Projet ASD1

JEU DU PENDU ET JEU MOTUS

Blidi Abderrahmane - Chouria Mohamed amine

ISG TUNIS | 1LNIG

Sommaire

Tableau des figures

I. Étude de l'existence

1) Principe du jeu "Le pendu":

Il s'agit de présenter un mot masqué et de proposer au joueur de trouver une à une les lettres qui le composent. Le nombre de tentatives est limité.

A chaque fois que le joueur choisit une lettre qui existe dans le mot, toutes les occurrences de celle-ci dans le mot sont dévoilées. Donc, le nombre de tentatives n'est pas décrémenté.

Quand il choisit une lettre qui ne se trouve pas dans le mot, un décompte est effectué jusqu'à l'échec. Donc, le nombre de tentatives est décrémenté.

2) Principe du jeu "Motus":

C'est le même principe que le jeu "le pendu". Sauf qu'ici, il s'agit de trouver un mot masqué dont on ne connaît que la première lettre. De plus, le joueur doit saisir successivement des mots entiers. Le programme l'informe des lettres bien placées, des lettres mal placées et de celles qui n'appartiennent pas au mot recherché.

II. SOLUTION

Il s'agit de développer un programme en langage C avec une interface qui permet l'utilisateur de choisir l'un des deux jeux : le pendu et motus. Le programme va limiter le temps pour le *joueur* d'où il va jouer contre la montre et également compter le score en basant sur le nombre des coups et l'utilisation d'aide.

Après chaque match du l'un des jeux l'utilisateur a le choix de :

- Rejouer
- Vérifier le score
- Afficher le classement du jeu
- Retour au menu principal
- Quitter l'application

III. Réalisation

- Dans la fonction main on a affiché le menu qui va demander de choisir un jeu et indiquer les options par des numéros. Selon le numéro saisit par l'utilisateur le programme va appeler l'un des procédures pendu, motus ou ext.

1) La procédure pendu :

D'abord on a affiché un message de bienvenue puis demandé le pseudo du joueur dans un contrôle de saisie pour éviter l'acceptation d'une chaine vide.

Aux variables globales on a déclaré un tableau de 100 chaines de caractères. Ces derniers sont les mots qu'on va utiliser dans les deux jeux

La variable rnd reçoit une variable aléatoire entre 0 et 99 (le nombre des mots -1). Ch reçoit la case du tableau dict d'indice rnd (ch est un mot aléatoire du tableau des mots)

```
srand(time(NULL));
rnd = rand()%100;
strcpy(ch,dict[rnd]);
```

Figure 1:le mot aléatoire

Puis on a déclaré et rempli un tableau alph avec les lettres alphabétiques par une boucle pour de 'A' à 'Z'. Quand le joueur tape une lettre correcte on va la déplacer dans le tableau par un « - ».

La variable ch2 représente le mot masqué qu'il révèle son identité petit à petit lorsque le joueur tape une lettre correcte

a) Le compteur :

Le principe du compteur est de faire la soustraction du [temps de système maintenant + 70 secondes (le temps de fin du jeu)] et [le temps du système maintenant]. Le résultat est le temps restant.

Tant que les coups, qui sont initialisé à 7 au début et décrémenté quand le joueur tape une fausse lettre, sont supérieur à 0 et le temps restant est supérieur à 0 et la chaine saisit par le joueur (ch2) différente à le mot à trouver(ch) le joueur peut donner une lettre.

```
gettimeofday (&tv, NULL);
int tf = (int)tv.tv_sec+70;
int coups=7,pts,aide=0,b;
char c;
while ((coups>0)&&((tf-(int)tv.tv_sec>0))&&(strcmp(ch,ch2)!=0))
  {
      if((int)tv.tv_sec<=0)
          break;
      printf("\n");
      printf("Il vous reste %d coups a jouer et %d secondes \n",coups,tf-(int)tv.tv_sec);
      printf("Quel est le mot secret ? %s \n ",ch2);
```

Figure 2:La boucle d'un match

La fonction gettimeofday () récupère l'heure du système.

L'application demande au joueur de saisir un caractère. Ce dernier ne peut être qu'une lettre entre A et Z majuscule ou minuscule ou «?» pour demander l'aide. Alors nous faisons face à deux cas.

```
if ((c=='?')&&(aide==0)) {
    aide=1;
    do {
      srand(time(NULL));
      rnd = rand()%strlen(ch);
    }
    while (ch2[rnd]!='*');
    ltr=ch[rnd];
    for (i=0;i<strlen(ch);i++) {
        if (ch[i]==toupper(ltr)) {
            ch2[i]=toupper(ltr);
        }
    }
    i=0;b=0;
    while ((i<26)&&(b==0)) {
            if (alph[i]==ltr) {
                alph[i]='-';
                b=1;
            }
            else i++;
        }
}
```

Figure 3: Le code d'aide

b) L'aide :

Le joueur a l'avantage d'utiliser un seul aide. En tapant «?», Sans changement du nombre des tentatives, l'application va afficher l'un des lettres du mot masqué ch2 qui le joueur n'a pas découvert encore (les lettre *) mais ça cause la perte d'un point de son score. Simultanément, cette lettre va être déplacer par un « - » dans le tableau alph

c) La donne d'une lettre correcte ou pas :

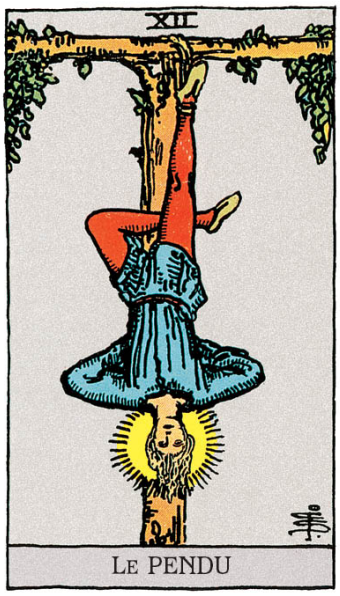
Dans ce cas on a appelé une fonction trouve qui a comme paramètre un caractère une chaine de caractère et un entier qui présente la taille du tableau. La fonction retourne 1 si le caractère est trouvé dans la chaine et 0 Sinon.

Si l'utilisateur donne une lettre correcte l'application va afficher l'un des lettres du mot masqué et la lettre va être déplacer par un « - » dans le tableau alph comme le cas du « ? » sinon les coups vont diminuer par 1.

```
int trouve(char c,char ch[50],int n) {
    int b=0, i=0;
    while ((i<n)&&(b==0)) {
        if (ch[i]==toupper(c)) {
            b=1;
        }
        i++;
    }
    return b;
}
```

Figure 4: La fonction trouve

d) L'affichage du pendu :



On a utilisé une méthode simple pour dessiner le pendu : avec des caractères simples tel que (/, |, \, _, o).

Le résultat final va s'afficher comme suit :

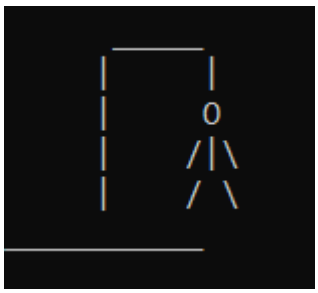


Figure 5: le pendu(dessin)

Chaque fois que le joueur donne une fausse lettre l'un des caractères du dessin va afficher. Alors selon les coups le programme va afficher un dessin.

```
switch (coups) {
    case 7 : printf("  _____\n  |    \n  |    \n  |    \n  |    \n  |_____\n\n");break;
    case 6 : printf("  _____\n  |   |\n  |    \n  |    \n  |    \n  |_____\n\n");break;
    case 5 : printf("  _____\n  |   |\n  |   O\n  |    \n  |    \n  |_____\n\n");break;
    case 4 : printf("  _____\n  |   |\n  |   O\n  |   |\n  |    \n  |_____\n\n");break;
    case 3 : printf("  _____\n  |   |\n  |   O\n  |  /|\n  |    \n  |_____\n\n");break;
    case 2 : printf("  _____\n  |   |\n  |   O\n  |  /|\\\n  |    \n  |_____\n\n");break;
    case 1 : printf("  _____\n  |   |\n  |   O\n  |  /|\\\n  |  /\n  |_____\n\n");break;
}
```

Figure 6: le pendu (code des dessins)

2) La procédure motus :

Le principe du motus ressemble à celle de pendu. Alors on a employé le même code du pseudo, avoir le mot à découvrir, la même boucle qui détermine la fin d'un match, l'aide avec une petite retouche, la même manière du calculer le score et le même affichage de la fin de match.

La différence c'est que le joueur doit proposer un mot entier.

Ici on a employé deux chaines de caractères chtest, qui va contenir les lettres en bonne position avec des « - » pour les lettres qui n'ont pas encore découvert, et une autre chaine chmp pour les lettres mal placé qui vont afficher entre des « [ ] ».

Dans les cas d'aide on va afficher l'un des lettres du mot masqué en permutant l'un des « - » dans chtest et le supprimer du chmp s'il existe. Pour supprimer la lettre d'aide du chmp une boucle du la lettre jusqu'à la fin du chaine chmp est utilisé. Chaque lettre va prendre la valeur de la lettre qui le suit et la dernière lettre va contenir « '\0' » pour indiquer la fin de la chaine.

```
i=0; trouve=0;
while ((i<strlen(chmp))&&(trouve==0)) {
 if (chmp[i]==chtest[rnd]) {
     for (j=i;j<strlen(chmp)-1;j++) {
         chmp[j]=chmp[j+1];
     }
     chmp[strlen(chmp)-1]='\0';
     trouve=1;
 }
 i++;
}
```

Figure 7: supprimer la lettre d'aide dans la chaine entre [ ]

Dans le cas de la proposition du joueur un mot. En début, on a vidé la chaine chmp et remplit chtest par des « - ». Après, les lettre ayant un indice correct vont être placé dans sa position dans chtest. Ensuite, Le remplissage du chmp. On va parcourir la chaine ch (donné par l'utilisateur). Avec chaque lettre de ch on va parcourir chrand (le mot aléatoire à découvrir) si la lettre existe dans chtest le programme le dépasse sinon si la lettre est correcte mal placé on va la placer dans chmp en respectant le nombre d'occurrence de cette lettre dans chrand.

Si le mot proposé n'est pas encore identique à le mot à découvrir les coups va diminuer

```
strcpy(chmp,"");
for (i=0;i<strlen(chrand);i++) {
   chtest[i]='-';
 }
for (i=0;i<strlen(chrand);i++) {
   if (toupper(ch[i])==chrand[i]) {
      chtest[i]=chrand[i];
    }
  }
 for (i=0;i<strlen(chrand);i++) {
    j=0; trouve=0;
    while ((j<strlen(chrand))&&(trouve==0)) {
      if (chrand[i]==toupper(ch[i])) {
          trouve=1;
      }
      else if (chrand[j]==toupper(ch[i])) {
          if (((occ(ch[i],chtest)+occ(ch[i],chmp)))<(occ(ch[i],chrand))) {
              strncat(chmp,&chrand[j],1);
              trouve=1;
          }
      }
       j++;
     }
 }
 if (strcmp(chtest,chrand)!=0) coups--;
```

Figure 8: le cas de la proposition d'un mot

3) La procédure ext :

La procédure ext est simple. Elle permet d'afficher un message lors de sortie de programme.

```
void ext () {
    system("cls");
    printf("||------------------------------------------------||\n");
    printf("||                  AU REVOIR!                    ||\n");
    printf("||------------------------------------------------||\n");
}
```

Figure 9: Le message d'au revoir

4) Les méthodes communes

- Dans notre programme, puisque les principes des deux jeux se rassemblent on a employé des méthodes et des fonctions communes :

- En début des fonctions ou procédures on a utilisé system (« cls ») Qui sert à effacer tout ce qu'il y a à l'écran.
- La fonction prédéfinit sleep (nbre de secondes) permet de faire attendre le programme pendant un certain nombre de secondes avant de continuer à exécuter les instructions.

- **L'Affichage de résultat :** en utilisant le même principe avec le changement des variables on fait face à trois cas :
    1. Si les deux chaines sont identiques et le temps n'a pas encore épuisé un message de félicitation va s'afficher avec le score
    2. Si le temps est épuisé et le joueur a encore des tentatives restantes le programme va afficher que le temps est épuisé
    3. Sinon, quand les tentatives se termine, un message de perte va s'afficher

```
if ((strcmp(chtest,chrand)==0)&&((tf-(int)tv.tv_sec>0))) {
    printf("\nGagne UwU Le mot secret etait bien : %s\nVotre score est : %i\n",chrand,pts);
}
else {
    if (((tf-(int)tv.tv_sec<=0))&&(coups>0)) {
        printf("\nTemps epuise >:( Le mot secret etait : %s\n",chrand);
        pts=0;
    }
    else {
        printf("\nPerdu >:( Le mot secret etait : %s\n",chrand);
    }
}
```

Figure 10: L'affichage de résultat

- **Les scores :** sont évalués selon les tentatives restantes. Lorsque le joueur utilise l'aide le score va décrémenter.

- **Remplissage des tableaux des pseudos et scores :** aux variables globales, on a déclaré 4 tableaux : pp et pm contenant respectivement les pseudos des jeux pendu et motus. Et sp et sm contenant respectivement les scores des pendu et motus. Avec 2 variables contenant la taille des tableaux (np pour les tableaux de pendu et nm pour les tableaux de motus)

  En utilisant une fonction pos déclaré au début du programme qui donne la position d'une chaine dans un tableau ou d'afficher -1 s'il n'existe pas, on va tester si le pseudo n'est pas trouvé dans le tableau des pseudos alors on va l'ajouter et insérer parallèlement le score dans le tableau des scores sinon on va additionner ce score avec le score ancien.

```
if (pos(pm,nm,pseudo)==-1) {
    strcpy(pm[nm],pseudo);
    sm[nm]=pts;
    nm++;
}
else {
   int o=pos(pm,nm,pseudo);
   sm[o]=sm[o]+pts;
}
```

Figure 11: remplissage des pseudos et scores

- **Fin jeu :** Cette fonction permet d'afficher un menu après chaque partie. Alors on l'appelle à la fin des procédures pendu et motus. Il affiche un menu comme il est représenté dans la figure12.

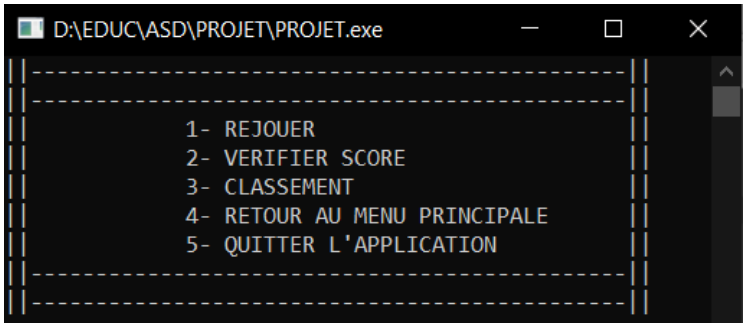


Figure 12: Le menu de la fin de jeu

Selon le numéro donné le programme va faire certaines instructions :

1. Rejouer en appelant la procédure pendue ou motus.
2. Afficher le score.
3. Afficher le classement en appelant la fonction classement : cette fonction va trier et afficher les deux tableaux des pseudos et score. On a employé la méthode de tri à bulles qui consiste à comparer

répétitivement les éléments consécutifs d'un tableau, et à les permuter lorsqu'ils sont mal triés.

4. Retourner au menu principal en appelant la fonction main ().
5. Quitter l'application en appelant la procédure ext.