LIFAP1 – TD 10 : Chaines de caractères suite

Objectifs : Utiliser les chaînes de caractères dans des programmes plus complexes, manipuler des chaînes de caractères dans des tableaux.

Jeu du mémo…

Le jeu "mémo" est un jeu de mémoire qui consiste à retrouver les paires d'images identiques dans une grille d'images retournées. Ici, les images seront représentées par les mots qu'elles signifient. La grille sera un tableau 2 dimensions.

	1	2	3	4
1	Lion	Chat	Chat	Poule
2	Lion	Chien	Vache	Oie
3	Poule	Oie	Canard	Tigre
4	Vache	Tigre	Canard	Chien

1. Écrire en notation algorithmique et en C la déclaration du tableau de taille TAILLE_GRILLE*TAILLE_GRILLE contenant les chaînes de caractères. TAILLE_GRILLE est une constante paire pour que le nombre de cases dans la grille soit pair aussi.

   En algo = Grille : tableau [TAILLE_GRILLE][TAILLE_GRILLE][15] de caractères
   En C/C++ = char Grille [TAILLE_GRILLE][TAILLE_GRILLE][15] ;

2. Écrire en C une procédure d'initialisation de la grille de jeu avec des "*"

```
void init_grille (char grille_mot[TAILLE_GRILLE][TAILLE_GRILLE][15])
{
  int i,j;

  for(i=0;i<TAILLE_GRILLE;i++)
  {
      for(j=0;j<TAILLE_GRILLE;j++)
          strcpy(grille_mot[i][j],"*");
  }
}
```

   Bien leur faire remarquer ici qu'on fait une copie du mot dans la grille par un **strcpy** et qu'on n'a pas le droit de faire grille_mot[i][j]="*" !!!

3. Écrire en C une procédure de remplissage de la grille. Cette procédure devra demander à l'utilisateur (TAILLE_GRILLE)² / 2 chaînes de caractères qui seront insérées aléatoirement dans la grille de jeu. Attention de bien vérifier que la case sélectionnée est vide avant d'insérer le mot.

```
void remplir_grille (char grille_mot[TAILLE_GRILLE][TAILLE_GRILLE][15])
{
  int num_paire,lig,col,i;
  char mot[15];

  for(num_paire=0;num_paire<(TAILLE_GRILLE*TAILLE_GRILLE)/2;num_paire++)
  {
```

```
        cout<<"donnez le mot "<<num_paire +1;
        cin >> mot;
        for (i=0;i<2;i++)   // pour placer 2 fois le même mot
        {
          do
          {
            lig=rand()%TAILLE_GRILLE;           // rand() = [0; MAX_INT]
            col=rand()%TAILLE_GRILLE;
          }
          while (strcmp(grille_mot[lig][col],"")!=0);   // strcmp renvoie 0 si identique
                                                        // sinon 1 ou -1
          //cout<<lig<<col;
          strcpy(grille_mot[lig][col],mot);
        }
    }
```

4. Écrire en C une procédure demandant au joueur de choisir deux cases et d'afficher le contenu de ces deux cases en les resituant dans la grille complète.

```
void affiche_choix (char grille_mot[TAILLE_GRILLE][TAILLE_GRILLE][15], char
solution[TAILLE_GRILLE][TAILLE_GRILLE][15], int l1,int c1, int l2, int c2)
{
  int i,j;
  for(i=0;i<TAILLE_GRILLE;i++)
  {
      for(j=0;j<TAILLE_GRILLE;j++)
      {
        if ( ((i==l1)&&(j==c1)) || ((i==l2)&&(j==c2)))
            cout<<grille_mot[i][j]<<"  ";
        else cout <<solution[i][j]<<"--";
      }
      cout<<endl;
  }
}

void choix_cases(char grille_mot[TAILLE_GRILLE][TAILLE_GRILLE][15], char
solution[TAILLE_GRILLE][TAILLE_GRILLE][15],int &l1,int &c1, int &l2, int &c2)
{
  cout<<"coordonnees de la premiere case";
  cin>>l1>>c1;
  cout<<endl;
  cout<<"coordonnees de la deuxieme case";
  cin>>l2>>c2;
  cout<<endl;
  affiche_choix(grille_mot,solution,l1,c1,l2,c2);
}
```

5. Écrire en C une fonction de vérification du choix de l'utilisateur. Si les deux cases choisies sont identiques la fonction renverra 0 sinon elle renverra 1.

```
int verification(char grille_mot[TAILLE_GRILLE][TAILLE_GRILLE][15], char
solution[TAILLE_GRILLE][TAILLE_GRILLE][15],int l1,int c1, int l2, int c2)
{
  if (strcmp(grille_mot[l1][c1],grille_mot[l2][c2])==0)
  {
    strcpy(solution[l1][c1],grille_mot[l1][c1]);        // pour version complète du jeu
    strcpy(solution[l2][c2],grille_mot[l2][c2]);        // pour version complète du jeu
    return 0;
  }
    else return 1;
```

```
}
```
Leur montrer ici l'utilisation de la comparaison de chaînes.

6. Simuler le jeu à deux joueurs jusqu'à ce que toutes les paires aient été trouvées.

Expliquer que l'on utilise une seconde grille « solution » pour afficher la grille partiellement découverte, et que pour cela, il faut modifier les sous-programmes précédents.

```
void choix_joueur (int &numjoueur)
{
    if (numjoueur==1)
      numjoueur=2;
      else numjoueur =1;
}

int main (void)
{
   int a1,a2,o1,o2;
   int nb_paires = TAILLE_GRILLE*TAILLE_GRILLE /2;
   int paires_trouvees=0,nb_coup=0;
   int joueur,points_joueur1=0,points_joueur2=0;
   char
memory[TAILLE_GRILLE][TAILLE_GRILLE][15],solution[TAILLE_GRILLE][TAILLE_G
RILLE][15];

   srand(time(NULL));
   init_grille(memory);
   init_grille(solution);
// affiche_grille(memory);
   remplir_grille(memory);
   affiche_grille(solution);
    joueur=rand()%2 + 1;

   while (paires_trouvees<nb_paires)
   {   cout<<"le joueur "<<joueur<<" joue "<<endl;
      choix_cases(memory,solution,a1,o1,a2,o2);
      if (verification(memory,solution,a1,o1,a2,o2)==0)
      {
         paires_trouvees++;
            cout<<"Le joueur "<<joueur<<" marque un point et totalise ";
         if(joueur==1) {
                                    points_joueur1++;
                                    cout<<points_joueur1<<" points"<<endl;
                                    }
                     else
                       {
                                     points_joueur2++;
                                     cout<<points_joueur1<<" points"<<endl;
                       }


      }
      affiche_grille(solution);
      choix_joueur(joueur);
   }

   if (points_joueur1>points_joueur2)
      cout<<"le joueur 1 a gagné avec "<<points_joueur1<<" points contre
   "<<points_joueur2<<" points."<<endl;
```

```
      else if (points_joueur2>points_joueur1)
                cout<<"le joueur 2 a gagne avec "<<points_joueur2<<" points contre
             "<<points_joueur1<<" points."<<endl;
                else cout<<"Egalite entre les deux joueurs !!!"<<endl;
   affiche_grille(solution);
 system("PAUSE");
   return 0;

}
```