Correction IS langage C

Elliot S. 2013/2014

Question 1

Il était demandé de compléter le fichier "mots_meles.h".

1) Macros

Code source 1 – Les macros TAILLEGRILLE et LGDICO

```
1 /* macro pour la taille de la grille */
2 #define TAILLEGRILLE 5
3
4 /* macro pour le nombre de mots a trouver */
5 #define LGDICO 5
```

2) Type DIRECTION

Code source 2 – Le type DIRECTION

```
1 /* type enumere pour choisir la direction : E=0, W=1, N=2, S=3, NE=4, SE=5, NW=6, ↩
      SW=7 */
2 typedef enum
3 { E, W, N, S, NE, SE, NW, SW } DIRECTION;
```

3) Type typeMot

Code source 3 – Le type typeMot

```
1 /* typeMot pour definir les mots a trouver ainsi que leur longueur et s'ils ont ↩
      ete trouves ou non */
2 typedef struct{
3     int tailleMot;
4     char mot[TAILLEGRILLE];
5     int trouve;
6 } typeMot;
```

Question 2

Il s'agissait ici d'écrire quelques fonctions dans le fichier "mots_meles.c".

1) Fonction verifCoord

Code source 4 – La fonction verifCoord

```
1  int verifCoord(int x, int y, int tailleGrille){
2      return (x >= 0) && (x < tailleGrille)
3              && (y >= 0) && (y < tailleGrille);
4  }
```

2) Fonction calculeDirection

Code source 5 – La fonction calculeDirection

```
1  typeCoord calculeDirection(DIRECTION dir){
2      typeCoord res;
3      switch(dir){
4          case E : res.x = 0; res.y = 1;
5                   break;
6          case W : res.x = 0; res.y = -1;
7                   break;
8          case N : res.x = -1; res.y = 0;
9                   break;
10         case S : res.x = 1; res.y = 0;
11                  break;
12         case NE : res.x = -1; res.y = 1;
13                  break;
14         case SE : res.x = 1; res.y = 1;
15                  break;
16         case NW : res.x = -1; res.y = -1;
17                  break;
18         case SW : res.x = 1; res.y = -1;
19                  break;
20     }
21     return res;
22 }
```

3) Fonction compareMots

Code source 6 – La fonction compareMots

```
1  int compareMots(char mot1[TAILLEGRILLE], int lgMot1, char mot2[TAILLEGRILLE], int ↩
       lgMot2){
2      if(lgMot1 != lgMot2)
3          return 0;
4      else{
5          int i = 0;
6          while (i < lgMot1 && mot1[i] == mot2[i])
7              i++;
8          return (i == lgMot1);
9      }
10 }
```

4) Fonction rechercheMot

Code source 7 – La fonction rechercheMot

```
1  int rechercheMot(typeMot motsATrouver[LGDICO], char mot[TAILLEGRILLE], int lgMot){
2      int i;
3      for(i = 0; i<LGDICO; i++)
4          if(compareMots(motsATrouver[i].mot, motsATrouver[i].tailleMot, mot, lgMot)↩
               )
5              return i;
6      return LGDICO;
7  }
```

5) Fonction extraireMot

Code source 8 – La fonction extraireMot

```
1  int extraireMot(char grille[TAILLEGRILLE][TAILLEGRILLE], int x, int y , int dx, ↩
       int dy, char mot[TAILLEGRILLE], int lgMot){
2      int i = 0;
3      while(i < lgMot && verifCoord(x + i*dx,y + i*dy,TAILLEGRILLE)){
4          mot[i] = grille[x + i*dx][y + i*dy];
5          i++;
6      }
7      return i;
8  }
```

6) Fonction affichеGrille

Code source 9 – La fonction affichеGrille

```
1  void afficheGrille(char grille[TAILLEGRILLE][TAILLEGRILLE], int grilleTrouve[↩
       TAILLEGRILLE][TAILLEGRILLE], typeMot motsATrouver[LGDICO]){
2      int i,j;
3      printf("mot meles :\n");
4      printf("   X ");
5      for (i = 0; i < TAILLEGRILLE; i++)
6          printf("%d  ", i);
7      printf("\n");
8      for (i = 0; i < TAILLEGRILLE; i++){
9          printf("   %d", i);
10         for (j = 0; j < TAILLEGRILLE; j++)
11             if(grilleTrouve[i][j])
12                 printf("(%c)", grille[i][j]);
13             else
14                 printf(" %c ", grille[i][j]);
15         printf("\t\t");
16         if(!(motsATrouver[i].trouve))
17             afficheMot(motsATrouver[i].mot, motsATrouver[i].tailleMot);
18         printf("\n");
19     }
20 }
```

7) Fonction toutTrouve

Code source 10 – La fonction toutTrouve

```
1  int toutTrouve(typeMot motsATrouver[LGDICO]){
2      int i = 0;
3      while(i < LGDICO && motsATrouver[i].trouve)
4          i++;
5      return (i == LGDICO);
6  }
```

Question 3

Il fallait corriger la fonction main dans le fichier "mots_meles.c".

1) Fonction main corrigée

Code source 11 – La fonction main corrigée

```c
#include <stdio.h>
#include "mots_meles.h" /* Erreur : le mauvais fichier .h etait inclus */

int main () {
    int i, j;
    int lg;
    int encore;
    typeCoup coup;
    char grilleJeu[TAILLEGRILLE][TAILLEGRILLE]={ {'A','I','N','T','J'},
                                                 {'R','F','H','S','E'},
                                                 {'K','A','H','N','U'},
                                                 {'P','A','H','O','T'},
                                                 {'E','S','A','C','Y'} };
    typeMot motsATrouver[LGDICO]={ {2, {'I', 'F'}, 0},
                                   {3, {'I', 'N', 'T'}, 0},
                                   {4, {'C', 'H', 'A', 'R'}, 0}, /* Erreur : il manquait une virgule */
                                   {4, {'C', 'A', 'S', 'E'}, 0},
                                   {5, {'C', 'O', 'N', 'S', 'T'}, 0} };
    int grilleTrouve[TAILLEGRILLE][TAILLEGRILLE]; /* Erreur : pas de .. mais [][] */
    char motATrouver[TAILLEGRILLE];

    for (i=0; i<TAILLEGRILLE; i++) /* Erreur : on met des ; dans les for, pas des virgules */
        for (j=0; j<TAILLEGRILLE; j++) /* Erreur : pas de ; a la fin de la ligne */
            grilleTrouve[i][j]=0;

    afficheGrille(grilleJeu, grilleTrouve, motsATrouver);

    do {
        coup=saisirCoup();
        lg=extraireMot(grilleJeu, coup.x, coup.y, coup.dx, coup.dy, motATrouver, coup.lg);
        i=rechercheMot(motsATrouver, motATrouver, lg);

        if (i<LGDICO)
        {
            printf("\nBravo, ");
            afficheMot(motATrouver, lg);
            printf(" trouve !!!\n\n");
            motTrouve(grilleTrouve, coup.x, coup.y, coup.dx, coup.dy, lg);
            motsATrouver[i].trouve=1;
        }
        else { /* Erreur : il manquait des accolades */
            afficheMot(motATrouver, lg);
            printf(" pas dans la grille\n");
        }

        afficheGrille(grilleJeu, grilleTrouve, motsATrouver);

        if (toutTrouve(motsATrouver)) {
            printf("BRAVO !!! Vous avez trouve tous les mots\n");
            encore=0;
        }
        else {
```

```
53              printf("continue ? (oui 1/ non 0) : ");
54              scanf("%d", &encore); /* 2 Erreurs : c'est un %d et il manquait un & ↩
                    devant encore */
55          }
56      } while (encore!=0);
57  }
```

Remarques

En corrigeant, pour vérifier mes résultats, j'ai codé complétement le programme de l'IS (qui, normalement, fonctionne sans problème). Si vous voulez, vous pouvez modifier des fonctions dans la source pour tester vos propres solutions. Normalement, les fichiers "mots_meles.h" et "mots_meles.c" vous sont fournis avec la correction. Pour compiler, il suffit d'utiliser gcc sur linux :

Code source 12 – Compiler le programme

```
1  gcc mots_meles.c -o mots_meles
```

Puis, pour lancer le programme en console linux :

Code source 13 – Lancer le programme

```
1  ./mots_meles
```

N'hésitez pas à me contacter pour des questions/remarques (à l'adresse elliot.sisteron@insa-rouen.fr). Bonne révisions et bonne chance pour votre futur examen,

Elliot.