**Project 1**

**<Wordle Game>**

**Cis-17C**

**Name: Anangafac Alabaweh**

**Date: 06/11/22**

**Profressor: Mark Lehr**

**Introduction**

I was drawn to the wordle game because it is a great way to test one's Primary English understanding and vocabulary. Furthermore, I usually experience word blocks in conversations, and Wordle has been an excellent resource for expanding my vocabulary. Moreover, Wordle is a simple game that tests one's ability to think critically.

             The game consisted of 838 lines, three different difficulty levels, a class, four queues, two stacks, and five lists. I spent sixteen days coding it. I used the wordle game from [the new york times website](https://www.nytimes.com/games/wordle/index.html) and mikhad.github.io/wordle/ GitHub repository as references.

**Approached to Development**

When I wrote the first engine for the game, it had a lot of errors; it frequently crashed without reason and could not determine duplicate values.

**The Approached  used for the current version Is;**

1. My first step was to create a new list called copy\_p2list and copy the word entered by the player into this list.

First, I created a function (process1) for determining whether a character is in the correct location.

If there is a character in the correct position, I replace it with an uppercase 'T' and store the character's current position in a new list. The comparison was accomplished by using two queues.

2)      After process1 is complete, the program proceeds to process 2. This is done by comparing the user's word with the current wordle word. Because I had already used process1 to determine if any words were in the correct position. In the process2 function, the characters are compared, and if they are not the same, the character is searched using copy\_p2list as the library. If the character is in the copy\_p2list, it is replaced with zero, followed by replacing zero with 'F'. This helps determine and eliminate faulty outputs.

Additionally, if the character is not found in the copy\_p2list list, it means the character is not part of the wordle word, so it is replaced with an asterisk "\*", which indicates the character is not part of the wordle puzzle.

After the process1 and process2 functions have completed executing, the word entered by the user is saved first in queue\_results, followed by the results from the copy\_p2list list. It is copied to the queue\_results and displayed to the player.

3)      Word\_gen: I used a hash in this function to randomly pick words from the hash, based on the game level entered by the user. Also in the word\_gen function, the array is recursively sorted before

4) Palindrome\_check: The palindrome check is an additional feature I added to the game; this function determines if the word entered by the player is a palindrome; I use the front of the stack and queue, compare them; If they are all the same, the word is a palindrome;

5) display\_queue: this function displays the word entered by the player and also the results after going through process1 and process2.

6) The four\_word, five\_word, and six\_word functions are for each level. There are four-word, five-word, and six-word levels. Depending on the level selected by the player, the user is prompted to enter the number of letters of words.  If the player's number of letters is less or more than the required amount, the game will keep prompting the player to enter the correct number of letter words. When the player enters a word with the correct number of letters, it is copied to mylist character by character. This is then passed to both processes 1 and 2 for processing; the result of the game is then copied from mylist to queue\_result using a range base for loop;

**Game Rules:**

-          Depending on the level of the game, the user has to enter the required number of words; if the user enters less or more, they won't be able to go to the next stage of the play.

-          The user  has only 6 attempts to determine the wordle word

-          The player has to enter only words in the English dictionary

**Results Display:**

**T**= The character is in the  wordle word and also in the correct position

F= The Character  is  in the wordle word but in the wrong position

**\***= The Character is not in the wordle word

If the player gets all T, for example, **"T T T T T,"** the player won. If there is any "**F**"in the result,  the player didn't win

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **Location** |  |
| **Iterator** | it | Random-Access iterator | Process1 |  |
|  | \*it, \*it1 | Insert iterator | Process1 |  |
|  | it2 | Random-Access iterator |  |  |
|  | \*it2 | Insert iterator | process2 |  |
|  | it3 | Bidirectional iterator | process2 |  |
|  | it\_toli++ | forward iterator | display results |  |
|  | mylist\_it | Random-Access iterator | fivewords, fourwords, sixwords |  |
|  | count\_it | forward iterator | fivewords, fourwords, sixwords |  |
| **list** | copy\_p2list | container that holds char values | process2 |  |
|  | mylist1 | container that holds char values and the wordle word | four\_words, five\_words, six\_words |  |
|  | mylist | list container that holds the char characters of the word entered by the user | Main, process1, process2, fivewords, fourwords, sixword |  |
|  | tocheli | integer list container that holds the position of every character | Main, process1, process2, fivewords, fourwords, sixword |  |
|  | results |  |  |  |
| **stack** | stk | stack/list container used to compare if the word entered by the user us a palidrome | Main, palidrome\_check |  |
| **Queue** | que | queue/list of char used to compare if the word entered by the use is palidrome | Main, palidrome\_check |  |
|  | que, que1 | use to compare the characters from mylist, and mylist1 | Main, process1, process2 |  |
|  | queue\_result | queue/list char container holds the results and word after every entry | Main |  |
| **Map** | words | Map/set associative container holds the wordle words for each level | word\_gen |  |
| **Set** | words | Map/set associative container holds the wordle words for each level | word\_gen |  |
| **Algorithm** | Advance | used to increment the iterator so it randomly selects a word from the set | word\_gen |  |
|  | find | used to the search if a character is part of the Wordle | process1, process2, |  |
|  | count | used to count the occurrence of T, if the player got the right word | main, process1, process2, fivewords, fourwords, sixword |  |
|  | sort | used to sort the Map/set-associative container | word\_gen |  |
|  | swap | used the clear the queue in the stack after every level play | clear |  |
| **integer** | trail\_count | hold the number of times the player has played | main |  |
|  | mylist\_ct | hold the number of times the player has played | Main, process1, process2, fivewords, fourwords, sixword |  |
|  | counter | hold the number of times the player has played | Main, process1, process2, fivewords, fourwords, sixword |  |
|  | n1 | variable passed in the display\_queue function | display\_queue |  |
| **string** | name | use to hold the words entered by the player | Main |  |
|  | line | use to the hold the words read the rules files | Main |  |
|  | word | use to hold the random word generated by the word\_gen function | fivewords, fourwords, sixword |  |
| **Char** | m1, m2 | use to hold the characters from the que and que1 before they are popped | process1, process2, |  |
|  | st,qu | use to hold the characters from the que and que1 before they are popped | Palidrome\_check |  |
|  | Welcome\_option | use to hold the player choice from the sub menu | Main |  |
|  | play\_option | use to hold the player choice from the main menu | Main |  |
| **Class** | AVL tree | This class is used to hold the results in an AVL tree | Main |  |
|  | Hash | This class is used to hash the words in the words array and also save thec content of the array in a hash | Main |  |
| **Recurrsion** | recursive bubble sort | This function recursively sort the array alphabetically from A to Z |  |  |
|  | lowerupper recursionn | This function recursively turn lowercase characters to uppercase |  |  |

Pseudocode:

**Process1**

Start

While loop start

Copy the characters with the wordle word characters

If the are the same, replace with T

Pop the queues

While loop end;

**Process2:**

Start  
 Copy the results from process1 to a new list

Compare the characters, if they are not the same,

Search for the character in the p1list;

If found, replace with F

Else replace with \*

End;

**Clear:**

Start

Create a new queue and swap it with the old queue;

End

**Palidrome\_check:**

Start

Compare the word enter by the use

Using the stack and queue

If the stack is equal to the queue

The word is a palindrome

else the word is not a palindrome

end

**Word\_gen:**

Start

Randomly select words from the set depending on the key

Return the word;

End

**Four\_words:**

Start

Prompt player to enter 4 letter word

Convert the word to characters and copy it to mylist

Call process1 and process2

Call the display function;

Clear all the containers

End;

**Five\_words:**

Start

Prompt player to enter 5 letter word

Convert the word to characters and copy it to mylist

Call process1 and process2

Call the display function;

Clear all the containers

End;

**six\_words:**

Start

Prompt player to enter 6 letter word

Convert the word to characters and copy it to mylist

Call process1 and process2

Call the display function;

Clear all the containers

End;

**Main:**

Start

Prompt user to select from menu

If the player selects 1:

Prompt the player to select the level they want to play

If 1: start the four-word function

If 2: start the five-word function

If 3: start the six-word function

If 4: quit the game:

If the player select 2:

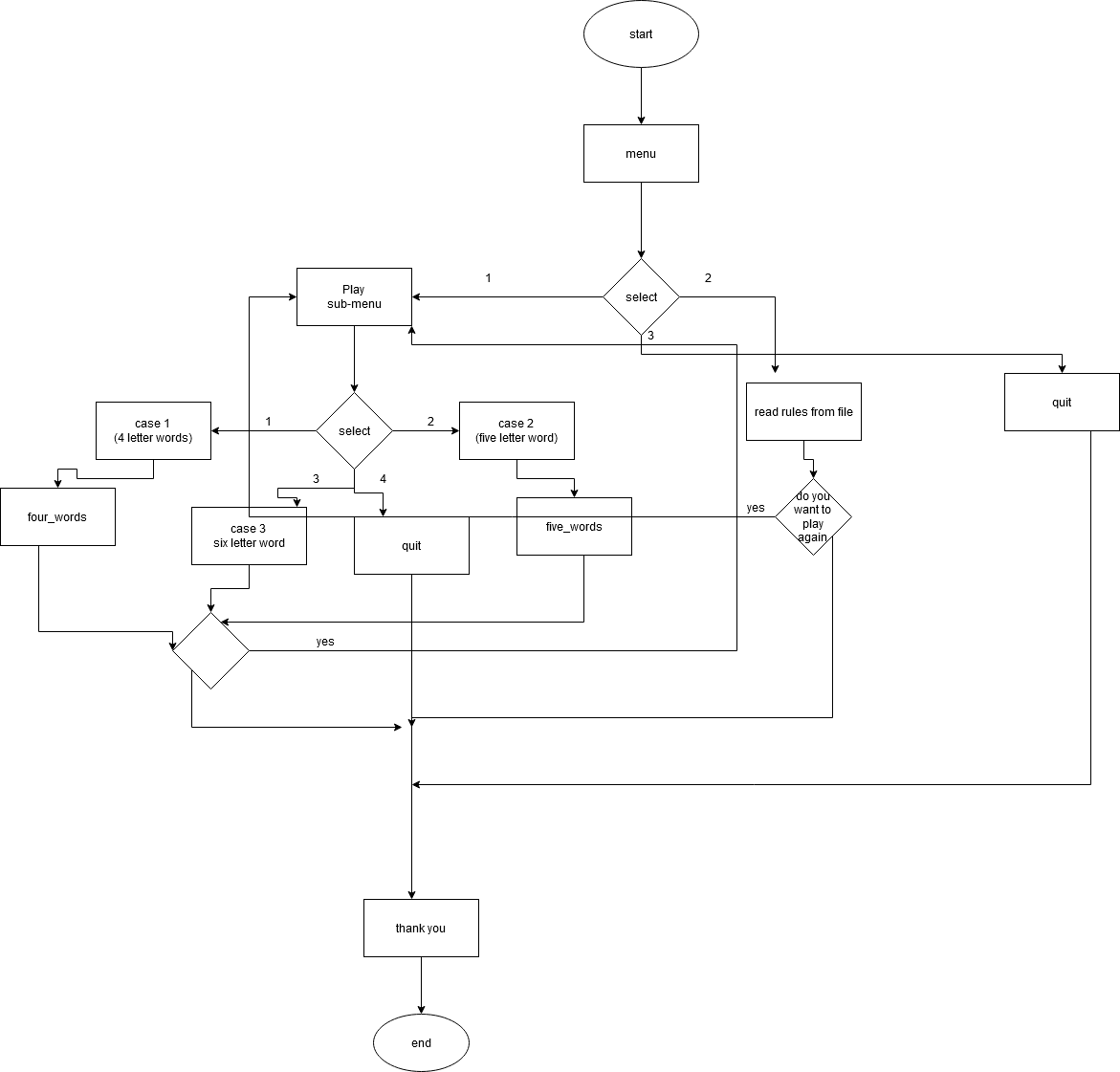
Display the rules of the game

If the player selects 3:

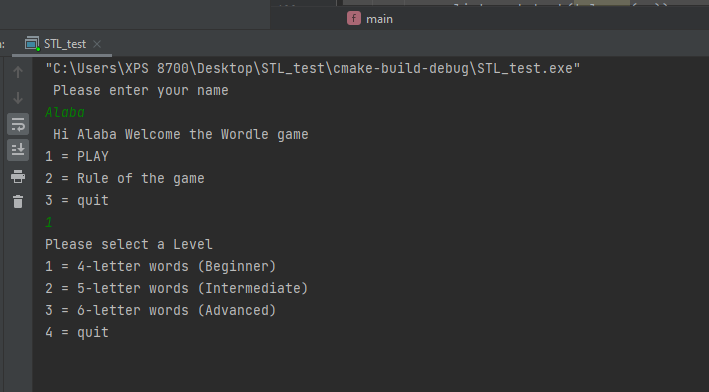
Quit

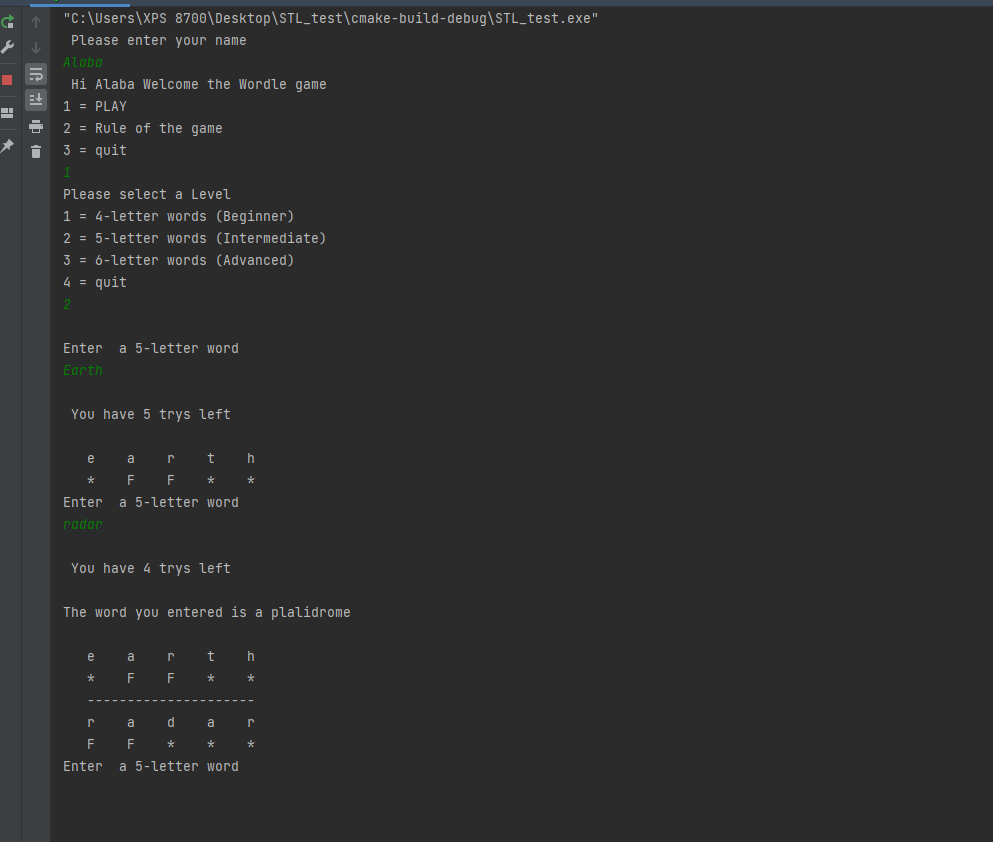
End

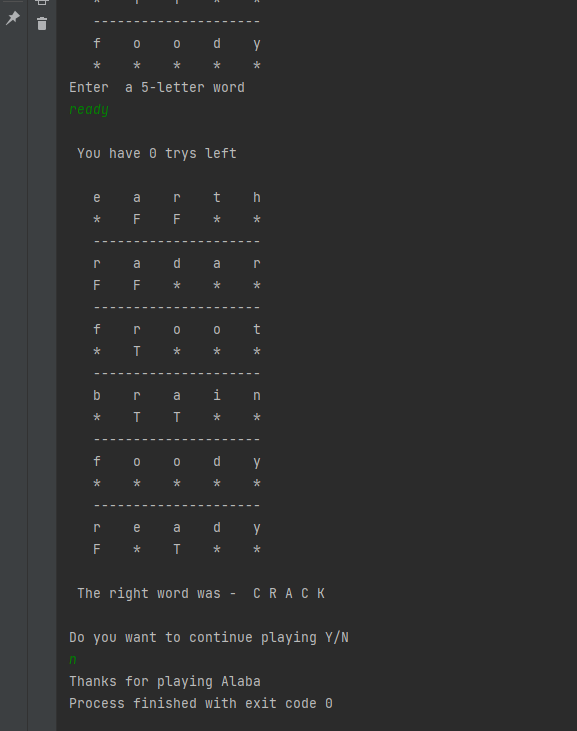
Sample Input/ output



**Sample input/output**







Checkoff Sheet Contents



**Reference:**

The C++ Standard Library: Second edition, Nicolai M Josuttis

C++ From control structures through Objects; Ninth edition, Tonny Gaddis

Wordle, New your times; <https://www.nytimes.com/games/wordle/index.html>

GeeksforGeeks; <https://www.geeksforgeeks.org>