Report(2017DS_Prog2)

Student ID:0310831

Name:簡銘賢

1. Pseudo Code:

1.**Construct** struct Node , set list<Node> and an iterator points to list<Node>.

2.**Read** a row from file to start_number by ifstream

3. **While** ifstream_data! =negative number

Do ifstream_data accesses to name of temp

Temp pushes back to list_chain

ENDWHILE

4.**While** getline !=NULL

DO

5**.** **If** (it is clockwise)

For I in 0 to size of word

If (iterator is end of list)

Then iterator is beginning of list

ENDIF

Push back the character to number which is

pointed by iterator

If (it is end of the word)

6. **Then** print the data to outputfile

If (the end of word is vowel)

Then clock_diretion doesn't change

Erase the gamer

Else

Then change the clock_diretion

Erase the gamer

ENDIF

ENDIF

ENDIF

7. **Else if** (it is anti-clockwise)

For I in 0 to size of word

If (iterator is beginning of list)

Then iterator is end of list

ENDIF

Push back the character to number which is

pointed by iterator

If (it is end of the word)

8. **Then** print the data to outputfile

If(the end of word is vowel)

Then clock_diretion doesn't change

Erase the gamer

Else

Then change the clock_diretion

Erase the gamer

ENDIF

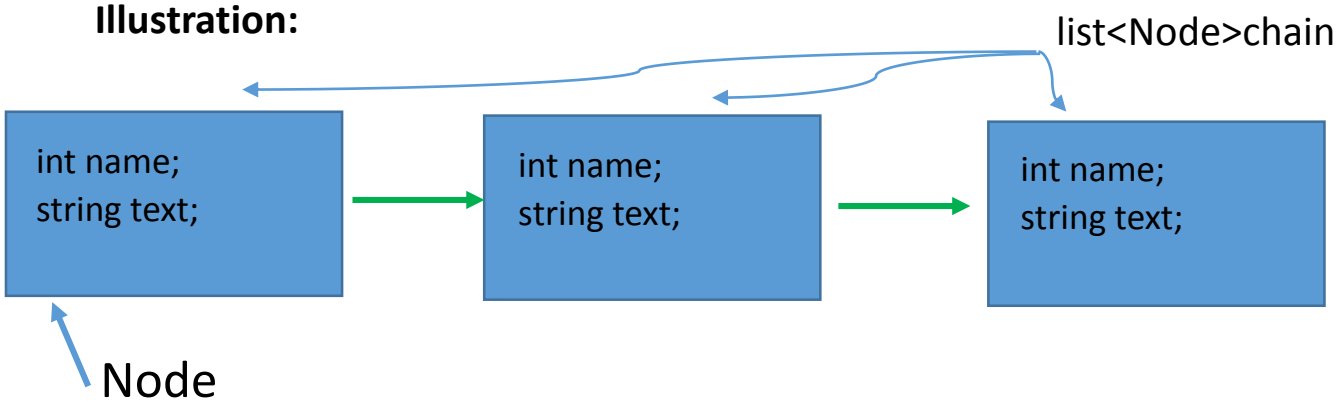
ENDIF

ENDIF

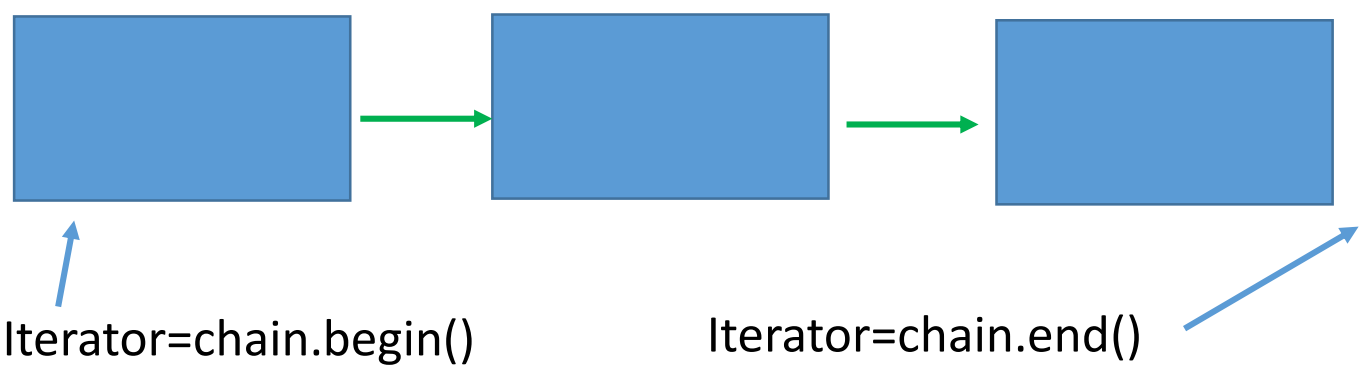
2. Approach Works:

1. Specially, I Construct a struct which is named “Node”, set list<Node> chain and an iterator points to list<Node>

Illustration:



2. Utilize the way of ifstream, we can get the chain.name.
3. Fill out the chain.name by loop.
4. We read the vocabulary row by row. Because each of word does not interact.
5. Determine if the clock is clockwise, and push back the char to chain.text. We also observe where the iterator is. If it is chain.end(), then we must move it to chain.begin()



6. If I read the end of word, we erase the gamer which receives the word. We also observe the end of word is vowel, and determine if change the direction of the list.
7. The direction is anti-clockwise, and we practice like step 5.
8. Like the step 6.

3. Time Complexity:

N: the number of people play the game

M: the max length of the vocabulary.

We must read the (N-1) rows vocabulary row by row. we Distribute every char of the vocabularies to each gamer, and we also know the max size of the word is M. So we can ensure time complexity of this program is equally less than (N-1)*M. This means the time complexity for big O is O(N*M).

4. The challenge in this work

Because many marginal conditions in this work, so I spent a big part of time arranging the order of these conditions. In addition, we have to aware of changing the iterator and ensure it is legal for the alteration, or might cause segment default error.