LIFAP1 – TD 11 : Encore des chaînes et jeu du mémory

Objectifs : Utiliser les chaînes de caractères dans des programmes plus complexes
Manipuler des chaînes de caractères dans des tableaux

En C/C++, bibliothèque contenant des sous-programmes facilitant la gestion des chaînes de caractères ➔ **string.h**

Quelques fonctionnalités

- lg = strlen (chaine) ➔ calcule et retourne la longueur de la chaine passée en paramètre
- strcat(ch1,ch2) ➔ concatène ch1 et ch2. Résultat dans ch1
- strcpy (ch1,ch2) ➔ copie ch2 dans ch1
- strcmp (ch1,ch2) ➔ compare ch1 et ch2. 0 si elles sont identiques, 1 si ch1> ch2, -1 sinon

1. Écrire l'algorithme d'une procédure qui prend une chaîne de caractères et construit une nouvelle chaine où toutes les voyelles de la chaîne donnée ont été supprimées.
   Exemple : sans_voyelle("programmation") ➔ "prgrmmtn"

```
Procédure sans_voyelle (mot : chaine de 20 caractères, voy : chaine de 20
caractères)
Précondition : aucune
Donnée / résultat : mot, voy
Variables locales : ind_mot, ind_voy :entier , lg : entier
Début
  ind_voy ← 0
  lg ← longueur(mot)
  Pour ind_mot allant de 0 à lg -1 par pas de 1 faire
    Si (mot[ind_mot]≠'a' et mot[ind_mot]≠'e' et mot[ind_mot]≠'i' et mot[ind_mot]≠'o' et
    mot[ind_mot]≠'u' et mot[ind_mot]≠'y')
    alors    voy[ind_voy] ← mot[ind_mot]
             ind_voy ← ind_voy +1
    FinSi
  Fin pour
  voy[ind_vod]='\0'
Fin
```

2. Écrire en langage algorithmique un sous-programme permettant de compter et renvoyer au programme appelant le nombre de majuscules, de minuscules et de voyelles dans une chaîne de caractères passée en paramètre.

```
Procédure min_maj_voy (mot : chaine de 20 caractères, nbmin : entier, nbmaj : entier,
nbvoy : entier)
Précondition : aucune
Donnée / résultat : mot, nbmin, nbmaj, nbvoy
Variables locales : ind_mot, lg : entier
Début
  nbmin ← 0
  nbmaj ← 0
  nbvoy ← 0
  lg ← longueur(mot)
  Pour ind_mot allant de 0 à lg -1 par pas de 1 faire
      Si (mot[ind_mot]='a' ou mot[ind_mot]='e' ou mot[ind_mot]='i' ou mot[ind_mot]='o'
      ou mot[ind_mot]='u' ou mot[ind_mot]='y' ou mot[ind_mot]='A' ou
      mot[ind_mot]='E' ou mot[ind_mot]='I' ou mot[ind_mot]='O' ou mot[ind_mot]='U'
      ou mot[ind_mot]='Y'
    alors     nbvoy ← nbvoy +1
    FinSi
```

```
        Si (mot[ind_mot]>='A' et mot[ind_mot]<='Z'
      alors    nbmaj ← nbmaj +1
          FinSi
        Si (mot[ind_mot]>='a' et mot[ind_mot]<='z'
      alors    nbmin ← nbmin +1
      FinSi
    Fin pour
Fin
```

Le jeu "mémory" 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/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/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/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 soit 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);
          }
      }
  }
```