LO21 - TP N°3
Liste de mots

Exercice n°1

Définition d'un type abstrait de donnée « liste doublement chaînée de mots ».

Question 1.

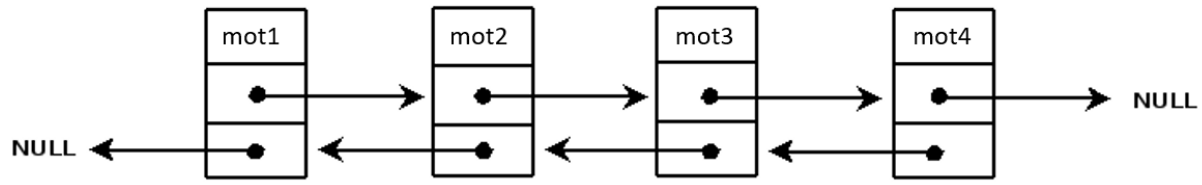
Écrire dans les fichiers « liste_mots.h » et « liste_mots.c » la définition d'une liste doublement chaînée de mots ainsi que les fonctions suivantes :

```
Dl_List  insertion_tete(Dl_List l, char* w);
Dl_List  insertion_queue(Dl_List l, char* w);
Dl_List  supprimer_tete(Dl_List l);
Dl_List  supprimer_queue(Dl_List l);
Dl_List reverse_list(Dl_List l);  */ Inverser une liste doublement chaînée/*

Dl_List deleteNodeAtGivenPos(Dl_List l, int n); */Supprimer un nœud d'une liste doublement chaînée à une position donnée/*

void affiche_liste(Dl_List l);
```

- Chaque élément de la liste est composé d'un pointeur vers la liste des caractères du mot, un pointeur vers l'élément suivant, et un pointeur vers l'élément précédent comme l'illustre la figure suivante :



On utilisera donc les définitions de type suivantes :

```
typedef struct elem {
  char* word;
  struct elem *next ;
  struct elem *prev ;
}  DListElement;

typedef    DListElement * Dl_List;
```

- Lors de l'insertion d'un mot dans la liste il est nécessaire d'allouer l'espace nécessaire au stockage du mot et de recopier le mot passé en paramètre dans cet espace. On utilisera pour cela les fonctions *malloc* et *strlen*, et également les librairies « stdlib.h » et « string.h ».

- Lors de la suppression d'un élément de la liste il est nécessaire de désallouer la mémoire du mot puis de l'élément.

Question 2.

Écrire dans un fichier « test_liste_mots_01.c » un programme qui permet de tester la liste doublement chaînée à l'aide du code suivant :

```
#include <stdio.h>
#include <stdlib.h>
#include "liste_mots.h"

int main (){

int n;
char* mot1 = "worry";
char* mot2 = "don't";
char* mot3 = "be";
char* mot4 = "happy";

Dl_List p = NULL;

/* Insertion des mots dans la liste */
p = insertion_tete(p, mot1);
p = insertion_tete(p, mot2);
p = insertion_queue(p, mot3);
p = insertion_queue(p, mot4);

/* Affichage de la liste */
affiche_liste(p);

/* Inverser la liste */
p=reverse_list(p);
affiche_liste(p);

printf("\nSelectionnez la position donnée du mot (element) à supprimer:"); scanf("%d",&n);
p = deleteNodeAtGivenPos(p, n);
affiche_liste(p);

/* Suppression de mots */
p = supprimer_tete(p);
p = supprimer_queue(p);
affiche_liste(p);

return 0 ;
}
```

Exercice n°2

Lecture de chaînes de caractères sur l'entrée standard.

Ecrire un programme permettant d'ajouter et de supprimer des mots à une liste doublement chaînée par des lectures sur l'entrée standard.

- L'exécution du programme aura l'allure suivante :

```
> La liste est vide.
> (1) ajouter en tête, (2) ajouter en queue, (3) supprimer en tête, (4) supprimer en queue, (0) Quitter.
> 1
> Mot à ajouter?
> un
```

> **La liste est : [un]**
> **(1) ajouter en tête, (2) ajouter en queue, (3) supprimer en tête, (4) supprimer en queue, (0) Quitter.**
> **2**
> **Mot à ajouter?**
> **deux**
> **La liste est : [un ; deux]**
> **(1) ajouter en tête, (2) ajouter en queue, (3) supprimer en tête, (4) supprimer en queue, (0) Quitter.**
> **4**
> **La liste est : [un]**
> **(1) ajouter en tête, (2) ajouter en queue, (3) supprimer en tête, (4) supprimer en queue, (0) Quitter.**
> **0**
> **Au revoir.**

- Afin de réaliser ce programme on utilisera un buffer de 100 caractères. La lecture sur l'entrée standard s'effectuera par la fonction « *fgets* » qui permet de limiter le nombre de caractères lu. Ainsi, pour ajouter un mot lu sur l'entrée standard dans la liste, on utilisera le code suivant :

```
void viderBuffer()
{
   int c = 0;
   while (c != '\n' && c != EOF)
   {
      c = getchar();
   }
}


int lire(char *Buffer)
{
   char *positionEntree = NULL;

   viderBuffer();

 /*On lit le texte saisi au clavier */

 if (fgets(Buffer, 100, stdin) != NULL)    /* Pas d'erreur de saisie ? (Buffer de 100 caractères) */
 {
      positionEntree = strchr(Buffer, '\n');   /* On recherche l'Entrée" */
      if (positionEntree != NULL)    /* Si on a trouvé le retour à la ligne */
      {
         *positionEntree = '\0';   /* On remplace ce caractère par \0   */
       }
      else
      {
         viderBuffer();
      }
      return 1;       /* On renvoie 1 si la fonction s'est déroulée sans erreur */
   }
   else
   {
      viderBuffer();
      return 0;         /* On renvoie 0 s'il y a eu une erreur   */
   }
}
```