```c
/*
 * Ce programme simule le jeu du mot le plus long. L'utilisateur joue
 * contre l'ordinateur qui a à sa disposition un dictionnaire (un
 * fichier) de 11691 mots classés par taille de 1 à 10 lettres, et
 * classés par ordre alphabétique
 *
 * @author Vincent Granet (vg@unice.fr)
 * 01/01/2023
 */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <time.h>
#include <ctype.h>

#define DICO_FILE "dico.txt"
#define MAXLETTRES 10
#define MAXNBMOTS 116971   // nb mots dans le dictionnaire DICO_FILE

typedef char mot[MAXLETTRES+1];  // type mot
typedef mot dictionnaire[MAXNBMOTS];     // type dictionnaire

/*
 * Rôle : renvoie le prochain caractère lu sur l'entrée standard
 */
int lireRep(void) {
  int rep=getchar(), c;
  do
    c = getchar();
  while (c != '\n' && c != EOF);
  return rep;
}

/* Antécédent : vOUc = 'v' ou 'c'
 * Rôle : renvoie une voyelle ou une  consonne tirée au hasard selon
 *        les tables de fréquences du jeu
 */
char tirageVoyellesConsonnes(const int vOUc) {
  static const int NBCONSONNES=20;
  static int NBVOYELLES=6;
  //
  static char consonnes[] = "bcdfghjklmnpqrstvwxz";
  static char voyelles[] = "aeiouy";
  static char tableFrequences[26] =
    {
        9, 2, 2, 3, 15, 2, 2, 2, 8, 1, 1, 5, 3, 6, 6, 2, 1, 6, 6, 6, 6, 2, 1, 1
, 1, 1
     //  a  b  c d   e   f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x
  y  z
    };
  //
  int c, fini = 0;
  do {
    c = (vOUc=='v') ? voyelles[rand()%NBVOYELLES] : consonnes[rand()%NBCONSONNES
];
    if (tableFrequences[c-'a']!=0) {
      tableFrequences[c-'a']--;
      fini=1;
    }
  } while (!fini);
  return c;
}

/*
```



```c
 * Rôle : afficher le tirage de nbLettres sur la sortie standard
 */
void afficherTirage(const char lesLettres[], const int nbLettres) {
  //
  int i=0;
  for ( ; i<nbLettres; i++)
    printf("%c ", lesLettres[i]);
  // compléter jusqu'à MAXLETTRES par des _
  // pour les lettres non encore tirées
  for ( ; i<MAXLETTRES; i++)
    printf("_ ");
  //
  printf("\n");
}

/* Rôle : fait le tirage aléatoire des voyelles et consonnes en
 *        fonction du choix des joueurs ; le booléen ordi indique si
 *        l'ordinateur joue en premier ou pas
 */
void tirageLettres(mot tirage , int ordi) {
  int i=0;
  do {
    int rep;
    if (ordi) {
      printf("L'ordinateur tire une lettre\n");
      rep = (rand()%2) == 0 ? 'c' : 'v';
    }
    else {
      printf("A vous. Voyelles ou consonnes (v/c) ? : ");
      while ((rep = lireRep()) != 'c' && rep != 'v')
        fprintf(stderr, "Recommencez! Répondez v ou c !\n");
      // la réponse valide
    }
    tirage[i++] = tirageVoyellesConsonnes(rep);
    afficherTirage(tirage, i);
    ordi = !ordi;
  } while (i<MAXLETTRES);
  // mettre le marqueur de fin de chaîne
  tirage[i++] = '\0';
}

/* Rôle : renvoie la position du caractère c non marqué dans
 *        l'ensemble des lettres du tirage, ou -1 si non trouvé
 */
int dans(const char c, const mot tirage, int marques[]) {
  for (int i=0; i<MAXLETTRES; i++)
    if (tirage[i]==c && marques[i]==0)
      return i;
  // la lettre n'est dans le tirage
  return -1;
}

/* Rôle : renvoie vrai si les lettres du MOT appartiennent au tirage
 */
int estInclus(const mot MOT, const mot tirage) {
  int m, marques[MAXLETTRES] = { 0 };
  int i=0;
  while (MOT[i]!='\0') {
    if ((m=dans(MOT[i], tirage, marques))==-1)
      // la lettre  n'est pas dans le tirage
      return 0;
    // Ok la lettre est dans le tirage, on la marque
    marques[m] = 1;
    i++;
  }
  return 1;
```

```c
}

/*
 *  Rôle : recherche si le mot le plus long mpl à partir du tirage de
 *         lettres passé en paramètre en parcourant le dictionnaire
 *         des mots de 10 lettres vers ceux de 1 lettre.
 * conséquent : renvoie 1 si mpl trouvé, et 0 sinon
 */
int chercherMotLePlusLong(const int lgDico, dictionnaire dico, const mot tirage,
 mot mpl)
{
  for (int i=lgDico-1; i>=0; i--) {
    if (estInclus(dico[i], tirage)) {
      // mot est le plus long
      strcpy(mpl, dico[i]);
      return 1;
    }
  }
  // pas de mot
  return 0;
}

/*
 * Antécédent : fDico fichier de texte qui contient les mots du dictionnaire
 *              1 mot par ligne
 * conséquent : tDico contient tous les mots lus dans le fichier fDico
 *              la fonction renvoie le nombre de mot lus
 */
int lireDictionnaire(const char fDico[], dictionnaire dico) {
  FILE *fd;
  // ouvrir le fichier fDico en lecture
  if ((fd=fopen(fDico, "r")) == NULL) {
    perror(fDico);
    exit(errno);
  }
  // lire tous les mots et les mettre dans le tableau tDico
  int nbMots=0;
  mot m;
  while (fscanf(fd,"%s", m)>0)
    // mettre le mot m dans le tableau tDico
    strcpy(dico[nbMots++], m);
  // fermer le fichier
  fclose(fd);
  // renvoyer le nb de mots lus
  return nbMots;
}

/*
 * Rôle : vérifie si le mot est dans le dictionnaire ou pas
 * Note : Algo : recherche dicothomique
 */
int motDansDico(const int n, const mot m, dictionnaire dico) {
  int gauche=0, droite = n-1;
  mot x;
  do {
    int milieu = (gauche+droite)/2;
    strcpy(x, dico[milieu]);
    //
    if (strcmp(m, x)>0)
      gauche = milieu+1;
    else
      droite = milieu;
    //
  } while (gauche<droite);
  // si mot est dans le dico, gauche est son indice
  return strcmp(m, dico[gauche])==0;
```



```c
}

/*
 * Rôle : fonction de comparaison pour qsort
 */
int cmp(const void *s1, const void *s2) {
  return strcmp(s1,s2);
}

/*
 * Rôle : crée un dictionnaire en ordre alphabétique à partir
 *        du dictionnaire ordonné par taille croissante des mots
 */
void  creerDicoOrdreAlpha(const int n, dictionnaire dicoOrdreAlpha , dictionnair
e dicoOrdreLgMot) {
  memcpy(dicoOrdreAlpha, dicoOrdreLgMot, sizeof(mot)*n);
  qsort(dicoOrdreAlpha, MAXNBMOTS, sizeof(mot), cmp);
}

/*
 * Rôle : écrit sur la sotie d'erreur std le message d'erreur msg
 *        et arrête l'application
 */
void erreur(const char msg[]) {
  fprintf(stderr, "Erreur : %s\n", msg);
  exit(EXIT_FAILURE);
}

/*
 * Rôle : lit sur l'E/S le mot du joueur, vérifie
 *        si la longueur du mot est valide
 * Conséquent : mj = mot du joueur lu
 */
void lireMotJoueur(mot mj) {
  char m[100]; // on voit large ;-)
  scanf("%s", m);
  if (strlen(m)>MAXLETTRES)
    erreur("mot trop long");
  //
  strcpy(mj, m);
}

/*
 * Rôle :  l'exécution du jeu proprement dite
 */
void jouer(const int lgDico, dictionnaire dicoOrdreAlpha , dictionnaire dicoOrdr
eLgMot) {
  //
  mot tirage, motDuJoueur;
  mot motLPL;
  int lgMotDuJoueur, ordiCommence=0;
  //
  srand(time(NULL));
  // tirage des lettres
  tirageLettres(tirage, ordiCommence);
  // l'ordi cherche le mot le plus long avec ce tirage
  if (!chercherMotLePlusLong(lgDico, dicoOrdreLgMot, tirage, motLPL))
    erreur("aucun mot avec ce tirage\n");
  //
  printf("\nProposez un mot avec le tirage précédent : ");
  // validité de l'input à vérifier
  lireMotJoueur(motDuJoueur);
  lgMotDuJoueur = strlen(motDuJoueur);
  // vérifier si le mot du joueur est valide
  if (!motDansDico(lgDico, motDuJoueur, dicoOrdreAlpha))
    erreur("Ce mot n'appartient pas à notre dictionnaire\n");
```

```
  // <
  printf("Bravo, votre mot de %d lettres est valide\n", lgMotDuJoueur);
  if (strlen(motLPL)>lgMotDuJoueur)
    printf("... mais l'ordinateur a trouvé un mot plus long : %s\n",motLPL);
  //
}

/*
 *
 */
int main(void) {
  // création des dictionnaires :
  // le 1er en ordre alphagbétique, le second par longueur de mots
  dictionnaire dicoOrdreLgMot, dicoOrdreAlpha;
  int lgDico = lireDictionnaire(DICO_FILE, dicoOrdreLgMot);
  creerDicoOrdreAlpha(lgDico, dicoOrdreAlpha, dicoOrdreLgMot);
  // lacer le jeu
  jouer(lgDico, dicoOrdreAlpha, dicoOrdreLgMot);

  return EXIT_SUCCESS;
}
```