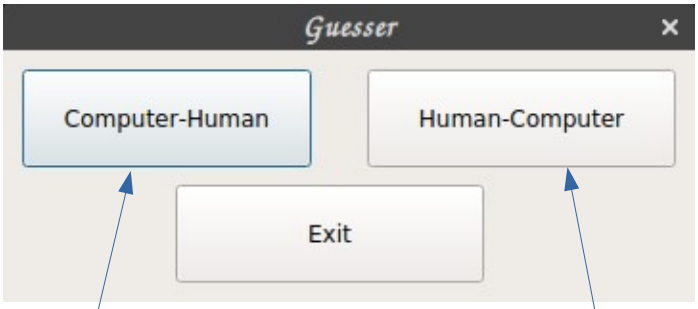
KARADENİZ TEKNİK ÜNİVERSİTESİ
Bilgisayar Mühendisliği
Proje Raporu

Ders: Algorıtmalar
Ders Sorumlusu: Prof. Dr. Vasıf NABIYEV
Dili: İngilizce

Problem Statement:

Word Guesser: This is a program that Guesses words depending on the selected mode.
There are three modes in the program which are:

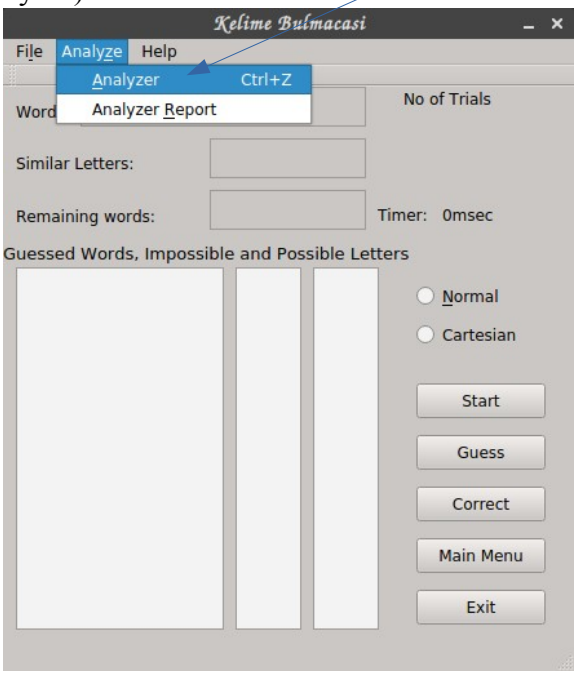
1. Computer-Human Program
2. Human-Computer Program
3. Computer-Computer Program (which is actually an analyzer).



Computer-Human Mode

Human-Computer Mode

Computer-Computer Mode

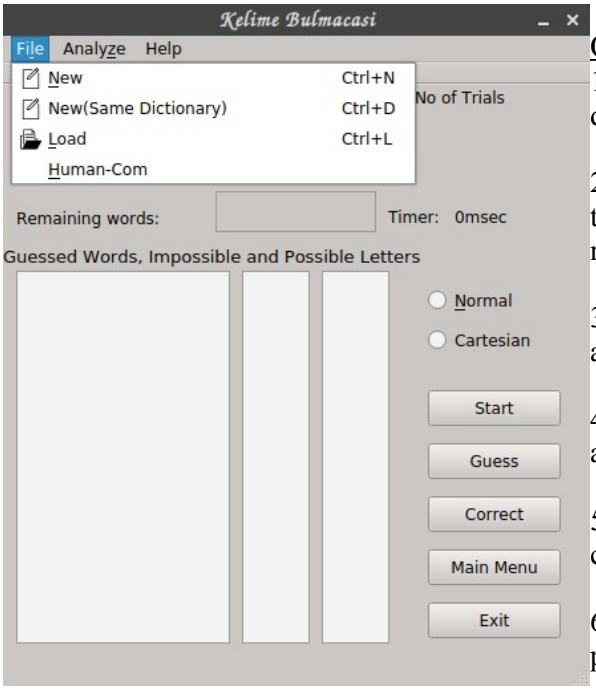


1. Computer-Human Program Mode:

In this mode, the user selects a word in his mind and the computer guesses the word within a certain number of trials specified by the Analyzer.

How to use this mode:

1. Load the dictionary from the file→load option on the menu bar.
2. Choose a word in your mind and click on the start button.
3. Click on the guess button so that Computer can guess your chosen word.
4. Follow any instruction given by the program.



Options:

1. New: This option is used to clear the program for a new dictionary to be loaded.

2. New(Same Dictionary): This option is used for clearing the program so that an already loaded dictionary can be reused.

3. Load: This option is used for loading a new dictionary after clearing the program.

4. Start Button: This button is used for starting the application after words has been loaded.

5. Guess Button: This button is used for informing the computer to guess the word the user chose.

6. Correct Button: This button is used for signaling the program that the user's guessed word has been found.

Algorithm:

1. Select the longest word in the list as the random word.
2. User inputs number of similar letters between random word and guessed word (This number can be calculated using Cartesian or Normal method of evaluation).
3. Find the weight of all the words in the dictionary with the random word and selects words that have the same weight with the weight inputted by the user.
4. Repeat from step 1 till the word is found.

Example: Using cartesian method of evaluation:
User Word: Brim
Dictionary:

invaders	Terra	Willows	Jeter	Grund
Beach	Hamlet	Myth	Adjust	Brim
Warning	Axis	Hapless	Aerial	Witches
Winston	Sulphur	Page	Oss	Fibre

Iteration 1:
Random Word- invaders
User Input- 2
Dictionary:

~~invaders~~	~~Terra-4~~	Willows-2	~~Jeter-3~~	~~Grund-3~~
Beach-2	Hamlet-2	~~Myth-0~~	Adjust-2	Brim-2
~~Warning-5~~	Axis-2	~~Hapless-4~~	~~Aerial-4~~	~~Witches-3~~
~~Winston-4~~	~~Sulphur-1~~	Page-2	Oss-2	~~Fibre-3~~

Iteration 2:
Random Word- Adjust
User Input- 0
New Dictionary:

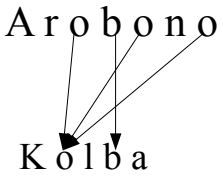
~~invaders~~	~~Terra~~	~~Willows-1~~	~~Jeter~~	~~Grund~~
Beach-0	~~Hamlet-1~~	~~Myth~~	~~Adjust~~	Brim-0
~~Warning~~	~~Axis-2~~	~~Hapless~~	~~Aerial~~	~~Witches~~
~~Winston~~	~~Sulphur~~	Page-0	~~Oss-2~~	~~Fibre~~

Iteration 3:
Random Word- Beach
User Input- 1
Dictionary:

~~invaders~~	~~Terra~~	~~Willows~~	~~Jeter~~	~~Grund~~
~~Beach~~	~~Hamlet~~	~~Myth~~	~~Adjust~~	Brim-1
~~Warning~~	~~Axis~~	~~Hapless~~	~~Aerial~~	~~Witches~~
~~Winston~~	~~Sulphur~~	~~Page-2~~	~~Oss~~	~~Fibre~~

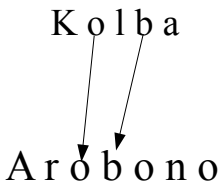
Result: The user Guessed word(Brim) has been found within 3 trials.

Cartesian Evaluation Method:
User Guessed Word: Arobono
Random Word: Kolba



Weight:4

Normal Evaluation Method:
User Guessed Word: Kolba
Random Word: Arobono



Weight:2

Data Structure:

```
struct Vector{
        string *Words;
        int characterfrequency[500];
        int noOfWords;
        int checkIfAnagram;
        int no_trials;
};
```

Main Function Pseudocode:

```
1. initialize Vector[30], n=0, iteration=0
2. ShowRandomWord()
3. input: UserWeight
4. for word in Vector[n]:
5.      weight=Evaluator(word)
6.      if weight=UserWeight:
7.              Vector[n+1].add(word)
8. n=n+1
9. if Vector[n] has a word:
10.     print word in Vector[n]
11. else if Vector[n] words are anagrams:
12.     print all words
13. else:
14.     display possible alphabets
15.     goto 2.
16. End
```

SelectRandomWords Function Pseudocode:

```
1. Randomword=first word in list
2. for word in Vector[n]:
3.         if length of word > length of Randomword:
4.                 Randomword=word
5. print Randomword
6. return
```

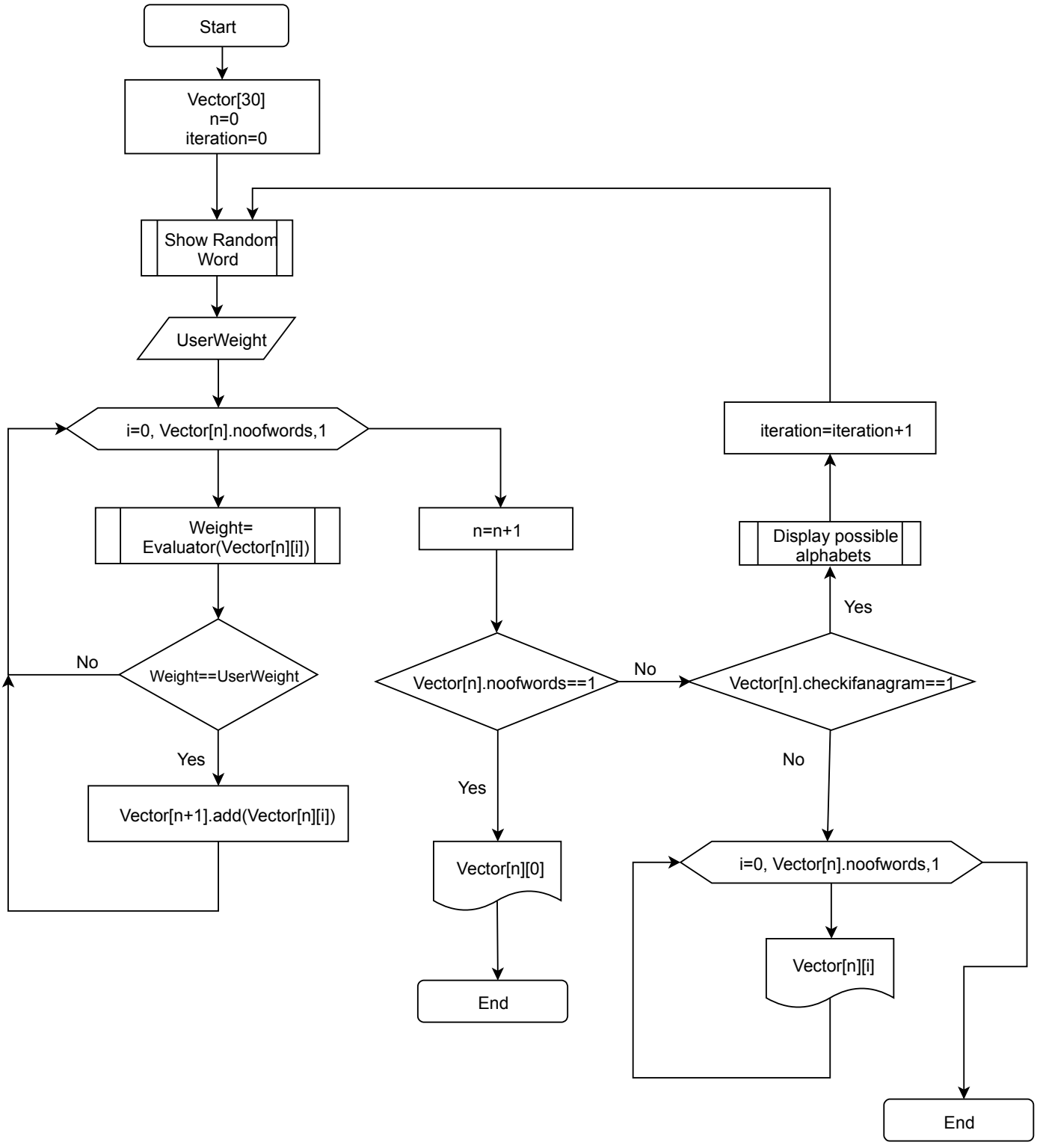
Evaluator(word, type) Function Pseudocode:

```
1. weight=0
2. for i in word:
3.         for j in Randomword:
4.                 if (type=cartesian):
5.                         if (i==j):
6.                                 weight=weight+1
7.                 else:
8.                         if (i==j):
9.                                 weight=weight+1
10.                                break
11. return weight
```

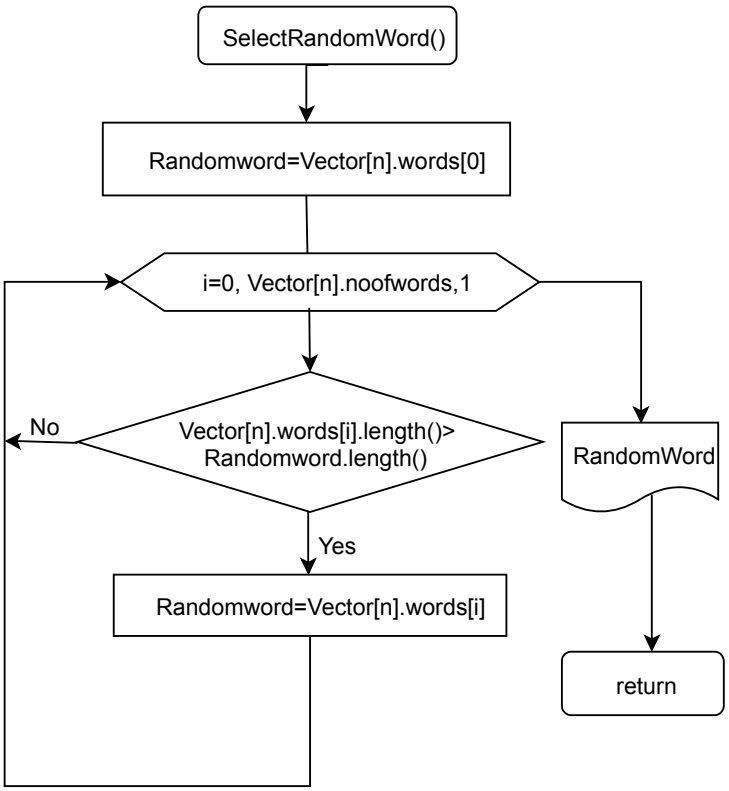
AlphabetProbability Function Pseudocode:

```
1. List="ĞğŞşÇçIıİiÖöÜü"
2. for i in range 500:
3.         char=ConverttoAsciiChar(i)
4.         if char is in List or (i>=65 and i<=90) or (i>=97 and i<122):
5.                 if Vector[n].characterfrequency[i]==0:
6.                         print char+”is not part of the word”
7.                 else:
8.                         print char+”is part of the word”
9. return
```

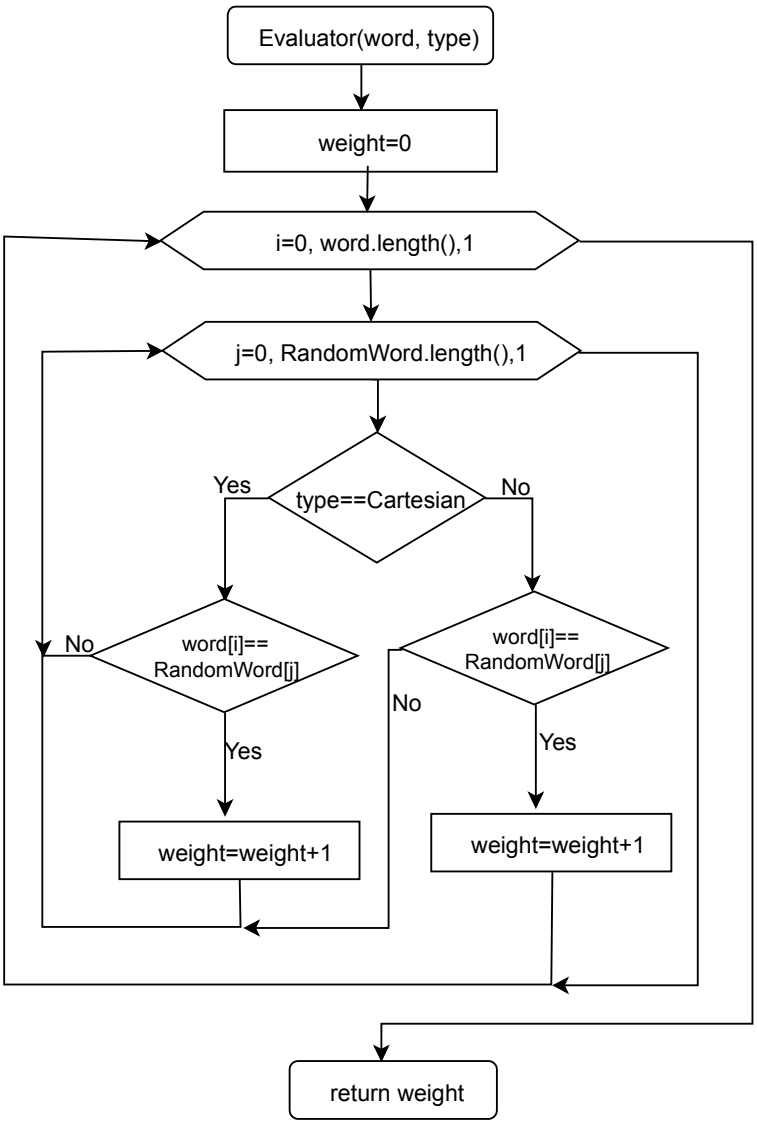
Main Program Flowchart



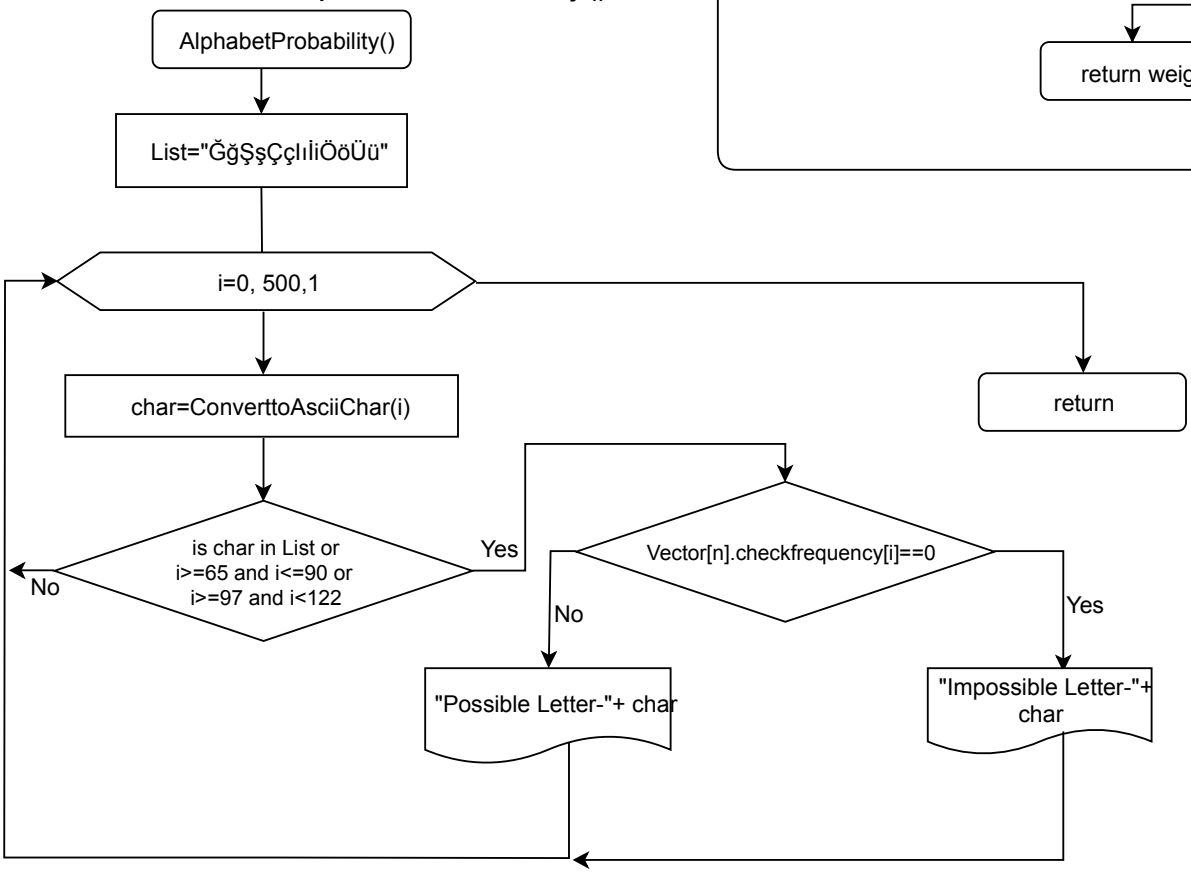
Flowchart: SelectRandomWords()



Flowchart: Evaluator(word, type)



Flowchart: AlphabetProbability()



Implementation:

The algorithm was implemented using the object oriented feature in C++ and QT platform Library for the program GUI design and development.

Files:

1. Vector.h and Vector.cpp:

A data structure called Vector was created for storing all information about the words in the dictionary. It stores:

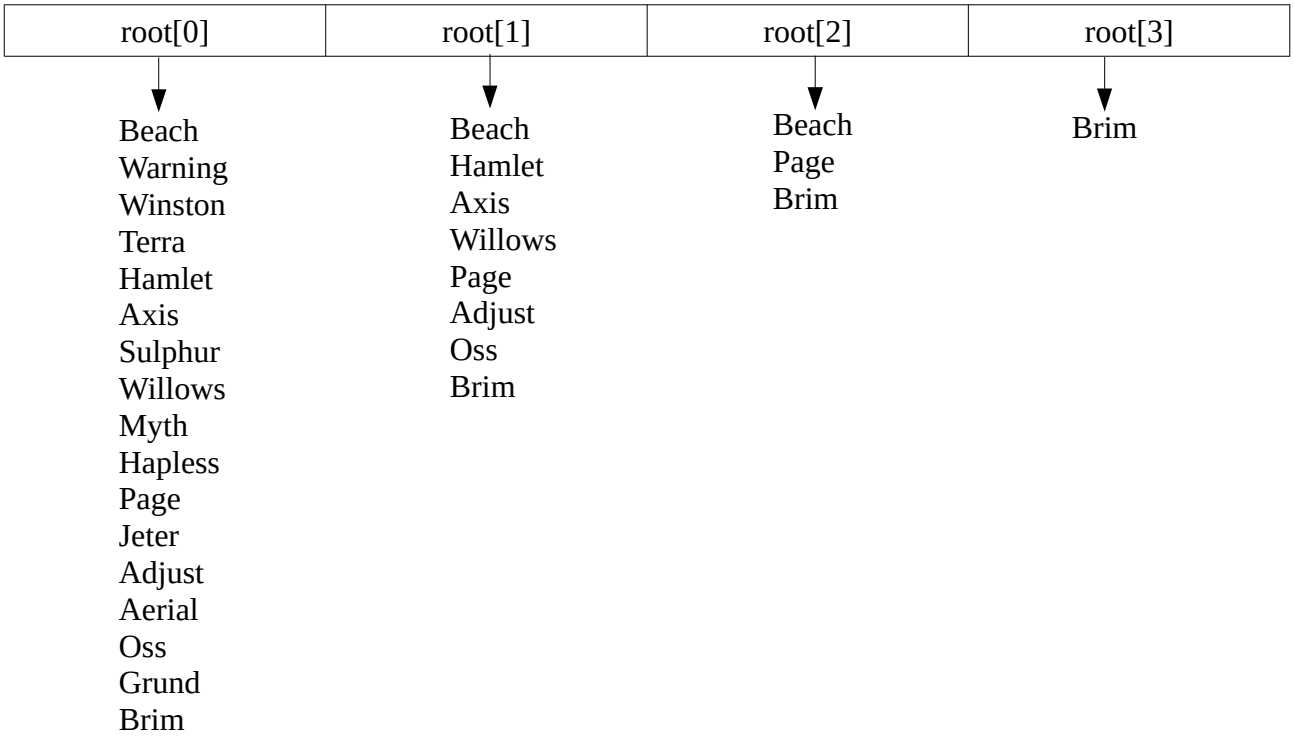
a. words from dictionary in an array
b. number of words in the dictionary
c. the frequency of each character in the dictionary

The data structure can also detect if the words are anagrams or not. An *anagram* is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. If the words are anagrams, the program outputs all the words because there is no optimum way we can find the word since all the words will sum up to the same weight when they are evaluated.

2. mainwindow.h, mainwindow.cpp and mainwindow.ui:

This file is the main file that carry out all the operation for the Computer-Human mode. Words **are not actually deleted according to our implementation**. Words are moved from one Vector object to another. This type of implementation makes tracking of word movement easy. The implementation is shown in the diagram below:

Declaration: **Vector *root[noOfTrials]**

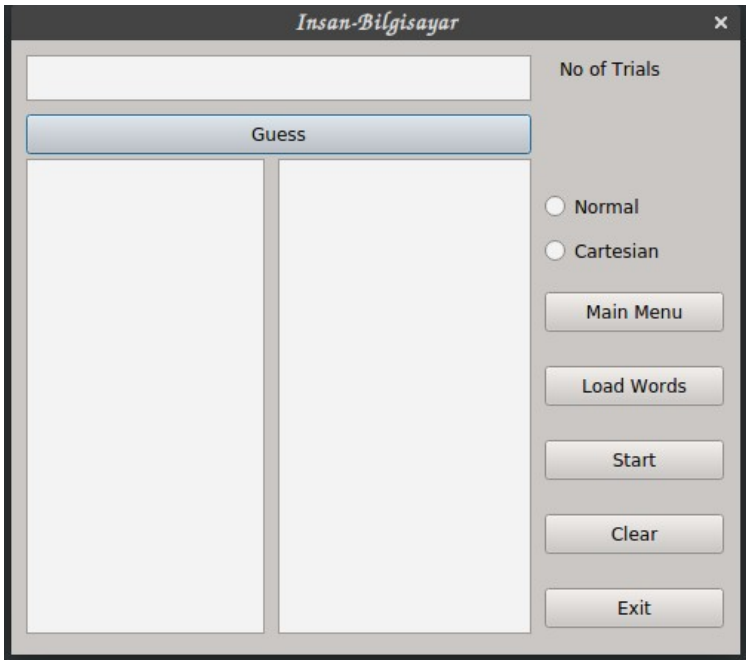


2. Human-Computer Program Mode:

In this mode, the computer selects a word and the user guesses the word the computer selected.

How to use this mode:

1. Load the dictionary from the Load Words Button.
2. Input the number of random words you want to load.
3. Click on the start button for the computer to select its word.
4. Guess the word and follow any instruction given by the program.



Algorithm:

1. User load certain number of words from dictionary.
2. Computer selects a random word.
3. User guesses a word.
4. Computer display the weight of the word the user guessed and the word it chose.
5. step 3 is repeated until the user guess the word correctly.

Main Program Pseudocode:

```
1. input: n
2. Load n words from dictionary into Words[]
3. iteration=0, ComputerWord=""
4. Pickword()
5. input: UserInput
6. if UserInput!=ComputerWord:
7.         weight=Evaluator(UserInput, type)
8.         print weight
9.         iteration=iteration+1
10.        go to 5.
11. else:
12.        print UserInput+“is correct”
13. End
```

Pickword Function Pseudocode:

1. i=rand()%noofwords
2. ComputerWord=Words[i]
3. return

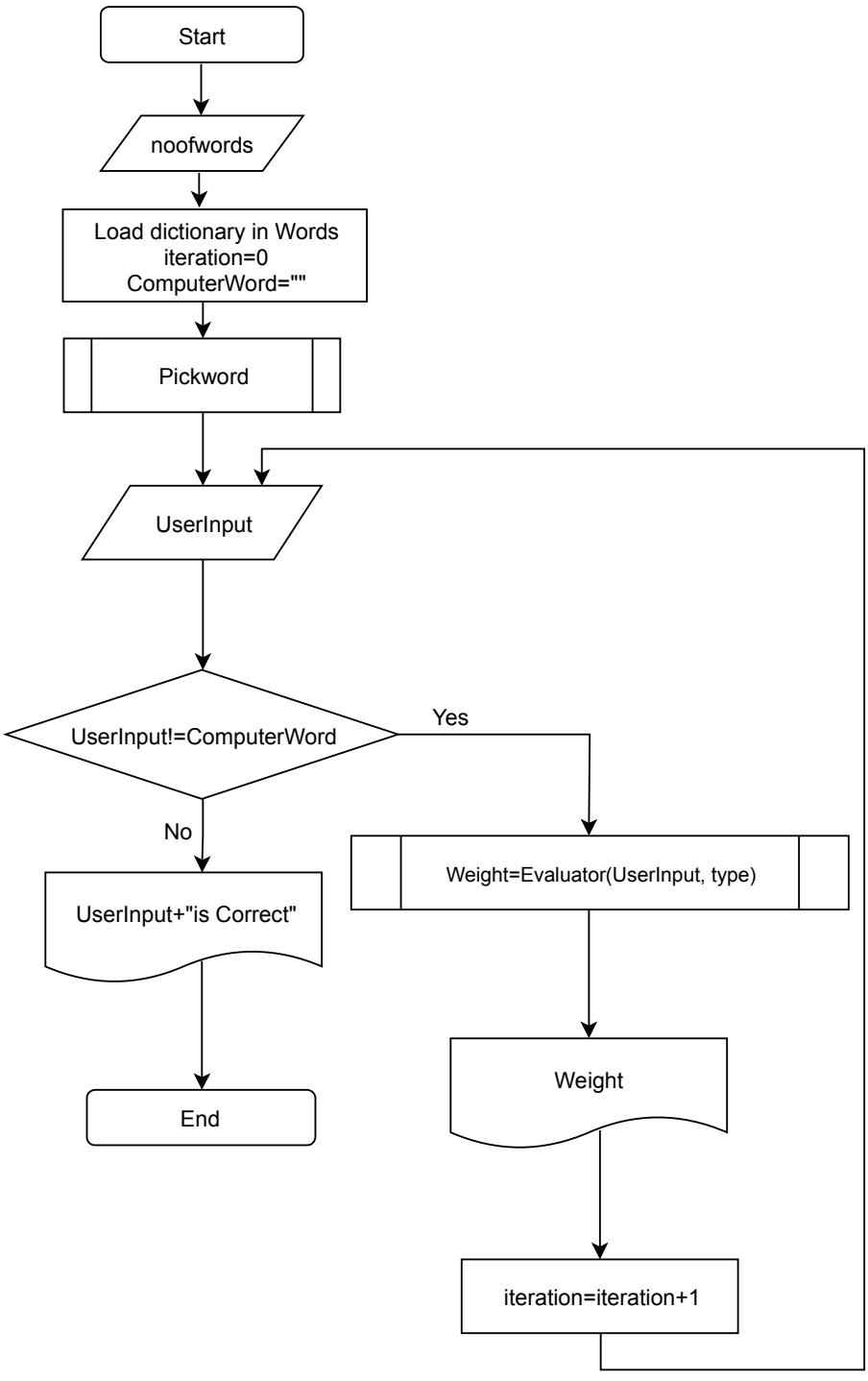
Note: rand() Function is a function that returns a random integer number

Files:

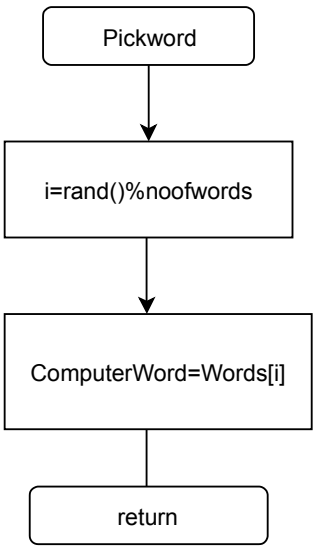
humanguess.h, humanguess.cpp and humanguess.ui:

The Evaluator Function that was used in the Computer-Human Mode was reused in this program for evaluating the weight of words.

Main Flowchart



Flowchart: Pickword()



3. Computer-Computer Mode (Analyzer):

In this mode, the computer selects a word and the computer guesses the word.
This mode is used for calculating the best and worst number of trials.
How to use this mode:
1. Load the dictionary.
2. Go to analyze and click on analyzer.
3. Input the number of random words to analyze.

This mode uses the same algorithm as the Computer-Human Mode. The only difference between the two modes implementation is that a function that imitates the human player was created in the **mainwindows.cpp** file to operate the Computer-Human Program.

The function that performs this operation is the **on_actionAnalysis_triggered()** function in the file.