Higher Institute of Engineering & Technology, El-Boheira

Computer Engineering Department

1st Year

2nd Semester

Programming Applications

CE222

Hangman game

Presented by:

Name	ID	Name in arabic
Ziad Ahmed	19035	زياد احمد محمد جابر أحمد
Mohamed Elzarka	19100	محمد يسري محمد الزرقا
Alaa El-maghlany	19156	ألاء سعيد محمد قطب المغلاني
Rahma Oshba	19184	رحمه خالد أحمد عشبة
Karim Akram	19206	كريم أكرم شوقي محمد قرعلي

Presented to: Dr \ Mahmoud Ismail

Hangman game

Problem description:

Hangman is a paper and pencil guessing game for two or more players. One player thinks of a word and the other tries to guess it by suggesting the letters. The word to guess is represented by a row of dashes, giving the number of letters. If the guessing player suggests a letter, which occurs in the word, the program writes it in all its correct positions. If the suggested letter does not occur in the word, the other player draws one element of the hangman diagram as a tally mark. The game is over when the guessing player completes the word, or guesses the whole word correctly.

In this program, the selection of a word, among a set of several words, depends on pseudo randomness.

Definition of pseudorandom:

Being or involving entities (such as numbers) that are selected by a definite computational process but that satisfy one or more standard tests for statistical randomness.

Hangman: is one of the first word games many children learn: the rules are few and simple, and the game—which, in its original pen-and-paper version, is usually a two-player affair—grows along with your (and your opponents') vocabulary. Another attraction may be that it is usually not too difficult to win, and when you win, your opponent does not really lose. They just have not chosen a challenging enough word for you to solve.

A quick refresher course, if you have forgotten: one player picks a word (either at random, or a word in a category), and puts short dashes on a piece of paper where the letters should be. Five blanks, it is a five-letter word. Eight blanks, it is an eight-letter word. The other player then guesses individual letters to spell the word, each correct guess goes in the appropriate space; each incorrect guess earns you a body part of the person—usually a stick figure. In a typical game, the doomed figure might have six parts—a head, spine, two legs, and two arms. If the stick figure is drawn before you figure out the word, you are "hung."
Another variation of the game is to allow the players to "build" part or all of the gallows, which provides more chances. The three hangman games reviewed here provide six chances (just the doomed), seven (the doomed and a noose), and 11 (a four-part platform, the noose, and the doomed) guesses.

How to play Hangman:

Guess letters to fill in the blanks before your little man gets hung out to dry.

Requires:

1) 2 to 4 players.
2) Pen and paper.

Game play:

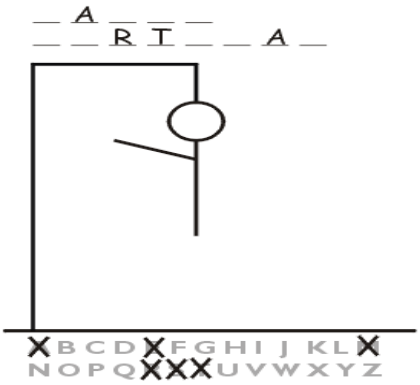
One player thinks of a word or phrase; the others try to guess what it is one letter at a time. The player draws a number of dashes equivalent to the number of letters in the word. If a guessing player suggests a letter that occurs in the word, the other player fills in the blanks with that letter in the right places. If the word does not contain the suggested letter, the other player draws one element of a hangman's gallows. As the game progresses, a segment of the gallows and of a victim is added for every suggested letter not in the word. The number of incorrect guesses before the game ends is up to the players, but completing a character in a noose provides a minimum of six wrong answers until the game ends. The first player to guess the correct answer thinks of the word for the next game.

Objective:

Guess the word/phrase before your man gets hung!

HANGMAN EXAMPLE:

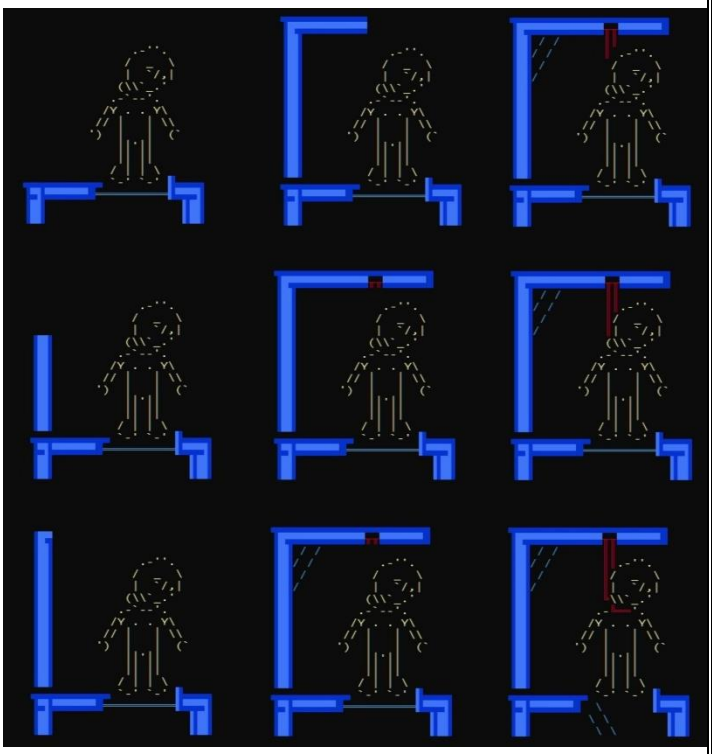
Here is an example of a hangman game in progress. The phrase is Happy Birthday, so 5 blank spaces were marked out for Happy, and below that 8 blank space were marked out for birthday. Three correct guesses were made: A, R and T. Three incorrect guesses were made: S, E, and M. Three body parts were added for the 3 incorrect guesses.



Important note

When we designed the game, we have implemented sound effects and some graphics (ASCII art), but they were extracted and removed from this particular version in order to focus on the main logic of the game and the flow of the code. In other words, it is a cleaner code, which is easier to understand and follow its flow.

However, You can download the full game with graphics and sound effects here: bit.do/hangman47

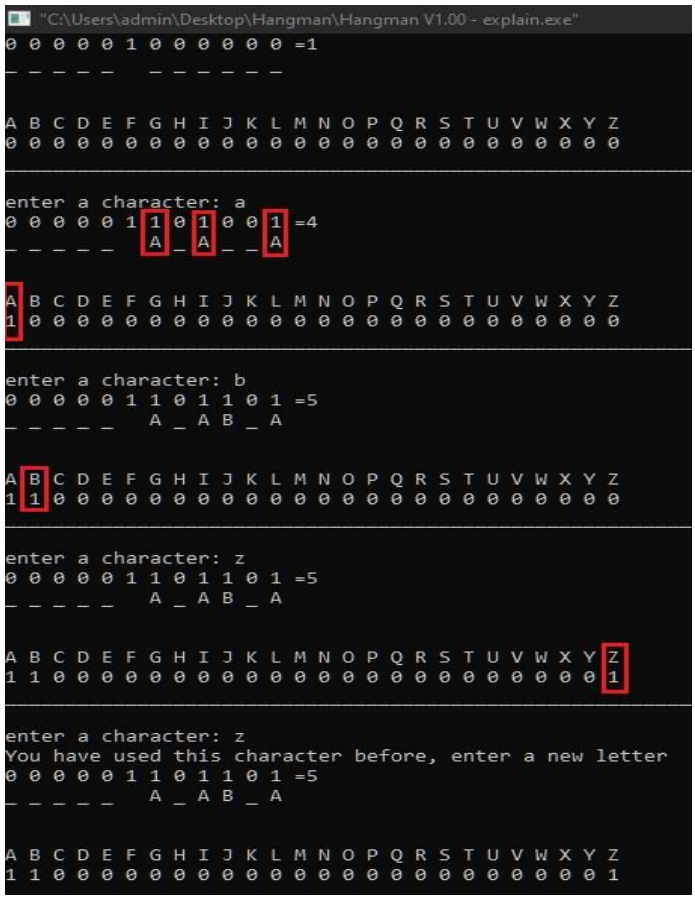


The core of the game

It depends on 2 elements:

A) A Boolean mirror of the English alphabet: it helps to prevent the player from using the same letter twice.

B) A Boolean mirror of the word: it helps to mark guessed letters as true so that you can show them. However, if a letter is marked false, a dash will be printed instead of the letter. In addition, the sum of its "trues" helps the program to realize if the player wins besides realizing if it was a right guess or a wrong guess.



- If all the letters are marked true (guessed), that means the player wins.
- If the sum was 4 then became 5 after the player input, that means a right guess.
- On the contrary, if the sum was 5 and still 5 after the player input, that means a wrong guess and the player loses a life.

Proposed model:

We will get this job done by implementing a c++ code, which will mainly depend on the following functions:

- **void import_from_file(string array[],char file[])**

A function to import text from a text file and assigns it to an array of strings. Each string contains a line of text.

- **unsigned int get_random_num(int minimum, int maximum)**

A function to generate a random number within a specific range [minimum, maximum] determined by the user.

- **int getindex(char t)**

A function to calculate the order of an English letter [A returns 0, B returns 1…, Z returns 25].

- **int num_of_guessed_chars(bool a[],int a_size)**

A function to count how many letters are marked as guessed in the Boolean mirror of the word.

- **char hint_func(string selected_word, bool did_you_guess_it[])**

A function that searches randomly in the Boolean mirror of the word until it find a character, which was not marked as guessed yet, and returns it.

- **int choose_difficulty()**

A function that enables the player to choose which level he wants.

- **char input_alphabitical_char(bool is_this_letter_used[])**

A function that enables the player to input an English letter, 1 (for a hint) or 0 (to surrender), then returns it. It also makes sure that this letter wasn't used before.

- **void print_word(string selected_word,bool did_you_guess_it[])**

A function to print the word (letters and dashes).

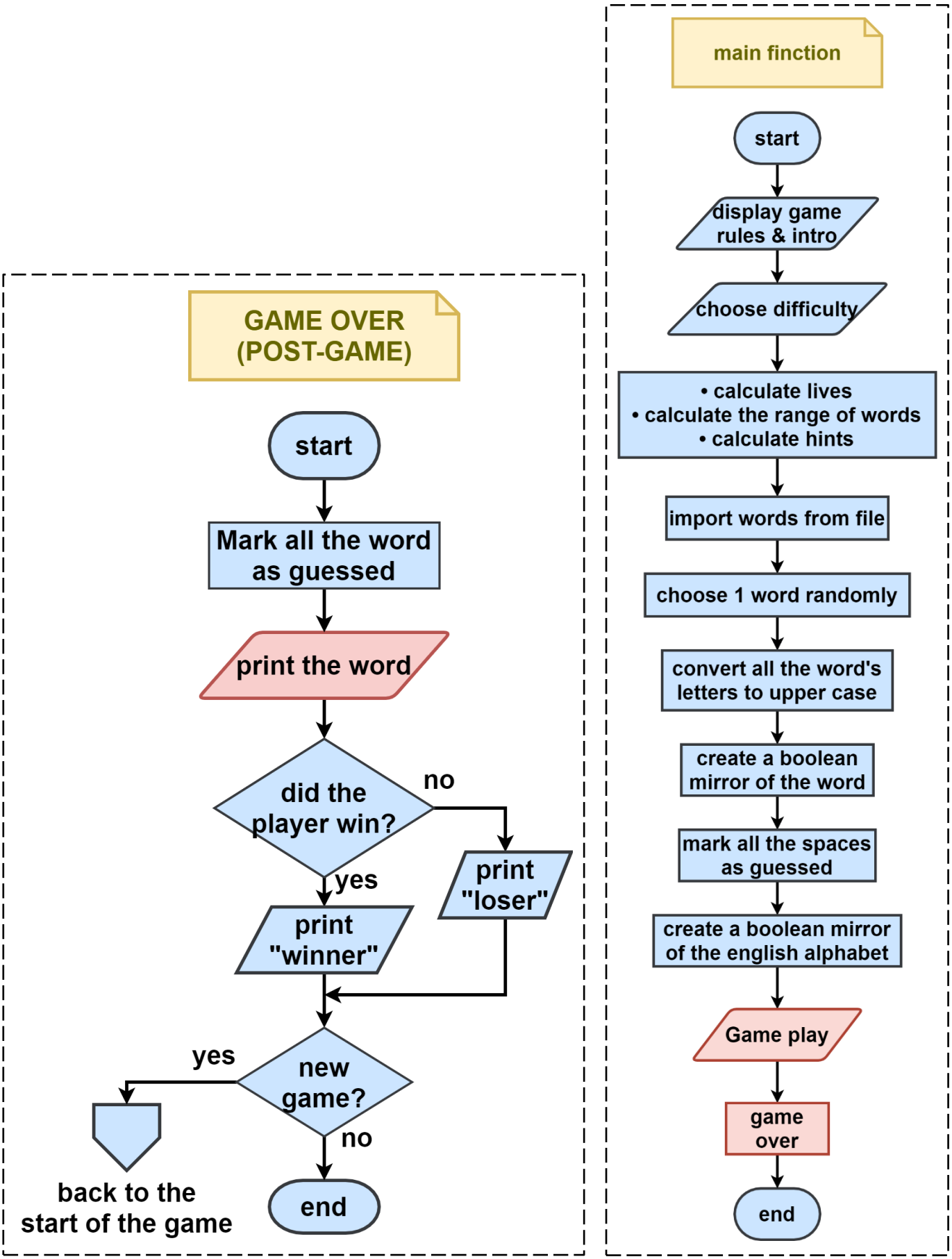
- **void intro()**

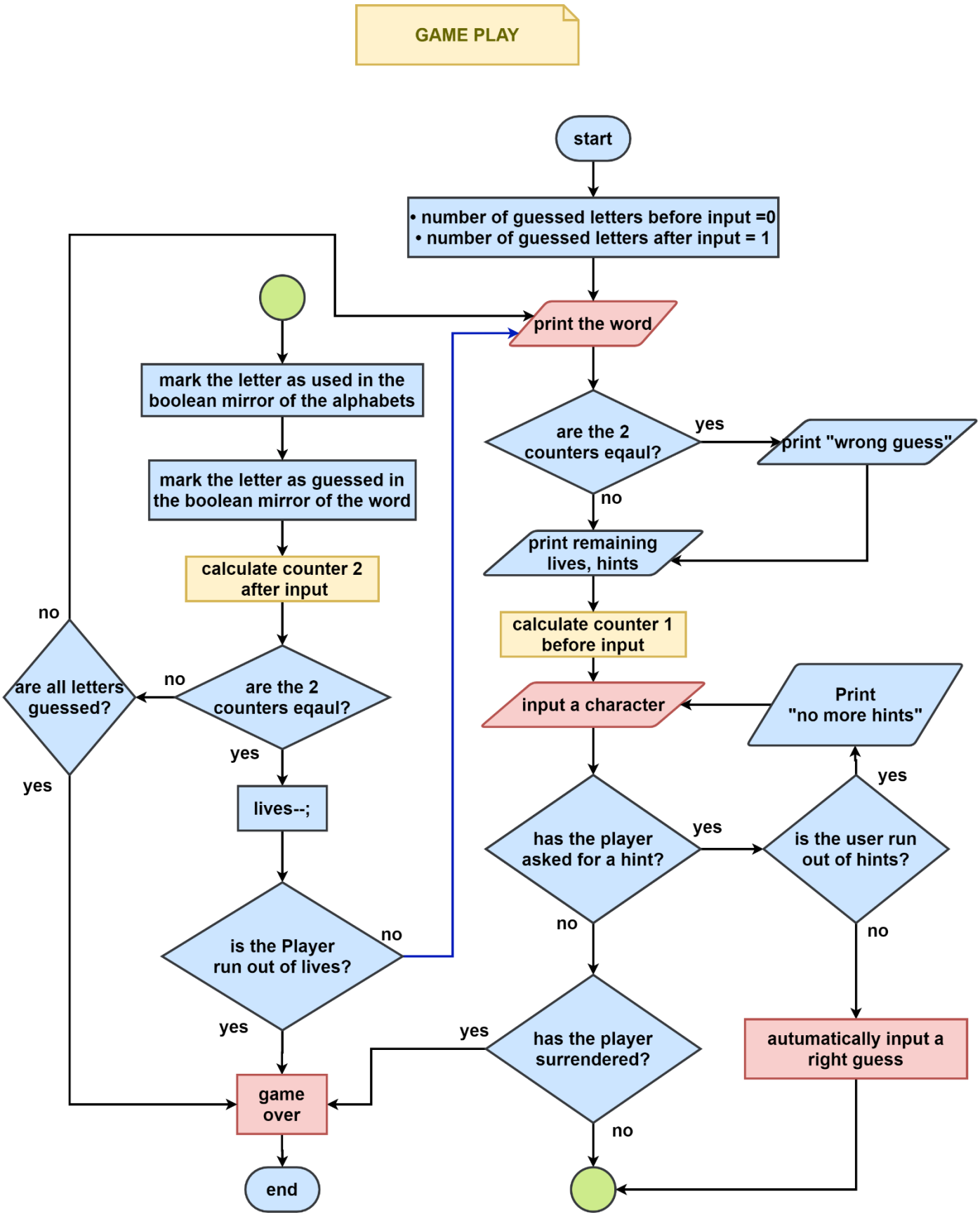
A function that imports the rules from a text file, then displays them.

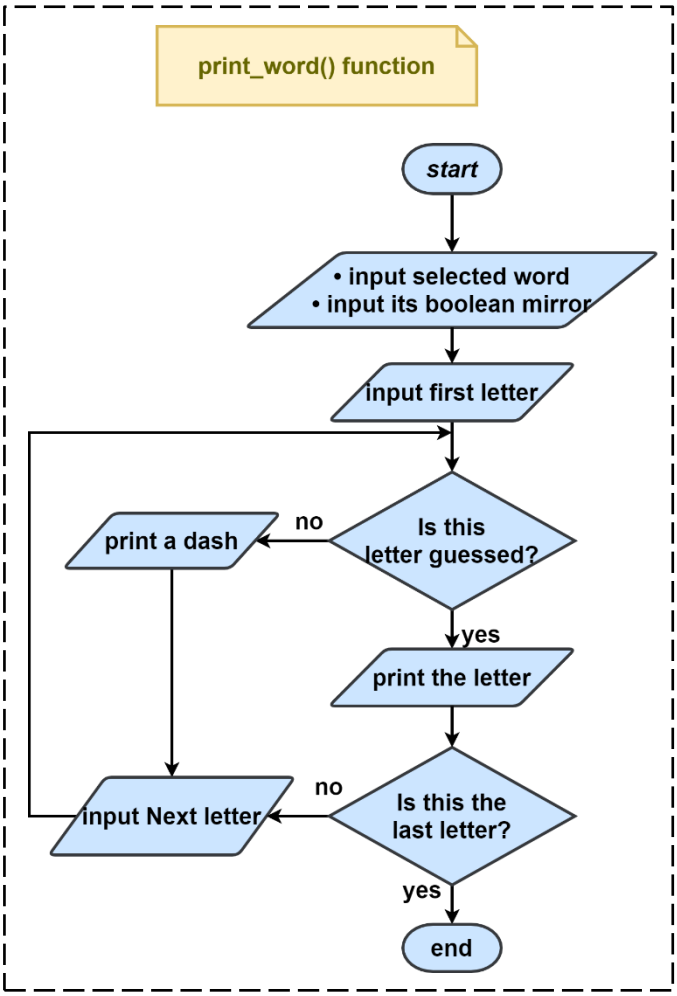
- **int main()**

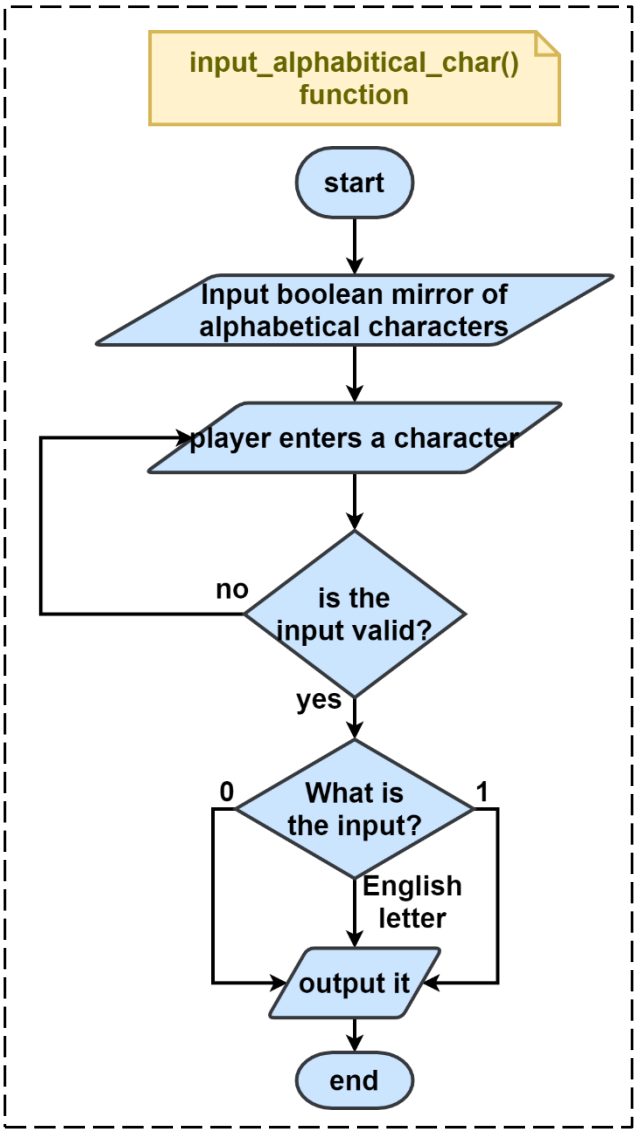
The start point of the program, which will call the other functions and proceed the game. It also declares vital variables such as 'did_you_guess_it[]' (a boolean mirror of the word) and 'is_this_letter_used[]' (a boolean mirror of the English alphabet). They are the main core of this code.

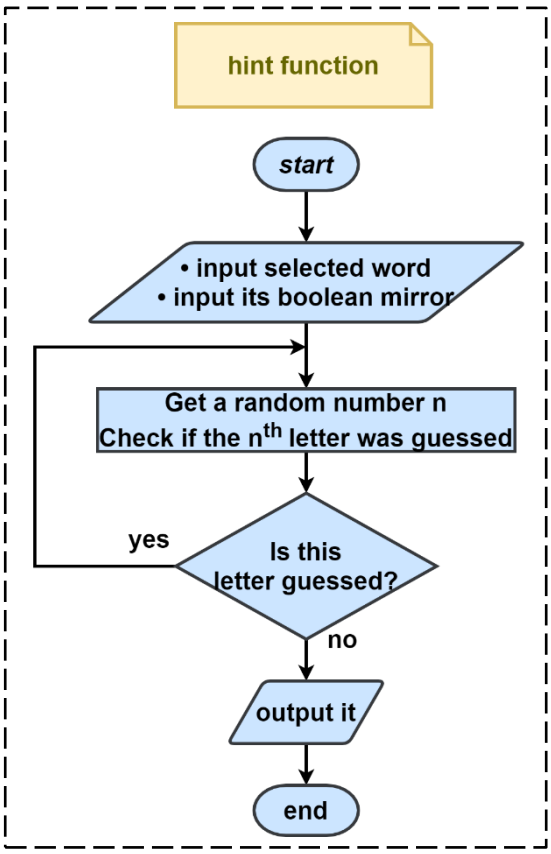
** FlowCharts**

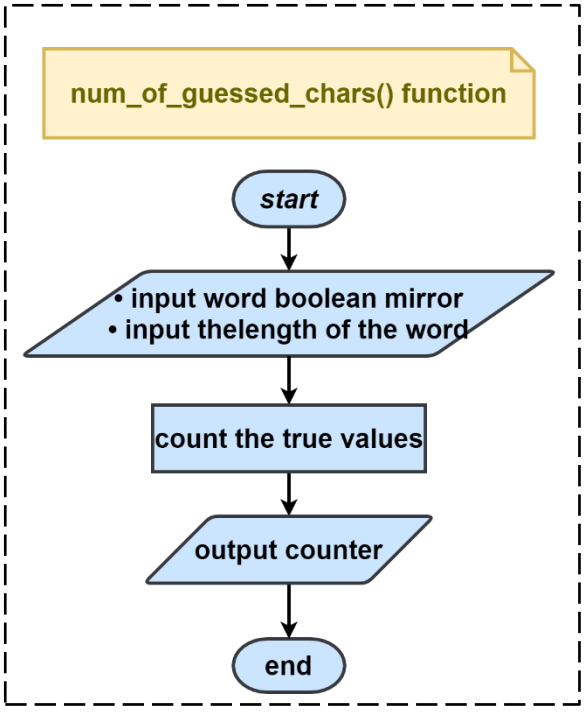

main finction
start
display game rules & intro
choose difficulty
• calculate lives
• calculate the range of words
• calculate hints
import words from file
choose 1 word randomly
convert all the word's letters to upper case
create a boolean mirror of the word
mark all the spaces as guessed
create a boolean mirror of the english alphabet
Game play
game over
end
GAME OVER (POST-GAME)
start
Mark all the word as guessed
print the word
did the player win?
no
yes
print "loser"
print "winner"
new game?
yes
no
back to the start of the game
end


GAME PLAY
start
• number of guessed letters before input =0
• number of guessed letters after input = 1
print the word
are the 2 counters eqaul?
yes
print "wrong guess"
no
print remaining lives, hints
calculate counter 1 before input
input a character
Print "no more hints"
has the player asked for a hint?
yes
is the user run out of hints?
yes
no
no
autumatically input a right guess
has the player surrendered?
yes
no
mark the letter as used in the boolean mirror of the alphabets
mark the letter as guessed in the boolean mirror of the word
calculate counter 2 after input
are the 2 counters eqaul?
no
yes
lives--;
is the Player run out of lives?
no
yes
are all letters guessed?
no
yes
game over
end


print_word() function
start
• input selected word
• input its boolean mirror
input first letter
Is this letter guessed?
no
print a dash
yes
print the letter
Is this the last letter?
no
input Next letter
yes
end


input_alphabitical_char() function
start
Input boolean mirror of alphabetical characters
player enters a character
is the input valid?
no
yes
What is the input?
0
1
English letter
output it
end


hint function
start
• input selected word
• input its boolean mirror
Get a random number n
Check if the nth letter was guessed
Is this letter guessed?
yes
no
output it
end


num_of_guessed_chars() function
start
• input word boolean mirror
• input thelength of the word
count the true values
output counter
end

Individuals Contributions:

Student	Contribution
Ziad Ahmed **19035**	- Flowcharts - Implemented functions: Import_from_file( ) intro()
Mohamed Elzarka **19100**	- Applied sound effects - Implemented functions: main() input_alphabitical_char()
Alaa El-maghlany **19156**	- Worked on documentation - Implemented functions: num_of_guessed_chars() print_word()
Rahma Oshba **19184**	- Worked on documentation - Implemented functions: get_random_num() hint_func()
Karim Akram **19206**	- designed the hangman - Implemented functions: getindex() choose_difficulty()

Implementation:

```
#include <iostream>
#include <string>
#include <stdlib.h> //rand() and srand() functions to get a random number
#include <ctime> //important to get a random number, because we use time as a seed
#include <fstream>


using namespace std;

//_________________FUNCTIONS PROTOTYPE_____________________________

void import_from_file(string array[],char file[]);
unsigned int get_random_num(int minimum, int maximum);
int getindex(char t);
int num_of_guessed_chars(bool a[],int a_size);
char hint_func(string selected_word, bool did_you_guess_it[]);

int choose_difficulty();
char input_alphabitical_char(bool is_this_letter_used[]);

void print_word(string selected_word,bool did_you_guess_it[]);
void intro();
int main()
//=====================================================================
{
    intro();
    bool newgame=1;
while(newgame)
{
//_____________DECLARATIONS & GAME INITIALIZATION_____________

    int difficulty=choose_difficulty();
    int lives=10-2*difficulty;  //a mathematical equation that calculates lives according to difficulty
    //according to difficulty, easy= 8, normal= 6, hard= 4 lives
    int set_start=20*difficulty-20; //a mathematical equation that determines the start of the set of the words you have chosen
    //according to difficulty, the words are divided into 1-20 easy, 20-40 normal, 40-60 hard
    int HINTS=4-difficulty; //a mathematical equation that determines the numbers of hints depending on the difficulty the player has used;

    string words[200];
    import_from_file(words,"Animals & countries.txt");
    string selected_word=words[get_random_num(set_start,set_start+19)];
```

```
42      for(int i=0;i<selected_word.size();i++)
  selected_word[i]=toupper(selected_word[i]);//converts the whole word
  to uppercase form
43
44      bool did_you_guess_it[100]={0}, is_this_letter_used[26]={0};
45      for(int i=0;i<selected_word.size();i++)if(selected_word[i]==' ')
  did_you_guess_it[i]=1; //The spaces are given to the player, guessing
  them isn't needed
46
47      char current_char;
48      int previous_counter=0,current_counter=1;
49      bool you_win=0, on=1;
50//_____________________GAMEPLAY_____________________________________
51      while(on)
52      {
53          print_word(selected_word,did_you_guess_it);
54          if(current_counter==previous_counter)cout<<"\n\nyou guessed
  wrong.";
55          cout<<"\n\nyou have "<<lives<<" lives left. Remember, you
  still have "<<HINTS<<" hints, enter '1' to use them or '0' to
  surrender\n";
56          previous_counter=
  num_of_guessed_chars(did_you_guess_it,selected_word.size()); //counts
  how many characters are guessed right before giving input.
57          current_char=input_alphabitical_char(is_this_letter_used);
58
59          while(current_char=='1') //Hint escape code
60          {
61              if(HINTS>0)
62              {
63                  HINTS--;
64
  current_char=hint_func(selected_word,did_you_guess_it);
65              }
66              else
67              {
68                  cout<<"\nYou ran out of hints\n";
69
  current_char=input_alphabitical_char(is_this_letter_used);
70              }
71          }
72          if(current_char=='0'){lives=0;break;} //escape code to exit
  the game
73          is_this_letter_used[getindex(current_char)]=1; //Mark the
  current character as used, so you won't be able to enter it again.
74          for(int
  i=0;i<selected_word.size();i++)if(current_char==selected_word[i])
  did_you_guess_it[i]=1; //Mark the letter as guessed, so the letter
  will be printed.
```

```
 75          current_counter=
   num_of_guessed_chars(did_you_guess_it,selected_word.size()); //counts
   how many characters are guessed right after giving input
 76          if(current_counter==previous_counter)lives--;
 77          if(lives==0)on=0;
 78          if(current_counter==selected_word.size())
 79          {
 80              on=0;
 81              you_win=1;
 82          }
 83
 84      }
 85//__________________GAME OVER (POST-GAME)_____________________
 86
 87      for(int i=0;i<selected_word.size();i++)did_you_guess_it[i]=1;
   //we well mark all letters as guessed, so the loser will be able to
   see the word
 88      print_word(selected_word,did_you_guess_it);
 89      if(you_win){cout<<"\n\n\tYOU WIN\n";}
 90      else {cout<<"\n\n\tYOU LOSE\n";}
 91
 92      string str99;
 93      cout<<"\n\nif you want to start a new game, enter 'N'\n";
 94      cin>>str99;
 95      str99[0]=toupper(str99[0]);
 96      newgame=0;
 97      if(str99=="N"){newgame=1;}
 98  }
 99      return 0;
100  }
101//=============================================================
102//___________________PROCESSING FUNCTIONS_____________________
103  void import_from_file(string array[],char file[])
104  {
105      int i=0;
106      ifstream in;
107      in.open(file);
108      while(in)getline(in,array[i++]);
109      in.close();
110  }
111
112  unsigned int get_random_num(int minimum, int maximum)
113  {
114  maximum++; //so the maximum value would be included
115  time_t nTime;
116  static bool first=1; //the value of "first" doesn't change or reset
   when the function is called again.
117  if(first)srand((unsigned) time(&nTime)); //prevents the code from
   reseeding. reseeding make the random function less reliable.
118  first=0;
119  return minimum + (rand() % (maximum-minimum));
```

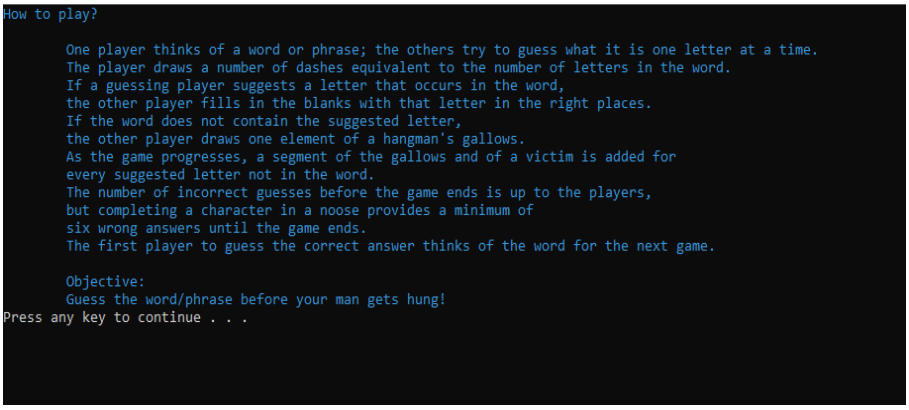
```
120  }
121
122  int getindex(char t)
123  {
124      t=toupper(t);
125      int u=t-65;
126      return u;
127  }
128  int num_of_guessed_chars(bool a[],int a_size)
129  {
130      int counter=0;
131      for(int i=0;i<a_size;i++)if(a[i])counter++;
132      return counter;
133  }
134
135  char hint_func(string selected_word, bool did_you_guess_it[])
136  {
137      int hint;
138      while(1)
139      {
140          hint=get_random_num(0,selected_word.size()-1);
141          if(!did_you_guess_it[hint])break;
142      }
143      return selected_word[hint];
144  }
145  //_____________INPUT FUNCTIONS_____________________________
146
147  int choose_difficulty()
148  {
149      cout<<"\nChoose the game difficulty\nKeep in mind:\n";
150      cout<<"\t\t - Hard: means even lives and really short hard
   words\n";
151      cout<<"\n\nHow difficult would you like the game?\n"<<endl;
152      cout<<"\t1- Easy   = 8 lives  + 3 hints (enter 1)\n";
153      cout<<"\t2- Normal = 6 lives  + 2 hints (enter 2)\n";
154      cout<<"\t3- Hard   = 4 lives  + 1 hint  (enter 3)\n";
155      cout<<"\nEnter your choice: ";
156      short int dif;
157      cin>>dif;
158      while (dif>3||dif<1||!cin)
159      {
160          cin.clear();
161          cin.ignore();
162          cout<<"enter 1, 2 or 3 only";
163          cout<<"\nEnter your choice: ";
164          cin>>dif;
165      }
166      return dif;
167  }
168
169  char input_alphabitical_char(bool is_this_letter_used[])
```

```
170  {
171       cout<<"\nEnter a character: ";
172       string str;
173       cin>>str;
174       if(str=="0")return '0';
175       if(str=="1")return '1';
176       while(str.size()!=1 || !isalpha(str[0]) ||
   is_this_letter_used[getindex(str[0])])
177       {
178            if(str.size()!=1)cout<<"\nYou have entered more than a
   letter, please enter only one English letter.\n";
179            else if(!isalpha(str[0]))cout<<"\nYou have entered an invalid
   value, please enter only one English letter.\n";
180            else if(is_this_letter_used[getindex(str[0])])cout<<"\nYou
   have used this letter before, try again.\n";
181            cout<<"\nEnter a character: ";
182            cin>>str;
183            if(str=="0")return '0';
184            if(str=="1")return '1';
185       }
186       str[0]=toupper(str[0]);
187       return str[0];
188  }
189  //___________ARTISTIC STUFF_____________________________________
190
191  void print_word(string selected_word,bool did_you_guess_it[])
192  {
193       cout<<"\t\t\t\t";
194       for(int i=0;i<selected_word.size();i++)
195       {
196            if(did_you_guess_it[i])cout<<selected_word[i]<<" ";
197            else cout<<"_ ";
198       }
199  }
200  void intro()
201  {
202  string rules[100];
203  import_from_file(rules,"rules.txt");
204  cout<<"How to play?"<<endl<<endl;
205  for(int i=0;i<15;i++)cout<<"\t"<<rules[i]<<endl;
206  }
```

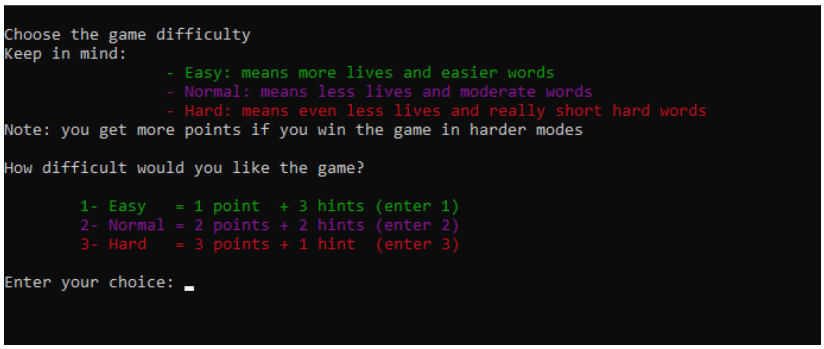
Sample program run:

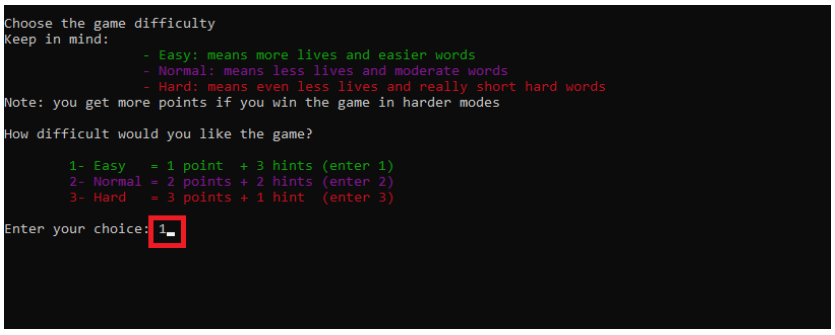
- The program begins with our names and game's name.

- Then game's intro and rules.



- The player can choose which level he wants.





- Now, you should guess letters to save him.

- If the letter you guessed was wrong, then you will lose a live

- If the letter is correct, it will be written in the blank space.

- It will send a message to alert you if you have used the letter before, entered many letters or entered non-English alphabet.

- If you guess correctly, you'll win.

- If you guess the word wrong, you will lose.

References:

1. Programming Applications Lectures & sections
2. https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/
3. http://www.cplusplus.com/forum/general/148263/
4. https://www.programiz.com/cpp-programming
5. http://forums.codeblocks.org/index.php/topic,11353.msg77369.html#msg77369
6. https://www.analyzemath.com/parabola/three_points_para_calc.html
7. https://www.ps2pdf.com/compress-wav
8. https://app.diagrams.net/
9. http://patorjk.com/software/taag/#p=display&f=ANSI%20Shadow&t=HANGMAN
10. https://www.browserling.com/tools/letter-frequency
11. http://www.unit-conversion.info/texttools/ascii/
12. ASCII-Table
13. https://www.joydeepdeb.com/tools/find-
14. https://docs.microsoft.com/en-us/previous-versions//dd743680(v=vs.85)?redirectedfrom=MSDN
15. http://www.cplusplus.com/forum/beginner/70856/
16. https://docs.microsoft.com/en-us/windows/console/console-screen-buffers
17. http://www.cplusplus.com/forum/beginner/5830/