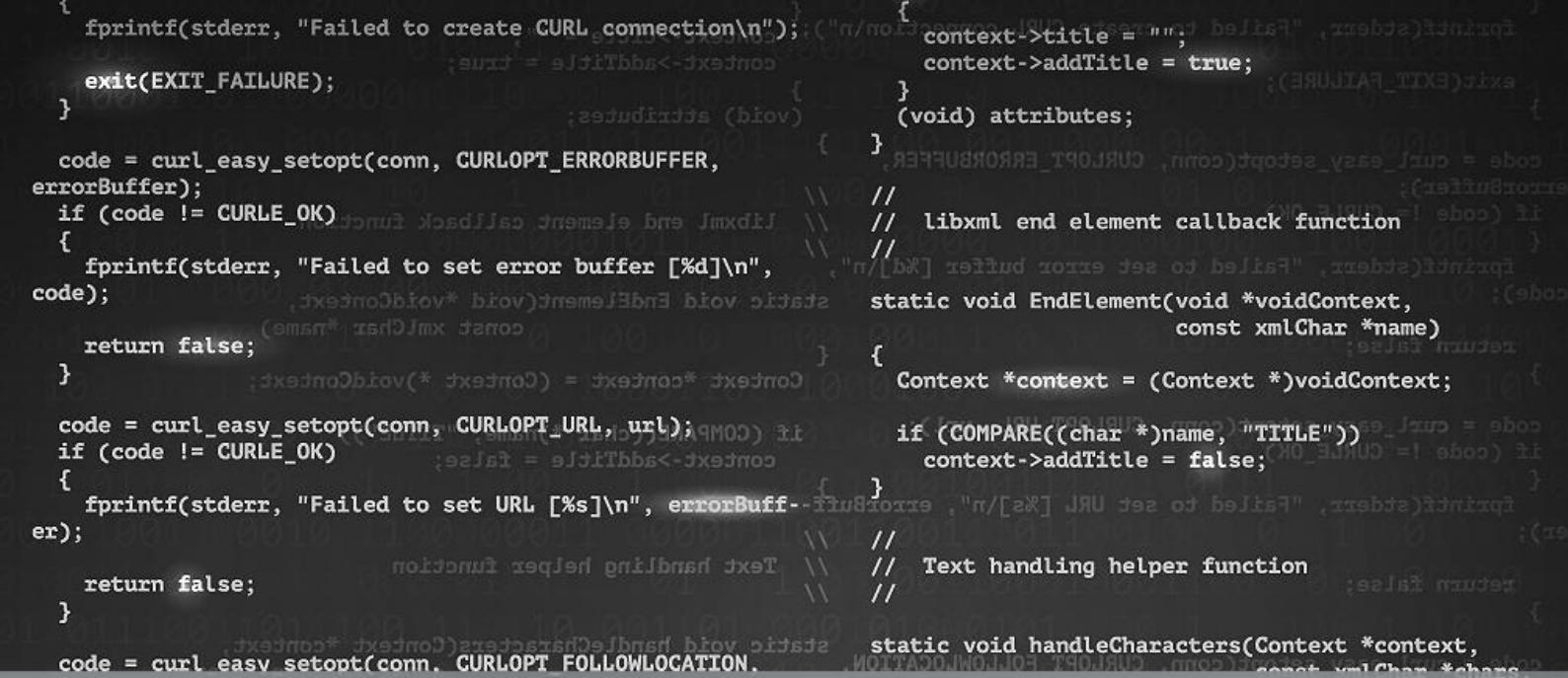

Institut Supérieur de Gestion de Tunis

RAPPORT DU PROJET ASD1

PRINCIPE DU JEU:Le joueur est appelé à trouver le mot valide le plus long à partir d'un ensemble de lettres alphabétiques choisies demanière aléatoire et ce au bout d'un nombre fini de tentatives.Un mot est valide s'il appartient à une liste de mots saisis à l'avance à partir d'un fichier texte qui représente le dictionnaire Français.

Réalisé par :
Timoumi Bassem
1 LNIG 04 _ TP 1

Encadré par :
Prof. Mahe Fraj

Année universitaire : 2023-2024

1- Les bibliothèques utilisées

<stdio.h> est un en-tête de bibliothèque standard en C fournissant des fonctions pour les opérations d'entrée/sortie, y compris la manipulation de fichiers. Il est essentiel pour les tâches impliquant l'entrée/sortie console et la manipulation de fichiers dans les programmes en langage C.

<string.h> est un en-tête de bibliothèque standard en C qui fournit des fonctions pour la manipulation de chaînes de caractères, y compris des opérations telles que la copie, la concaténation, la comparaison et la recherche. Il est crucial pour la manipulation d'arrays de caractères et de chaînes de caractères dans les programmes en langage C.

<stdlib.h> est un en-tête de bibliothèque standard en C qui fournit des fonctions liées à l'allocation de mémoire, à la génération de nombres aléatoires, aux fonctions de conversion, et à d'autres utilitaires généraux. Il comprend des fonctions telles que malloc, free, atoi et rand, jouant un rôle clé dans la gestion de la mémoire et les opérations utilitaires de base.

<time.h> est un en-tête de bibliothèque standard en C qui fournit des fonctions pour manipuler la date et l'heure. Il inclut des fonctions pour obtenir l'heure actuelle, formater les valeurs temporelles et calculer les différences entre les moments dans le temps. Il est couramment utilisé pour les opérations liées au temps en programmation en langage C.

<ctype.h> est un en-tête de bibliothèque standard en C qui fournit des fonctions pour tester et manipuler des caractères. Il inclut des fonctions telles que isalpha, isdigit, toupper et tolower, qui sont utilisées pour vérifier et transformer des caractères. Cet en-tête est particulièrement utile pour les opérations basées sur les caractères et la classification des caractères.

<windows.h> est un fichier d'en-tête pour l'API Windows en programmation C et C++, fournissant des fonctions pour des opérations spécifiques à Windows telles que la création d'interfaces graphiques (GUI), la gestion des messages, et des tâches liées au système. Son utilisation rend le code spécifique à Windows et peut ne pas être portable sur différents systèmes d'exploitation.

<conio.h> est un fichier d'en-tête en programmation C qui signifie "console input/output". Il fournit des fonctions comme getch() et _kbhit() pour les opérations d'entrée/sortie basées sur la console. Cependant, il est important de noter que <conio.h> n'est pas un en-tête de bibliothèque C standard et n'est pas portable entre différents compilateurs ou plates-formes.

2- Les fonctions prédéfinies utilisées

strlen() est une fonction en C qui calcule la longueur d'une chaîne de caractères. Elle prend une chaîne en entrée et compte le nombre de caractères dans la chaîne jusqu'à ce qu'elle rencontre le caractère de fin de chaîne nul ('\0'). Le résultat est la longueur de la chaîne, en excluant le caractère nul.

strcmp() est une fonction en C qui compare deux chaînes de caractères. Elle prend deux chaînes en entrée et renvoie une valeur entière qui indique la relation entre les deux chaînes. La valeur de retour est zéro si les chaînes sont égales, une valeur négative si la première chaîne est lexicographiquement inférieure à la deuxième, et une valeur positive si la première chaîne est lexicographiquement supérieure à la deuxième.

strcpy() est une fonction en C utilisée pour copier le contenu d'une chaîne dans une autre. Elle prend deux arguments de chaîne : la chaîne de destination (où le contenu doit être copié) et la chaîne source (le contenu à copier).

_kbhit() est une fonction en C utilisée pour déterminer si une touche a été pressée sur le clavier. Elle est souvent utilisée dans des situations où l'on souhaite vérifier une saisie clavier sans bloquer l'exécution du programme.

Sleep() est une fonction en C, particulièrement sur les systèmes Windows, qui est utilisée pour mettre en pause l'exécution d'un programme pendant un nombre spécifié de millisecondes.

strchr() est une fonction en C qui recherche la première occurrence d'un caractère spécifié dans une chaîne de caractères. Si le caractère est trouvé, elle renvoie un pointeur vers la première occurrence du caractère dans la chaîne. Si le caractère n'est pas trouvé, elle renvoie un pointeur nul (NULL).

srand() est une fonction en C qui initialise le générateur de nombres aléatoires, en fournissant une valeur de graine pour la fonction rand(). Cette graine influence la séquence de nombres pseudo-aléatoires générée par rand().

system() est une fonction en C qui vous permet d'exécuter des commandes système à partir d'un programme. Elle prend une chaîne de caractères en tant qu'argument, représentant la commande à exécuter.

3- Implémentation

Cette section du code fonctionne en parcourant le fichier ligne par ligne afin de déterminer le nombre total de lignes. Ensuite, elle utilise la fonction rewind() pour réinitialiser le pointeur de fichier au début, puis elle alloue dynamiquement de la mémoire à l'aide de malloc() pour stocker ces lignes dans une matrice. Cette approche est nécessaire lorsque la taille de la matrice est trop grande pour contenir toutes les lignes du fichier, d'où la nécessité d'une allocation dynamique.

*main.c

```c
{
    FILE *f;
    int nb = 0;
    char fin;
    char ligne[27];

    f = fopen("dictionnaire.txt", "r");
    if (f == NULL) {
        fprintf(stderr, "erreur fichier \n");
        return 1;
    }


    while ((fin = fgetc(f)) != EOF) {
        if (fin == '\n') {
            nb++;
        }
    }
    rewind(f);
    char (*matrix)[27] = malloc(nb * sizeof *matrix);
    if (matrix == NULL) {
        fprintf(stderr, "Memory allocation error\n");
        fclose(f);
        return 1;
    }


    for (int i = 0; i < nb; i++) {
        if (fgets(ligne, sizeof(ligne), f) != NULL) {
            if (ligne[strlen(ligne) - 1] == '\n') {
                ligne[strlen(ligne) - 1] = '\0';
            }
            strupr(ligne);
            strcpy(matrix[i], ligne);
        }
    }

    fclose(f);
```

3- Implémentation

Cette section du code fonctionne en générant de manière aléatoire un caractère (consonne ou voyelle) en fonction du choix de l'utilisateur. Elle utilise la fonction rand() pour sélectionner un caractère aléatoire dans une chaîne prédéfinie, offrant ainsi à l'utilisateur la possibilité de faire ce choix de manière aléatoire.

main.c

```
37
38      char lettres [70]="";
39      char lettresobli [70]="";
40      char choix2;
41      int i;
42      for(i=1;i<=taille;i++){
43      do{
44      printf("Caractére %i : Consonne (c/C) ou Voyelle (v/V)= ",i);
45      scanf(" %c",&choix2);}
46      while(choix2!='c' && choix2!='C' && choix2!='v' && choix2!='V');
47      printf("\n");
48
49
50      char caractere;
51      if(choix2=='c'||choix2=='C'){
52      char consonnes[] = "BCDFGHJKLMNPQRSTVWXYZ";
53      int taillecons = sizeof(consonnes) - 1;
54      caractere = consonnes[rand() % taillecons]; }
55      else{
56      char voyelles[] = "AEIOU";
57      int taillevoy = sizeof(voyelles) - 1;
58      caractere = voyelles[rand() % taillevoy];
59      }
60
61
62      char temp1[7] = {' ',' ',' ',' ',' ',caractere, '\0'};
63      char temp2[7] = {caractere,'\0'};
64      strcat(lettres, temp1);
65      strcat(lettresobli, temp2);
66      };
67      printf("%s\n",lettres);
68
```

3- Implémentation

Cette section du code se consacre à la recherche de la chaîne la plus longue constituée des caractères aléatoires présents dans la matrice. Pour cela, elle utilise la fonction strchr(). Le programme procède ensuite à la suppression de l'indice du caractère trouvé dans la chaîne des lettres aléatoires, garantissant ainsi que chaque caractère est utilisé sans dépasser la quantité initialement spécifiée.

main.c

```
109
110
111         int k;
112         int max=0;
113         char ch[28];
114         int b;
115         int h;
116         char lettreoblitempo[28];
117         for (int j = 0; j < nb; j++) {
118             strcpy(ch,matrix[j]);
119
120             b=1;
121             strcpy(lettreoblitempo,lettresobli);
122             for (h = 0; h < strlen(ch); h++) {
123
124                 if (strchr(lettreoblitempo, ch[h]) == NULL) {
125                     b = 0;  // il ya un character qui n'existe pas dans lettresobli
126                     break;
127                 }
128
129
130                 else {
131                     char *resultat = strchr(lettreoblitempo, ch[h]);
132                     size_t position = resultat - lettreoblitempo;
133                     for (int p = position; p < strlen(lettreoblitempo) - 1; ++p) {
134                             lettreoblitempo[p] = lettreoblitempo[p + 1];
135                     }
136                     lettreoblitempo[strlen(lettreoblitempo) - 1] = '\0';
137                 }
138
139
140             }
141
142         if (b==1 && strlen(ch)>max){
143                 max=strlen(ch);
144
145
146             }
147
148         }
```

3- Implémentation

Cette partie du programme réalise une tâche similaire à celle précédemment mentionnée, mais avec une différence clé : au lieu de simplement récupérer la longueur du mot le plus long, le programme identifie maintenant ce mot lui-même. Il extrait le mot le plus long de la matrice en utilisant la même logique décrite précédemment, puis le stocke dans un tableau dédié.

```
151
152         char t[30][28];
153         char t3[999][28];
154         int y=0;
155         int m=0;
156         for (int j = 0; j < nb; j++) {
157         strcpy(ch,matrix[j]);
158
159         int b=1;
160         strcpy(lettreoblitempo,lettresobli);
161         for (h = 0; h < strlen(ch); h++) {
162             if (strchr(lettreoblitempo, ch[h]) == NULL) {
163                 b = 0;  // il ya un character qui n'existe pas dans lettresobli
164                 break;
165             }
166
167             else {
168                 char *resultat = strchr(lettreoblitempo, ch[h]);
169                 size_t position = resultat - lettreoblitempo;
170                 for (int p = position; p < strlen(lettreoblitempo) - 1; ++p) {
171                         lettreoblitempo[p] = lettreoblitempo[p + 1];
172                 }
173                 lettreoblitempo[strlen(lettreoblitempo) - 1] = '\0';
174
175             }
176
177
178             }
179             if(b==1){
180             strcpy(t3[m],ch);
181             m++;}
182         if (b==1 && strlen(ch)==max){
183                 printf("%s \n",ch);//juste pour tester et connaitre les mot les plus longs
184                 strcpy(t[y],ch);
185                 y++;
186
187             }
188         }
```

3- Implémentation

Dans cette section, le programme applique une méthode similaire à celle utilisée précédemment sur la matrice, mais cette fois-ci sur les mots fournis par l'utilisateur. Il utilise ces mots pour créer différentes réponses du programme en fonction des entrées de l'utilisateur.

*main.c

```
196     char reponse[26];
197     i=0;
198     int n=0;
199     int v=1;
200     while(i<taille && v==1){
201         i++;
202         printf("%i-   ",i);
203         scanf("%s",reponse);
204         k=0;
205         while (reponse[k] != '\0') {
206         reponse[k] = toupper(reponse[k]);
207         k++;
208         }
209
210         strcpy(lettreoblitempo,lettresobli);
211
212         b=1;
213         for (h = 0; h < strlen(reponse); h++) {
214
215
216                 if (strchr(lettreoblitempo, reponse[h]) == NULL) {
217                 b = 0;  // il ya un character qui n'existe pas dans lettresobli
218                 printf("Le mot que vous avez saisi est incorrect!\n");
219                 break;
220
221                 }
222                 else {
223                     char *resultat = strchr(lettreoblitempo, reponse[h]);
224                     size_t position = resultat - lettreoblitempo;
225                     for (int p = position; p < strlen(lettreoblitempo) - 1; ++p) {
226                         lettreoblitempo[p] = lettreoblitempo[p + 1];
227                 }
228                 lettreoblitempo[strlen(lettreoblitempo) - 1] = '\0';
229
230                 }
231             }
```

3- Implémentation

Cette section de code est dédiée au minuteur dans l'option 2 du jeu, qui partage des similitudes avec la première option, mais avec l'ajout d'une contrainte de temps. Elle utilise la fonction _kbhit() pour détecter si l'utilisateur appuie sur n'importe quelle touche du clavier. Si aucune touche n'est enfoncée pendant un laps de temps défini, le programme passe automatiquement à la tentative suivante, et ainsi de suite.

main.c

```
463     char t2[9][28];
464     char reponse[26];
465     int timeout = taille; // Timeout en secondes
466     int elapsed;
467     i=0;
468     int n=0;
469     int v=1;
470     while(i<taille && v==1){
471         i++;
472         printf("%i-   ",i);
473         elapsed = 0;
474         int test=1;
475
476
477
478         while (elapsed < timeout && test==1) {
479         Sleep(1000); // Sleep pour 1 seconde (1000 millisecondes)
480
481         if (_kbhit()) {
482         scanf("%s",reponse);
483         test=0;
484
485         }
486         elapsed++;
487         }
488         if (elapsed==timeout){
489         printf("\nVous avez dépassé le temps permis pour une tentative qui est de %is!.\n",taille);}
490
491
492
493
494         k=0;
495         while (reponse[k] != '\0') {
496         reponse[k] = toupper(reponse[k]);
497         k++;
498         }
```

3- Implémentation

Cette section de code correspond à l'option 3 du jeu, introduisant un autre joueur pour jouer en duel. Elle sollicite la volonté de jouer un autre tour et, à la fin, affiche le gagnant. Le score de chaque joueur est calculé simplement en fonction de la longueur du mot le plus long deviné.

*main.c

```
963     int choixf;
964     printf("1- Afficher le/les mot(s) le/les plus long(s)\n");
965     printf("2- Afficher tous les mots possibles\n\n");
966     do{
967
968     printf("                              Votre choix:");
969     scanf("%i",&choixf);}
970     while(choixf>2 || choixf<1);
971     printf("\n");
972     if (choixf==1){
973        for(int l=0;l<y;l++){
974            printf("%s \n",t[l]);
975        }
976     }
977     else{
978        if(choixf==2){
979            for(int l=0;l<m;l++){
980            printf("%s \n",t3[l]);
981        }
982
983        }
984     }
985     do{
986     printf("\n\nVous voulez continuez? Réponder par 'o' si oui et 'n' sinon = ");
987     scanf(" %c",&choixx);}
988     while (choixx!='o' && choixx!='O' && choixx!='n' && choixx!='N');
989
990    }
991
992     while (choixx=='O' || choixx=='o');
993
994     if (choixx=='n' || choixx=='N'){
995        if (score1==score2){
996            printf("égalité!");
997        }
998        else{
999
1000       if (score1>score2){
1001           printf("\nj1 est le vainqueur!");
1002       }
1003       else{printf("\nj2 est le vainqueur!");}
1004    }
1005    }
```

NB

Le programme intègre une validation de saisie pour les choix d'options et les données fournies par l'utilisateur, assurant ainsi la gestion appropriée des entrées utilisateur.

```
82      printf("Caractére %i : Consonne (c/C) ou Voyelle (v/V)= ",i);
83      scanf(" %c",&choix2);}
84      while(choix2!='c' && choix2!='C' && choix2!='v' && choix2!='V');
85      printf("\n");
86
```

```
61      printf("                                    Votre choix = ");
62      scanf("%i",&choix);}
63      while(choix>3 || choix<0);
64
65      if(choix==1){
66      int taille;
67      do{
68      printf("\n\nTapez la taille (7/8/9) du mot le plus long à retrouver = ");
69      scanf("%i",&taille);
70      if (taille>9 || taille<7){
71      printf("La taille que vous avez saisie est erronée!\n");};
72      }
73      while(taille>9 || taille<7);
74      printf("\n");
```