Yuk Chun Yin 6D 26

2020 HKDSE ICT SBA

Software Development: Documentation

| Objectives

* A clear output format that can be easily controlled by user (Path Navigation)
* Convenient for using and reading (Readability)
* The instruction should be precise and concise (Correct Wording)
* Handle all of the errors (Debugging)
* Short/Clear instruction

| Conception of doing the program

* Designing the UI (Should be clear)
* Function (All of the requirements)
* User-Friendly

| Preliminary Investigation

The system request is developing a text analyzer.

So, we need to collect requirements.

The basic function of the system:

* Extract some single words from a selectable text passage / many passages in a folder.
* Search for a given keyword / phrase in a passage(s).
* Replace the searched keyword / phrase in a passage(s).
* Count the occurrences of all the words or given word /phrase.
* Check the spellings of all the words in a passage(s).

Due to the requirements is given, there is no need to collect additional information through Interviews, Questionnaires, Documentation Reviews, Observation etc…

| Feasibility Study

Technical Feasibility

* Contain small size of spaces:

In this software, there is no need to store any text or file because the main idea of this software is analyzing the text which the user inputted, so it is not necessary to build a database or server and the size of problem will be much smaller. Not too much human resources is needed. Simply 1 to 2 people can finish all the works.

In conclusion, the possibility of developing the software is high.

* Only basic programming concepts needed

Just only know a little bit of programming language then can perform the algorithm design

* Conclude small amount of personnel

Not too much Work loading

The system only required small amount of functions.

There are no database, so database administrator not needed

There are no output device except the monintor like (printers , network equipment ,server),so technician not needed

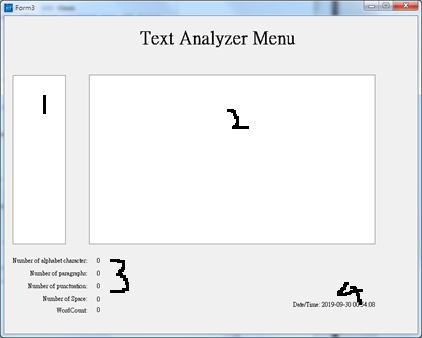
 There is only a software/application only , doesn’t have another facilities,so computer operator not needed  
   
   
Economic Feasibility

Not a product for commercial or any organization

Open Source Software / Freeware🡪 no benefits in economic way

In conclusion , not feasible economically.

| System Design – Interface of the system



① The place of displaying the filename , filecount.

② Displaying the content of text file

③ Displaying the status/information of the text file

④ Date

Consistency of color scheme

Color of the background: Gray

Color of the text of labels,datetime: Black

Color of the listbox: White

As the color of those color mentioned in below is very similar, so it have a better and clear User Interface for user to see or focus at a long period of time and it also won’t damage the eye severely.

The placement of different elements on a User Interface:

In Hong Kong , most of people are reading from left to right ,so all of the text displayed in the system will be reading from left to right for fitting the user’s habit.

Alignment of fields.

The system adapted right alignment for the text shown in the Interface ,so it have a consistency of alignment for user to have a better readability.

Usage of control objects:

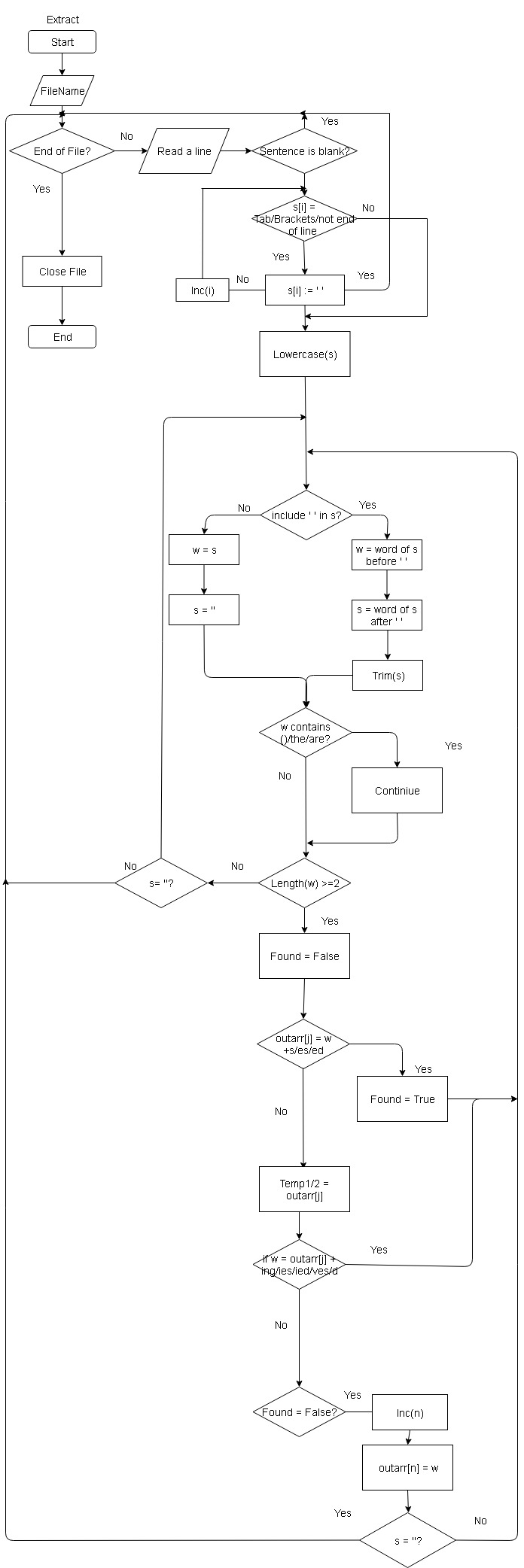
In the system, there are have many kinds of object for user to use, such as LIstboxs,labels,buttons…etc. It built up a user-friendly interface for user because the system will look redundant for only 1 kind of object.

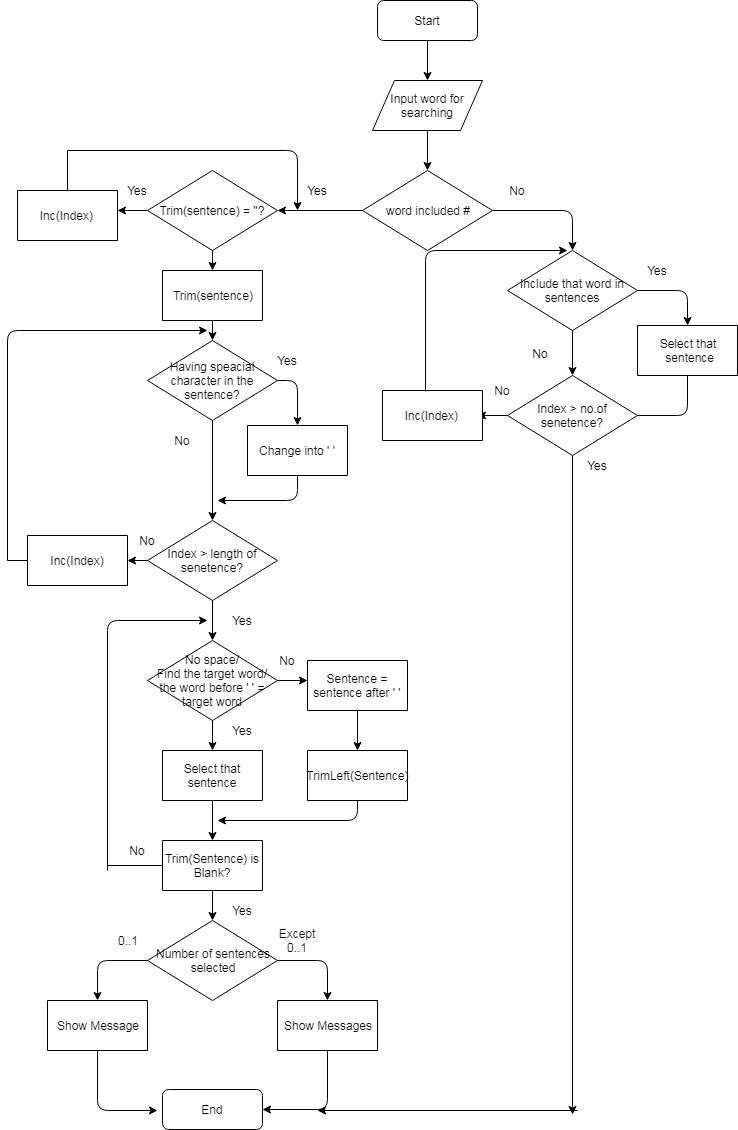
Wording of the system:

Not only the labels in the interface , but also the buttons are maintaining a concise wording and show the usefulness of those function in a simplest way.

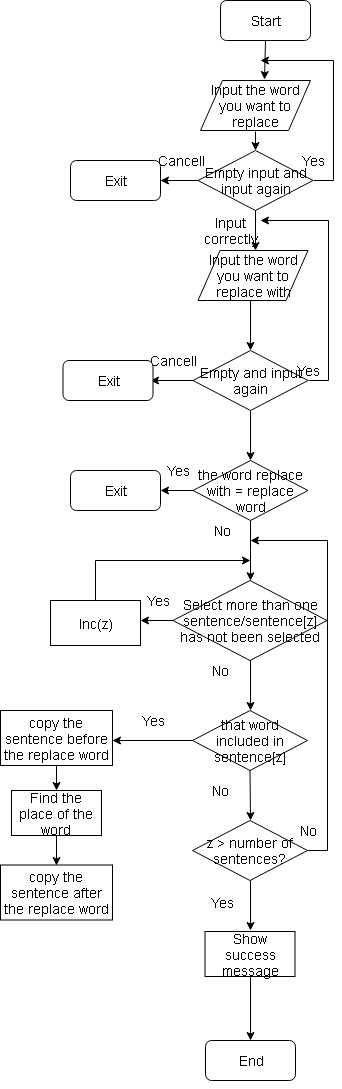
| System Design – Flowchart

Extract:

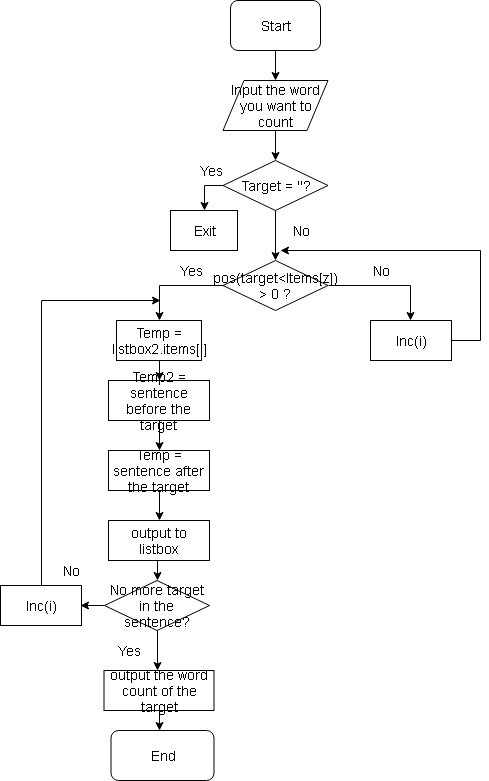


Search:

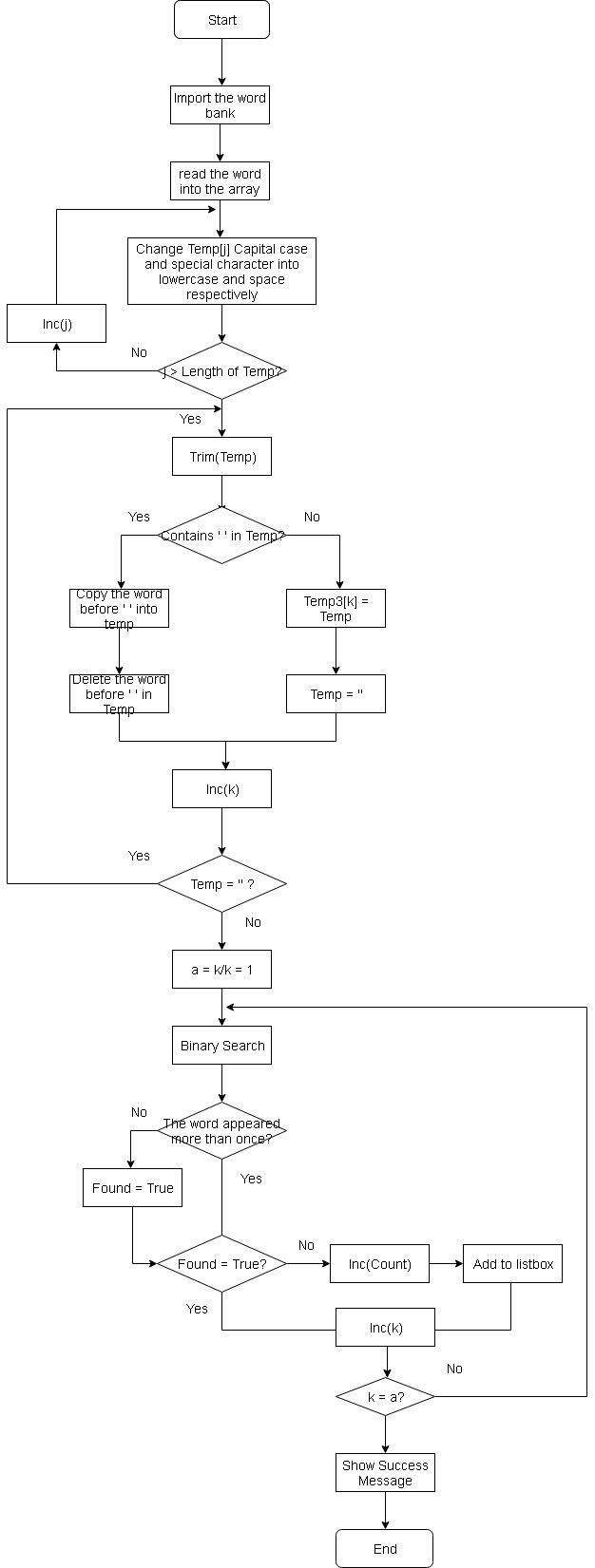
Replace:



SpellCount:



SpellCheck:



|System Testing (Extract)

Extract Dry Run:

Here is the pseudocode of Extract:

10 WHILE not eof(file) DO

20 Input(file,sentence)

30 IF Trim(sentence) = Blank THEN

40 Continue

50 FOR i := 1 to Length(Sentence) do

60 IF (Sentence[i] = Tab) or (Sentence[i] = Special Character) THEN

70 Sentence[i] <-- Space

80 END IF

90 END FOR

100 Sentence <-- LowerCase(Sentence)

110 REPEAT

120 IF Sentence included ' ' THEN

130 w <-- the string before ' '

140 Sentence <-- the string after ' '

150 Sentence <-- TrimLeft(Sentence)

160 ELSE

170 w <-- Sentence

180 Sentence <-- ' '

190 END IF

200 IF w equal to '()' or The or the or are or Are THEN

210 Continue

220 QuickSort // Trace it in another dry run

230 BinarySearch // Trace it in another dry run

240 IF the word is not exist THEN

250 Length(Outarr) <-- Length(Outarr) + 1 // outarr is an array

260 Outarr[Last Item] <-- W

270 END IF

280 UNTIL Sentence = ' '

290 END WHILE

We will trace the program part by part.

First,

50 FOR i := 1 to Length(Sentence) do

60 IF (Sentence[i] = Tab) or (Sentence[i] = Special Character) THEN

70 Sentence[i] <-- Space

80 END IF

90 END FOR

Case 1 : Let “S+c-h\*o/o l” be the test data.

Expected result should be ‘S c h o o l’.

|  |  |  |
| --- | --- | --- |
| Times of running in Line 20 | (Sentence[i] = Tab) or (Sentence[i] = Special Character) | Sentence[i] |
| 0 | Undefined | S+c-h\*o/o l |
| 1 | No | S+c-h\*o/o l |
| 2 | Yes | S c-h\*o/o l |
| 3 | No | S c-h\*o/o l |
| 4 | Yes | S c h\*o/o l |
| 5 | No | S c h\*o/o l |
| 6 | Yes | S c h o/o l |
| 7 | No | S c h o/o l |
| 8 | Yes | S c h o o l |
| 9 | No | S c h o o l |
| 10 | Yes | S c h o o l |

Another test data also prepared to check the accuracy of this algorithm.

|  |  |  |  |
| --- | --- | --- | --- |
| Test Data  (Valid Data) | Sentence[i] | | |
| Test Result | Expected Result | Correct |
| I go to school by bus | I go to school by bus | I go to school by bus | Yes |
| IWeYouThey | IWeYouThey | IWeYouThey | Yes |
| +-\*/+-\*/+-\*/ | (Twelve of Space) | (Twelve of Space) | Yes |
| abcdefg+--+gfedcba | abcdefg gfedcba | abcdefg gfedcba | Yes |

As the program handled all of the string, so there are only integer/real/ Boolean data type become the invalid data.

There are no boundary case/critical values ,so there is no needed to be tested.

Next,

110 Repeat

120 IF Sentence included ' ' THEN

130 w <-- the string before ' '

140 Sentence <-- the string after ' '

150 Sentence <-- TrimLeft(Sentence)

160 ELSE

170 w <-- Sentence

180 Sentence <-- ' '

190 END IF

…………

280 Until Sentence = ‘’

Case1 :Let sentence 🡨 ‘ I go to school by bus’

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the program | w <-- the string before ' ' | Sentence <-- the string after ' ' | Sentence <-- TrimLeft(Sentence) |
| 0 | Undefined | Undefined | Undefined |
| 1 | I | ‘ go to school by bus’ | ‘go to school by bus’ |
| 2 | Go | ‘ to school by bus’ | ‘to school by bus’ |
| 3 | To | ‘ school by bus’ | ‘school by bus’ |
| 4 | School | ‘ by bus’ | ‘by bus’ |
| 5 | By | ‘ bus’ | ‘bus’ |
| 6 | Bus | ‘’ | ‘’ |

Case 2 :Sentence🡨‘igotoschoolbybus’

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the program | w <-- the string before ' ' | Sentence <-- the string after ' ' | Sentence <-- TrimLeft(Sentence) |
| 0 | Undefined | Undefined | Undefined |
| 1 | igotoschoolbybus | ‘’ | ‘’ |

Case 3 : Sentence 🡨 ‘’

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the program | w <-- the string before ' ' | Sentence <-- the string after ' ' | Sentence <-- TrimLeft(Sentence) |
| 0 | Undefined | Undefined | Undefined |
| 1 | ‘’ | ‘’ | ‘’ |

Thirdly,

200 IF w equal to '()' or The or the or are or Are THEN

210 Continue

|  |  |  |  |
| --- | --- | --- | --- |
| Test Data  (Valid Data) | w | | |
| Test Result | Expected Result | Correct |
| ((((()))) | Exit | Exit | Yes |
| Fsfsfsf | Continue | Continue | Yes |
| TheareerahehT | Continue | Continue | Yes |
| are | Exit | Exit | Yes |
| #$%^ | Continue | Continue | Yes |

Fourthly,

QuickSort

10 i := Top;

20 j := Bottom;

30 k := outarr[(Top + Bottom) div 2];

40 repeat

50 While k > outarr[i] do

60 inc(i);

70 While k < outarr[j] do

80 dec(j);

90 if i <= j then

100 begin

110 Temp := outarr[i];

120 outarr[i] := outarr[j]; // Swapping

130 outarr[j] := Temp;

140 dec(j);

150 inc(i);

160 end;

170 until i > j;

180 if Top < j then

190 Quicksort(Top,j);

200 if i < Bottom then

210 QuickSort(i,Bottom);

Suppose

Test data/outarr 🡨[‘I ’,’go’ ,’to ‘,’ school ’,’ by ’,’ bus ’] // I go to school by bus

10 i := Top;

20 j := Bottom;

30 k := outarr[(Top + Bottom) div 2];

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the subprogram | i := Top | J := Bottom | k := outarr[(Top + Bottom) div 2]; |
| 0 | Undefined | Undefined | Undefined |
| 1 | 0 | 5 | To(outarr[2]) |

40 repeat

50 While k > outarr[i] do

60 inc(i);

…………

170 Until I <j

|  |  |  |
| --- | --- | --- |
| Times of running the subprogram | K > outarr[i] | I |
| 0 | Undefined | 0 |
| 1 | Yes (‘To’ >’ I’) | 1 |
| 2 | Yes (‘To’ > ‘go’) | 2 |
| 3 | No (‘To’ = ‘go’) | 2 |

40 repeat

………..

70 While k < outarr[j] do

80 dec(j); I go to school by bus

………..

170 Until I < j

|  |  |  |
| --- | --- | --- |
| Times of running the subprogram | K < outarr[j] | j |
| 0 | Undefined | 5 |
| 1 | No (‘To’ >’ bus’) | 5 |

Fifthly,

40 repeat

50 While k > outarr[i] do

60 inc(i);

70 While k < outarr[j] do

80 dec(j);

90 if i <= j then

100 begin

110 Temp := outarr[i];

120 outarr[i] := outarr[j]; // Swapping

130 outarr[j] := Temp;

140 dec(j);

150 inc(i);

160 end;

170 Until I <j

Suppose

Test data/outarr 🡨[‘I ’,’go’ ,’to ‘,’ school ’,’ by ’,’ bus ’] // I go to school by bus

The expected result is [‘bus ’,’by’ ,’go ‘,’ i ’,’ school ’,’ tp ’] // bus by go I school to

Trace Table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Times of running the subprogram | I <= j | Top < j | I < Bottom | i | j | Outarr |
| 0 | Undefined | N/A | N/A | 2 | 5 | // I go to school by bus |
| 1 | Yes(2<5) | N/A | N/A | 5 | 4 | // I go to bus by school |
| 2 | No(5 > 4) | Yes | N/A | 1 | 1 | // bus go I school by to |
| 3 | No(1 = 1) | N/A | N/A | 1 | 0 | // bus go I school by to |
| 4 | Yes(0 < 1) | No | Yes | 3 | 2 | // bus go by school I to |
| 5 | No(3 > 2) | Yes | N/A | 4 | 3 | // bus by go school I to |
| 6 | No (4 >3) | No | No | 4 | 3 | // bus by go I school to |

Sixthly, Binary Search

10 H := 0;

20 L := Length(outarr) - 1;

30 Found := False;

40 repeat

50 M := (H + L) div 2;

60 if w > outarr[M] then // Binary Search

70 H := M + 1

80 else if w < outarr[M] then

90 L := M - 1

100 else

110 Found := True;

120 until (H > L) or Found;

Suppose the w (’gogogo’ Outarr([’bussub’ ,’byyb’ , ’gogogo’ , ’gowentgone’,‘HiBye’,’schoolschool’,’totototototo’,]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Times of running the subprogram | H | M | L | Outarr[M] | W > outarr[M] | Found |
| 0 | Undefined | Undefined | Undefined | Undefined | Undefined | Undefined |
| 1 | 0 | 3 | 6 | gowentgone | No | False |
| 2 | 0 | 2 | 4 | gogogo | No | True |

|System Testing (Search)

PseudoCode

10 Count ( 0

20 Form3.Listbox2.Multiselect ( True

30 Form3.Listbox1.MultiSelect ( False

40 Input Word

50 if Word contains # in the beginning then

60 for j ( 0 to Form3.Listbox2.Items.Count - 1

70 Temp ( Form3.Listbox2.Items[j]

80 repeat

90 if Temp contains Word then

100 Count ( Count + 1

110 Delete(Temp,1,pos(Word,Temp) + Length(Word))

120 Form3.Listbox2.Selected[j] ( True

130 End if

140 Until Temp doesn’t contain Word

150 End Repeat

160 End For

170 else

180 Delete the first character of Word

190 For j ( 0 to Form3.Listbox2.Items[j] do

200 if Temp doesn’t contain Word then

210 Continue

220 End if

230 Temp ( Trim(Form3.Listbox2.Items[j]) + ‘ ’

240 For I ( 1 to Length(Temp) do

250 if Temp[i] contains special character then

260 Temp[i] ( ‘’

270 End if

275 End for

280 if Last character of Temp <> ‘ ‘ then

290 Temp ( Temp + ‘ ‘

300 End if

310 repeat

320 if (Temp[pos(Word,Temp) - 1] = ' ') and (Temp[pos(Word,Temp) + Length(Word)] = ' ') then

330 Count ( Count + 1

340 Form3.Listbox2.Selected[j] ( True

350 End if

360 Delete(Temp , pos(Word,Temp) + Length(Word))

370 Until Temp doesn’t contain Word

380 End Repeat

385 End for

390 if Count = 0 or 1 then

400 ShowMessage(No results has found)

410 Form3.Listbox2.MultiSelect ( False

420 else

430 ShowMessage(Count + ‘Results have found’)

440 End if

We will trace the program partly

First,

60 for j ( 0 to Form3.Listbox2.Items.Count - 1

70 Temp ( Form3.Listbox2.Items[j]

80 repeat

90 if pos(Word,Temp) > 0 then

100 Count ( Count + 1

110 Delete(Temp,1,pos(Word,Temp) + Length(Word))

120 Form3.Listbox2.Selected[j] ( True

130 End if

140 Until pos(Word,Temp) = 0

150 End Repeat

160 End For

Case 1: Let Word ( o

Temp ( I go to school by bus

|  |  |  |
| --- | --- | --- |
| Times of running the program | Count := Count + 1 | Temp |
| 0 | 0 | I go to school by bus |
| 1 | 1 | To school by bus |
| 2 | 2 | School by bus |
| 3 | 3 | Ol by bus |
| 4 | 4 | L by bus |
| 5 | 4 | L by bus |

Second,

180 Delete the first character of Word

190 For j ( 0 to Form3.Listbox2.Items[j] do

200 if Temp doesn’t contain Word then

210 Continue

220 End if

Case 1 (Valid): Let Temp (’school school school bus bus bus’

Word (’#afl;jk’

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the subprogram | Word | Pos(Word,Temp) | Exit? |
| 0 | #Afl;jk | Undefined | Undefined |
| 1 | Afl;jk | 0 | Exit |

Case 2 (Valid): Let Temp (’school school school bus bus bus

Sine Cosine Tangent’

Word (’Sine’

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the subprogram | Word | Pos(Word,Temp) | Exit? |
| 0 | Sine | Undefined | Undefined |
| 1 | Sine | 0 | Exit |
| 2 | Sine | 1 | Continue |

Third

240 For I ( 1 to Length(Temp) do

250 if Temp[i] contains special character then

260 Temp[i] ( ‘’

270 End if

Case 1 : Let Temp ( ‘x/\*/\*-/+2@$#’

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the subprogram | Temp[i] | ExecRegExpr(‘\W’,Temp[i]) | Temp |
| 0 | Undefined | Undefined | ‘x/\*/\*-/+2@$#’ |
| 1 | X | No | ‘x/\*/\*-/+2@$#’ |
| 2 | / | Yes | ‘x \*/\*-/+2@$#’ |
| 3 | \* | Yes | ‘x /\*-/+2@$#’ |
| 4 | / | Yes | ‘x \*-/+2@$#’ |
| 5 | \* | Yes | ‘x -/+2@$#’ |
| 6 | - | Yes | ‘x /+2@$#’ |
| 7 | / | Yes | ‘x +2@$#’ |
| 8 | + | Yes | ‘x 2@$#’ |
| 9 | 2 | No | ‘x 2@$#’ |
| 10 | @ | Yes | ‘x 2 $#’ |
| 11 | $ | Yes | ‘x 2 #’ |
| 12 | # | Yes | ‘x 2 ’ |

280 if Last character of Temp <> ‘ ‘ then

290 Temp ( Temp + ‘ ‘

Test Case 1: Temp (’sinep’

|  |  |  |
| --- | --- | --- |
| Times of running the subprogram | Temp[Last] = ‘ ’? | Temp |
| 0 | Undefined | ‘sinep’ |
| 1 | No | ‘sinep ’ |

Test Case 2: Temp (’leechunkit ’

|  |  |  |
| --- | --- | --- |
| Times of running the subprogram | Temp[Last] = ‘ ’? | Temp |
| 0 | Undefined | ’leechunkit ’ |
| 1 | Yes | ’leechunkit ’ |

310 repeat

320 if (Temp[pos(Word,Temp) - 1] = ' ') and (Temp[pos(Word,Temp) + Length(Word)] = ' ') then

330 Count ( Count + 1

340 Form3.Listbox2.Selected[j] ( True

350 End if

360 Delete(Temp , pos(Word,Temp) + Length(Word))

370 Until Temp doesn’t contain Word

Test Case 1: Let Temp (’ 學而時習之 不亦說乎 ’

Word (’ 不亦說乎 ’

|  |  |  |
| --- | --- | --- |
| Times of running the subprogram | (Temp[pos(Word,Temp) - 1] = ' ') and (Temp[pos(Word,Temp) + Length(Word)] = ' ') | Delete(Temp , pos(Word,Temp) + Length(Word)) |
| 0 | Undefined | Undefined |
| 1 | Yes | ’學而時習之’ |

|System Testing (Replace)

10 Input str

20 If str = ‘’ then

30 Exit

40 End if

50 Input Replace\_Word

60 If Replace\_Word = ‘’ then

70 Repeat

80 Tell the user he doesn’t input the word

90 Yes : Enter again

100 No:Exit

110 Cancel:Exit

120 Until Replace\_Word is not blank

130 End Repeat

140 End if

150 If Replace\_Word = str then

160 ShowMessage(inputted the same word)

170 ShowMessage(Try Again)

180 End if

190 For z ( 0 to Form3.Listbox2.Count – 1do

200 Temp ( Form3.Listbox2.Items[z]

210 If Form3.Listbox2.SelCount > 1 and Form3.Listbox2.Selected[z] = False then

220 Continue

230 End if

240 If Form3.Listbox2.Items[z] included str then

250 Temp3 ( ‘’

260 Temp2 ( Form3.Listbox2.Items[z]

270 Repeat

280 Temp ( copy(Temp2, 1,pos(str,temp2) - 1)

290 Temp2 (copy(Temp2,pos(str,Temp2) + Legnth(str),Length(Temp2))

300 Temp3 ( Temp3 + Temp + Replace\_Word

310 Until Temp2 doesn’t include str

320 End Repeat

330 End for

First,

20 If str = ‘’ then

30 Exit

Trace Table

|  |  |  |  |
| --- | --- | --- | --- |
| Test Data  (Valid Data) | str | | |
| Test Result | Expected Result | Correct |
| ‘fsfgsgsagas’ | Continue | Continue | Yes |
| ‘’ | Exit | Exit | Yes |
| ‘0.127896745’ | Continue | Continue | Yes |
| ‘fas358fgasf.352#%$’ | Continue | Continue | Yes |

Next,

50 Input Replace\_Word

60 If Replace\_Word = ‘’ then

70 Repeat

80 Tell the user he doesn’t input the word

90 Yes : Enter again

100 No:Exit

110 Cancel:Exit

120 Until Replace\_Word is not blank

130 End Repeat

140 End if

Test Case 1: Let Replace\_Word ( ‘’ &’’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Times of running the subprogram | Replace\_Word | Input again? | Replace\_Word  (second time) | Input again?  (second time) |
| 0 | Undefined | Undefined | N/A | N/A |
| 1 | ‘’ | Yes | ‘’ | Yes |

Test Case 2: Let Replace\_Word ( ‘I go to school by bus’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Times of running the subprogram | Replace\_Word | Input again? | Replace\_Word  (second time) | Input again?  (second time) |
| 0 | Undefined | Undefined | N/A | N/A |
| 1 | ‘I go to school by bus’ | No | N/A | N/A |

Test Case3 : Let Replace\_Word (’’ & ‘I go to school by bus’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Times of running the subprogram | Replace\_Word | Input again? | Replace\_Word  (second time) | Input again?  (second time) |
| 0 | Undefined | Undefined | N/A | N/A |
| 1 | ‘’ | Yes | ‘I go to school by bus’ | No |

Test Case 4 : Let Replace\_Word (’’ & ‘(Tab)’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Times of running the subprogram | Replace\_Word | Input again? | Replace\_Word  (second time) | Input again?  (second time) |
| 0 | Undefined | Undefined | N/A | N/A |
| 1 | ‘’ | Yes | ‘(Tab)’ | Yes |

Thirdly,

150 If Replace\_Word = str then

160 ShowMessage(inputted the same word)

170 ShowMessage(Try Again)

Valid test Case 1: Let Replace\_Word ( ‘’ hahaha’’

Str ( “ hehehe”

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the subprogram | Replace\_Word | str | ShowMessage? |
| 0 | Undefined | Undefined | N/A |
| 1 | hahaha | hehehe | No |

Valid Test Case 2: Let Replace\_Word ( ‘’ wryyyyyyy’’

Str ( “ wryyyyyyy”

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the subprogram | Replace\_Word | str | ShowMessage? |
| 0 | Undefined | Undefined | N/A |
| 1 | wryyyyyyy | wryyyyyyy | Yes |

Due to the input box can be inputted the string data type only , so there are no invalid data to check the accuracy of this subprogram.

After that ,

210 If Form3.Listbox2.SelCount > 1 and Form3.Listbox2.Selected[z] = False then

220 Continue

Test Case 1:

Let form3.Listbox2 ( ‘Thomas Edison was born on 11 February 1847. He was one of the outstanding geniuses of technology and he obtained patents for more than one thousand inventions including the electric light bulb, the record player and an early type of film projector. He also created the world's first industrial research laboratory’ (2 Lines Selected)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Times of running the subprogram | Items[i] | SelCount | Selected[z] | Exit? |
| 0 | 0 | 2 | Undefined | Undefined |
| 1 | 1 | 2 | False | Exit |
| 2 | 2 | 2 | True | Continue |
| 3 | 3 | 2 | False | Exit |
| 4 | 4 | 2 | True | Continue |

Test Case2:

Let form3.Listbox2 (’I go to school by bus

You go to school on foot

Aeiou

IYouWeThey ‘ (0 Line Selected)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Times of running the subprogram | Items[i] | SelCount | Selected[z] | Exit? |
| 0 | 0 | 0 | Undefined | Undefined |
| 1 | 1 | 0 | False | Continue |
| 2 | 2 | 0 | False | Continue |
| 3 | 3 | 0 | False | Continue |
| 4 | 4 | 0 | False | Continue |

270 Repeat

280 Temp ( copy(Temp2, 1,pos(str,temp2) - 1)

290 Temp2 (copy(Temp2,pos(str,Temp2) + Legnth(str),Length(Temp2))

300 Temp3 ( Temp3 + Temp + Replace\_Word

310 Until Temp2 doesn’t include str

Case 1: Let Temp2 ( ‘ you go to school on foot ‘

str (‘on foot’

Replace\_Word ( ‘ by bus’

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the table | Temp ( copy(Temp2, 1,pos(str,temp2) - 1) | Temp2 (copy(Temp2,pos(str,Temp2) + Length(str),Length(Temp2)) | Temp3 ( Temp3 + Temp + Replace\_Word |
| 0 | Undefined | You go to school on foot | ‘’ |
| 1 | You go to school | ‘’ | You go to school by bus |

Case2: Let Temp2 ( ‘go go go go go go go’

Str ( ‘go’

Replace\_Word (’bus’

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the subprogram | Temp ( copy(Temp2, 1,pos(str,temp2) - 1) | Temp2 (copy(Temp2,pos(str,Temp2) + Length(str),Length(Temp2)) | Temp3 ( Temp3 + Temp + Replace\_Word |
| 0 | Undefined | go go go go go go go’ | ‘’ |
| 1 | ‘’ | ‘go go go go go go’ | ‘bus’ |
| 2 | ‘’ | ‘go go go go go’ | ‘ bus bus’ |
| 3 | ‘’ | ‘go go go go’ | ‘bus bus bus’ |
| 4 | ‘’ | ‘go go go’ | ‘bus bus bus bus’ |
| 5 | ‘’ | ‘go go’ | ‘bus bus bus bus bus’ |
| 6 | ‘’ | ‘go’ | ‘bus bus bus bus bus bus’ |
| 7 | ‘’ | ‘’ | ‘bus bus bus bus bus bus bus’ |

In the following evaluation, ‘ I go to school by bus ‘will becomes the test data (Valid)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Str | Replace\_Word | Expected Result | Output Result | Correct |
| I go to | School by bus | School by bus school by bus | School by bus school by bus | Yes |
| Abccba | Abcabc | I go to school by bus | I go to school by bus | Yes |
| Sub yb loohcs ot og i | I go to school by bus | I go to school by bus | I go to school by bus | Yes |
| Bus | ####### | I go to school by ####### | I go to school by ####### | Yes |
| I bus | Bus | I go to school by bus | I go to school by bus | Yes |
| o | Jjj | I gjjj tjjj schjjjjjjl by bus | I gjjj tjjj schjjjjjjl by bus | Yes |
| I go to school | 我上學 | 我上學 by bus | 我上學 by bus | Yes |

|System Testing (SpellCount)

10 Input target

20 if Target is blank then

30 Exit

35 End if

40 Form5.Listbox1.Clear

50 for z ( 0 to Form3.Listbox2.Items.Count – 1 do

60 if Selection of Form3.Listbox2 > 1 and Form3.Listbox2.Selected[z] = False then

70 Temp ( Form3.Listbox2.Items[z]

80 Temp3 ( ‘’

90 repeat

100 Temp2 ( the string before target

110 Temp ( the string after target

120 Add the location of target

130 Until Temp doesn’t contain Target

140 End Repeat

150 End if

160 End for

170 Add the Count to the listbox

180 Show Form

Testing

20 if Target is blank then

30 Exit

Case 1:Let Target ( ‘gay’

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the program | Target | Blank? | Exit? |
| 0 | Undefined | Undefined | Undefined |
| 1 | gay | No | Continue |

Case 2:Let Target ( ‘’

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the program | Target | Blank? | Exit? |
| 0 | Undefined | Undefined | Undefined |
| 1 | ‘’ | Yes | Exit |

90 repeat

100 Temp2 ( the string before target

110 Temp ( the string after target

120 Add the location of target

130 Until Temp doesn’t contain Target

140 End Repeat

Case 1:Let Form3.Listbox2.Items (’I go to bus school bus by bus’

Target (’ bus’

|  |  |  |  |
| --- | --- | --- | --- |
| Times of running the program | Target | Temp | Temp2 |
| 0 | Undefined | Undefined | Undefined |
| 1 | Bus | I go to | School bus by bus |
| 2 | Bus | School | By bus |
| 3 | Bus | By | Bus |

|System Testing (SpellCheck)

PesudoCode

10 Form5.Listbox1.Clear

20 Length of WordBank ( 0

30 Length of Temp3 ( 1

40 assignfile(Bank, WordBank.txt)

50 reset(Bank)

60 While not eof(Bank) do

70 Length of WorkBank ( Length of WordBank + 1

80 read the vocab into the array

90 End While

100 CloseFile(Bank)

110 for I ( 0 to Form3.Listbox2.Items.Count -1 do

120 if Trim(form3.Listbox2.Items[i]) is not blank then

130 Temp ( LowerCase(Trim(Form3.Listbox2.Items[i]))

140 else

150 Continue

160 End if

170 for j ( 1 to Length(Temp) do

180 if Temp[j] contains special character then

190 Change into ‘ ‘

200 End if

210 repeat

220 Temp ( Trim(Temp)

230 if Temp contains space then

240 Last character of Temp3 ( copy(Temp,1,pos(‘ ‘ ,Temp) - 1)

250 Temp ( copy(Temp, pos(‘ ‘ ,Temp) + 1,Length(Temp))

260 else

270 Last term of Temp3 ( Temp

280 Temp ( ‘‘

290 End if

300 val(Last term of Temp3,Number ,Code)

310 if Code = 0 then

320 Last term of Temp3 ( ‘’

330 Continue

340 End if

350 if Last term of Temp3 <> ‘’ then

360 Length of Temp3 ( Length of Temp + 1

370 Until Temp = ‘’

380 End Repeat

390 Binary Search // Evaulate in the next dry run

400 for I ( 0 to Form5.Listbox1.Items.Count -1 do

410 if Form5.Listbox1.Items[i] = Temp3[k] then

420 Continue

430 End if

440 End For

450 if Found = False then

460 Add Temp3[k] into Form5.Listbox

470 End if

480 if ItemsCount of Form5.LIstbox1 = 1 then // 1 line is used to display the count

490 ShowMessage(No mistake)

500 else

510 ShowMessage(Done)

520 if ItemsCount of Form5.Listbox1 = 2

530 ShowMessage(Count + Mistake)

540 else

550 ShowMessage(Count + Mistakes)

560 End if

570 Display the Count in the listbox

580 Show the Form5

590 End if

Trace Table

We will trace the table partly

First,

110 for I ( 0 to Form3.Listbox2.Items.Count -1 do

120 if Trim(form3.Listbox2.Items[i]) is not blank then

130 Temp ( LowerCase(Trim(Form3.Listbox2.Items[i]))

140 else

150 Continue

So , in the latter evaluation , we will not discuss with the logic of grammar checking

Let the passage in the below become the test data in the listbox:(valid)

“

among the modules of the school attendance system we mentioned before ,

we can first test the attendance record searching module to see whether it works properly .

For example ,if we input the information about a student, such as his/her student ID,name or class ,

the corresponding student attendance records should be displayed .

Alternatively,we can input a certain date into the system and see

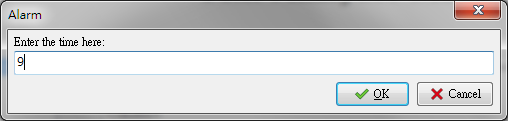
whether it displays the correct list of students who are late for school that day “

|  |  |  |
| --- | --- | --- |
| Times of running the program/i | Trim(form3.Listbox2.Items[i]) | LowerCase(Trim(Form3.Listbox2.Items[i])) |
| 0 | Undefined | Undefined |
| 1 | ‘among the modules of the school attendance system we mentioned before ,’ | ‘among the modules of the school attendance system we mentioned before ,’ |
| 2 | ‘we can first test the attendance record searching module to see whether it works properly .’ | ‘we can first test the attendance record searching module to see whether it works properly .’ |
| 3 | ‘For example ,if we input the information about a student, such as his/her student ID,name or class ,’ | ‘for example ,if we input the information about a student, such as his/her student id,name or class ,’ |
| 4 | ‘the corresponding student attendance records should be displayed .’ | ‘the corresponding student attendance records should be displayed .’ |
| 5 | ‘Alternatively,we can input a certain date into the system and see’ | ‘alternatively, we can input a certain date into the system and see’ |
| 6 | ‘’ | ‘’ |
| 7 | ‘whether it displays the correct list of students who are late for school that day’ | ‘whether it displays the correct list of students who are late for school that day' |

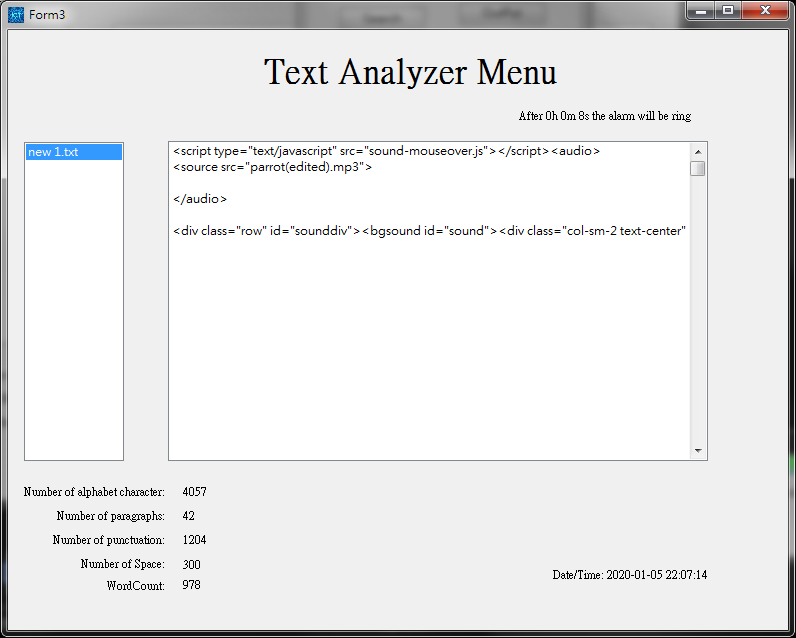
|System Testing (Additional Function)

In the following ,I will evaluate the usage of the alarm function and test whether it will output a proper message and function to the user.

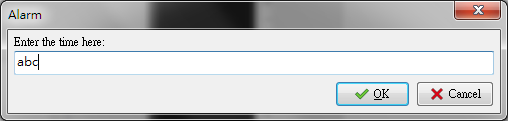
Test Case 1:Numeric Number (0-9)



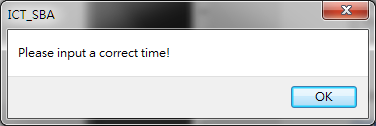
Outputted result:

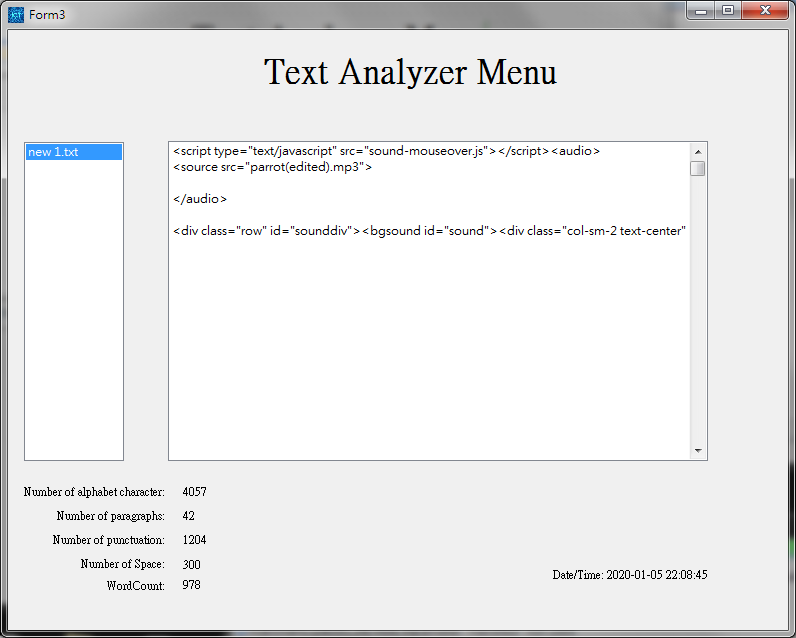


Test Case 2: alphabet



Outputted result:

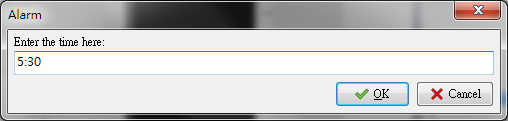




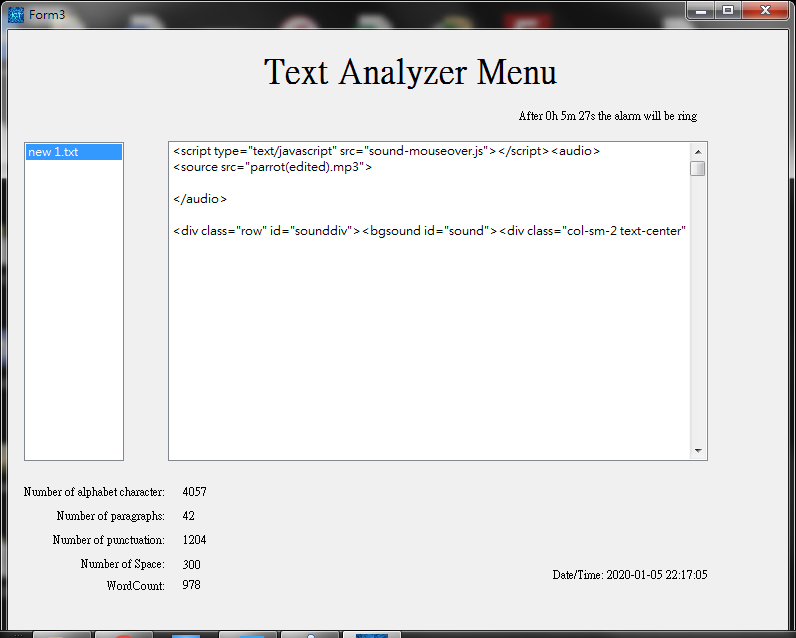
Nothing happened

Test Case 3: (Numeric Number)

Inputted value:

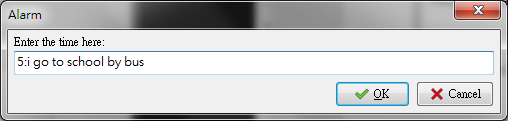


Outputted result

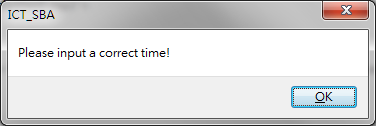


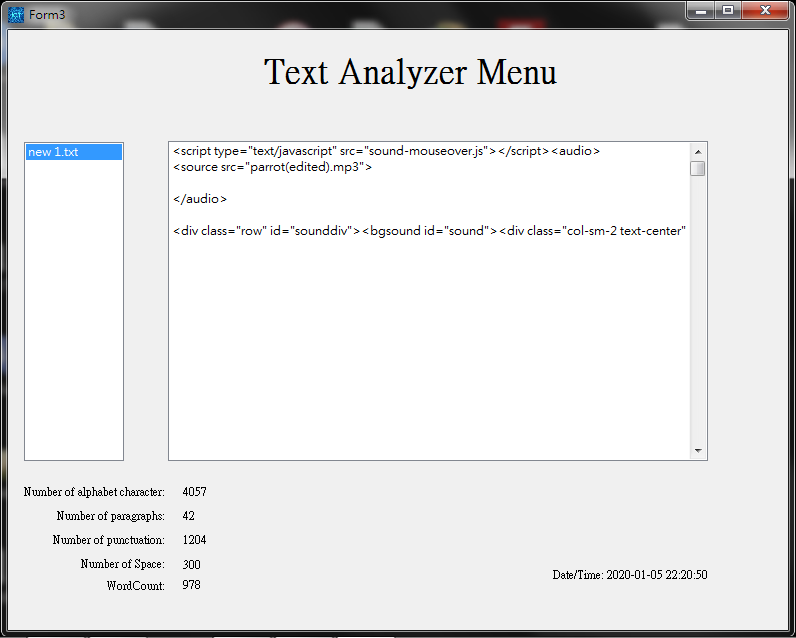
Test Case 4: (Numeric Number+ alphabet)

Inputted value:



Outputted result:

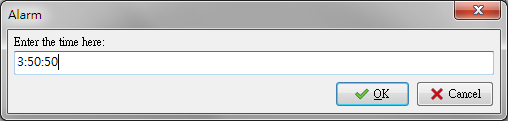




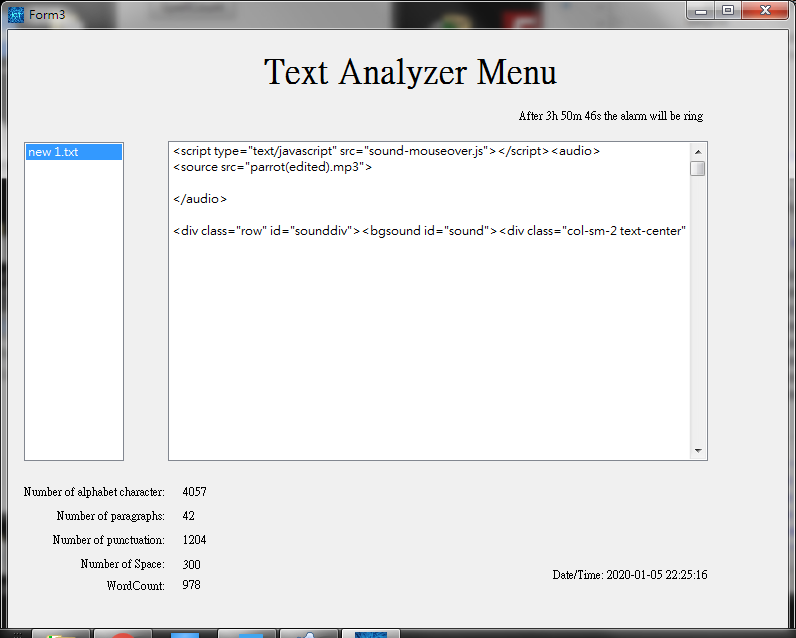
Nothing happened

Test Case 5 (Numeric Number)

Inputted value:

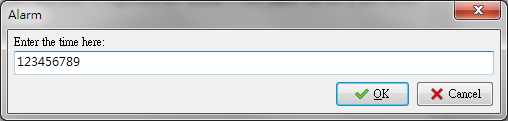


Outputted result:

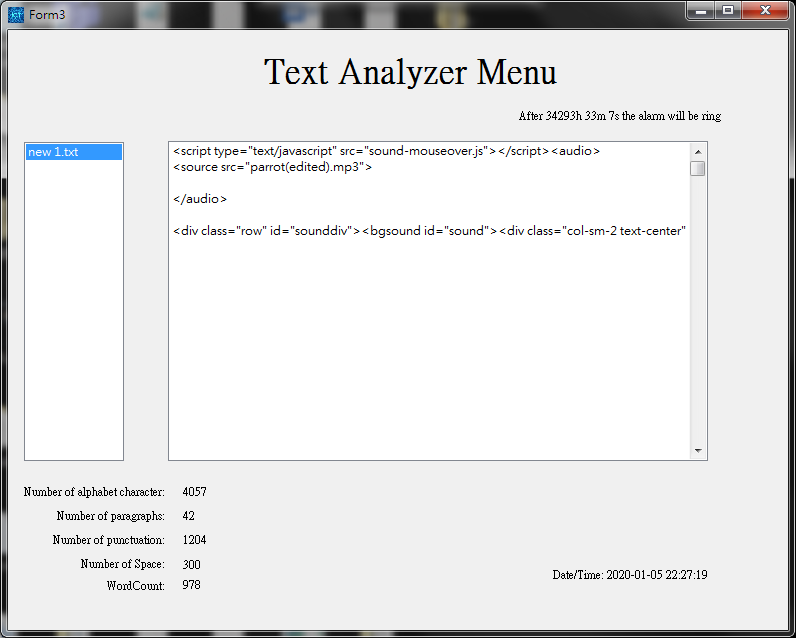


Test Case 6: (Numeric Number)

Inputted value:

e

Outputted result:



Hour:123456789/3600 = 34293…1989

Minute:1989/60 = 33…9

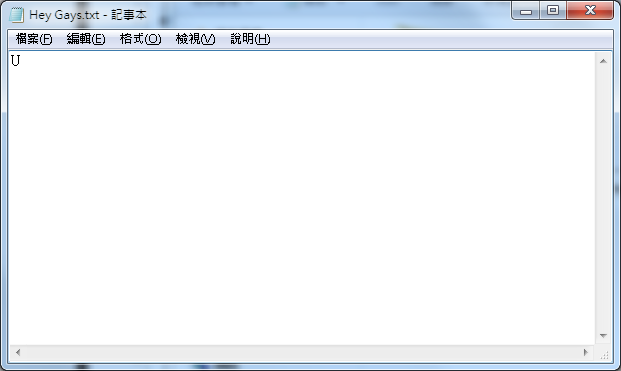
Result:

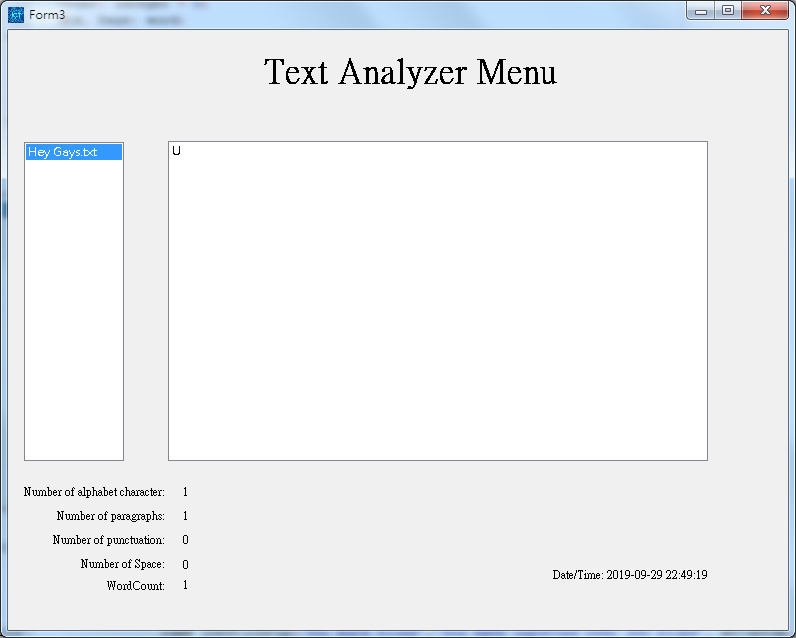
|  |  |  |  |
| --- | --- | --- | --- |
| Inputted Value | Outputted Result | Expected Result | Correct  (±5s error) |
| 9 | 7s | 9s | Yes |
| Abc | Invalid | Invalid | Yes |
| 5:30 | 5m27s | 5m30s | Yes |
| 5:I go to school by bus | Invalid | Invalid | Yes |
| 3:50:50 | 3h50m46s | 3h50m50s | Yes |
| 123456789 | 34293h33m7s | 34293h33m9s | Yes |

|System Testing (Performance Time Test)

In the following evaluation , we will determine the length of time used by the system to process data and the user will receive a reasonable time or not.

Test Case 1: 1 character only .

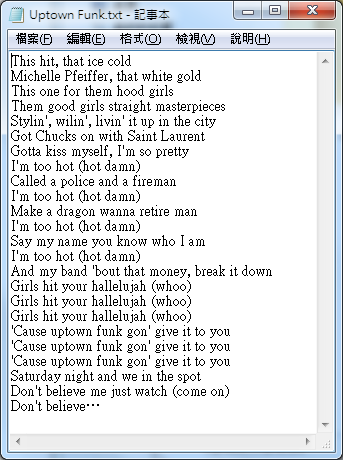


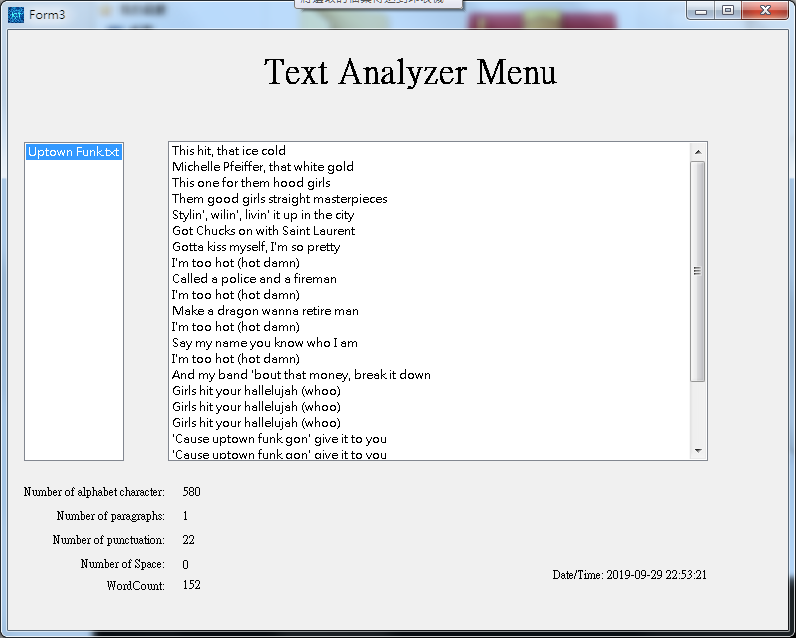


Required Time : <1 second.



Test Case 2: A passage .

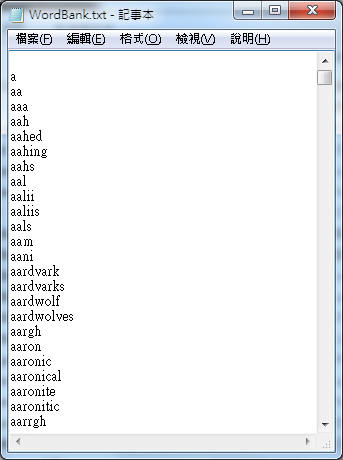




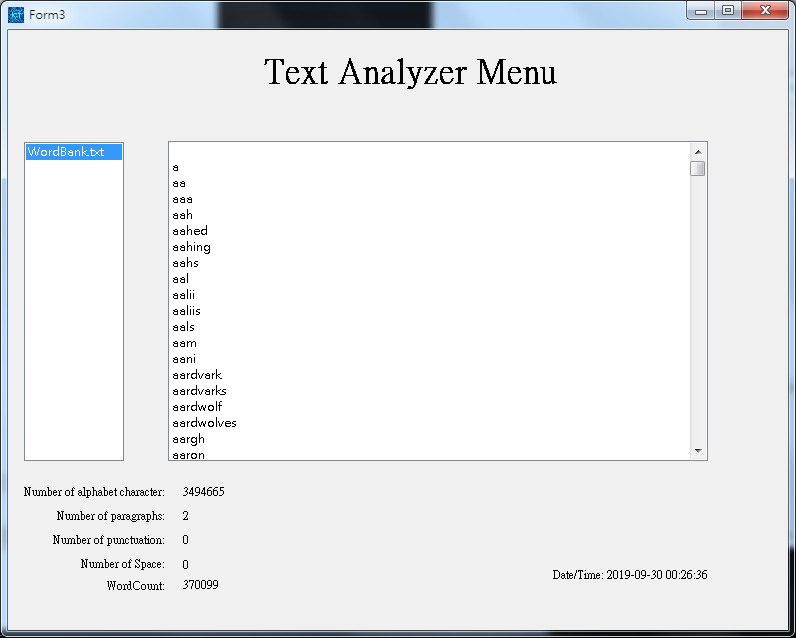
Required Time : <1 second.



Test Case 3: Dictionary



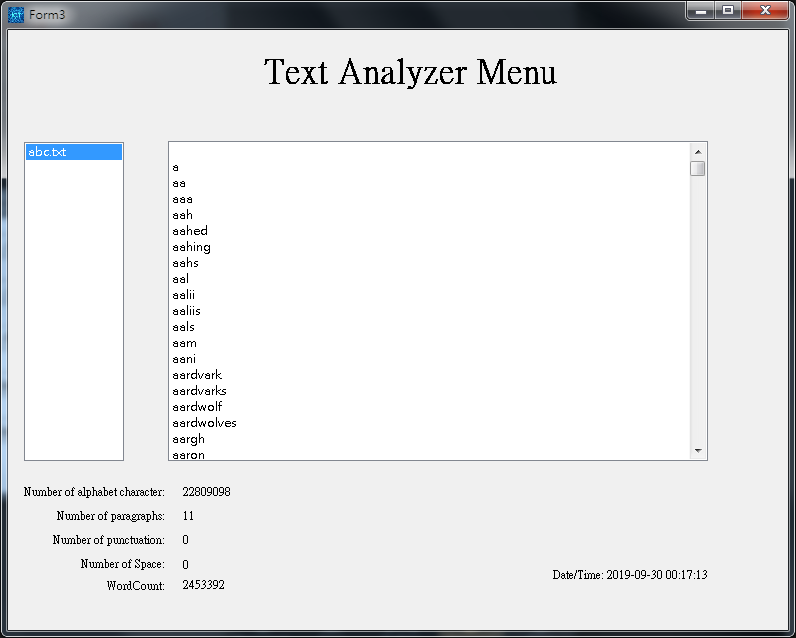
~200000 words dictionary





Required Time :11-12s

Test Case 4 : Word Dictionary x7



Required Time : around 6.30 minutes

Result:

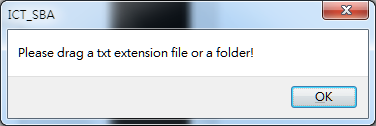
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case | Alphabet | Word | Time | Word/Time | Reasonable? |
| Character | 1 | 1 | <1s | A lot | Yes |
| Passage | 980 | 152 | <1s | A lot | Yes |
| Dictionary | 3494665 | 370099 | 11.30s | 32182word/s | Yes |
| 7 Dictionary | 22809098 | 2453392 | 6.30mins | 6291 word/s | No |

In a nutshell, the user should avoid inputting a bunch of file with many words , as it will increase the workload of the system drastically.

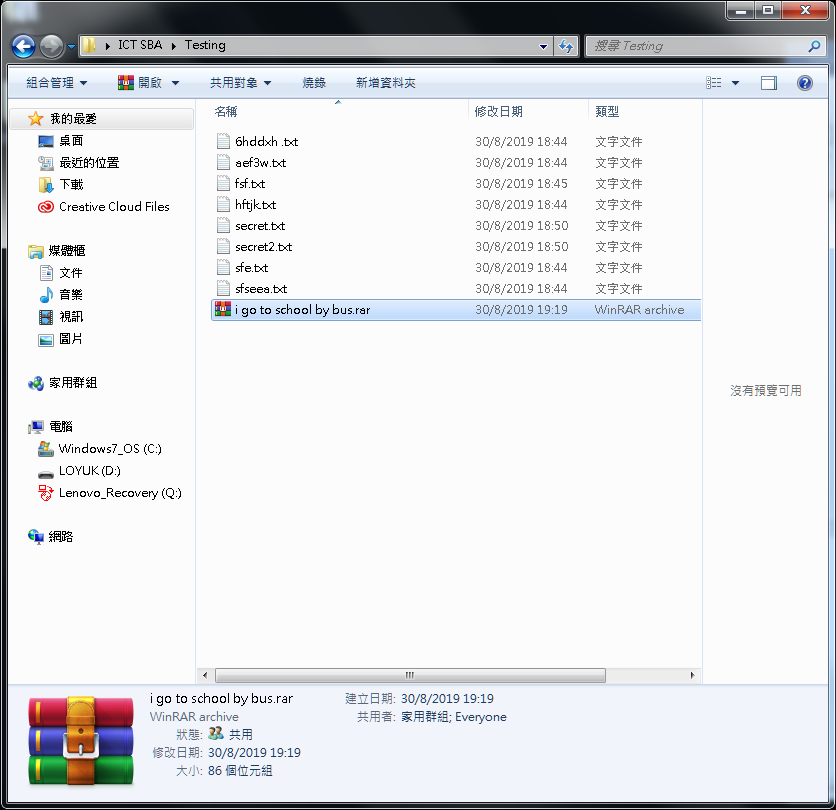
|Data Validation (Extract)

In this text analyzer, we can not drag or select the document which is not a txt file.

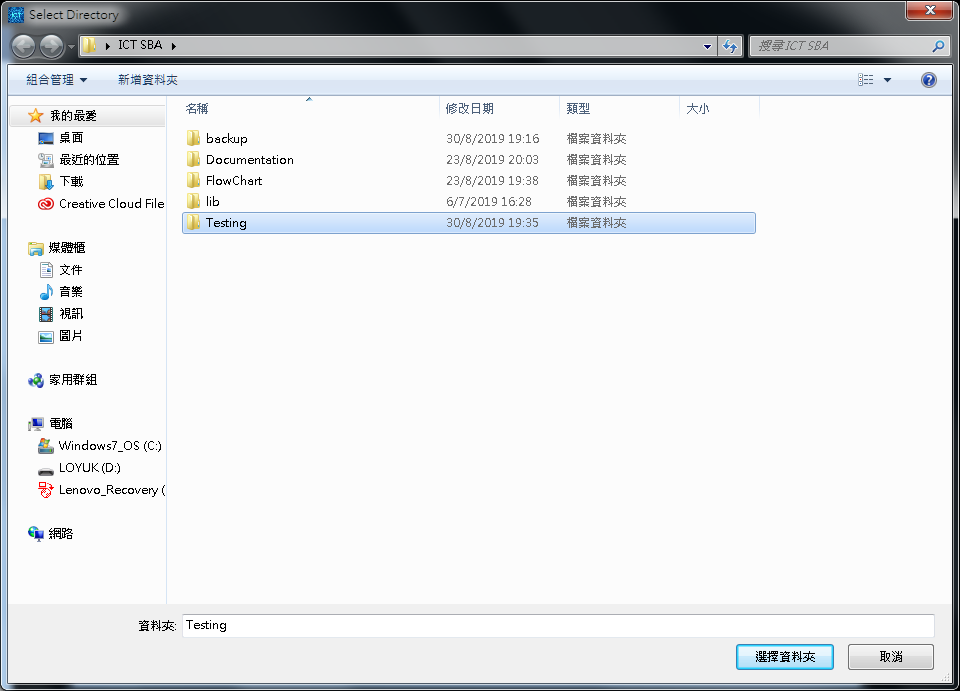
In order to avoid GIGO , when we inputted those invalid file , it will pop up a warning for inputting a wrong data.



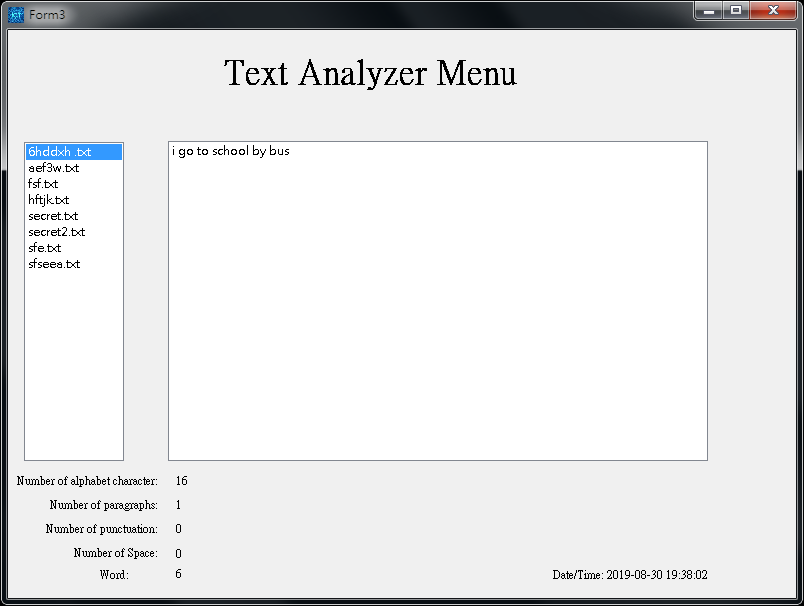
If a folder contains different of extension like “ exe ,docx ,zip ”, it will not go into the text analyzer or occur any errors.



Contains a rar file



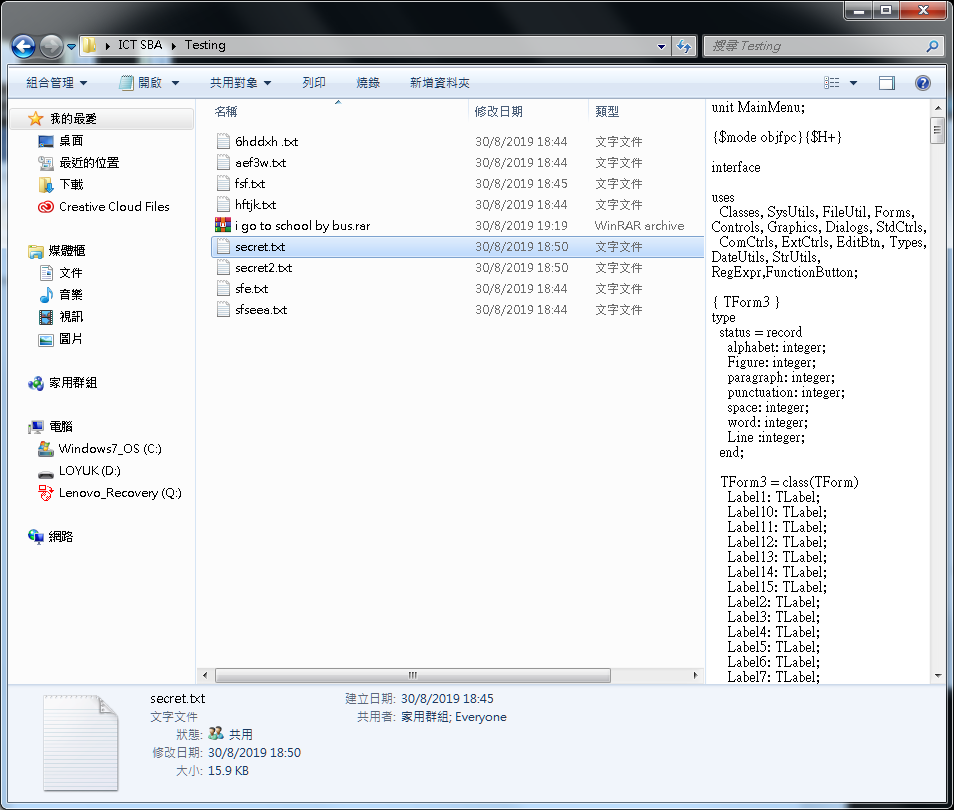
Selecting…



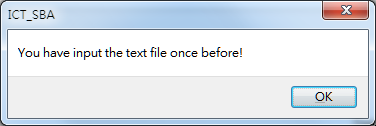
The rar file haven’t inputted in

Inputted Repeatedly

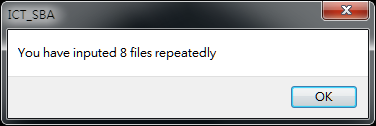
If user inputted a text file more than once, the analyzer will have some hints to tell user.



Select one of the file which is inputted in before

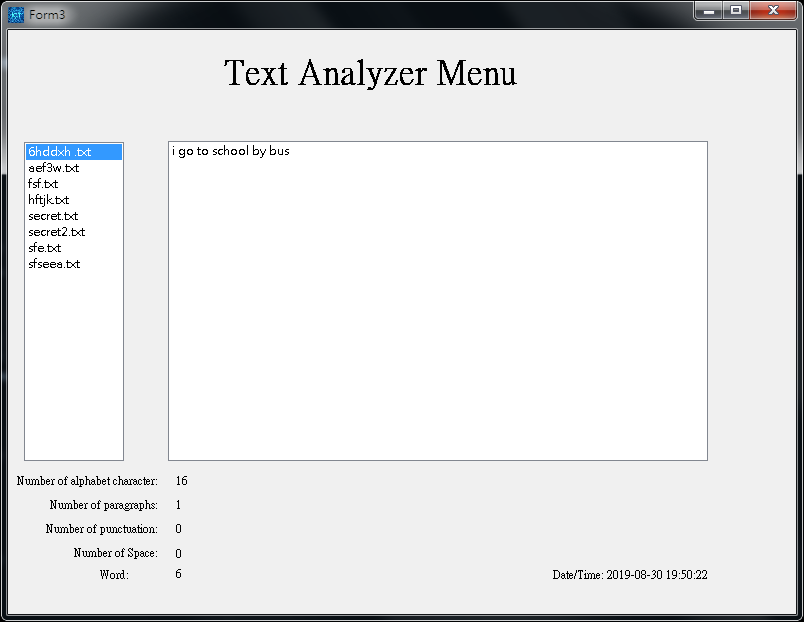


It will tell you the file have been inputted

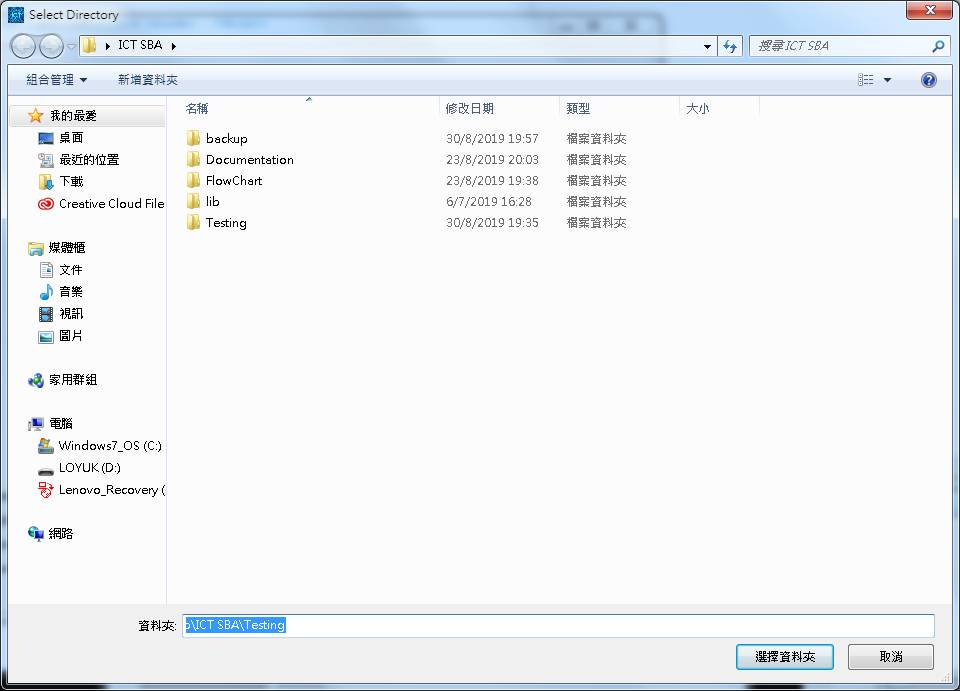


A file also.

The user should be reminded the analyzer can not double click the listbox in the middle when user have inputted a text file before.



Should click the blank side of the user interface



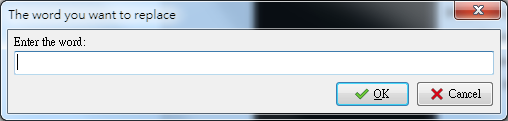
In this testing, I will use different of data to test the logic of analyzer in the table below.

|  |  |  |
| --- | --- | --- |
| Name | Expected Output | Output |
| Hello? | Exit(Invalid) | Exit(Invalid) |
| Abc123456789 | Abc123456789.txt | Abc123456789.txt |
| >?>?>?>?>? | Exit(Invalid) | Exit(Invalid) |
| SFJKLDSJ.txt | SFJKLDSJ.txt | SFJKLDSJ.txt |
| 273849723489.txt | 273849723489.txt | 273849723489.txt |
| How are you | How are you.txt | How are you.txt |
| 123456789123456789.txt | 123456789123456789.txt | 123456789123456789.txt |
| .txt | .txt.txt | .txt.txt |

According to the chart in the above , we can know that the GIGO is basically ran out of this program.

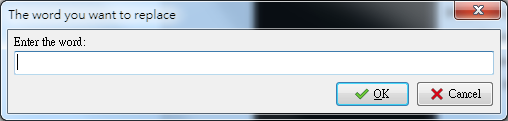
|Data Validation (Replace)

The user can input the word they want to replace .



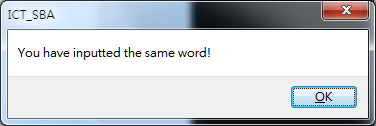
If the user blanked the input box ,the program will be automatically exit;

Then, enter the word you want to replace with in the passage.

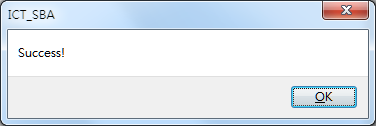


Also, if the user had blanked the inputbox , will also exit the program.

If the user inputted the same word in different input box .It will show message to the user to remind them they had inputted the same word and ask them to try again.

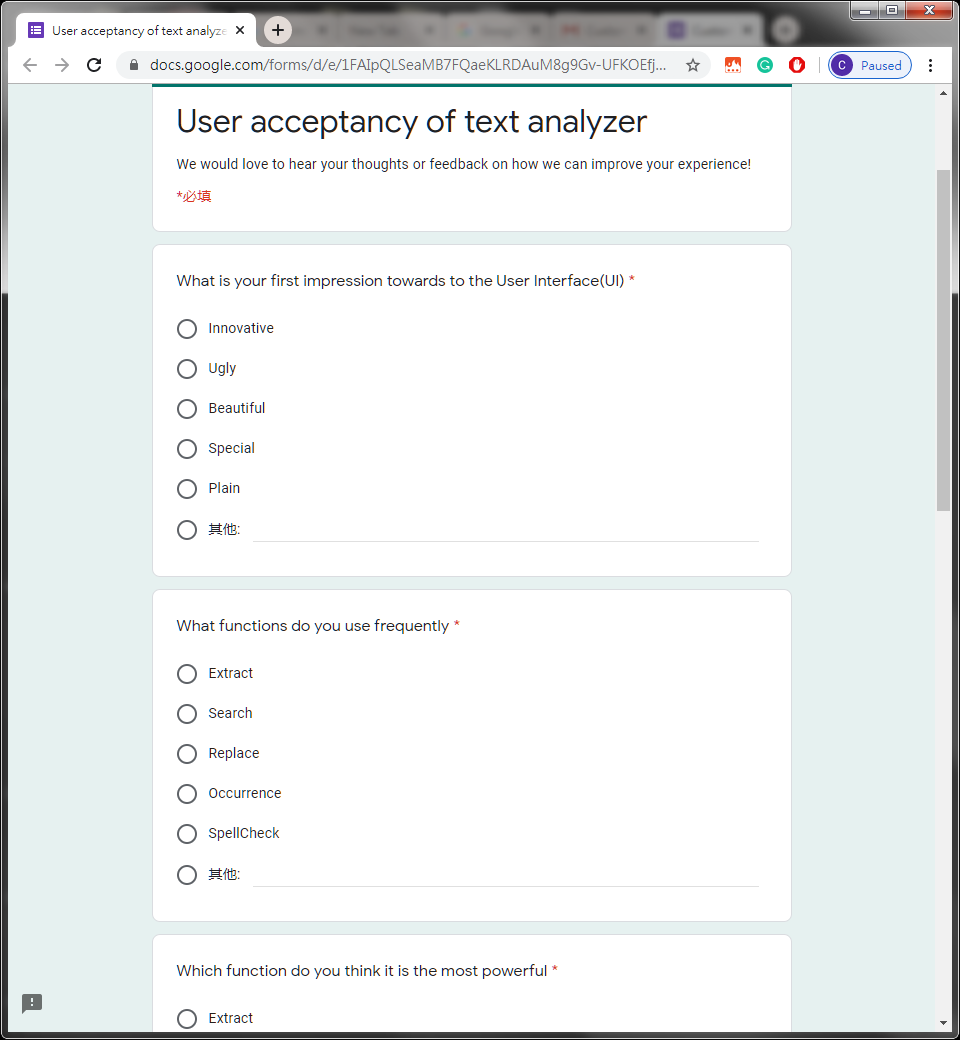


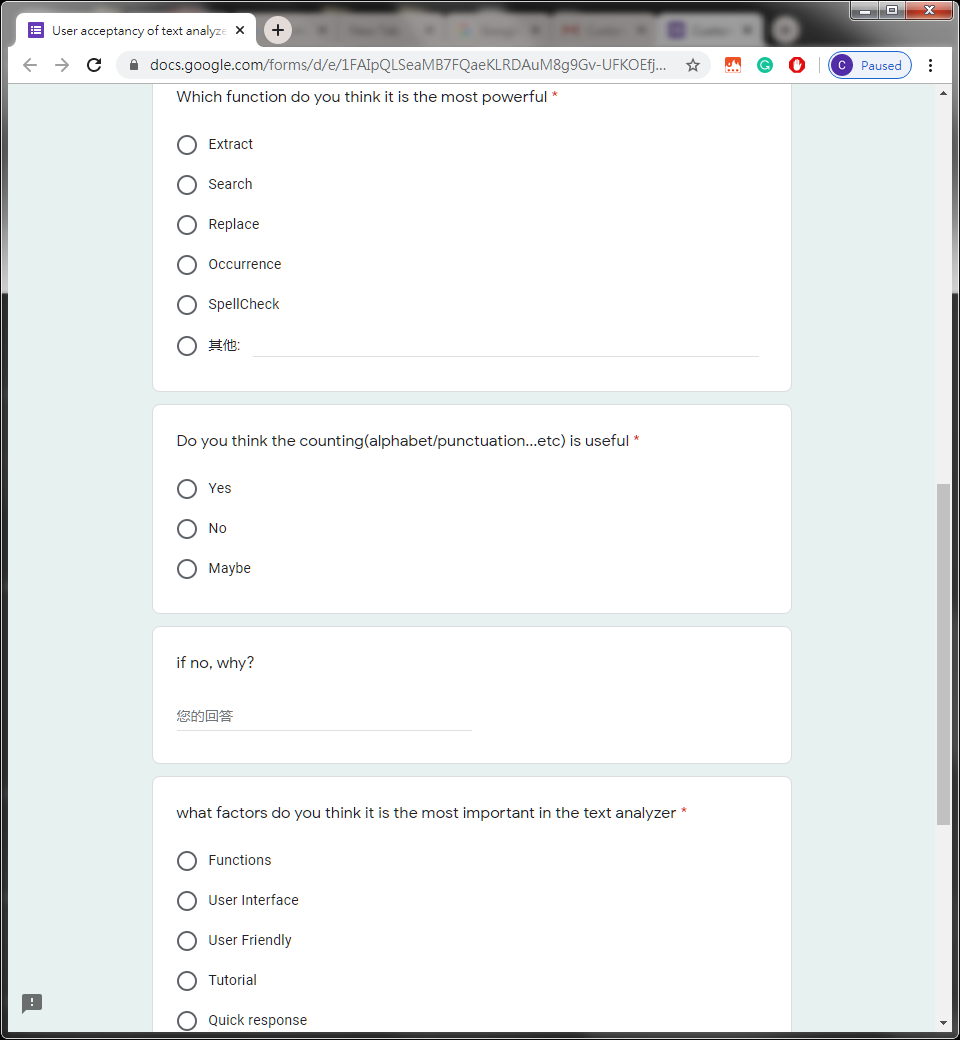
If the user input the word in the input box appropriately, the system will show a success message to user.



User acceptancy of the program:

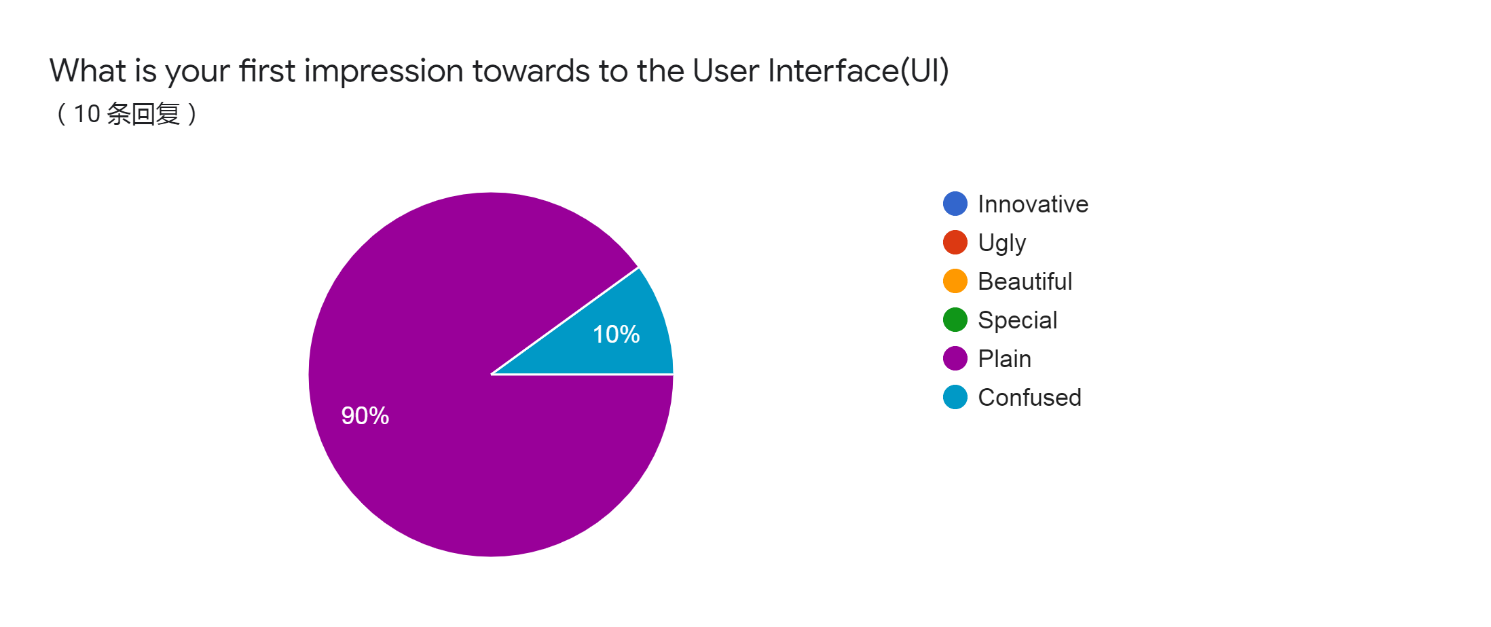
A form is created to collect user requirements.

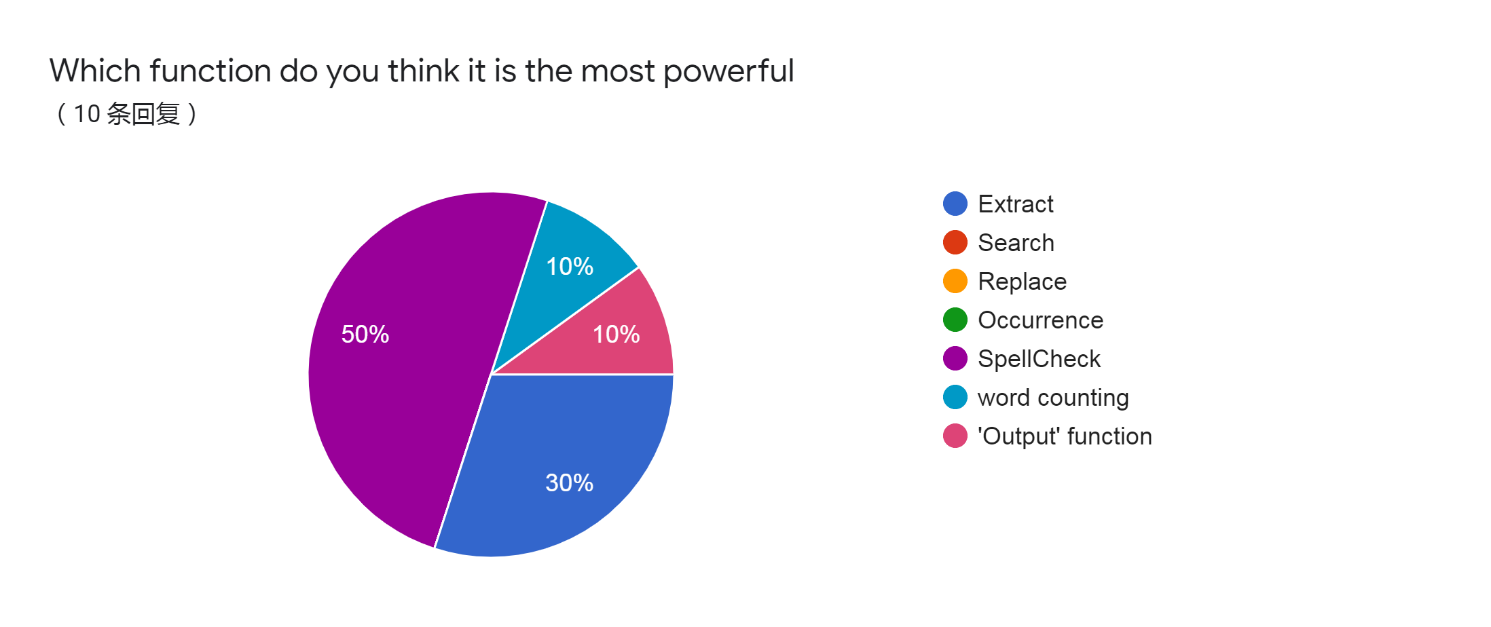
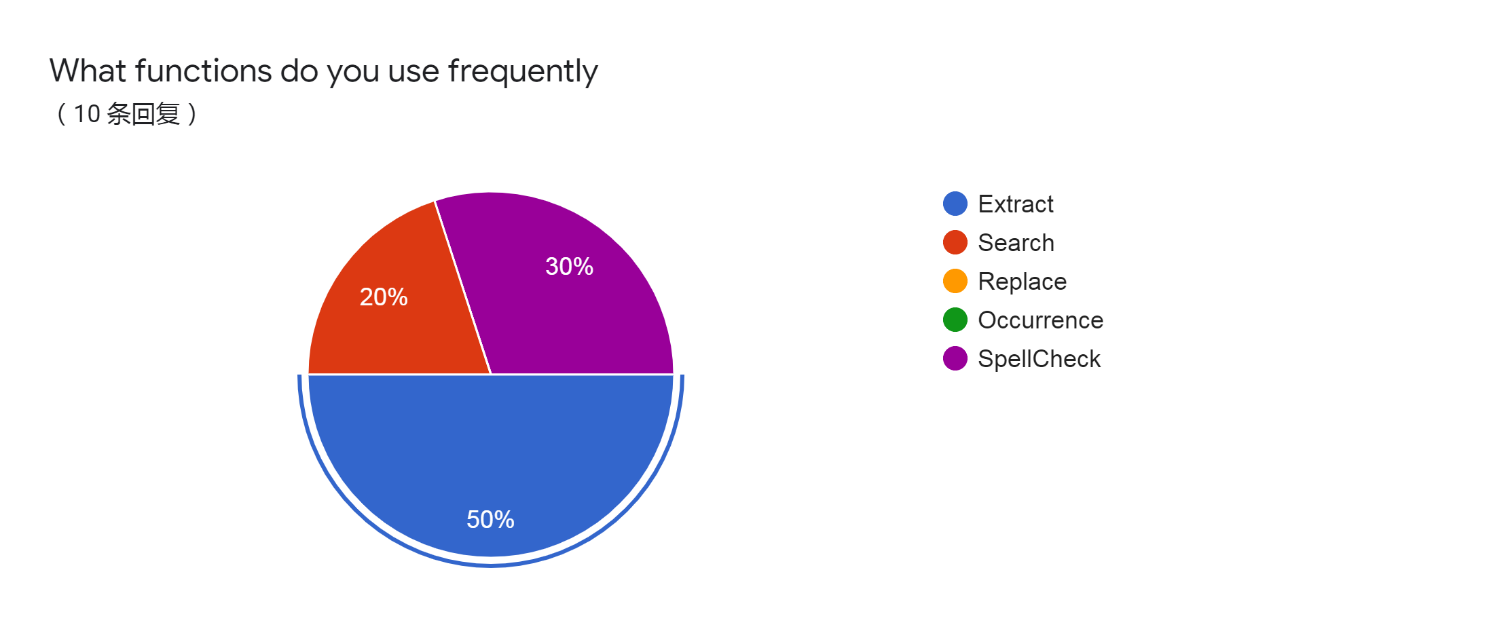


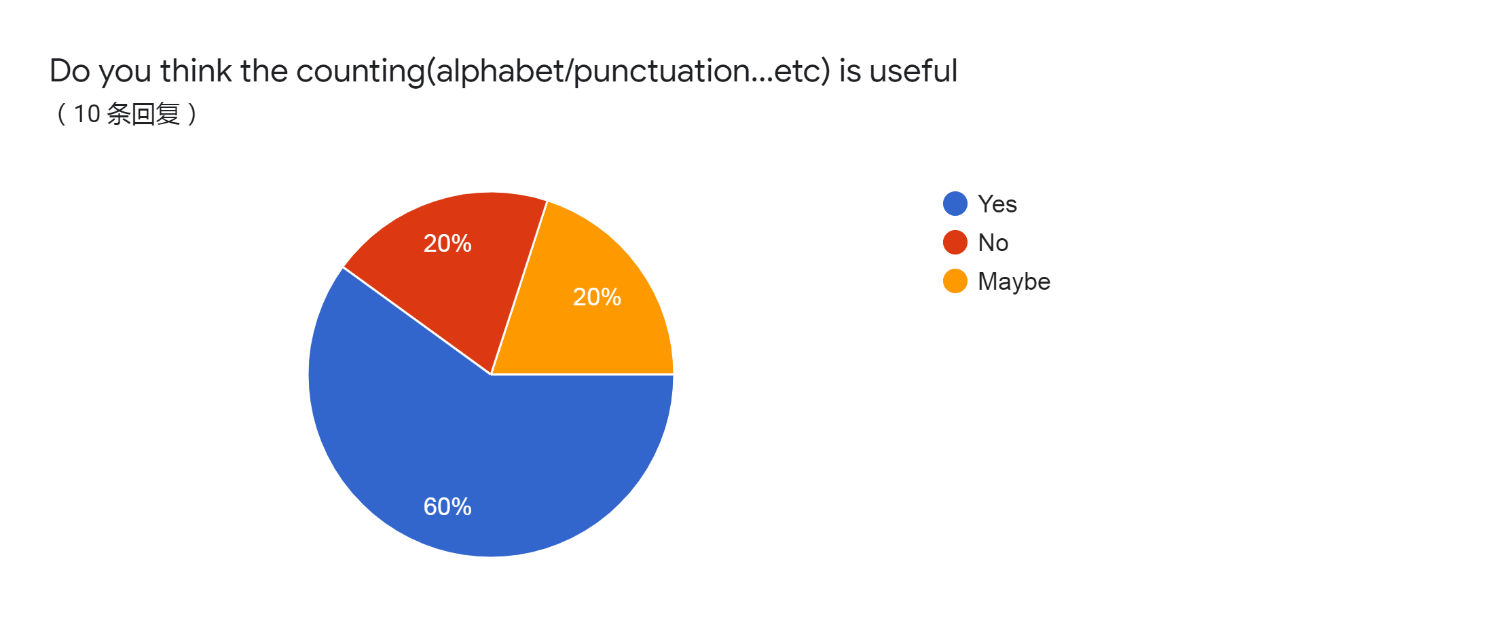


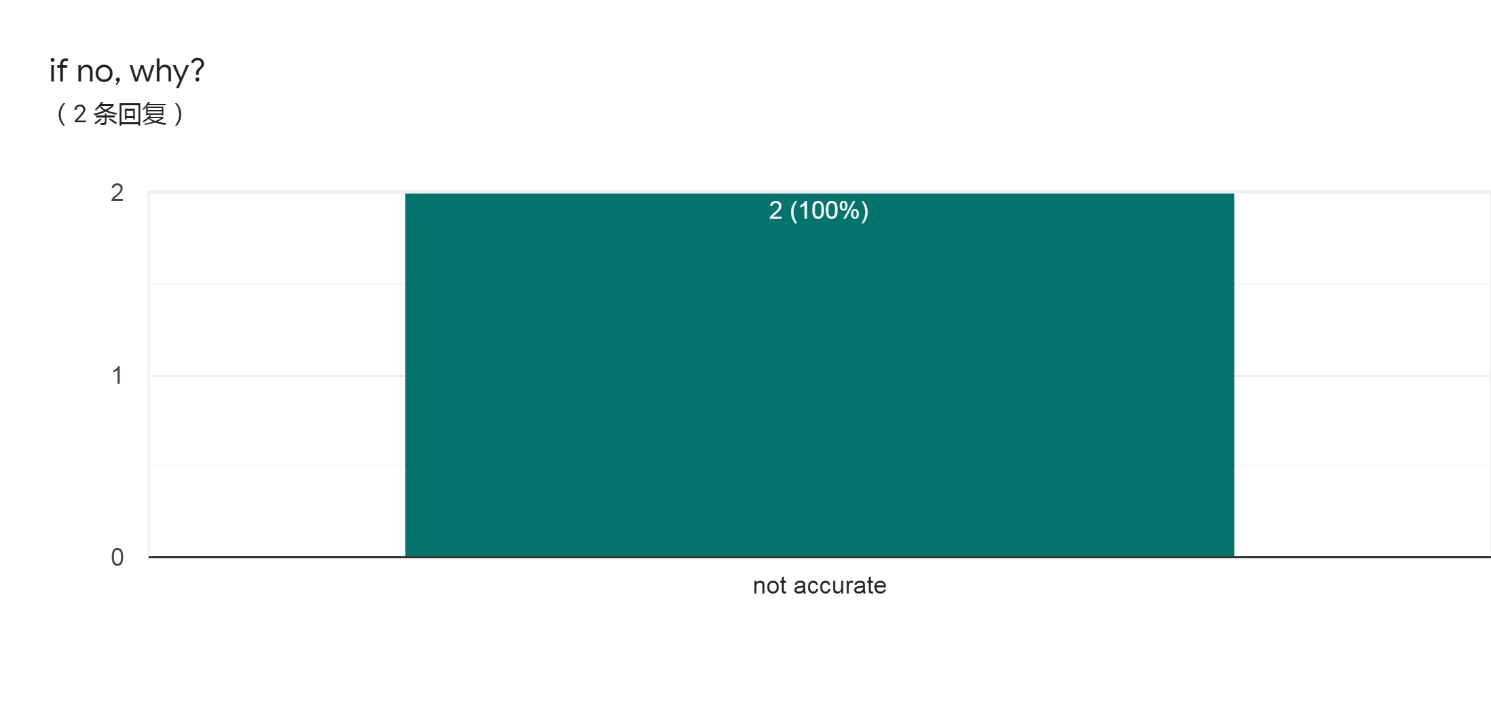


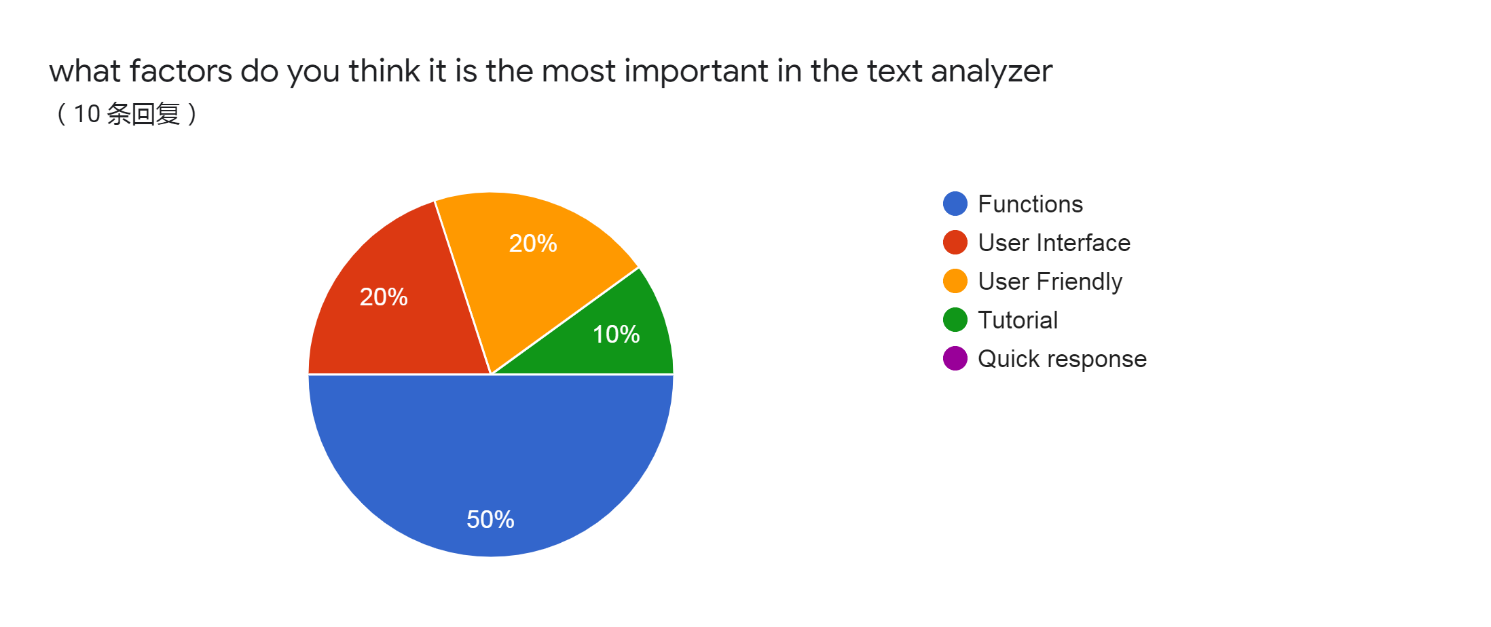
10 user’s feedback is collected.











Feedbacks:

the time has a 1-2s delay

improve the accuracy of the word counting

support other file extension

support Chinese word

make the UI become more attractive

support amendment in the text analyzer

support other file format

include more functions

always lagging

the user manual is quite good for us to learn with a new text analyzer

To conclude,

Supporting more file extension is needed

More attractive UI

Include more functions

Having optimization on the word counting

Add some instructions on the text analyzer to tell user how to use it

|Pros and Cons of the system design

We have collected the user requirements and in the following , I will list some of the system drawbacks during my testing and evaluation.

Pros:

Extract :

1.Quick

2.Convinent

3.Clearly Visible to the user

4.Have a scroll bar to check the content

5.The output file had Sorted

6.Can use the Mouse wheel to slide down the listbox

7.Can select the text file which user specifically selected

Search :

1.Highlight the sentence that in word included

2.MultiSelect is available

3.Data Validation

4.Display the occurrences of the result

5.Can only search a part of a text

Replace:

1.Data Validation

2. User can select replace a whole word or character in the string

3.Display message to tell user the function is finished

4.Can select the line of listbox and the program will only consider the selected one

5.Chinese text can also be replace

Spell Count:

1.Having data validation

2. If user click the small one of listbox,the bigger listbox will highlight the place according to the place

3. Display the place of the word specifically

SpellCheck:

1.WordBank is large enough to check the grammar or tense of a word

2.Using dynamic array , so that it will not lead to memory leaks

3.The numerical number will not be considered

4.Using binary search to check the appearance of a word (more efficient)

5.Display the number of result and if the result is 0 it won’t display the listbox ,just show the message ’ doesn’t find’

Cons:

Extract:

1.Cannot extract Chinese word because it will become garbled message in the output file

2.When the program is executing a huge size of text, there is no tips or hints to tell the user about when the program is done

3.Can only select a file to enter the program, not a txt file independently

4.Only English alphabet can extract precisely (if we extract the words like Chinese , the word will become garbled)

Search:

1.Chinese text can not be searched

2.After the searaching, if the user click the mouse , all the highlights will be disabled

3.Only blue highlighted color can be chose

4.Can’t locate the result accurately(only highlight the whole line)

5.Only the number of lines in the results and the accurate position will not be displayed

Replace:

1.If the user need to replace lots of word , they may not know what exactly changes in the passage , as the display doesn’t highlight the replaced results.

2.No matter the program changes many words or doesn’t , the program will still display ‘success’ when it is completed.

3.Can’t save into the original file

4.Can’t change back to the original words

5.The order of the input word can be more clear

6.Won’t highlight the replaced word

7.Blue colour highlighter only

SpellCount:

1.Will highlight many of sentences at the same time (the user may feel confused)

2.The user don’t know what the number of the column is

3.Sometimes the form of spell count will be displayed behind the menu

4.Chinese text can not be counted accurately

5.The character inside the word may also be counted

6.The specific position of the result will not be displayed inside the listbox

(only highlight the whole sentence)

SpellCheck:

1.Only can checking the spelling of the words , but not grammar

2.Some of the old and rare words or short form will also be marked as a wrong word(words in shakespare)

3.The number of vocab may be enough and it will lead to the wrong checking of program

4.Can’t select the wrong word through the spellcheck form

5.Can not change the wrong in the new listbox directly

6.Redundant reminder (such as Done)



|Reflections:

Before I do the SBA , I felt confused about coding , so I need search for the solution in the internet, However,

after coding for a month, the typing skills and the logic of my mind is getting improved.

It can be a revision of the text book, as most of the programming skills need to be done by our hands ,which is very crucial that the knowledge have immersed into my brain or not.

I have learnt about different experience of getting bugs, so the next time will the error more efficiently.

Learn a programming language in a deeper way.so if I need to learn a programming language from scratch, from the experience of doing SBA project , I can learn other programming language in a efficiently way.

When I was doing the analyzer, it develops my patient and logic with no doubt, as this text analyzer becomes the first software in my life ,it encourages me to step into the IT-related job.

Inside the SBA, I have experienced of doing IT-related job. When I am in the society , I will be more competitive than other ICT students.

|Appendix

Source code of the system:

unit FunctionButton;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls,

Menus, ExtCtrls, LCLType, Buttons, ComCtrls, EditBtn;

type

{ TForm4 }

TForm4 = class(TForm)

Alarm: TButton;

OutPut: TButton;

SelectDirectoryDialog1: TSelectDirectoryDialog;

PopupMenu2: TPopupMenu;

Extract: TButton;

Label2: TLabel;

Search: TButton;

Replace: TButton;

SpellCount: TButton;

SpellCheck: TButton;

procedure AlarmMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: integer);

procedure ExtractClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure OutPutClick(Sender: TObject);

procedure ReplaceClick(Sender: TObject);

procedure SearchClick(Sender: TObject);

procedure SpellCheckClick(Sender: TObject);

procedure SpellCountClick(Sender: TObject);

procedure ContinueClick(Sender: TObject);

procedure PauseClick(Sender: TObject);

procedure StopClick(Sender: TObject);

procedure RestartClick(Sender: TObject);

private

{ private declarations }

public

{ public declarations }

word: string;

Pointer: integer;

Time: string;

Days :integer;

Hour: integer;

Minute: integer;

Second: integer;

Cnt: integer;

end;

var

Form4: TForm4;

n: integer;

Button: array of TButton;

Counter: integer;

outfile: TextFile;

outarr,WordBank: array of string;

Cal :array of TButton;

Alarm\_Option: array of TMenuItem;

path : TMenuItem;

Re\_s,Re\_m,Re\_h,Re\_d: integer;

implementation

uses MainMenu, RegExpr, WorkingPanel;

{$R \*.lfm}

{ TForm4 }

procedure QuickSort(Top,Bottom:integer);

var

i,j:integer;

k,temp : string;

begin

i := Top;

j := Bottom;

k := outarr[(Top + Bottom) div 2];

repeat

While k > outarr[i] do

inc(i);

While k < outarr[j] do

dec(j);

if i <= j then

begin

Temp := outarr[i];

outarr[i] := outarr[j];

outarr[j] := Temp;

dec(j);

inc(i);

end;

until i > j;

if Top < j then

Quicksort(Top,j);

if i < Bottom then

QuickSort(i,Bottom);

end;

procedure Extracting(Folder: string);

var

Code ,No ,i ,H ,M ,L: integer;

infile :Text;

s, w: string;

Found: boolean;

begin

try

Assign(infile, Folder);

reset(infile);

SetLength(outarr,1);

outarr[0] := '';

while not eof(infile) do

begin

readln(infile, s);

if Trim(s) = '' then

continue;

for i := 1 to Length(s) do

if (s[i] = chr(9)) or (ExecRegExpr('\W',s[i])) then

s[i] := ' ';

s := AnsiLowerCase(s);

repeat

if pos(' ',s) > 0 then

begin

w := copy(s, 1, pos(' ', s) - 1);

s := copy(s, pos(' ', s) + 1, length(s));

s := TrimLeft(s);

end

else begin

w := s;

s := '';

end;

val(w,No,Code);

if ExecRegExpr('^()$', w) or (Code = 0) or (Length(w) = 1) then // Continue if w = (.....)

continue;

if outarr[0] <> '' then

begin

H := 0;

L := Length(outarr) - 1;

Found := False;

repeat

M := (H + L) div 2;

if w > outarr[M] then // Binary Search

H := M + 1

else if w < outarr[M] then

L := M - 1

else

Found := True;

until (H > L) or Found;

if not Found then

begin

SetLength(outarr,Length(outarr) + 1);

outarr[Length(outarr) - 1] := w;

end;

end

else

outarr[0] := w;

until s = '';

end;

finally

QuickSort(0,Length(outarr) - 1);

Close(infile);

end;

end;

procedure WordExtract;

var

F: TSearchRec;

Input: string;

i:integer;

begin

try

try

SetLength(outarr,0);

Input := InputBox('Input FileName', 'Write down the out-put file name:', '');

if ExecRegExpr('[?\/:"<>]', Input) = True then

begin

ShowMessage('Please write down a valid file name');

Exit;

end;

if Input = '' then

begin

case QuestionDlg('Oops', 'You haven''t input the Out-put file name',mtCustom, [mrYes, 'Use the default file name', mrIgnore, 'Exit'], '') of

mrYes: Input := 'Default.txt';

mrCancel: Exit;

mrIgnore: Exit;

end;

end

else if pos('.txt', Input) = 0 then

Input := Input + '.txt';

if (pos('#',Input) = 1) and (Length(Input) > 1) and (Form3.SelectDirectoryDialog1.Execute = True) then

begin

Delete(Input,1,1);

assign(outfile,Form3.SelectDirectoryDialog1.FileName + '\' + Input);

end

else

Assign(outfile,GetCurrentDir + '\' + Input); // Add a location for one text file only and self-selected location

rewrite(outfile);

for i := 0 to Form3.ListBox1.Items.Count - 1 do

if Form3.Listbox1.Selected[i] = True then

Extracting(Form3.Location[i]);

for i := 0 to Length(outarr) - 1 do

writeln(outfile, outarr[i]);

except

ShowMessage('couldn''t open the folder!');

ShowMessage('Please try again!');

end;

finally

Close(outfile);

ShowMessage('The file have been saved into ' + GetCurrentDir + '\' + Input);

end;

end;

procedure TForm4.SearchClick(Sender: TObject);

var

i,j,Count: integer;

Temp: string;

begin

try

Count := 0;

Form3.Listbox2.MultiSelect := True;

Form3.Listbox1.MultiSelect := False;

Form3.Listbox2.ClearSelection;

Word := InputBox('WordSearch', 'Enter Here', '');

if Word = '' then

Exit;

if ExecRegExpr('^#',word) = False then

for j := 0 to Form3.Listbox2.Items.Count - 1 do

begin

Temp := Form3.Listbox2.Items[j];

repeat

if pos(Word,Temp) > 0 then

begin

inc(Count);

Delete(Temp,1,pos(Word,Temp) + Length(Word));

Form3.Listbox2.Selected[j] := True;

end

until pos(Word,Temp) = 0;

end

else begin

if Length(Word) > 1 then

Delete(word, 1, 1);

for j := 0 to Form3.Listbox2.Items.Count - 1 do

begin

if pos(Word,Temp) = 0 then

continue;

Temp := Trim(Form3.Listbox2.Items[j]) + ' ';

for i := 1 to Length(Temp) do

if ExecRegExpr('\W',Temp[i]) = True then

Temp[i] := ' ';

if Temp[Length(Temp)] <> ' ' then

Temp := Temp + ' ';

repeat

if (Temp[pos(Word,Temp) - 1] = ' ') and (Temp[pos(Word,Temp) + Length(Word)] = ' ') then

begin

inc(Count);

Form3.Listbox2.Selected[j] := True;

end;

Delete(Temp,1,pos(Word,Temp) + Length(Word));

until pos(Word,Temp) = 0;

end;

end;

case Count of

0:begin

ShowMessage('No result has found');

Form3.Listbox2.MultiSelect := False;

end;

1:begin

ShowMessage(IntToStr(Count) + ' Results have found');

Form3.Listbox2.MultiSelect := False;

end;

else

ShowMessage(IntToStr(Count) + ' Results have found');

end;

except

end;

end;

procedure TForm4.SpellCheckClick(Sender: TObject);

var

Bank: Text;

Number,code ,i ,Hi, Middle, Bottom, j: integer;

Temp: string;

Temp3: array of string;

k: integer = 0;

Found: boolean;

begin

try

Form5.Listbox1.Clear;

SetLength(WordBank,0);

SetLength(Temp3,1);

AssignFile(Bank, GetCurrentDir + '\WordBank.txt');

reset(Bank);

while not EOF(Bank) do

begin

SetLength(WordBank,Length(WordBank) + 1);

readln(Bank, WordBank[Length(WordBank) - 1]); // Inserting the vocabulary

end;

CloseFile(Bank);

for i := 0 to Form3.Listbox2.Items.Count - 1 do

begin

if Trim(Form3.Listbox2.Items[i]) <> '' then

Temp := AnsiLowerCase(Trim(Form3.Listbox2.Items[i]))

else

continue;

for j := 1 to length(Temp) do

if ExecRegExpr('\W', Temp[j]) = True then

Temp[j] := ' ';

repeat

Temp := Trim(Temp);

if pos(' ', Temp) > 0 then

begin

Temp3[Length(Temp3) - 1] := copy(Temp, 1, pos(' ', Temp) - 1);

Temp := copy(Temp, pos(' ', Temp) + 1, length(Temp));

end

else begin

Temp3[Length(Temp3) - 1] := Temp;

Temp := '';

end;

val(Temp3[Length(Temp3) - 1],Number,Code);

if Code = 0 then

begin

Temp3[Length(Temp3) - 1] := '';

continue;

end;

if Temp3[Length(Temp3) - 1] <> ' ' then

SetLength(Temp3,Length(Temp3) + 1);

until Temp = '';

end;

for k := 0 to Length(Temp3) - 1 do

begin

Hi := 1;

Bottom := Length(WordBank) - 1;

Found := False;

repeat

Middle := (Hi + Bottom) div 2;

if Temp3[k] > WordBank[Middle] then // Binary Search

Hi := Middle + 1

else if Temp3[k] < WordBank[Middle] then

Bottom := Middle - 1

else

Found := True;

until (Hi > Bottom) or Found;

for i := 0 to Form5.Listbox1.Items.Count - 1 do

if Form5.Listbox1.Items[i] = Temp3[k] then

Continue;

if Found = False then

begin

Form5.ListBox1.Items.Add(Temp3[k]);

end;

end;

finally

with Form5.Listbox1.Items do

begin

if Count = 1 then

ShowMessage('There are no mistakes in this passage!')

else begin

ShowMessage('Done');

if Count = 2 then

ShowMessage('There are ' + IntToStr(Count - 1) + ' mistake in the passage!')

else

ShowMessage('There are ' + IntToStr(Count - 1) + ' mistakes in the passage!');

Form5.Label1.Caption := 'SpellCheck';

Add('Count: ' + IntToStr(Count - 1));

Form5.Show;

end;

end;

end;

end;

procedure TForm4.SpellCountClick(Sender: TObject);

var

Target: string = '';

z: integer;

Count: integer = 0;

Temp,Temp3: string;

Temp2: string = '';

begin

Target := Inputbox('Please write down the word you want to count', '', '');

if Target = '' then

Exit;

Form5.Listbox1.Clear;

for z := 0 to Form3.Listbox2.Items.Count - 1 do

begin

if (Form3.listbox2.SelCount > 1) and (Form3.Listbox2.Selected[z] = False) then

continue;

if pos(Target,Form3.Listbox2.Items[z]) > 0 then

begin

Temp := Form3.Listbox2.Items[z];

Temp3 := '';

repeat

Temp2 := copy(Temp, 1, pos(Target, Temp) + length(Target) - 1);

Temp := copy(Temp, pos(Target, Temp) + length(Target), length(Temp));

Form5.Listbox1.Items.Add('(' + IntToStr(z) + ':' +

IntToStr(pos(Target, Temp2) + length(Temp3)) + ')');

inc(Count);

Temp3 := Temp3 + Temp2;

until pos(Target, Temp) = 0;

end;

end;

if Form5.Listbox1.Items.Count = 0 then

ShowMessage('The word doesn''t occur in the passage!')

else begin

Form5.Label1.Caption := 'SpellCount';

Form5.Listbox1.Items.Add('Count: ' + IntToStr(Count));

Form5.Show;

end;

end;

procedure TForm4.ExtractClick(Sender: TObject);

begin

WordExtract;

end;

procedure TForm4.FormCreate(Sender: TObject);

var

Count: integer = 0;

i: integer;

begin

Visible := False;

Constraints.MaxHeight := 275;

Constraints.MinHeight := 275;

Constraints.MinWidth := 275 ;

Constraints.MaxWidth := 275;

Extract.Caption := 'Extract';

Search.Caption := 'Search';

Replace.Caption := 'Replace';

SpellCount.Caption := 'SpellCount';

SpellCheck.Caption := 'SpellCheck';

OutPut.Caption := 'OutPut';

with Extract do

begin

BorderSpacing.Bottom := 20;

BorderSpacing.Top := 20;

BorderSpacing.Left := 20;

BorderSpacing.Right := 40;

Parent := Self;

inc(Count);

end;

with Search do

begin

BorderSpacing.Top := 20;

AnchorSide[akTop].Side := asrBottom;

AnchorParallel(akLeft, 0, Extract);

AnchorSide[akTop].Control := Extract;

Count := Count + 1;

end;

with Replace do

begin

BorderSpacing.Top := 20;

AnchorSide[akTop].Side := asrBottom;

AnchorParallel(akLeft, 0, Search);

AnchorSide[akTop].Control := Search;

Count := Count + 1;

end;

with SpellCount do

begin

BorderSpacing.Top := 20;

AnchorSide[akTop].Side := asrBottom;

AnchorParallel(akLeft, 0, Replace);

AnchorSide[akTop].Control := Replace;

Count := Count + 1;

end;

with SpellCheck do

begin

BorderSpacing.Top := 20;

AnchorSide[akTop].Side := asrBottom;

AnchorParallel(akLeft, 0, SpellCount);

AnchorSide[akTop].Control := SpellCount;

Count := Count + 1;

end;

SetLength(Alarm\_Option,3);

for i := 0 to 3 do

begin

Alarm\_Option[i] := TMenuItem.Create(PopUpMenu2);

PopUpMenu2.Items.Add(Alarm\_Option[i]);

Alarm\_Option[i].Tag := i;

case i of

0:Alarm\_Option[i].Caption := 'Pause';

1:Alarm\_Option[i].Caption := 'Stop';

2:Alarm\_Option[i].Caption := 'Restart';

3:Alarm\_Option[i].Caption := 'Continue';

end;

end;

// Select the form size

end;

procedure TForm4.OutPutClick(Sender: TObject);

var

outputfile:text;

i:integer;

begin

SelectDirectoryDialog1.Execute;

assignfile(outputfile,SelectDirectoryDialog1.FileName + '\LogOutPut.txt');

rewrite(outputfile);

for i := 0 to Form3.Listbox1.Items.Count - 1 do

with Form3.Content[i] do

begin

writeln(outputfile,Form3.Listbox1.Items[i]);

writeln(outputfile,' Alphabet:' + inttostr(alphabet));

writeln(outputfile,' Figure:' + inttostr(Figure));

writeln(outputfile,' Paragraph:' + inttostr(Paragraph));

writeln(outputfile,' Punctuation:' + inttostr(Punctuation));

writeln(outputfile,' Space:' + inttostr(Space));

writeln(outputfile,' Word:' + inttostr(Word));

writeln(outputfile,' Line:' + inttostr(Line));

end;

writeln(outputfile);

writeln(outputfile,'Edited Time:' + DateTimetoStr(Now));

closefile(outputfile);

ShowMessage('The data has been saved in ' + SelectDirectoryDialog1.FileName);

end;

procedure TForm4.ReplaceClick(Sender: TObject);

var

str: string = '';

Replace\_Word: string = '';

z: integer;

temp3, temp, temp2: string;

begin

try

str := Inputbox('The word you want to replace', 'Enter the word:', '');

if str = '' then

Exit;

Replace\_Word := Inputbox('Replace with "' + str + '"', 'Enter the word:', '');

if Replace\_Word = '' then

repeat

case QuestionDlg('Oops', 'You haven''t input the word you want to replace with',

mtCustom, [mrYes, 'Input Again', mrNo, 'Exit'], '') of

mrYes: Replace\_Word :=

Inputbox('Replace with "' + str + '"', 'Enter the word:', '');

mrCancel: Exit;

mrNo: Exit;

end;

until (Replace\_Word <> '');

if (Replace\_Word = str) then

begin

ShowMessage('You have inputted the same word!');

ShowMessage('Please try again');

Exit;

end;

for z := 0 to Form3.Listbox2.Count - 1 do

begin

Temp := Form3.Listbox2.Items[z];

if (Form3.listbox2.SelCount > 1) and (Form3.Listbox2.Selected[z] = False) then

continue;

if pos(str, Form3.Listbox2.Items[z]) > 0 then

begin

temp3 := '';

temp2 := Form3.Listbox2.Items[z];

repeat

temp := copy(temp2, 1, pos(str, temp2) - 1);

temp2 := copy(temp2, pos(str, temp2) + length(str), length(temp2));

Temp3 := Temp3 + Temp + Replace\_Word;

until pos(str, temp2) = 0;

Form3.Listbox2.Items[z] := Temp3 + Temp2;

end;

end;

ShowMessage('Success!');

except

end;

end;

procedure TForm4.AlarmMouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: integer);

var

Temp: string;

i:integer;

begin

if Button = mbLeft then

begin

Second := 0;

Minute := 0;

Hour := 0;

Days := 0;

Time := Inputbox('Alarm', 'Enter the time here:', '');

if Time = '' then

Exit;

if Time = '0' then

begin

ShowMessage('Please input a correct time!');

Exit;

end;

for i := 1 to Length(Time) do

if Time[i] in [' ','0'..'9',':'] = False then

ShowMessage('Please input a correct time!');

Temp := Time;

repeat

if pos(':', Temp) > 0 then

begin

Temp := copy(Temp, pos(':', Temp) + 1, Length(Temp));

Inc(Cnt);

end;

until pos(':', Temp) = 0;

case Cnt of

0: Second := StrToInt(Time);

1:

begin

Minute := StrToInt(copy(Time, 1, pos(':', Time) - 1));

Second := StrToInt(copy(Time, pos(':', Time) + 1, Length(Time)));

end;

2:

begin

Hour := StrToInt(copy(Time, 1, pos(':', Time) - 1));

Time := copy(Time, pos(':', Time) + 1, Length(Time));

Minute := StrToInt(copy(Time, 1, pos(':', Time) - 1));

Second := StrToInt(copy(Time, pos(':', Time) + 1, Length(Time)));

end

else begin

ShowMessage('You''ve inputted a wrong number.Please try again!');

Exit;

end;

end;

if Second > 60 then

begin

Minute := Minute + Second div 60;

Second := Second mod 60;

end;

if Minute > 60 then

begin

Hour := Hour + Minute div 60;

Minute := Minute mod 60;

end;

Re\_s := Second;

Re\_m := Minute;

Re\_h := Hour;

Form3.Timer3.Enabled := True;

Form3.Label15.Visible := True;

end

else

with PopUpMenu2 do

begin

PopUp(Mouse.CursorPos.X, Mouse.CursorPos.Y);

Items[0].OnClick := @PauseClick;

Items[1].OnClick := @StopClick;

Items[2].OnClick := @RestartClick;

Items[3].OnClick := @ContinueClick;

end;

end;

procedure TForm4.PauseClick(Sender: TObject);

begin

if Form3.Timer3.Enabled then

Form3.Timer3.Enabled := False;

end;

procedure TForm4.StopClick(Sender: TObject);

begin

if Form3.Timer3.Enabled then

begin

Hour := 0;

Minute := 0;

Second := 0;

Form3.Label15.Visible := False;

end;

end;

procedure TForm4.RestartClick(Sender: TObject);

begin

Case QuestionDlg('Confirm','Are you sure to restart the Timer?',mtCustom,[mrYes,'Yes',mrNo,'No'],'') of

mrYes:;

mrNo:Exit;

mrCancel:Exit;

mrIgnore:Exit;

end;

if Form3.Timer3.Enabled then

begin

Hour := Re\_s;

Minute :=Re\_m;

Second := Re\_h;

end;

end;

procedure TForm4.ContinueClick(Sender: TObject);

begin

if (((Hour > 0) or (Minute > 0) or (Second > 0)) and Form3.Label15.Visible = True)then

Form3.Timer3.Enabled := True;

end;

end.

unit MainMenu;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls,

ComCtrls, ExtCtrls, EditBtn, Menus, Types, DateUtils, StrUtils, RegExpr,

FunctionButton;

{ TForm3 }

type

status = record

alphabet: integer;

Figure: integer;

paragraph: integer;

punctuation: integer;

space: integer;

word: integer;

Line: integer;

end;

TForm3 = class(TForm)

Label1: TLabel;

Label10: TLabel;

Label11: TLabel;

Label12: TLabel;

Label13: TLabel;

Label14: TLabel;

Label15: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label6: TLabel;

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

ListBox1: TListBox;

ListBox2: TListBox;

ListBox3: TListBox;

PopupMenu1: TPopupMenu;

SelectDirectoryDialog1: TSelectDirectoryDialog;

Timer1: TTimer;

Timer2: TTimer;

Timer3: TTimer;

procedure FormClick(Sender: TObject);

procedure FormDropFiles(Sender: TObject; const FileNames: array of string);

procedure FormKeyDown(Sender: TObject; var Key: word; Shift: TShiftState);

procedure Label4DblClick(Sender: TObject);

procedure Label6DblClick(Sender: TObject);

procedure ListBox1Click(Sender: TObject);

procedure ListBox1DblClick(Sender: TObject);

procedure ListBox2DblClick(Sender: TObject);

procedure ListBox2MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: integer);

procedure ListBox2MouseWheelDown(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: boolean);

procedure ListBox2MouseWheelUp(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: boolean);

procedure FormCreate(Sender: TObject);

procedure ListBox1MouseWheelDown(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: boolean);

procedure ListBox1MouseWheelUp(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: boolean);

procedure Timer1Timer(Sender: TObject);

procedure Timer2Timer(Sender: TObject);

procedure Timer3Timer(Sender: TObject);

private

public

Content: array of status;

Location: array of string[100];

FN: string;

Last: integer;

end;

var

Form3: TForm3;

Open: TDateTime;

First: boolean = True;

Count: integer = 0;

Hrs, Days: word;

Screen: TMemo;

i: integer;

procedure FindFile(Path: string);

procedure FileStatus;

implementation

{$R \*.lfm}

{ TForm3 }

procedure FileStatus;

var

Temp: string;

k: integer;

blank: boolean;

Txt: TextFile;

begin

for i := 0 to Length(Form3.Content) - 1 do

with Form3.Content[i] do

begin

alphabet := 0;

paragraph := 0;

punctuation := 0; // initialization

space := 0;

word := 0;

Figure := 0;

Line := 0;

end;

for i := 0 to Length(Form3.Location) - 1 do

with Form3.Content[i] do

begin

Assign(Txt, Form3.Location[i]);

reset(Txt);

if EOF(Txt) then

Continue;

Blank := False;

while not EOF(Txt) do

begin

readln(Txt, Temp);

Inc(Line);

if Trim(Temp) = '' then

begin

Blank := True;

Continue;

end

else if Blank then

begin

Blank := False;

Inc(Paragraph);

end;

Temp := TrimRight(Temp);

word := word + WordCount(Temp, StdWordDelims);

for k := 1 to length(Temp) do

begin

if Temp[k] in ['~', '!', '@', '#', '$', '%', '^', '&', '\*','(', ')', '\_', '-', '+', '=', '\', '/', '?', '.', ',', '>', '<', '{','}', '[', ']', ':', ';', '"', '|', '`'] then

Inc(Punctuation)

else if Temp[k] in ['a'..'z', 'A'..'Z'] then

Inc(Alphabet)

else if (Temp[k] = '') or (Temp[k] = chr(9)) then

Inc(Space)

else if Temp[k] in ['0'..'9'] then

Inc(Figure);

end;

end;

Inc(Paragraph);

Closefile(Txt);

end;

with Form3.Content[0] do

begin

Form3.Label7.Caption := IntToStr(Alphabet);

Form3.Label8.Caption := IntToStr(Paragraph);

Form3.Label9.Caption := IntToStr(Punctuation);

Form3.Label11.Caption := IntToStr(Space);

Form3.Label14.Caption := IntToStr(word);

end;

end;

procedure FindFile(Path: string);

var

Mask: string;

i,j,Count: integer;

Found: boolean;

begin

try

Count := 0;

Form3.Last := Length(Form3.Location);

if ExtractFileExt(Form3.FN) = '.txt' then

begin

SetLength(Form3.Content, Length(Form3.Content) + 1);

SetLength(Form3.Location, Length(Form3.Location) + 1);

Form3.Listbox1.Items.Add(ExtractFileName(Path));

Form3.Location[Length(Form3.Location) - 1] := Path;

end

else

begin

if Path = 'C:\' then

begin

ShowMessage('This File is too big');

ShowMessage('Try to input another file!');

Exit;

end;

Mask := '\*.txt';

if FindAllFiles(Path, '\*', True).Count >= 100 then

case QuestionDlg('Too much files', 'You have inputted over 100 files',

mtCustom, [mrYes, 'I Know', mrNo, 'Press Wrong'], '') of

mrYes: ;

mrCancel: Exit;

mrNo: Exit;

end;

Form3.Listbox3.Items.Clear;

Form3.Listbox3.Items := FindAllFiles(Path, Mask, True);

if Form3.ListBox3.Items.Count = 0 then

begin

ShowMessage('There are no text file in your folder');

Exit;

end;

for i := 0 to Form3.Listbox3.Items.Count - 1 do

begin

Found := False;

j := 0;

repeat

if Form3.Listbox1.Items.Count = 0 then

break;

if Form3.Listbox1.Items[j] =

ExtractFileName(Form3.Listbox3.Items[i]) then

Found := True

else

Inc(j);

until (Found = True) or (j + 1 > Form3.Last);

if Found = True then

Inc(Count)

else

begin

Form3.Listbox1.Items.Add(ExtractFileName(Form3.Listbox3.Items[i]));

SetLength(Form3.Location, Length(Form3.Location) + 1);

Form3.Location[Form3.Last + i] := Form3.Listbox3.Items[i];

SetLength(Form3.Content, Length(Form3.Content) + 1);

end;

end;

if Count = 1 then

ShowMessage('You have inputed ' + IntToStr(Count) + ' file repeatedly')

else if Count > 1 then

ShowMessage('You have inputed ' + IntToStr(Count) + ' files repeatedly');

end;

FileStatus;

finally

Form3.Listbox2.Items.LoadFromFile(Form3.Location[0]);

Form3.Listbox1.Selected[0] := True;

Form3.Last := Length(Form3.Location);

Form4.Show;

Form4.Alarm.Enabled := True;

end;

end;

procedure TForm3.ListBox1MouseWheelDown(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: boolean);

begin

try

if (Listbox1.ItemIndex < Length(Content)) and (Listbox2.Visible) then

begin

Listbox1.Selected[Listbox1.ItemIndex + 1] := True;

Listbox2.Items.LoadFromFile(Location[Listbox1.ItemIndex]);

with Content[Listbox1.ItemIndex] do

begin

Label7.Caption := IntToStr(Alphabet);

Label8.Caption := IntToStr(Paragraph);

Label9.Caption := IntToStr(Punctuation);

Label11.Caption := IntToStr(Space);

Label14.Caption := IntToStr(word);

end;

end

except

end;

end;

procedure TForm3.FormCreate(Sender: TObject);

begin

SetLength(Form3.Location, 0);

Listbox3.Visible := False;

AllowDropFiles := True;

SelectDirectoryDialog1.InitialDir := GetCurrentDir;

Listbox2.Visible := True;

Label1.Caption := 'Text Analyzer Menu';

Label2.Caption := 'Number of alphabet character:';

Label3.Caption := 'Number of paragraphs:';

Label4.Caption := '';

Label5.Caption := 'Number of punctuation:';

Label6.Caption := '';

Label7.Caption := '0';

Label8.Caption := '0';

Label9.Caption := '0';

Label10.Caption := 'Number of Space:';

Label11.Caption := '0';

Label13.Caption := 'WordCount:';

Label14.Caption := '0';

Label15.Caption := '';

Label1.Alignment := taCenter;

Label4.Alignment := taCenter;

Label6.Alignment := taCenter;

Label15.Caption := '';

Label7.AutoSize := True;

Label8.AutoSize := True;

Label9.AutoSize := True;

Label11.AutoSize := True;

Timer1.Interval := 1000;

Timer2.Interval := 1000;

Timer3.Interval := 1000;

Timer3.Enabled := False;

Label6.Visible := False;

KeyPreview := True;

Constraints.MaxHeight := 600;

Constraints.MinHeight := 600;

Constraints.MaxWidth := 780;

Constraints.MinWidth := 780;

end;

procedure TForm3.ListBox1MouseWheelUp(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: boolean);

begin

try

if (Listbox2.Visible = True) and (Listbox1.ItemIndex > 0) then

begin

Form3.Listbox2.Items.LoadFromFile(Form3.Location[Listbox1.ItemIndex - 1]);

Listbox1.Selected[Listbox1.ItemIndex - 1] := True;

with Content[Listbox1.ItemIndex] do

begin

Label7.Caption := IntToStr(Alphabet);

Label8.Caption := IntToStr(Paragraph);

Label9.Caption := IntToStr(Punctuation);

Label11.Caption := IntToStr(Space);

Label14.Caption := IntToStr(word);

end;

end

except

end;

end;

procedure TForm3.Timer1Timer(Sender: TObject);

begin

Label4.Caption := 'Date/Time: ' + FormatDateTime('yyy-mm-dd hh:nn:ss', Now);

end;

procedure TForm3.Timer2Timer(Sender: TObject);

begin

Inc(Count);

if Count = 60 then

begin

Count := 0;

Inc(Hrs);

end;

if Hrs = 24 then

begin

Hrs := 0;

Inc(Days);

end;

Label6.Caption := 'Working for: ' + TimeToStr(EncodeTime(Days, Hrs, Count, 0));

end;

procedure TForm3.Timer3Timer(Sender: TObject);

begin

with Form4 do

begin

if Second > 0 then

Dec(Second)

else if Minute > 0 then

begin

Dec(Minute);

Second := 59;

end

else if Hour > 0 then

begin

Dec(Hour);

Minute := 59;

Second := 59;

end

else if (Hour = 0) and (Minute = 0) and (Second = 0) then

begin

// Make a alarm

end;

Label15.Caption := 'After ' + IntToStr(Hour) + 'h ' + IntToStr(Minute) +

'm ' + IntToStr(Second) + 's the alarm will be ring';

end;

end;

procedure TForm3.ListBox2MouseWheelDown(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: boolean);

begin

try

with Listbox2 do

begin

if ItemIndex = Content[Listbox1.ItemIndex].Line - 1 then

Selected[Content[Listbox1.ItemIndex].Line - 1] := True;

if (SelCount > 0) and (ItemIndex < Content[Listbox1.ItemIndex].Line - 1) then

Selected[ItemIndex + 1] := True;

if SelCount = 0 then

Selected[0] := True;

end;

except

end;

end;

procedure TForm3.Label6DblClick(Sender: TObject);

begin

Label6.Visible := False;

Label4.Visible := True;

end;

procedure TForm3.ListBox1Click(Sender: TObject);

var

Shifted: TShiftState;

i,j,k,l: integer;

begin

K := Listbox1.Items.Count;

L := Listbox1.ItemIndex;

if K > 0 then

begin

for i := 0 to Length(Form3.Location) - 1 do

if not FileExists(Form3.Location[i]) then

Form3.Location[i] := '';

if (i = 0) and (Length(Form3.Location) = 1) and (Form3.Location[i] = '')then

begin

Listbox1.Items.Clear;

Form3.Location[i] := '';

SetLength(Form3.Location, 0);

Exit;

end

else begin

j := 0;

repeat

if Form3.Location[j] = '' then

for i := j + 1 to Length(Location) do

if Form3.Location[i] = '' then

break

else if Form3.Location[i] <> '' then

begin

Form3.Location[j] := Form3.Location[i];

break;

end;

Inc(j);

until j = Length(Location);

end;

Listbox1.Items.Clear;

for i := 0 to Length(Form3.Location) - 1 do

Listbox1.Items.Add(ExtractFileName(Form3.Location[i]));

SetLength(Form3.Location, i + 1);

if K > Listbox1.Items.Count then

begin

ShowMessage('Some of the file is not exist in the original file');

ShowMessage('Please re-select the file');

Listbox2.Items.Clear;

Label7.Caption := '0';

Label8.Caption := '0';

Label9.Caption := '0';

Label11.Caption := '0';

Label14.Caption := '0';

Exit;

end;

if Listbox1.Items.Count = 0 then

begin

Listbox2.Items.Clear;

ShowMessage('The text file is not exist in the default path:');

ShowMessage('Those file will be deleted from the listbox');

Label7.Caption := '0';

Label8.Caption := '0';

Label9.Caption := '0';

Label11.Caption := '0';

Label14.Caption := '0';

Exit;

end

else begin

FileStatus;

Shifted := GetKeyShiftState;

if ssCtrl in Shifted then

ListBox1.MultiSelect := True;

Listbox2.Items.LoadFromFile(Form3.Location[L]);

Listbox1.Selected[L] := True;

with Content[L] do

begin

Label7.Caption := IntToStr(Alphabet);

Label8.Caption := IntToStr(Paragraph);

Label9.Caption := IntToStr(Punctuation);

Label11.Caption := IntToStr(Space);

Label14.Caption := IntToStr(word);

end;

end;

end;

end;

procedure TForm3.ListBox1DblClick(Sender: TObject);

begin

if SelectDirectoryDialog1.Execute = True then

FindFile(SelectDirectoryDialog1.FileName);

end;

procedure TForm3.ListBox2DblClick(Sender: TObject);

begin

if (Listbox1.Items.Count = 0) and (SelectDirectoryDialog1.Execute = True) then

FindFile(SelectDirectoryDialog1.FileName);

end;

procedure TForm3.ListBox2MouseDown(Sender: TObject; Button: TMouseButton;

Shift: TShiftState; X, Y: integer);

begin

if Button = mbRight then

PopUpMenu1.PopUp(Mouse.CursorPos.X, Mouse.CursorPos.Y);

end;

procedure TForm3.Label4DblClick(Sender: TObject);

begin

Label4.Visible := False;

Label6.Visible := True;

end;

procedure TForm3.FormDropFiles(Sender: TObject; const FileNames: array of string);

var

i: integer;

F: TSearchRec;

begin

FN := '';

for i := Low(FileNames) to High(FileNames) do

FN := FN + FileNames[i];

for i := 0 to Length(Location) - 1 do

if FN = Location[i] then

begin

ShowMessage('You have input the text file once before!');

Listbox1.Selected[i] := True;

Listbox2.Items.LoadFromFile(Location[i]);

with Content[i] do

begin

Label7.Caption := IntToStr(Alphabet);

Label8.Caption := IntToStr(Paragraph);

Label9.Caption := IntToStr(Punctuation);

Label11.Caption := IntToStr(Space);

Label14.Caption := IntToStr(word);

end;

Exit;

end;

if (ExtractFileExt(FN) = '.txt') or

(FindFirst(FN + '\\*.\*', F.Attr and faAnyFile, F) = 0) then

begin

FindFile(FN);

Form4.Alarm.Enabled := True;

Form4.Visible := True;

end

else

ShowMessage('Please drag a txt extension file or a folder!');

end;

procedure TForm3.FormClick(Sender: TObject);

var

i:integer;

begin

for i:= 0 to Listbox2.Items.Count - 1 do

Listbox2.Selected[i] := False;

if Listbox2.MultiSelect = True then

Listbox2.MultiSelect := False;

if Listbox1.MultiSelect = True then

begin

Listbox1.MultiSelect := False;

Listbox2.Items.LoadFromFile(Location[Listbox1.ItemIndex]);

with Content[Listbox1.ItemIndex] do

begin

Label7.Caption := IntToStr(Alphabet);

Label8.Caption := IntToStr(Paragraph);

Label9.Caption := IntToStr(Punctuation);

Label11.Caption := IntToStr(Space);

Label14.Caption := IntToStr(word);

end;

end;

end;

procedure TForm3.FormKeyDown(Sender: TObject; var Key: word; Shift: TShiftState);

var

j: integer;

begin

if ((Key = Ord('S')) or (Key = Ord('s'))) and (Shift = [ssCtrl]) then

try

for j := 0 to Listbox1.Items.Count - 1 do

begin

if FileSize(Location[j]) > 0 then

Listbox2.Items.SaveToFile(Location[j]);

end;

finally

ShowMessage('Please restart the program to get the latest data');

end;

if (Shift = [ssCtrl]) and ((Key = ord('q')) or (Key = ord('Q'))) and (Form3.Listbox1.Items.Count > 0) then

Form4.Show;

end;

procedure TForm3.ListBox2MouseWheelUp(Sender: TObject; Shift: TShiftState;

MousePos: TPoint; var Handled: boolean);

begin

try

with Listbox2 do

begin

if ItemIndex = 0 then

Selected[0] := True;

if (SelCount > 0) and (ItemIndex >= 1) then

Selected[Listbox2.ItemIndex - 1] := True;

if SelCount = 0 then

Selected[Content[Listbox1.ItemIndex].Line - 1] := True;

end;

except

end;

end;

end.

unit WorkingPanel;

{$mode objfpc}{$H+}

interface

uses

Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs, StdCtrls;

type

{ TForm5 }

TForm5 = class(TForm)

Label1: TLabel;

ListBox1: TListBox;

procedure ListBox1Click(Sender: TObject);

private

public

end;

var

Form5: TForm5;

implementation

uses MainMenu,FunctionButton;

{$R \*.lfm}

{ TForm5 }

procedure TForm5.ListBox1Click(Sender: TObject);

var

j: integer;

begin

try

case Label1.Caption of

'SpellCount':

if Listbox1.ItemIndex <> Listbox1.Count - 1 then

begin

val(copy(Listbox1.Items[Listbox1.ItemIndex], 2, pos(':', Listbox1.Items[Listbox1.ItemIndex]) - pos('(', Listbox1.Items[Listbox1.ItemIndex]) - 1), j);

Form3.Listbox2.Selected[j] := True;

end

else

Listbox1.ClearSelection;

'SpellCheck': if Listbox1.ItemIndex <> Listbox1.Count - 2 then

for j := 0 to Form3.Listbox2.Items.Count - 1 do

if pos(Listbox1.Items[Listbox1.ItemIndex], Form3.Listbox2.Items[j]) > 0 then

begin

Form3.Listbox2.Selected[j] := True;

Form3.Listbox2.MultiSelect := True;

end

else

Listbox1.ClearSelection;

end;

except

end;

end;

end.