

**C Project Report**

**Rescue Apples**

**Group 1**

**TAQ-Phy Dairy co. LTD**

**Jilin University, Changchun**

**July 7, 2019**

**CONTENTS**

[1 Abstract 1](#_Toc13435469)

[2 Introduction 1](#_Toc13435470)

[3 Problem Statement 2](#_Toc13435471)

[3.1 Interface System & Function Choice 2](#_Toc13435472)

[3.2 Setting 2](#_Toc13435473)

[3.3 Game 3](#_Toc13435474)

[3.3.1 Game Wordlist Creation 3](#_Toc13435475)

[3.3.2 Game Control 3](#_Toc13435476)

[3.4 Score 3](#_Toc13435477)

[3.4.1 Record Historical Score 3](#_Toc13435478)

[3.4.2 Show Historical Score 4](#_Toc13435479)

[4 Analysis 4](#_Toc13435480)

[4.1 How to play more than one time in an execution 4](#_Toc13435481)

[4.2 How to control interface jump 5](#_Toc13435482)

[4.3 How to set, save and apply setting parameters 5](#_Toc13435483)

[4.4 How to realize animation effect of falling apples 6](#_Toc13435484)

[4.5 How to judge the state of game 6](#_Toc13435485)

[4.5.1 Input Judgement 6](#_Toc13435486)

[4.5.2 Position Judgement 7](#_Toc13435487)

[4.6 How to realize word operation 7](#_Toc13435488)

[4.7 How to create play wordlist 8](#_Toc13435489)

[5 Design 9](#_Toc13435490)

[5.1 User Interface 9](#_Toc13435491)

[5.2 data flow diagram 10](#_Toc13435492)

[5.3 Hierarchy of Source Files & Head Files 11](#_Toc13435493)

[5.4 Flow Chart of Pivotal Functions 13](#_Toc13435494)

[5.4.1 main body 13](#_Toc13435495)

[5.4.2 Setting interface 13](#_Toc13435496)

[5.4.3 Score Interface 14](#_Toc13435497)

[5.4.4 Game 15](#_Toc13435498)

[5.4.5 Create game wordlist 16](#_Toc13435499)

[5.4.6 Game Body 17](#_Toc13435500)

[5.4.7 Write data to file 18](#_Toc13435501)

[6 Division of Labor 19](#_Toc13435502)

[6.1 Zheng Wenjun 19](#_Toc13435503)

[6.1.1 Programing 19](#_Toc13435504)

[6.1.2 Coding 19](#_Toc13435505)

[6.1.3 Presentation 20](#_Toc13435506)

[6.2 Chao Yuxin 20](#_Toc13435507)

[6.2.1 Programing 20](#_Toc13435508)

[6.2.2 Coding 21](#_Toc13435509)

[6.2.3 Presentation 21](#_Toc13435510)

[6.2.4 Others 22](#_Toc13435511)

[6.3 Li yanyuan 22](#_Toc13435512)

[6.3.1 Programing 22](#_Toc13435513)

[6.3.2 Coding 22](#_Toc13435514)

[6.3.3 Presentation 23](#_Toc13435515)

[7 Implementation 23](#_Toc13435516)

[7.1 Technical Approaches & Methods Used 23](#_Toc13435517)

[7.1.1 Algorithm 23](#_Toc13435518)

[7.1.2 File 25](#_Toc13435519)

[7.1.3 Linked list 32](#_Toc13435520)

[7.1.4 Array 35](#_Toc13435521)

[7.1.5 Graphics and animation programming 35](#_Toc13435522)

[7.2 Detail Optimizing 40](#_Toc13435523)

[7.2.1 Robustness 40](#_Toc13435524)

[7.2.2 Editable Lexicon 41](#_Toc13435525)

[8 test 42](#_Toc13435526)

[9 Result & Conclusion 46](#_Toc13435527)

# 1 Abstract

Our game is a typing practice game. It is for those who want to improve their typing skills or test their typing skills. In our game, the users can play game as many times as they want and change the game parameters as they like. They can also inquiry their score in the score interface. Their score and game parameters won’t be missed even if they game is closed.

# 2 Introduction

Most of us may be familiar with typing practice games. In which， the users type the words shown on the screen in a particular box. But most of the games can’t change the game parameters and can’t even show the users’ grades. The words are created randomly.

Our inspiration comes from a typing practice game called Kingsoft TypeEasy. Like most typing practice games, it has the deficiency I stated just now, but it change the form of the game. During this game, with a word in it, the apple falls from the top of the game window, and the users are required to type the words in apple before the apples falls to the ground. We think it’s quite interesting and innovative, so we adopt this design.

And our aim is to overcome the deficiency of most typing practice games and make our game more functional and efficient. Meanwhile, we’d also like to improve the robustness and beauty of our game.

There are four interfaces in our game. Firstly, when the program is opened, after a short animation, the users came into the main interface. And there are three sub-interfaces in it, which are game, score and setting interface. Users are able to change the game parameters in the setting interface and inquiry their typing speed and accuracy in the last ten times in the score interface. The game interface is the main body of the game. The scores and game data won’t be lost even the game is closed.

# 3 Problem Statement

## 3.1 Interface System & Function Choice

First of all, we need to create a main interface to welcome the user and to give them entries to the other functions, including setting, game and score.

## 3.2 Setting

In this interface, we are supposed to create four sliders which enable the user to adjust the values of the falling speed of apples, the length of game, background music, the biggest length of words, and whether there is sound effect during the game.

## 3.3 Game

### 3.3.1 Game Wordlist Creation

Before the game start, we have to create the game wordlist by selecting words from all wordlist according to previous errors. These words will appear on the apple in order during the game.

### 3.3.2 Game Control

In the game, we need to realize the falling of apples, each with a word on it. Then, we have to decide the rule of game: If the user typed correct characters, the characters will be shown in the input box at the bottom of the interface. When the characters in the input box are identical to the word on the apple, this apple will be replaced by a new apple with a new word. If the user typed wrong characters, nothing will happen. If the user fail to type the word correctly before it reach the ground, this apple will also be replaced by a new apple with a new word. Whenever a new apple appears, the input box should be cleared.

In addition, we need to decide the condition of the end of game. The first condition is the number of word in the game equals to the game length chose in Setting interface. The second condition is the user pressed ESC button on the keyboard.

## 3.4 Score

### 3.4.1 Record Historical Score

As soon as the user enter into the score interface, the program have to read the game data of the latest 10 games, which means these data need to be recorded during the game.

### 3.4.2 Show Historical Score

The program need to draw two line charts to show the typing speed and typing accuracy rate.

# 4 Analysis

## 4.1 How to play more than one time in an execution

Normally, the program ends spontaneously after its run is finished, which decides that the program can execute only once. But to make this game more practical and functional, we hope that the users can play this game as many times as they want and end the game when they think the practice is enough. So that means the users run program once and execute game several times.

To solve this problem, we divide the body of the program into two parts, which are main function and Main\_Interface function. The main function can invoke Main\_Interface function. And Main\_Interface function can create the main interface, through which we can enter game, score, setting interface. And in these interface, we can return to the main interface, and the users can do it once and once again, so it can solve the problem perfectly.

## 4.2 How to control interface jump

There are many interfaces in our game, such as main interface, game interface, score interface, etc. So there is a problem we can’t avoid facing, which is how to switch the interface and realize the interface jump.

Naturally, we hope to make our game more relevant to the real games. And in most real games, the users can control the interface through the mouse. To realize it, we apply the GetMouseMsg function, which can get the coordinate information of mouse click. So we can jump to certain interface according to the specific area of mouse click.

## 4.3 How to set, save and apply setting parameters

In our game ,we design a setting interface. In the setting interface, the users can change some game parameters, including the song, the falling velocity of the apple, the length of the word , etc. In this part, the users can change the parameter if they like, meanwhile, the parameter data won’t lost after the end of the game, and when the game starts, historical parameters will be used. In a word, that is how to change, save and apply the parameter data.

To realize our vision, we also use the GetMouseMsg function to get the click information of the mouse and use these data to finish drawing the setting interface once and once again . Then we transfer the coordinate data to the setting data and save them in Setting.txt. When the game starts again, the data will be read from the Setting.txt and transferred to coordinate data, and these data will be used to finish initial drawing and the setting data will be applied in the game. Above all, we choose to use GetMouseMsg function and establish a Setting.txt file to solve this problem.

## 4.4 How to realize animation effect of falling apples

When the game start, the apples are supposed to fall from the top of the window.

To realize the animation effect, we used a for loop.Before each cycle, the vertical coordinate value of the apple plus one. In the cycle, the previous apple picture will be covered by a rectangle whose color is identical to the background color. Then, a new apple will be drawn at the new coordinate. Afterwards, we employed Sleep() function to make the apple stay for a certain length of time. As a result, when the loop start, the user can see the falling effect because of the persistence of vision.

## 4.5 How to judge the state of game

### 4.5.1 Input Judgement

During the game, the user can type in characters continuously at any time, and the program have to respond to user’s input immediately. Therefore, the judgement of input becomes crucial.

To solve this problem, we add a judgement in the for loop, using kbhit() function to verdict whether the user have pressed the keyboard in the last cycle. Then, getche() function is used to receive the character in the keyboard buffer. If the character is right, we also have to use strcmp() function to verdict whether the whole word has been typed correctly.

### 4.5.2 Position Judgement

If the user fail to type the word correctly before the apple falls on the ground, this apple will be removed and a new one will be generated. Therefore, the judgement of apple’s position should be added in the loop.

So, we add a judgement at the end of each loop, comparing the vertical coordinate value of the apple with the vertical coordinate value of the window, which realizes the effects we mentioned above.

## 4.6 How to realize word operation

As a typing practice game, countless words will be involved in this program. The program should not only create a list of play words for game, but also test whether words were typed correctly as well as record the mistake word and its error location. In addition, words containing historical error fragment are to be sort out for repeated practice.

Therefore we put forward a method that operate all word by linked list. We are inclined to engage linked list rather than other tool including string array and 2-dimension array for many reasons. Firstly, it is much more flexible. For example, when the program read words saved in file, an array with a certain length possesses a size limitation so the size should be set big enough which may be too lavish to deal with a small number of words. What’s more, it will be too complicated to deleted some certain words in an array which means all word besides it must be moved in this process.

However, linked list fits our needs perfectly. One kind of linked list will be defined to store words read from lexicon as well as providing them to game. Another kind of linked list is to record error and the third one can store error fragment for searching certain words in lexicon.

## 4.7 How to create play wordlist

Being able to provide users with high-efficiency practice by adjusting play wordlist according to historical error is one of the advantages of this game. We would like to make the program sort out words in the lexicon which contain the “error fragment” of the latest two (if possible) or one times automatically.

If the amount of these part of word is bigger than a certain percentage of the setting all node amount, only a part of this wordlist will be used in play wordlist and the other part will be composed by some word abstracted from all wordlist randomly.

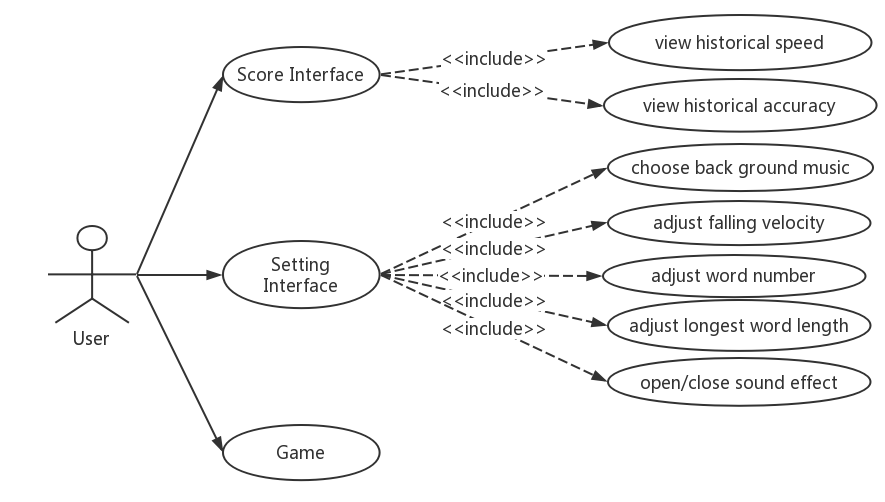
That is to say, we have to realize the following three steps: 1) string searching 2) compare number and count the length of linked list 3) create 2 linked list with required length and joint than together 4) randomize the linked list.

# 5 Design

## 5.1 User Interface

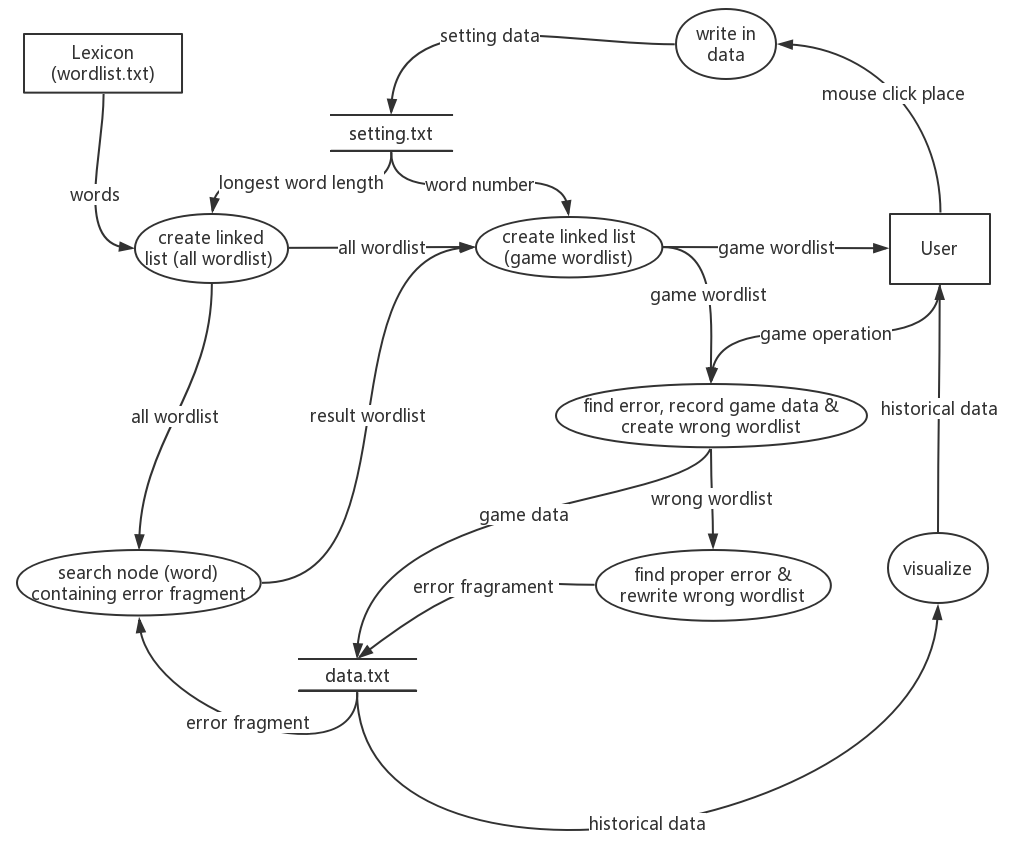
Form the aspect of user, after viewing a brief opening animation, he will face a main interface, three button named “Game”, “Score” and “Setting” are lying in the windows.

In score interface, he can inspect average typing speed and average accuracy of latest 10 runs of games in the form of broken line chart. In setting interface, he can adjust several parameters including “back ground music”, “falling velocity”, “word number”, “longest word length”, “sound effect”. In game part, he can play the game followed by showing current accuracy.

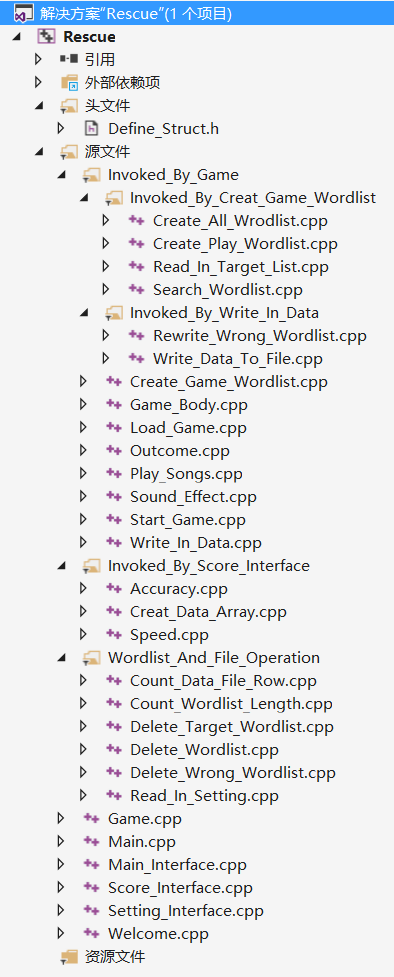


## 5.2 data flow diagram

Lexicon, setting parameter and historical data including error fragment are stored in “wordlist.txt”, “setting.txt”, “data.txt”. Linked list are used to convey data and in word operation.



## 5.3 Hierarchy of Source Files & Head Files

1 head file should be established to define all kind of linked list needed.

Define\_Struct.h

6 most fundamental and pivotal source files will be established:

Main.cpp

Welcome.cpp

Main\_Interface.cpp

Game.cpp

Score\_Inverface.cpp

Setting\_Interface.cpp

To support the file above run successfully 3 series of second-class-called file should be involved:

1) Wordlist\_And\_File\_Operation:

Count\_Data\_File\_Row.cpp

Count\_Wordlist\_Length.cpp

Delete\_Target\_Wordlist.cpp

Delete\_Wordlist.cpp

Delete\_Wrong\_Wordlist.cpp

Read\_In\_Setting.cpp

2) Invoked\_By\_Score\_Interface

Creat\_Data\_Array.cpp

Accuracy.cpp

Speed.cpp

3) Invoked\_By\_Game

Create\_Game\_Wordlist.cpp

Load\_Game.cpp

Start\_Game.cpp

Game\_Body.cpp

Write\_In\_Data.cpp

Outcome.cpp

Play\_Songs.cpp

Sound\_Effect.cpp

3.1) Invoked\_By\_Creat\_Game\_Wordlist

Create\_All\_Wrodlist.cpp

Read\_In\_Target\_List.cpp

Search\_Wordlist.cpp

Create\_Play\_Wordlist.cpp

3.2) Invoked\_By\_Write\_In\_Data

Rewrite\_Wrong\_Wordlist.cpp

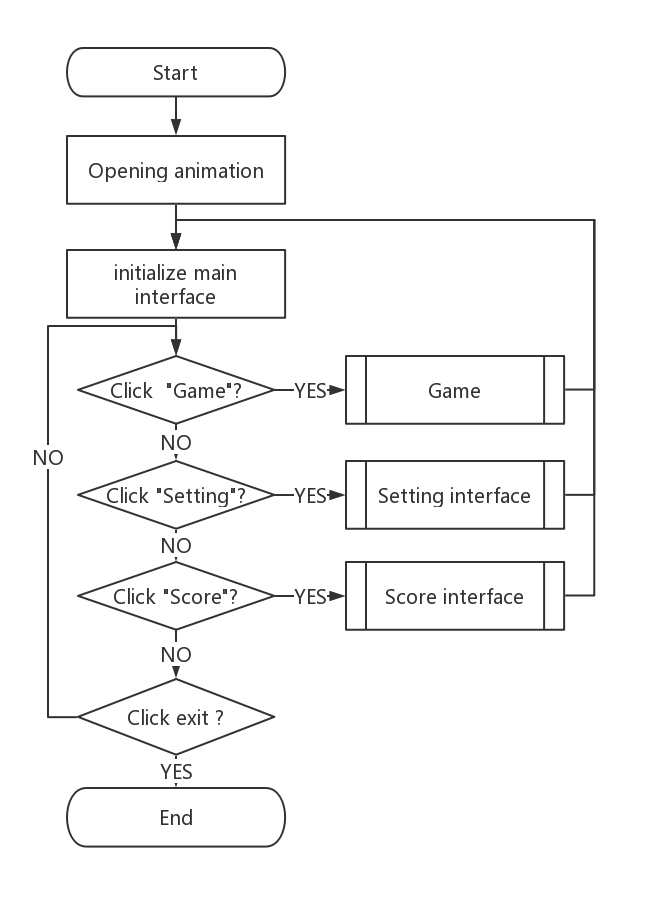
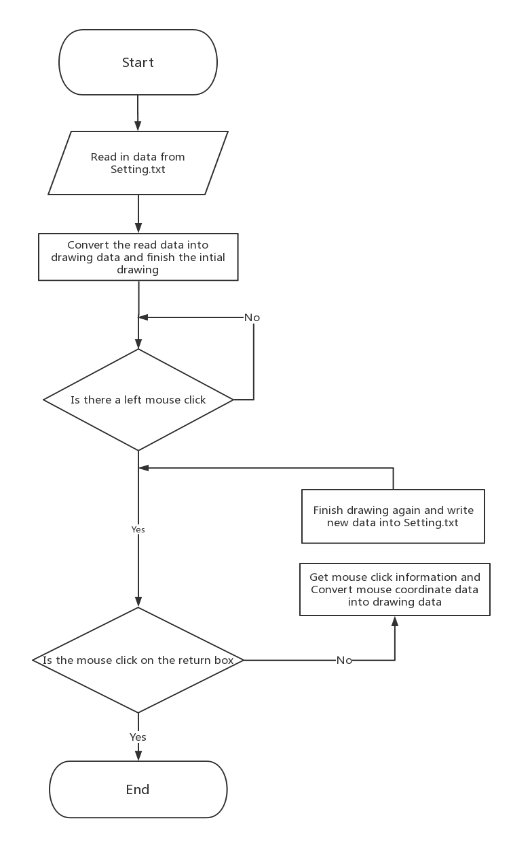
Write\_Data\_To\_File.cpp

## 5.4 Flow Chart of Pivotal Functions

### 5.4.1 main body

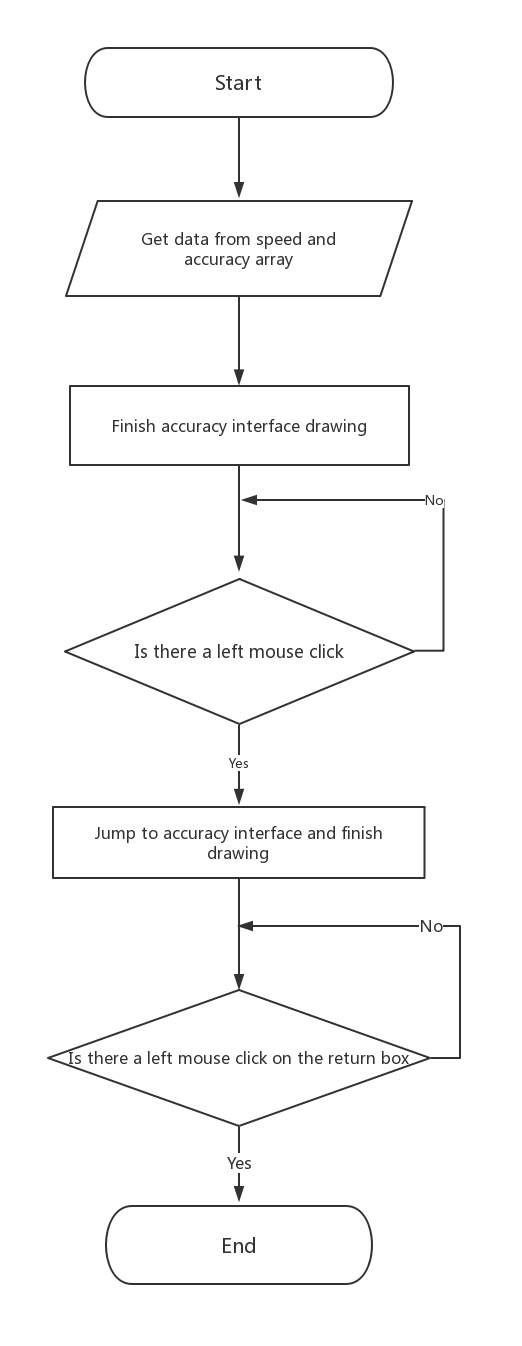
In the main interface, user can jump into three functional part by click button.

### 5.4.2 Setting interface

The loop enable user mouse click to change all kind of setting and click “exit” is a signal to return to main interface and save the changes.

### 5.4.3 Score Interface

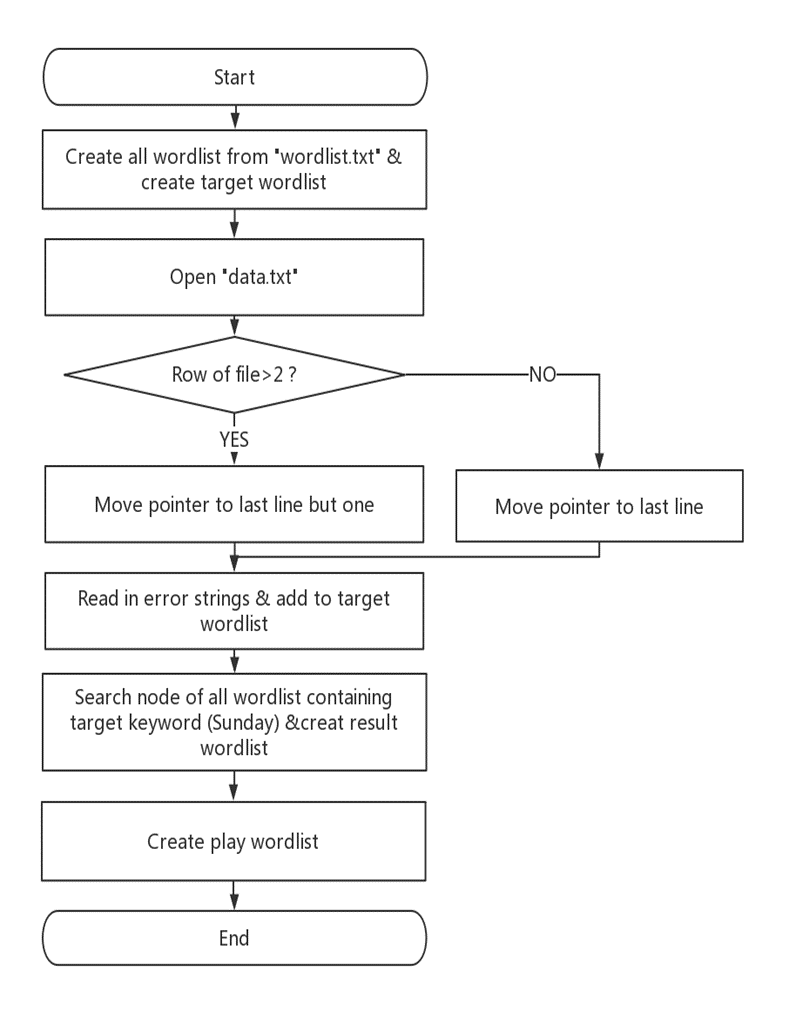
In this part, two broken line chart will be shown on by one when user click mouse anywhere within window.



### 5.4.4 Game

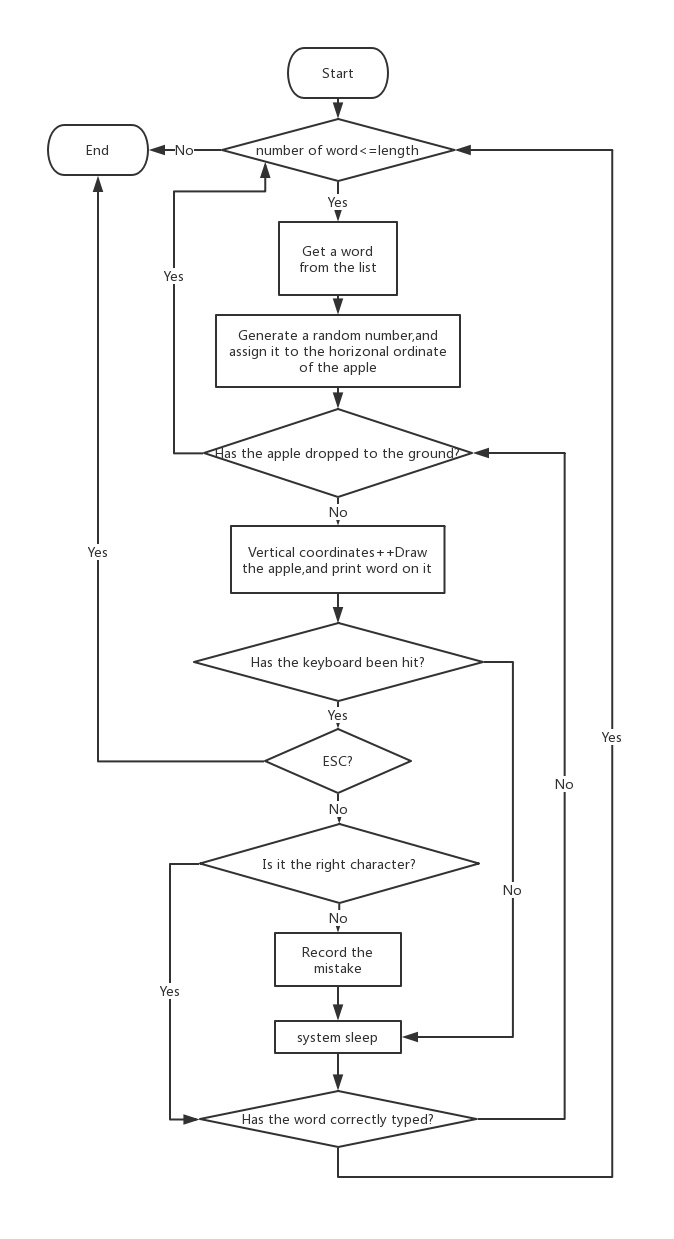
### 5.4.5 Create game wordlist

We only get data from the last one or two (if have) row, which will be used in game word list creation.



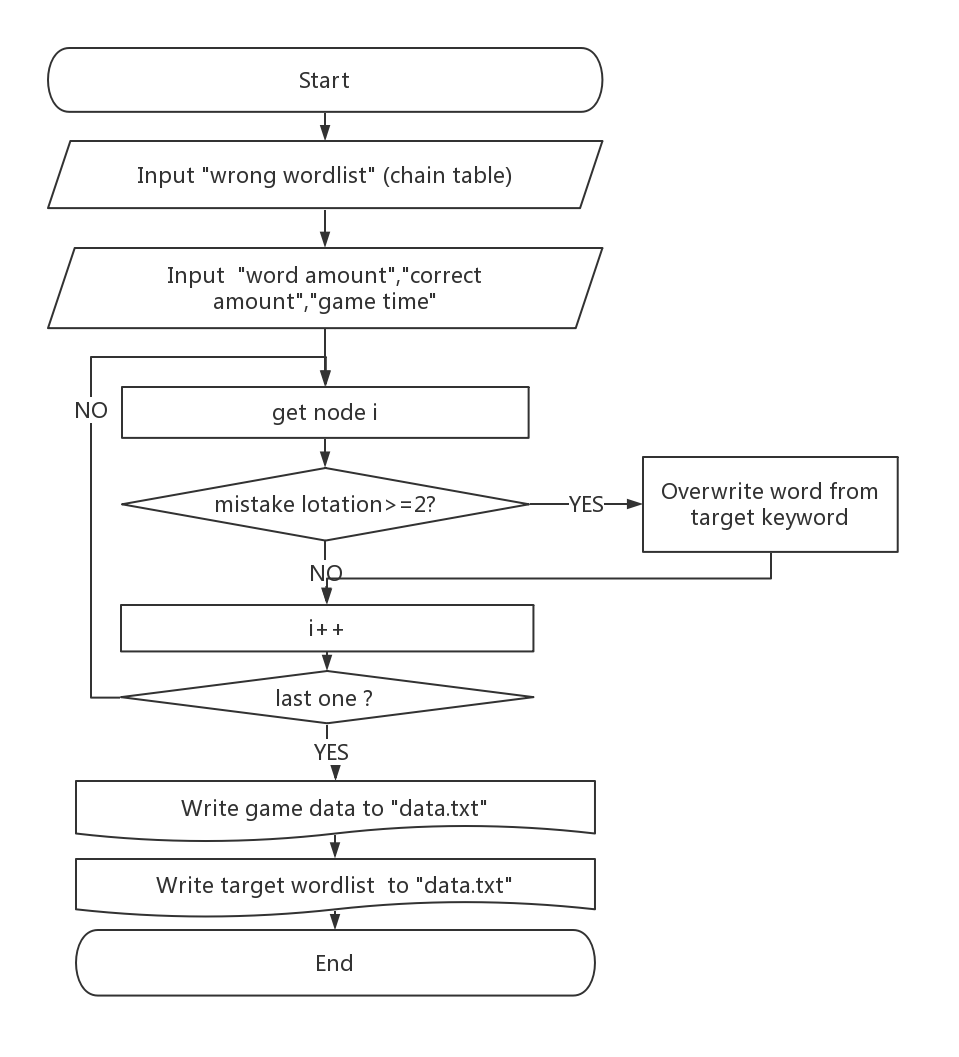
### 5.4.6 Game Body

Two loops are engaged in game. The big one make the apple picture falling continuously, the small one can check whether input correct.



### 5.4.7 Write data to file

There is a judgement in this part, only errors that meet the criteria are recorded, that is, the error location is bigger than 2, for example, in word “abcdef”, the mistake happened on ‘a’ and ‘b’ will never be recorded.



# 6 Division of Labor

All team members are responsible for the proportional and appropriate work

/\*ATTENTION: NO RELATIONSHIP BETWEEN FUNCTION AMOUNT AND CONTRIBUTION COMMENT OF SOURSE CODE POVIDE MORE INFORMATION\*/

## 6.1 Zheng Wenjun

### 6.1.1 Programing

Create organizational structure for the program & overall design

### 6.1.2 Coding

(implement providing game wordlist and storing game data)

Count\_Data\_File\_Row.cpp

Count\_Data\_File\_Row(void)

Count\_Wordlist\_Length.cpp

Count\_Wordlist\_Length(WORDLIST \* )

Delete\_Target\_Wordlist.cpp

Delete\_Target\_Wordlist(TARGET\_WORDLIST\*)

Delete\_Wordlist.cpp

Delete\_Wordlist(WORDLIST \* )

Delete\_Wrong\_Wordlist.cpp

Delete\_Wrong\_Wordlist(WRONG\_WORDLIST \* )

Create\_All\_Wrodlist.cpp

Create\_All\_Wrodlist(int)

Create\_Play\_Wordlist.cpp

Create\_Play\_Wordlist(int, int, RES…LIST\*,ALL…LIST \*)

Create\_All\_Part(int, int, int, ALL\_WORDLIST \*)

Create\_Play\_Wordlist(int, int,RES…LIST\*,ALL…LIST \*)

Randomize\_Play\_Wordlist(int, PLAY\_WORDLIST\* )

Read\_In\_Target\_List.cpp

Read\_In\_Target\_List(int)

Search\_Wordlist.cpp

Search\_Wordlist(TARGET\_WORDLIST\*, ALL\_WORDLIST\*)

int FindIndex(char \*target, char temp)

Rewrite\_Wrong\_Wordlist.cpp

Rewrite\_Wrong\_Wordlist(WRONG\_WORDLIST \*, int \*)

Write\_Data\_To\_File.cpp

Write\_Data\_To\_File(WRO…LIST\*, int, int, int, int, int)

### 6.1.3 Presentation

First presentation

## 6.2 Chao Yuxin

### 6.2.1 Programing

Provide technical support in area of “easyx” and multimedia materials

### 6.2.2 Coding

(implement the core part of the program--- the whole game)

Game.cpp

void Game(void)

Read\_In\_Setting.cpp

Read\_In\_Setting(int\* , int\*, int\*,int\*, int \*)

Game\_Body.cpp

Game\_Body(int,int,int,int,int,PL…ST\*,WR…ST\*\*,WR…ST\*)

Load\_Game.cpp

Load\_Game(int song)

Outcome.cpp

Outcome(int numberOfSuccessWords, int length)

Play\_Songs.cpp

Play\_Songs(int song)

Sound\_Effect.cpp

Sound\_Effect(int soundEffect)

Start\_Game.cpp

Start\_Game()

Write\_In\_Data.cpp

Write\_In\_Data(WRONG\_WORDLIST\*, int, int, int)

### 6.2.3 Presentation

Second presentation

### 6.2.4 Others

Usage of Githu

## 6.3 Li yanyuan

### 6.3.1 Programing

Provide project ideas & puts forward overall requirement

### 6.3.2 Coding

(interface system and setting system & usage of mouse function)

Define\_Struct.h

Main.cpp

main(void)

Welcome.cpp

Welcome(void)

Main\_Interface.cpp

Main\_Interface(void)

Score\_Interface.cpp

void Score\_Interface(void)

Setting\_Interface.cpp

void Setting\_Interface(void)

void Load\_Setting(int a, int b, int c, int d, int song)

Accuracy.cpp

Accuracy(int \* accuracyArray)

Laod\_Accuracy\_Frame(void)

Creat\_Data\_Array.cpp

void Creat\_Data\_Array(int\*, int\*, int)

Speed.cpp

Laod\_Speed\_Frame(void)

Speed(int \* speedArray, int \* accuracyArray)

### 6.3.3 Presentation

Final presentation

# 7 Implementation

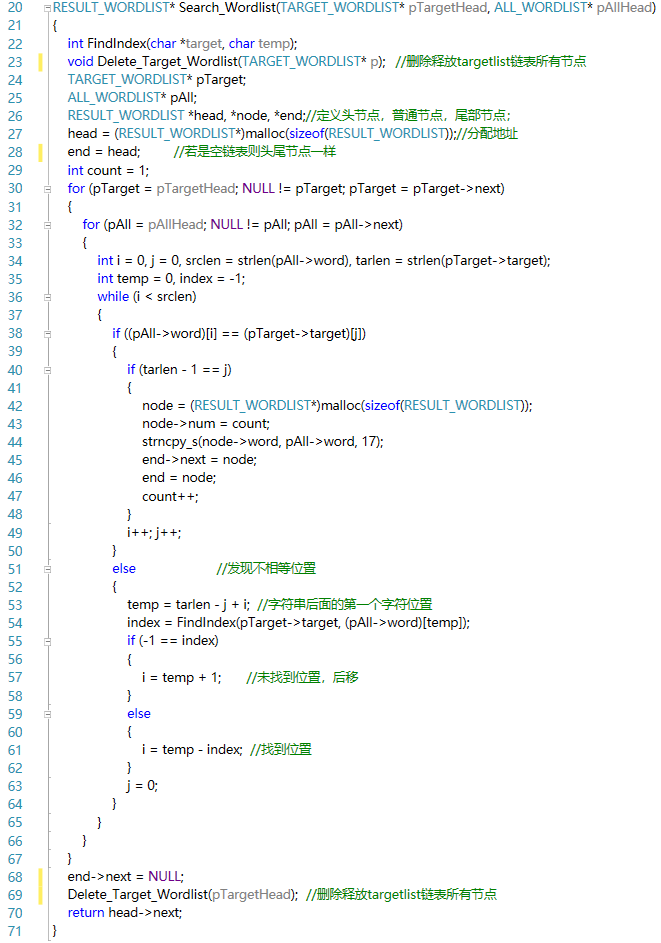
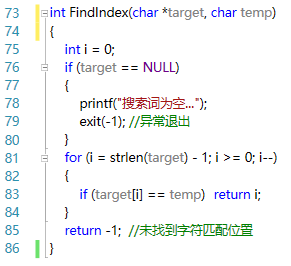
## 7.1 Technical Approaches & Methods Used

### 7.1.1 Algorithm

As ALL\_WORDLIST and TARGET\_WORDLIST have been defined and created in “Define\_Struct.h” as the flowing chart



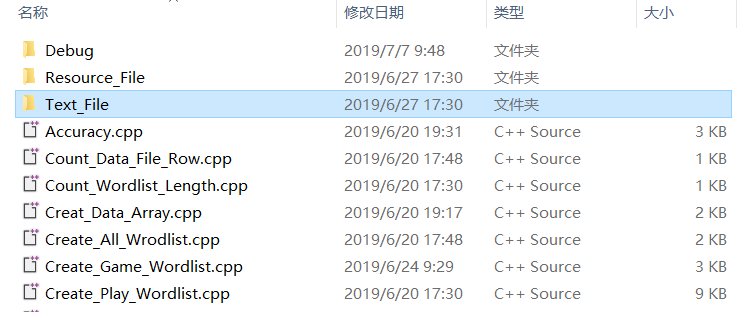
The next step is to find words (word[17]) that contained error fragment (target[4]) in ALL\_WORDLIST. Therefor we used Sunday Algorithm which has been involved in “Search\_Wordlist.cpp”. To implement this part, two function are contained in these source file. (to make screen capture easier, the code format has been adjusted)

And function “FindIndex”

### 7.1.2 File

File (in format of “txt”) can save some important data in disk forever which means that the data will never vanish when the software is closed. So, three necessary composed our file system.

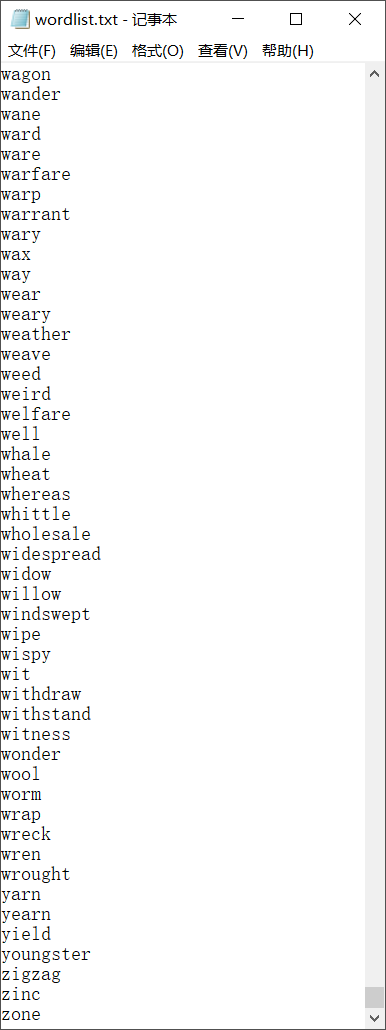
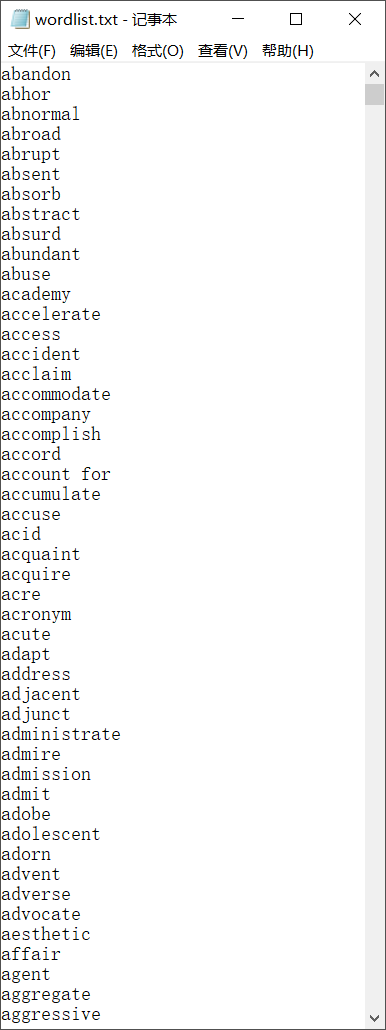
All text file has been store in a special folder named “Text\_File”.



And three text file are contained in this floder.



**7.1.2.1 wordlist.txt**



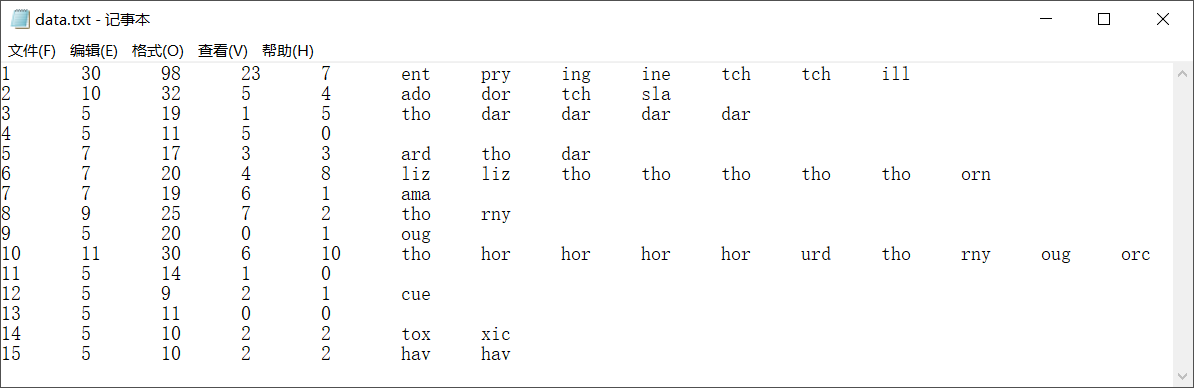
Wordlist is the entity of lexicon more than 2000 words are stored in the file. Every time when user run the game, the program will open this file and read these word one by one and joint them as a linked list named “ALL\_WORDLIST”. That is to say, this file will never be changed or wrote in. In addition each word occupied one row and divided by ‘\n’, which made it possible for user to edit his own lexicon (see 6.2).

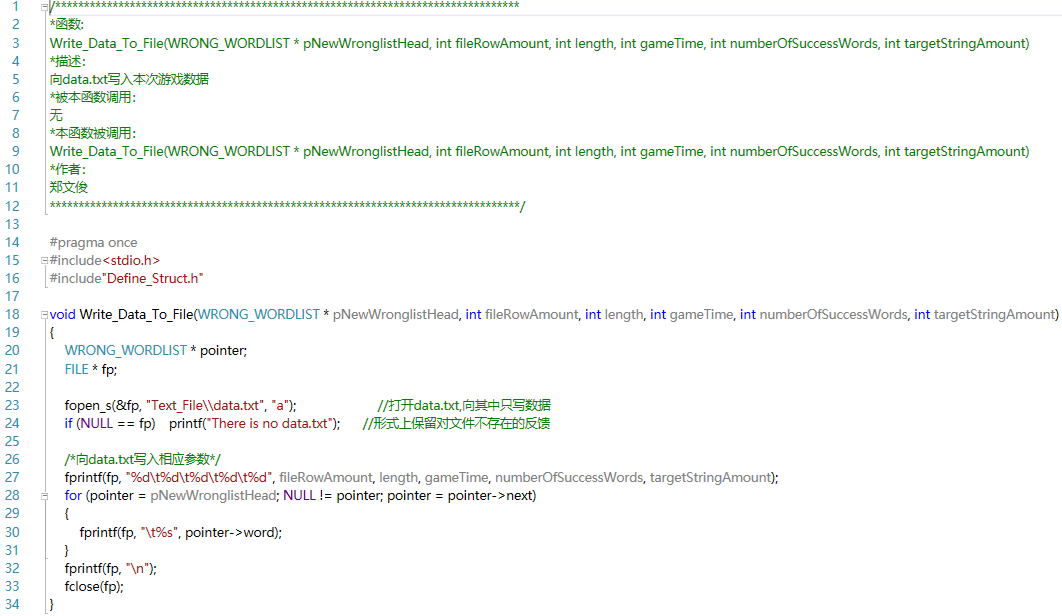
**7.1.2.2 data.txt**

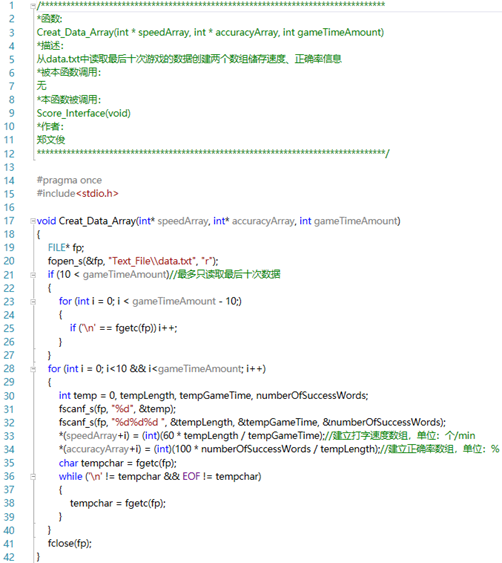
All historical game data and error fragment will be stored in this file.

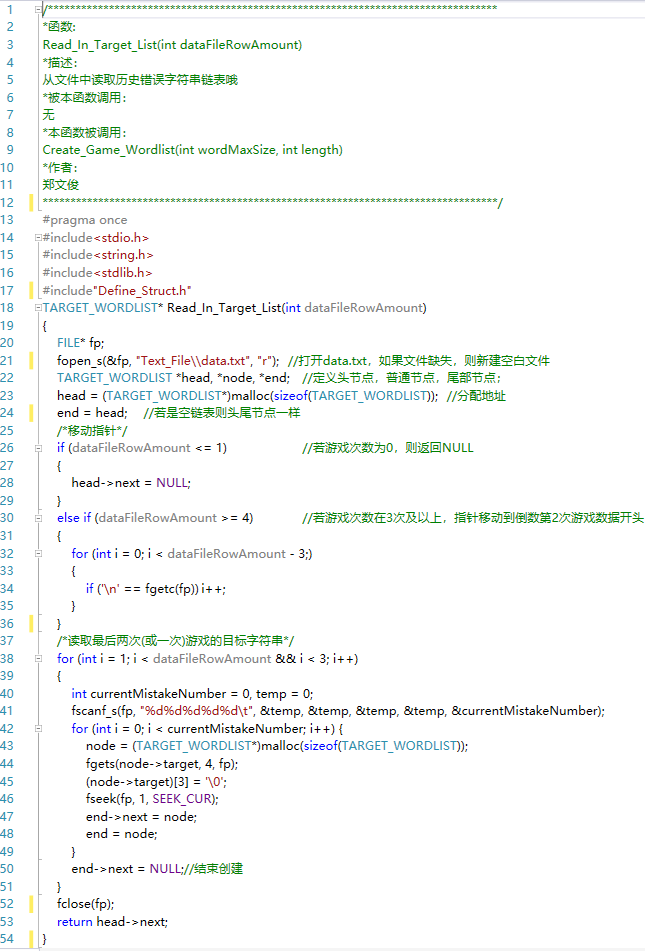
Each line store a special data:

(int) game time number; (int) word number; (int) time(second); (int) correct number; (int)target fragment number; (char(\*3))string1; (char(\*3))string2; (char(\*3))string3……

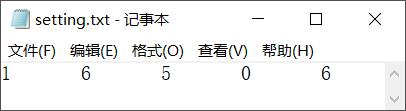
Each game’s runs will write a row in this file as the first number add one after the other. While two part of the program will be able to read data from it that is “Score\_Interface” and “Create\_Game\_Wordlist”

Write data to “data.txt” (pNewWronglistHead’s creation has been done)

Read in the 2nd to 4th data of last ten times (at most) as two array in order to plot broken line chart in Score\_Interface.

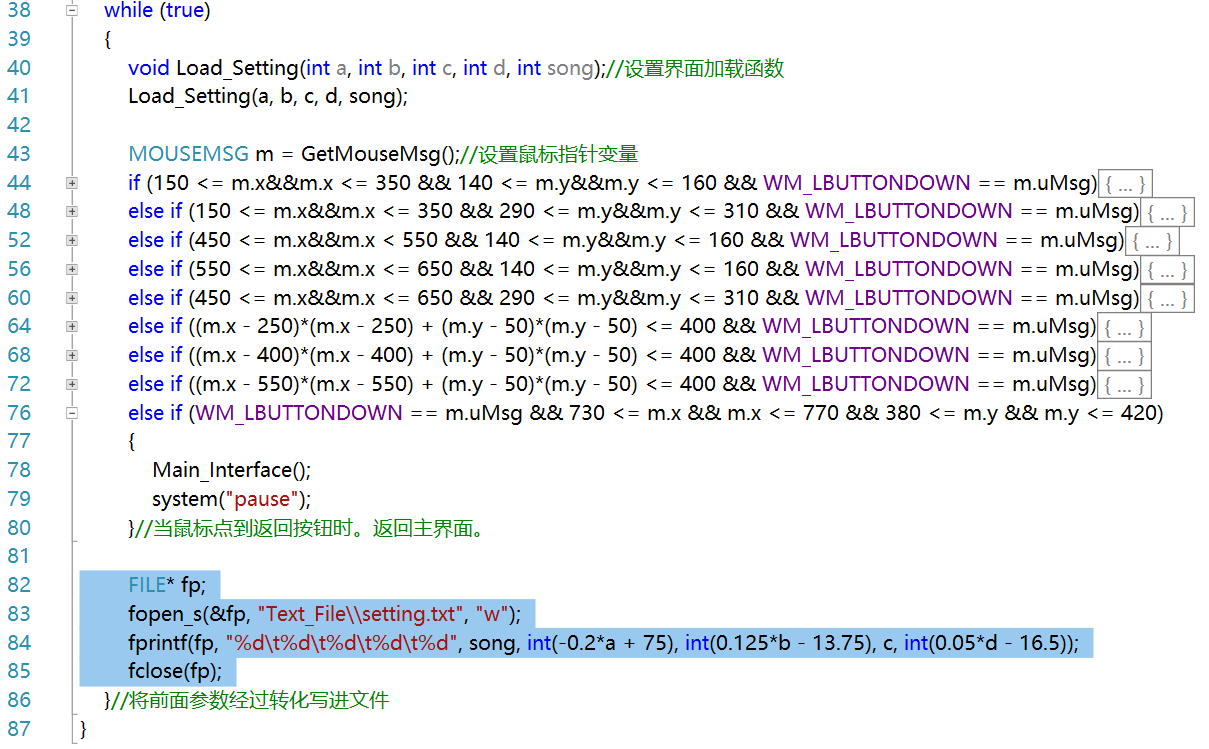
Read in the target strings for searching as which has been mentioned in 6.1.1 (to make screen capture easier, the code format has been adjusted )

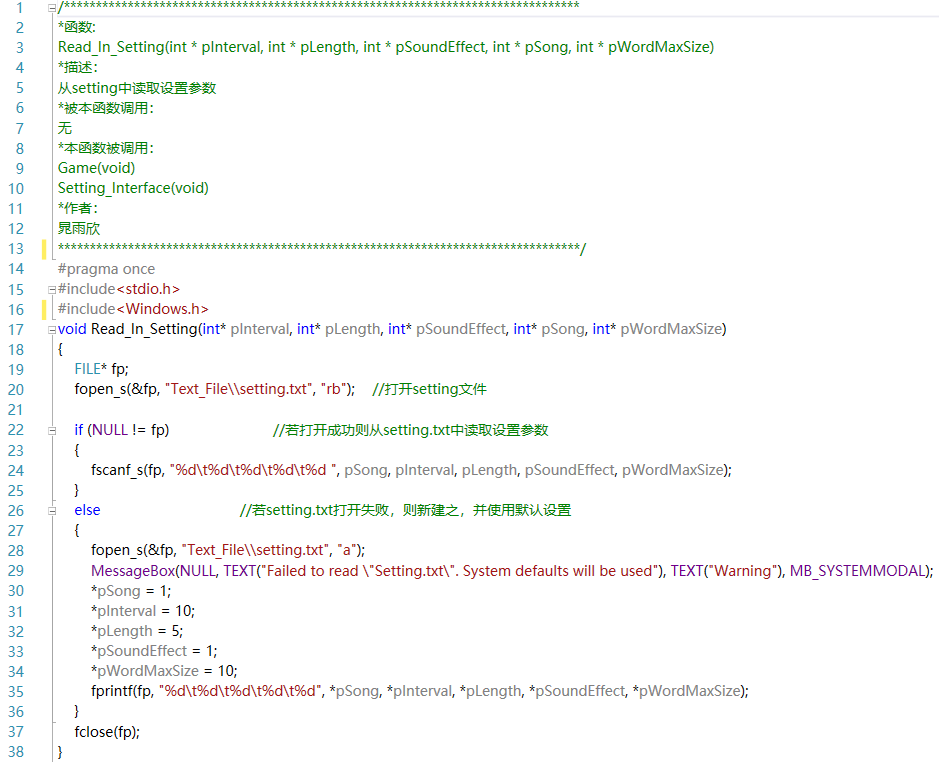
**7.1.2.3setting.txt**



5 number in this file is a match to 5 setting in “Setting\_Interface”. To be more specifically, they are:

(int)Song; (int) time interval; (int)word number; (int)sound effect; (int) longest word length

void Setting\_Interface(void) will write data to this file and the new setting parameters will cover the former one which means every time the user change the setting, this file will be rewrite.

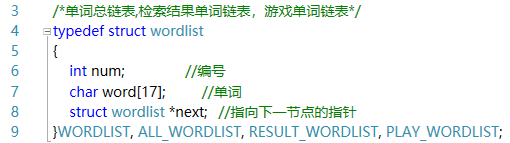
To establish a proper game, when a new game begin, the program will read what contain in this file. (from void Read\_In\_Setting(int\* pInterval, int\* pLength, int\* pSoundEffect, int\* pSong, int\* pWordMaxSize) )

### 7.1.3 Linked list

There are 3 types of linked list with 6 names used in every area of words operation.

**7.1.3.1 Definition of linked list**

All lined list are defined in a head file named “Define\_Struct.h”

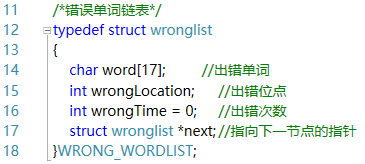


WORDLIST General term, used in general operation like counting length and deleting

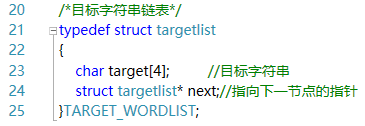
ALL\_WORDLIST store lexicon, created from wordlist.txt

RESULT\_WORDLIST store words containing historical error fragments, the result of searching

PLAY\_WORDLIST provided to game, combined with (maybe part of) RESULT\_WORDLIST and random part of ALL\_WORDLIST



WRONG\_WORDLIST store the mistake words and error location, created during the game process, will be rewrite to error fragment later



TARGET\_WORDLIST store the error fragment, created from data.txt, used to search words in ALL\_WORDLIST

**7.1.3.2 Operation of linked list**

Count\_Wordlist\_Length.cpp

To judge how many words in Game wordlist come from RESULT\_WORDLIST and ALL\_WORDLIST, the length of these linked list must be counted.

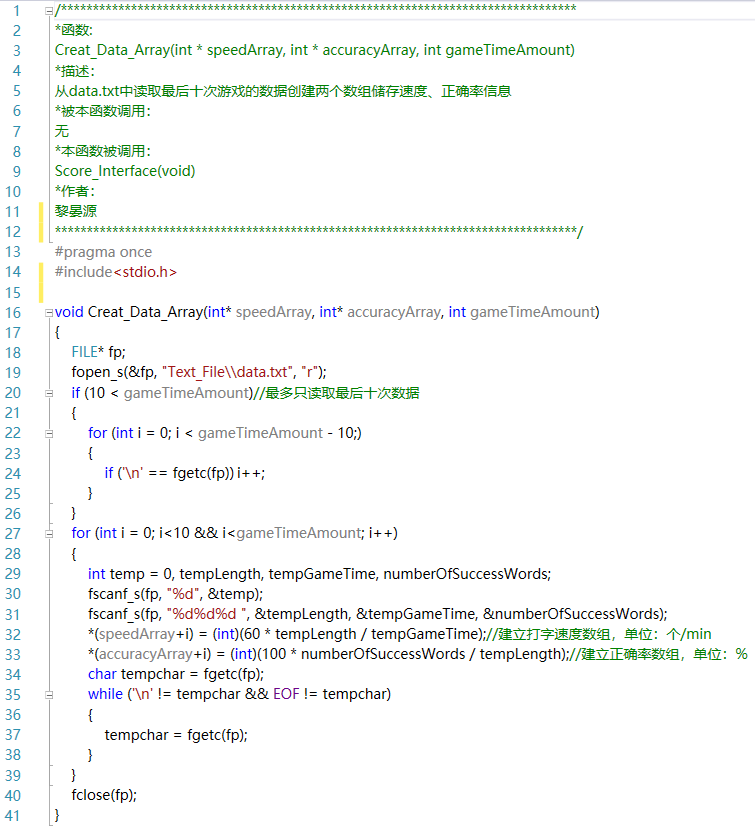
Create\_Play\_Wordlist.cpp

The standard of proportion is 2/3, which means that if the length of RESULT\_WORDLIST is bigger than 2/3 counterpart of length (as setting), any excess will be deleted randomly, else all member of it will be engaged. Then rest part will be filled by words from wordlist randomly.

### 7.1.4 Array

2 arrays will be used to store number data from data.txt which will be finally provided to Score\_Interface.

Creat\_Data\_Array.cpp



### 7.1.5 Graphics and animation programming

**7.1.5.1 game animation**

This is a part of Game\_Body.cpp the animation of falling apples is implemented in this way:

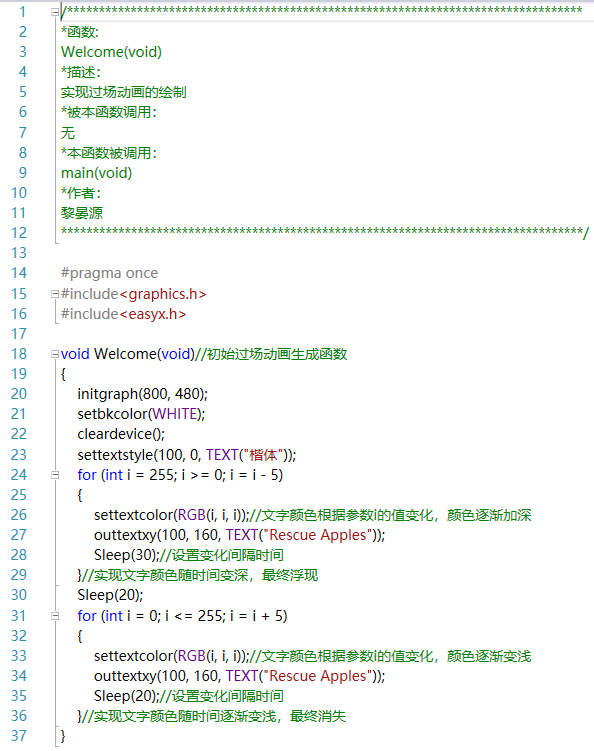
Insert a picture of apple at a certain place->wait for a certain time-> insert the picture at one pixel beneath-> insert a color block to cover the former picture of apple.

Consequently, the falling velocity can be decided by the interval time’s setting.



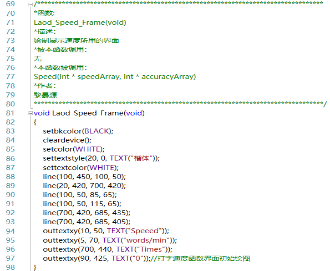
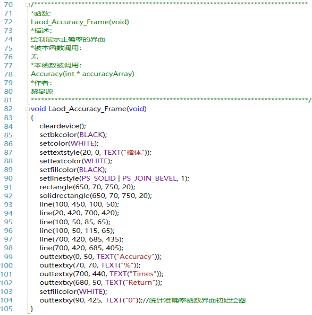
**7.1.5.2 Opening Animation**

Every 30 milliseconds write a new, deeper black “RESCUE APPLES”. After stopping 20 milliseconds, the text fades away



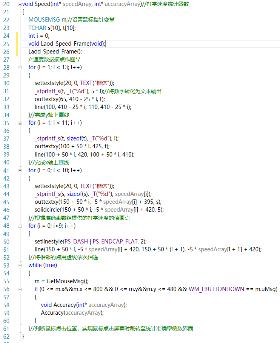
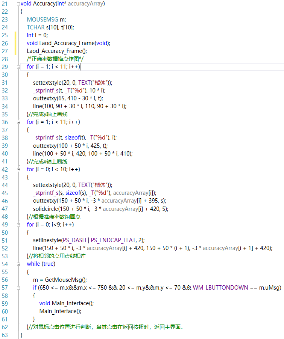
**7.1.5.3 Show historical score**

Both accuracy & speed showing based on the following steps:



1) Draw the background and coordinate system

2) Plot points on the coordinate system using data of array and joint then together by broken line.

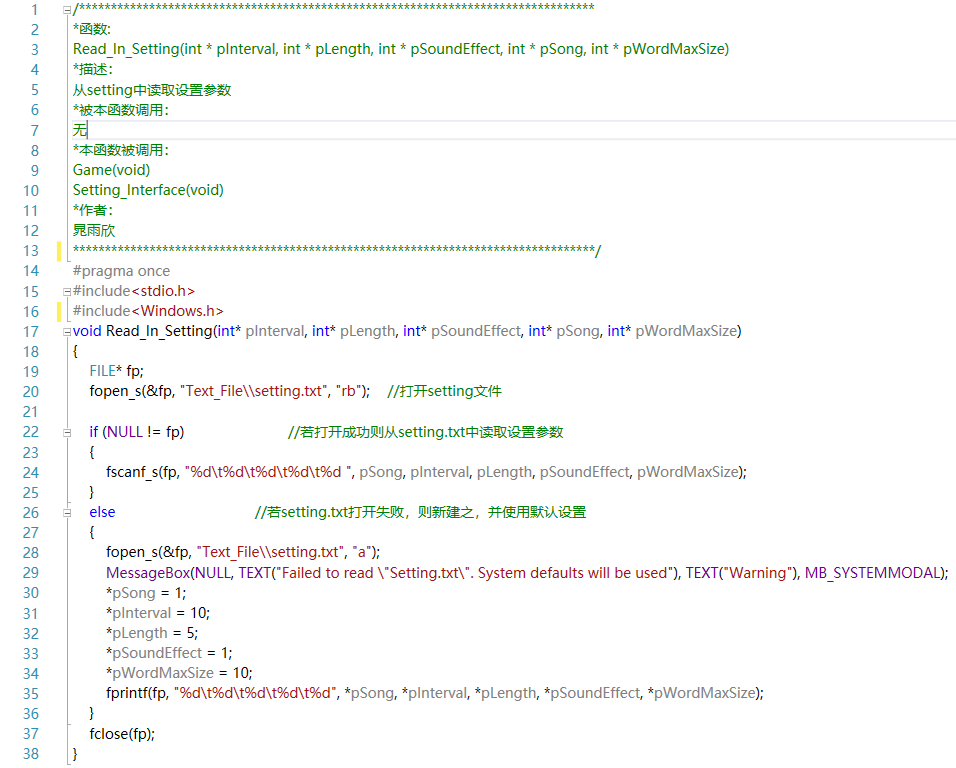


## 7.2 Detail Optimizing

### 7.2.1 Robustness

Even some file was missing for unknown reasons, the program is also able to run and fix the problem automatically.

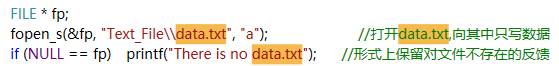
**7.2.1.1 Missing setting.txt**

If the program is failed to open setting.txt at a given position, it will create a new one and write in some default parameters as well as pops up a prompt window.

**7.1.1.2 Missing data.txt**

The program open data.txt in the ways of “r” and “a”. so even this file is missing creating a blank one is OK.





### 7.2.2 Editable Lexicon

For the form of wordlist.txt is fixed, user can edit the lexicon by his own will. He can deleted or add words which are required to divided by ‘\n’ or key “Enter”.

# 8 test

|  |  |  |  |
| --- | --- | --- | --- |
| INTERFACE | INPUT | EXPECT RESULT | RESULT |
| Main Interface | Click on the “Setting” button(setting.txt not missing) | Enter the Setting interface | Enter the Setting interface |
| Click on the “Setting” button(setting.txt missing) | Enter the Setting interface,and pop out a warning message box | Enter the Setting interface,and pop out a warning message box |
| Click on the “Game” button(setting.txt not missing) | Enter the Game interface | Enter the Game interface |
| Click on the “Setting” button(setting.txt missing) | Enter the Game interface,and pop out a warning message box | Enter the Game interface,and pop out a warning message box |
| Click on the “Score” button | Enter the Score interface | Enter the Score interface |
| Setting Interface | Click on the blue circle which represents the second music | The triangle on the blue circle change into double lines | The triangle on the blue circle change into double lines |
| Click on a certain point on the first slider | The value of falling velocity change | The value of falling velocity change to “80” |
| Click on a certain point on the second slider | The state of sound effect changes to “open” | The state of sound effect changes to “open” |
| Click on a certain point on the third slider | The value of word number change | The value of word number change to “5” |
| Click on a certain point on the forth slider | The value of longest word length change | The value of longest word length change to “10” |
| Click on the return button | Return to main interface | Return to main interface |
| Game Interface | Press “a”/ ”1”/ ”?” before the game start | The game start with the first apple falling down | The game start with the first apple falling down |
| Press “r” when the word on the apple is “ragged”(The input box is empty) | The character “r” appear in the input box at the bottom of window | The character “r” appear in the input box at the bottom of window |
| Press “d” when the word on the apple is “ragged”(The input box is “ragge”) | Apple with “ragged” disappears before a new apple appears, and the input box is cleared | Apple with “ragged” disappears before a new apple appears, and the input box is cleared |
| Press ESC(before the game) | Return to main interface | Return to main interface |
| Press “m” when the word on the apple is “condition”(The input box is “co”) | The computer generates a beep(sound effect),the content of input box remains the same | The computer generates a beep(sound effect),the content of input box remains the same |
| Press ESC(during the game) | Return to main interface | Return to main interface |
| Fail to type “condition” before the apple falls to the ground(The input box is “co”) | Apple with “condition” disappears before a new apple appears, and the input box is cleared | Apple with “condition” disappears before a new apple appears, and the input box is cleared |
| The game comes to an end(The user rescued 3 apples out of 5) | The result of game “Your accuracy is 60.00%,good job!” shows | The result of game “Your accuracy is 60.00%,good job!” shows |
| Score Interface | Enter the Score interface | A new point appears on the first chart | A new point appears on the first chart(10,25) |
| Click on the screen | Switch to the second chart with a new point | Switch to the second chart with a new point(10,60%) |

# 9 Result & Conclusion

In general, our game reaches our goal and can run smoothly. Except some details, our game basically meets the requirements of our original design. It overcomes the deficiency of most typing practice games. Users can not only change the game condition, but also get the feedback from the game. It can also test the users’ typing skills and find out what kind of word are usually typed mistakenly and give the users corresponding exercise. So it means our game can provide abundant game experience.

But there are still some details needs to be improved. One of which is the flicker of the falling apple in game interface, till now, we still can’t figure out why the apples are sometimes flashing and sometimes not flashing. What’s more, in the game interface, only one apple falls at a time now, to make our game more functional, realizing many apples falls at a time is essential.

Finally thanks to my group members, without whom, the game can’t be made. Thanks to professor Xu Hao and professor AT, without them, we wouldn’t have such a chance to anneal and improve ourselves. Thanks to everyone who helped us make our game. Thank you!