//Total number of guesses
    int gs_cor;//Number of correct guesses
    int gs_miss;//Number of missed guesses (all time)
```

```
float pgs_cor;//Percentage of guesses correct
float pgs_misd;//Percentage of guesses missed
//Flags
bool running;//A flag used to determine if the current instance of hangman is over
bool is_match;//A flag used to determine if the guessed letter was in the answer
bool was_gsd;//A flag used to determine if the user already guessed this letter
//File streams and variables
fstream stats;//Stream for the statistics file
ifstream w_file;//Stream for the words file
int w_ld = 0;//The number of words loaded (cannot exceed W_MAX)

//Begin setup for the game
//Load the necessary data for statistics and words
//Try to open the statistics file
stats.open("statistics.dat",ios::in);
if(!stats){
    //If there is no statistics file, create a statistics file and write
    //default values for the statistics to it
    gms_tot = gms_won = gs_tot = gs_cor = 0;
    stats.open("statistics.dat",ios::out);
    stats<<flush;
    stats<<gms_tot<<endl<<gms_won<<endl<<gs_tot<<endl<<gs_cor<<endl;
}
else{
    //If there is a statistics file, load the current statistics
    stats>>gms_tot>>gms_won>>gs_tot>>gs_cor;
}

//Close the statistics file
stats.close();

//Try to open the words file
w_file.open("words.dat");
if (!w_file){//If unable to load words, exit with a fatal error
    cout<<"Fatal Error: unable to open words.dat\n";
    exit(EXIT_FAILURE);
}

//Load up to W_MAX size of words (default is 1000)
while(w_ld<W_MAX&&w_file>>words[w_ld])
    w_ld++;

//Close the words file
w_file.close();

//Seed the random number generator
srand(static_cast<int>(time(0)));

//Format statistics output to two decimal places
cout<<fixed<<showpoint<<setprecision(2);

//End setup for the game

//Output a welcome message
cout<<"Welcome to Hangman!\n";

//Enter main menu loop
do{
    //Show the game menu
    cout<<"--------------------Main Menu--------------------\n";
    cout<<"1.       Start a new game of hangman\n";
    cout<<"2.        Show the rules for hangman\n";
    cout<<"3.       Show your hangman statistics\n";
    cout<<"4.                   Quit\n";

    //Get users choice
    cout<<"Choose an option from above (1-4): ";
    cin>>m_chse;
    cout<<endl;

    //Ignore all input except the first character
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    //Determine what to do depending on the user's choice
    switch(m_chse){
        case('1'):{
            //Pick a random word from the words loaded and get its length
            answer = a_so_far = words[rand()%w_ld];
            a_len = answer.length();
            //Set the answer found by the user so far to empty
```

```
for(int i = 0;i<a_len;i++)
    a_so_far[i] = '_';
//Set the number of guesses and missed guesses to zero
gs_num = gs_misd = 0;
//Set the guesses so far to empty
for(int i = 0;i<GS_MAX;i++)
    guesses[i] = ' ';
guesses[GS_MAX] = '\0';
//Run the main game
running = true;

//Enter main game loop
while(running){
    //Output the number of guesses left, how much of the answer
    //the user has gotten, and what the user has guessed so far
    cout<<"Number of guesses left: "<<(MGS_MAX-gs_misd)<<endl;
    cout<<"Answer so far: ";
    for(int i = 0;i<a_len;i++)
        cout<<a_so_far[i]<<" ";
    cout<<endl;
    cout<<"Letters guessed so far: ";
    for(int i = 0;i<gs_num;i++)
        cout<<guesses[i]<<" ";
    cout<<endl;

    //Validate the guess, i.e., make sure it is in the alphabet
    //and that it hasn't been guessed before
    do{
        //Get a guess from the user
        cout<<"Guess a letter: ";
        cin>>guess;

        //Ignore all input except the first character
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        //Validation
        guess = tolower(guess);//Convert guess to lower case
        was_gsd = false;//Assume the guess was not already guessed
        //Check to see if the guess was already guessed
        for(int i = 0;i<gs_num;i++){
               if(guesses[i] == guess){
                   was_gsd = true;
                   break;//Break out if it was guessed already
           }
        }
        //If it was guessed already, print that the user guessed
        //that letter already
        if(was_gsd)
            cout<<"Oops! You already guessed that letter\n";
        //If the guess isn't in the alphabet, print an error message
        if(!isalpha(guess))
            cout<<"Invalid input! Please enter a letter from a to z\n";
    }while(was_gsd || !isalpha(guess));

    //Once we have valid input, continue the game as normal
    guesses[gs_num] = guess;//Add the guess to guesses
    gs_num++;//Increment guesses for this game by one

    //Check to see if the guess is part of the answer
    is_match = false;//Assume guess is not part of the answer
    for(int i = 0;i<a_len;i++){
        //If the guess is part of the answer, then update
        //the part of the answer the user has gotten so far
        //and indicate there was a match
        if(answer[i] == guess){
            a_so_far[i] = guess;
            is_match = true;
        }
    }

    //If the guess was in the answer, increment the
    //correct guesses
    if(is_match)
        gs_cor++;
    //If it wasn't, then increment the missed guesses
    else
        gs_misd++;

    //If the answer the user has found so far is the same as
    //the answer, then the user wins
```

```
                if(!(answer.compare(a_so_far))){
                    //Output a winning message
                    cout<<"Congratulations, you've won!\n";
                    cout<<"The word was: "<<answer<<endl;
                    //Increment the games won
                    gms_won++;
                    //Set the status of the game to not running
                    running = false;
                }

                //If the guesses missed is equal to the max number of
                //missed guesses, the user loses
                if(gs_misd >= MGS_MAX){
                    //Output a losing message
                    cout<<"You have been hung!\n";
                    cout<<"The word was: "<<answer<<endl;
                    //Set the status of the game to not running
                    running = false;
                }
                cout<<endl;
            }
            //Game is finished
            //Update the guess total by adding the total guesses for this game
            gs_tot += gs_num;
            //Increment the total games counter by one
            gms_tot++;
            break;
        }
        case('2'):{
            //Output the rules of hangman
            cout<<"----------------------Rules----------------------\n";
            cout<<"Objective:   Determine a word given its length.\n";
            cout<<"Constraints: You may only guess one letter at\n";
            cout<<"             a time, and you are only allowed\n";
            cout<<"             five wrong guesses.\n\n";
            break;
        }
        case('3'):{
            //Calculate games lost, guesses missed, percentage of games won
            //and percentage of guesses correct
            gms_lst = gms_tot-gms_won;
            gs_miss = gs_tot-gs_cor;
            if(gms_tot > 0){
                pgms_won = (static_cast<float>(gms_won)/gms_tot)*DEC_PCT_CNV;
                pgms_lst = 100-pgms_won;
            }
            else
                pgms_won = pgms_lst = 0;
            if(gs_tot > 0){
                pgs_cor = (static_cast<float>(gs_cor)/gs_tot)*DEC_PCT_CNV;
                pgs_misd = 100-pgs_cor;
            }
            else
                pgs_cor = pgs_misd = 0;

            //Output statistics
            cout<<"--------------------Statistics-------------------\n";
            cout<<"Games played:                  "<<setw(6)<<gms_tot<<endl;
            cout<<"Games won:                     "<<setw(6)<<gms_won<<endl;
            cout<<"Games lost:                    "<<setw(6)<<gms_lst<<endl;
            cout<<"Percentage of games won:       "<<setw(6)<<pgms_won<<"%\n";
            cout<<"Percentage of games lost:      "<<setw(6)<<pgms_lst<<"%\n";
            cout<<"Total guesses:                 "<<setw(6)<<gs_tot<<endl;
            cout<<"Correct guesses:               "<<setw(6)<<gs_cor<<endl;
            cout<<"Missed guesses:                "<<setw(6)<<gs_miss<<endl;
            cout<<"Percentage of correct guesses: "<<setw(6)<<pgs_cor<<"%\n";
            cout<<"Percentage of missed guesses:  "<<setw(6)<<pgs_misd<<"%\n\n";
            break;
        }
        case('4'):{//If 4 quit the program
            //Output a quitting message
            cout<<"Goodbye!\n";
            break;
        }
        default:{//Otherwise the input is invalid
            //Output that the input is not valid and ask again
            cout<<"Invalid input, please enter a number from 1 to 4.\n\n";
        }
    }
}while(m_chse != '4');
```

```
    //Open the stats file to write the current statistics
    stats.open("statistics.dat",ios::out);

    //Write statistics to statistics.dat
    stats<<gms_tot<<endl<<gms_won<<endl<<gs_tot<<endl<<gs_cor<<endl;

    //Close the statistics file
    stats.close();

    //Exit program
    return 0;
}
```