Correction DS langage C

Elliot S. 2013/2014

Question 1

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

1) Macro max(a,b)

Code source 1 – La macro max

```
1 /* macro pour le maximum de a et b */
2 #define max(a,b) (((a)<(b))?(b):(a))
```

2) Type typeJeu

Code source 2 – Le type typeJeu

```
1 /* type pour definir un jeu de mots meles */
2 typedef struct {
3     int tailleGrille; // taille de la grille
4     char** grille; // matrice dynamique de caracteres
5     int** grilleTrouve; // matrice dynamique de booleens
6     int tailleDico; // nombre de mots a trouver
7     typeMot* motsATrouver; // tableau dynamique des mots a trouver
8 } typeJeu;
```

Question 2

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

1) Fonction coupValide

Code source 3 – La fonction coupValide

```
1  int coupValide(typeCoup coup, int tailleGrille){
2      int xi = coup.cinit.x, yi = coup.cinit.y, xf = coup.cfinal.x, yf = coup.cfinal↩
           .y;
3      int dansGrille = (xi >= 0) && (xi < tailleGrille) && (yi >= 0) && (yi < ↩
           tailleGrille) && (xf >= 0) && (xf < tailleGrille) && (yf >= 0) && (yf < ↩
           tailleGrille);
4      int memeHorizontale = (yi == yf);
5      int memeVerticale = (xi == xf);
6      int memeDiagonale = (max(xf - xi, xi - xf) == max(yf - yi, yi - yf));
7      return (dansGrille && (memeVerticale || memeHorizontale || memeDiagonale));
8  }
```

2) Fonction saisirCoup

Code source 4 – La fonction saisirCoup

```
1  typeCoup saisirCoup(int tailleGrille){
2      typeCoup res;
3      do{
4          printf("Debut du mot (x,y) : ");
5          scanf("%d,%d", &(res.cinit.x), &(res.cinit.y));
6          printf("fin du mot (x,y) : ");
7          scanf("%d,%d", &(res.cfinal.x), &(res.cfinal.y));
8      }while(!coupValide(res, tailleGrille));
9      return res;
10 }
```

3) Fonction extraireMot

Code source 5 – La fonction extraireMot

```
1  char* extraireMot(char **grille, typeCoup coup){
2      int i;
3      int xi = coup.cinit.x, yi = coup.cinit.y, xf = coup.cfinal.x, yf = coup.cfinal↩
           .y;
4      int dx = xf - xi, dy = yf - yi;
5      int lgRes = max(max(dx, -dx), max(dy, -dy)) + 1;
6      char* res = (char*)malloc((lgRes+1)*sizeof(char));
7
8      if (dx != 0)
9          dx = ((dx<0)?(-1):1);
10     if (dy != 0)
11         dy = ((dy<0)?(-1):1);
12
13     for(i = 0 ; i < lgRes ; i++)
14         res[i] = grille[xi + i*dx][yi + i*dy];
15
16     res[lgRes] = '\0';
17
18     return res;
19 }
```

4) Fonction toutTrouve

Code source 6 – La fonction toutTrouve

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

Question 3

Il fallait coder la fonction d'allocation de la matrice de char.

1) Fonction alloueGrilleChar

Code source 7 – La fonction alloueGrilleChar

```
1  char ** alloueGrilleChar(int tailleGrille){
2      int i;
3      char** res = (char**)malloc(tailleGrille*sizeof(char*));
4      char* tmp = (char*)malloc(tailleGrille*tailleGrille*sizeof(char));
5      if (res == NULL || tmp == NULL)
6          return NULL;
7      for(i = 0 ; i < tailleGrille ; i++)
8          res[i] = &tmp[i*tailleGrille];
9      return res;
10 }
```

Question 4

Il était demandé de coder la fonction d'affichage du jeu.

1) Fonction afficheJeu

Code source 8 – La fonction afficheJeu

```
1  void afficheJeu(typeJeu unJeu){
2      int i,j;
3      printf("mot meles :\n");
4      printf("   X ");
5      for (i = 0; i < unJeu.tailleGrille; i++)
6          printf("%d  ", i);
7      printf("\n");
8      for (i = 0; i < unJeu.tailleGrille; i++){
9          printf("   %d", i);
10         for (j = 0; j < unJeu.tailleGrille; j++)
11             if(unJeu.grilleTrouve[i][j])
12                 printf("(%c)", unJeu.grille[i][j]);
13             else
14                 printf(" %c ", unJeu.grille[i][j]);
15         printf("\t\t");
16         if(!(unJeu.motsATrouver[i].trouve)){
17             j = 0;
18             while (unJeu.motsATrouver[i].mot[j]){
19                 printf("%c", unJeu.motsATrouver[i].mot[j]);
```

```
                j++;
            }
        }
        printf("\n");
    }
}
```

Question 5

Il fallait ici s'occuper de fonctions relatives à la persistance du jeu.

1) Fonction chargeJeu

Code source 9 – La fonction chargeJeu

```
int chargeJeu(typeJeu *unJeu){
    int trouve;
    FILE* fic = fopen("GRILLE.DATA", "r");
    if (fic != NULL){
        unJeu->grille = lireGrille(fic, &(unJeu->tailleGrille));
        fscanf(fic, "[TROUVE=%d]\n", &trouve);
        unJeu->grilleTrouve = alloueGrilleInt(unJeu->tailleGrille);
        int i,j;
        if (trouve){
            for(i = 0; i < unJeu->tailleGrille ; i++){
                for (j = 0; j < unJeu->tailleGrille ; j++)
                        fscanf(fic, "%d ", &(unJeu->grilleTrouve[i][j]));
                fscanf(fic, "\n");
            }
        }
        else{
            for(i = 0; i < unJeu->tailleGrille ; i++)
                for (j = 0; j < unJeu->tailleGrille ; j++)
                        unJeu->grilleTrouve[i][j] = 0;
        }
        unJeu->motsATrouver = lireMotsATrouver(fic, unJeu->tailleGrille, &(unJeu->↩
            tailleDico));
    }
    else return 0;
    fclose(fic);
    return 1;
}
```

2) Fonction lireMotsATrouver

Code source 10 – La fonction lireMotsATrouver

```
typeMot * lireMotsATrouver(FILE *fic, int tailleGrille, int *tailleDico){
    typeMot* res;
    int i,j;
    fscanf(fic, "[TAILLEDICO=%d]\n", tailleDico);
    res = (typeMot*)malloc((*tailleDico)*sizeof(typeMot));
    for (i = 0 ; i < *tailleDico ; i++){
        res[i].mot = (char*)malloc(tailleGrille*sizeof(char));
        fscanf(fic, "%s %d\n", res[i].mot, &(res[i].trouve));
    }
    return res;
}
```

3) Fonction sauveJeu

Code source 11 – La fonction sauveJeu

```
1  int sauveJeu(typeJeu unJeu){
2      FILE* fic = fopen("GRILLE.DATA", "w");
3      if (fic != NULL){
4          int i, j, trouve = 0;
5          fprintf(fic, "[TAILLEGRILLE=%d]\n", unJeu.tailleGrille);
6          for(i = 0 ; i < unJeu.tailleGrille ; i++){
7              for(j = 0 ; j < unJeu.tailleGrille ; j++)
8                  fprintf(fic, "%c ", unJeu.grille[i][j]);
9              fprintf(fic, "\n");
10         }
11         i = 0;
12         while (i < unJeu.tailleDico && !trouve){
13             trouve = trouve || unJeu.motsATrouver[i].trouve;
14             i++;
15         }
16         fprintf(fic, "[TROUVE=%d]\n", trouve);
17         if (trouve){
18             for(i = 0; i < unJeu.tailleGrille ; i++){
19                 for (j = 0; j < unJeu.tailleGrille ; j++)
20                     fprintf(fic, "%d ", unJeu.grilleTrouve[i][j]);
21                 fprintf(fic, "\n");
22             }
23         }
24         fprintf(fic, "[TAILLEDICO=%d]\n", unJeu.tailleDico);
25         for (i = 0; i < unJeu.tailleDico; i++){
26             fprintf(fic, "%s %d\n", unJeu.motsATrouver[i].mot, unJeu.motsATrouver[↩
                   i].trouve);
27         }
28     }
29     else return 0;
30     fclose(fic);
31     return 1;
32 }
```

Remarques

En corrigeant, pour vérifier mes résultats, j'ai codé complétement le programme du DS (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.