Documentation du Code – Wordle Solver

Nom Étudiant

2025

Introduction

Le mini-projet *Wordle* en langage C a pour objectif d'appliquer des concepts algorithmiques dans un contexte pratique en réalisant à la fois une version jouable du jeu et un solveur intelligent. Le solveur doit deviner un mot secret de 5 lettres en utilisant uniquement le retour fourni après chaque tentative, ce qui nécessite une stratégie efficace et des structures de données adaptées. Au-delà de l'implémentation, le projet met également l'accent sur l'analyse : comprendre comment le solveur réduit le dictionnaire de mots possibles, comment cela influence les performances, ainsi que le comportement global de l'algorithme en termes de complexité temporelle et spatiale. Ce rapport résume les idées essentielles de notre approche, explique nos choix de conception et illustre l'implémentation à travers des exemples de code documenté.

Le code du programme

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "wordleGame.h"

#define WORD_LEN 6
#define MAX_ATTEMPTS 6
#define MAX_WORDS 1000000

//  viter  stack overflow     tableau global
char word[MAX_WORDS][WORD_LEN];

// fonction pour lire des mots d un fichier
int read_dictoinary(char Words[][WORD_LEN], const char *filename)
   {
    FILE *f = fopen(filename , "r");
    if( f == NULL){
        printf("erreur!!");
        return 0;
    }

    int count = 0;
```

```
23
24      while(fscanf(f , "%5s", Words[count]) != EOF){
25
26          // convertir en minuscules
27          for(char *p = Words[count]; *p; p++){
28              if(*p >= 'A' && *p <= 'Z'){
29                  *p += 32;
30              }
31          }
32
33          count++;
34          if(count >= MAX_WORDS){
35              break;
36          }
37      }
38
39      fclose(f);
40      return count;
41  }
42
43  // procedure feedback
44  void feedback(char guess[], char word[]){
45      for(int i = 0; i < 5; i++){
46          if(guess[i] == word[i]){
47              printf("%c est dans le mot et dans la bonne position\
                    n", guess[i]);
48          }
49          else {
50              int found = 0;
51              for(int j = 0; j < 5; j++){
52                  if(guess[i] == word[j]){
53                      found = 1;
54                      break;
55                  }
56              }
57
58              if(found)
59                  printf("%c est dans le mot mais a la mauvaise
                        place\n", guess[i]);
60              else
61                  printf("%c ne se trouve pas dans le mot\n", guess
                        [i]);
62          }
63      }
64  }
65
66  int main(){
67      int count = read_dictoinary(word, "words.txt");
68      if(count == 0){
69          return 1;
70      }
```

```c
    // choisir aleatoirement un mot du fichier
    srand(time(NULL));

    char tragect[WORD_LEN];
    strcpy(tragect , word[rand() % count]);

    printf("======WORDLE GAME====== \n");
    printf("vous avez %d tentative pour deviner le mot secret. \n
       ", MAX_ATTEMPTS);

    for(int attempt = 0; attempt < MAX_ATTEMPTS; attempt++) {
        char guess[WORD_LEN];
        printf("tentative %d : ", attempt + 1);
        scanf("%5s", guess);
        feedback(guess, tragect);
        if (strcmp(tragect , guess) == 0){
            printf("BRAVO!, vous avez trouve le mot '%s' \n",
               tragect);
            return 0;
        }
    }

    printf("PERDU, le mot etait : %s", tragect);

    return 0;
}
```

a. Strategy Description

b. Data Structure Justification

c. Complexity Analysis

d. Documentation des Fonctions

1. `read_dictoinary()`

Rôle : Cette fonction lit un fichier contenant des mots de 5 lettres et les stocke dans un tableau 2D global. Elle renvoie le nombre total de mots lus.

Entrées :

— `Words` : tableau 2D où seront stockés les mots

— `filename` : nom du fichier contenant les mots

Sortie : Nombre total de mots valides trouvés dans le fichier.

Logique d'algorithme :

— Ouvre un fichier en lecture

— Lit chaque mot avec `fscanf()`

— Convertit toutes les lettres en minuscules

— S'arrête lorsque le tableau est plein ou que le fichier est terminé

2. `feedback()`

Rôle : Affiche un retour pour chaque lettre entrée :
— bonne lettre + bonne position
— bonne lettre mais mauvaise position
— lettre absente
Entrées :
— `guess[]` : mot proposé par l'utilisateur
— `word[]` : mot secret à deviner
Logique :
— Parcourt toutes les lettres du mot proposé
— Compare la lettre à la même position
— Sinon, cherche si la lettre apparaît ailleurs dans le mot secret

3. `main()`

Rôle : Fonction principale qui :
— charge la liste des mots
— choisit un mot aléatoire
— gère les tentatives de l'utilisateur
— appelle la fonction `feedback()`
— annonce victoire ou défaite
Composants principaux :
— Générateur aléatoire avec `srand()`
— Sélection aléatoire d'un mot dans le dictionnaire
— Boucle de tentatives (6 essais maximum)

Exemple de Code Commenté (Snippet)

```
1  // Retourne le nombre de mots lus depuis le fichier
2  // Parametres :
3  //   Words[][] : tableau 2D ou stocker les mots
4  //   filename  : nom du fichier dictionnaire
5  // Retour :
6  //   nombre total de mots lus
7  int read_dictionary(char Words[][WORD_LEN], const char *filename)
       {
8      FILE *f = fopen(filename, "r");
9      if (!f) {
10         printf("Erreur␣d'ouverture␣!\n");
11         return 0;
12     }
13
14     int count = 0;
15     // Lecture mot par mot
16     while (fscanf(f, "%5s", Words[count]) != EOF) {
```

```
17          // Conversion en minuscules
18          for (char *p = Words[count]; *p; p++) {
19              if (*p >= 'A' && *p <= 'Z')
20                  *p += 32;
21          }
22          count++;
23      }
24      fclose(f);
25      return count;
26  }
```