Wordle Game

Group 6 :

Edward Alvin Kusmanto : 2501963141

Galih Kamulyan : 2201807772

Jairo Siegers : 2502042971

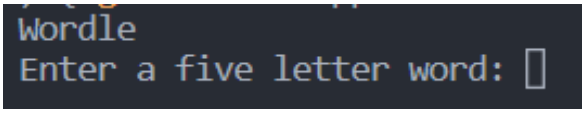
Table of Contents

I. Program Description

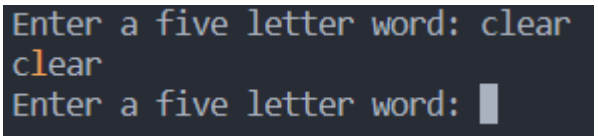
This is a simple “wordle” game running on C++ console. “Wordle” is a simple game where you have 6 tries to guess a word that contains 5 letters. Every failed attempt will grant you a hint, only if the letters of the word you’ve inputted are also contained in the word you have to guess. For example, if you’ve inputted the word chair whereas the word you have to guess is chill, then the game will grant you the hint which is the letter “c” and “h” since both of the words have both of those letters. If you’ve guessed the correct word in under 7 tries, you win.

It uses a 5 letter word dictionary to get a word for us to guess, to verify if the word we inputted exists, and to verify if the word we’ve inputted is correct. Each letter that has been inputted to guess the answer gets verified through an algorithm whether they are contained in the string you have to guess, either in the correct position or the wrong position. There will be a clue printed under the word you’ve inputted, green-coloured letter means the correct letter in the correct place, orange-coloured letter means the answer contains that letter but in the wrong place, and white-coloured letter means it the letter is not contained in the answer. For example, the answer is chair. If we were to guess the word “castle”, the clue would be “castle” since the letter C is the correct letter in the correct placement, the letter A is contained in the answer’s word but at the incorrect placement, the letter S, T, and E that isn’t present in the answer. The algorithm will grant you a win if you have guessed the correct string in under 6 tries.

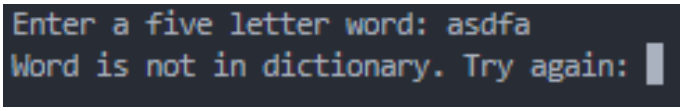
II. Application Flow



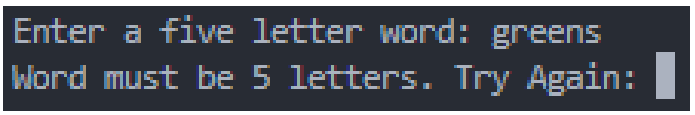
Start of the game.
Player must input a five letter word as their first try.



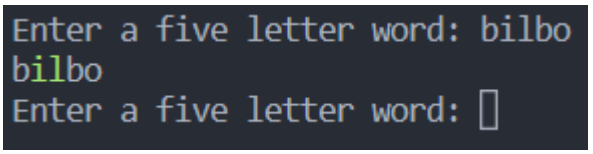
If there is a correct letter out of position.
Player will then input another letter until their last try.



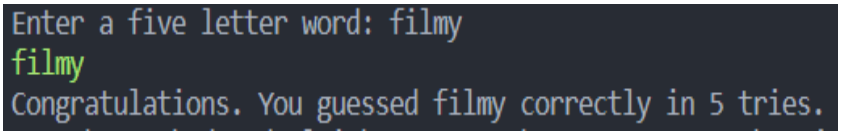
If the word entered by the user isn't in the dictionary, it will prompt the user to enter another word.



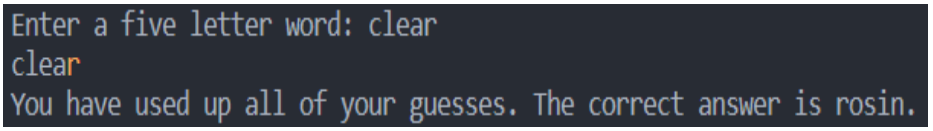
If the word entered by the user isn't a 5 letter word, it will prompt the user to enter another word.



If there is a correct letter in the correct position.
Player will then input another letter until their last try.



If the word is guessed correctly.
Player wins the game.



If the word has not been guessed correctly in 6 tries, it will show a message of defeat and show the answer.
Player loses the game.

III. Lessons that Have Been Learned

In this project, our team has done moderately well considering the fact that some of the algorithms we've made were learned independently. For example, the algorithm that verifies the guess string chars so it would return

either a green-coloured letter (correct letter in the correct placement) and an orange-coloured letter (correct letter in the wrong placement). We've revamped the code twice to find a better way to iterate through the string and verify them into the algorithm logic. Also, the algorithm that keeps the game running until the answer is guessed correctly was pretty challenging. We had problems doing this sort of algorithm coding, but after a lot of trial and error, we've succeeded.

IV. Code Explanation

```
string getRandomWord(string file)
{
    ifstream in(file);
    vector<string> words;
    string word;

    srand(time(0));

    while (in >> word)
    {
        words.push_back(word);
    }

    return words[rand() % words.size()];
}
```

This function is used to take a word from a txt file. srand(time(0)) is used to take a random value every time we run the file. Otherwise, we will get the same value over and over again each time we run the file.

```
bool checkWord(string file, string guess)
{
    ifstream in(file);
    vector<string> words;
    string word;
    bool check = false;
    int i = 0;

    while (in >> word)
    {
        words.push_back(word);
    }

    for (int i = 0; i < words.size(); i++)
    {
        if (guess == words[i])
        {
            check = true;
        }
    }
    return check;
}
```

This function is used to check if the word is in the dictionary or not. If it is, it will return true; else, it will return false.

```
void changeColor(int desiredColor)
{
    SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), desiredColor);
}
```

This function is used to change the font colour of the guesses so that it displays if that letter is in the right position or otherwise.

```
bool validateAnswer(string answer, string guess)
{
    vector<char> clue = {'-', '-', '-', '-', '-'};
    string clues;
    vector<bool> flags = {false, false, false, false, false};

    for (int i = 0; i < 5; i++)
    {
        if (guess[i] == answer[i])
        {
            clue[i] = 'G';
            flags[i] = true;
        }
    }

    for (int i = 0; i < 5; i++)
    {
        if (clue[i] == '-')
        {
            for (int j = 0; j < 5; j++)
            {
                if (guess[i] == answer[j] && !flags[j])
                {
                    clue[i] = 'Y';
                    flags[j] = true;
                    break;
```

We first create a vector named clue which will store 5 ‘-’ for its value. It will represent the colour of the letters. We then create a string which we will use to display the clues that have been checked. The flags variable will ensure that a duplicate letter will not show to be correct in case if there’s no double of that letter in the answer. The first for loop will iterate through the guess and check if there is any correct letter in the correct position and turns that position in the clue into ‘G’. The second for loop along with a nested one inside of it will check if there is a correct letter in an incorrect position and turns that position in the clue into ‘Y’.

```
for (int i = 0; i < 5; i++)
{
    clues.push_back(clue[i]);
}

for (int i = 0; i < 5; i++)
{
    if (clues[i] == 'G')
    {
        changeColor(10);
        cout << guess[i] << RESET;
    }
    else if (clues[i] == 'Y')
    {
        changeColor(14);
        cout << guess[i] << RESET;
    }
    else
    {
        cout << RESET << guess[i];
    }
}
cout << endl;
```

This for loop will append the clue in every position into clues and it changes the font of that letter according to the validation algorithm of the function.

```
if (clues == "GGGGG")
{
    return true;
}
else
{
    return false;
}
```

It will then return true if all the letters of the answer have been guessed correctly and return false if the answer has not been guessed completely.

```
int main()
{
    string guess;
    int tries = 0;
    int guesses = 6;
    bool guessed_correctly = false;
    string answer = getRandomWord("words.txt");

    cout << "Wordle" << endl;

    while (tries < 6 && !guessed_correctly)
    {
        cout << "Enter a five letter word: ";
        cin >> guess;
        while (guess.length() != 5){
            cout << "Word must be 5 letters. Try Again: ";
            cin >> guess;
        }
        while (checkWord("words.txt", guess) == false){
            cout << "Word is not in dictionary. Try again: ";
            cin >> guess;
        }
        guessed_correctly = validateAnswer(answer, guess);
        tries++;
        guesses--;
        cout << "Tries left: " << guesses << endl;
    }
```

The main function is where the program will run its logic. It will first create a guess which will take an input from the user. It will get an answer using the getRandomWord() function that we have created. It will then loop using a while loop with the parameters if tries is less than 6 and the word hasn't been guessed correctly. It will take the user input and validate if the length of the word guessed is equal to 5 or not. If it is a 5 letter word, it will check the guess using the checkWord() function that we have created before to ensure that the word is a valid one from the dictionary. If it is, then we will iterate through the guess and validate which letters are there inside the answer using the validateAnswer() function.

```
if (guessed_correctly)
{
    cout << "Congratulations. You guessed " << answer << " correctly in " << tries << " tries." << endl;
}
else
{
    cout << "You have used up all of your guesses. The correct answer is " << answer << "." << endl;
}
```

If the word has been guessed correctly it will show the message of congratulations and if not, if it has used up all 6 tries, it will show the answer and terminate the program.

V. Alternative Data Structure

An alternative data structure that can be used in making a wordle game is array. With the same capabilities of vectors, arrays are able to be traversed and elements inside the array can be accessed by using index. The main difference of the 2 is the sizing of them. Vector is dynamic meaning that the size of the vector doesn't need to be stated before. We use this advantage to get the words from "words.txt" into a vector so that the size doesn't need to be declared at the start. Although arrays are more efficient in accessing the elements inside, in our case of using the vectors to hold 5 elements inside it, doesn't make that much of a difference in both time and space complexity.

VI. Project Link

https://github.com/fl-sll/DataStructureFP

VII. References

- https://www.daniweb.com/programming/software-development/threads/30942/getting-words-from-txt-file
- https://www.javatpoint.com/how-to-split-strings-in-cpp
- https://github.com/dwyl/english-words
- https://github.com/ggleblanc2/wordle
- https://www.youtube.com/watch?v=MvX4tVETjHk
- https://github.com/SimonVutov/WordleGUI/blob/main/GUI.java
- https://www.thewordfinder.com/wordle-solver/