Université Moulay Ismail
Faculté des Sciences et Techniques Errachidia
Département d'Informatique
Responsable Pr. B. Aksasse

AU 2019/2020
Filière MIP/ S4
Module I143

Corrigé de la Série 2 TD Structures de données en C : les listes chaînées

Pour répondre aux deux questions de l'exercice de cette deuxième série, nous allons créer trois fichiers. Le premier **liste.h** dont lequel on va mettre les définitions des structures et les prototypes des fonctions. Dans le deuxième fichier **liste.c,** on va y mettre les définitions de toutes les fonctions (le code des fonctions). Dans le troisième ficher **liste_run.c**, on met la fonction **main( )** dont laquelle on fait appel aux différentes fonctions déjà construites dans **liste.c**

Exercice 1

```
//**********contenu du fichier liste.h*****************//
struct Maillon{
int donnee;
struct Maillon *suivant;
};

 typedef struct Maillon Element ;

struct liste{
Element *debut;
Element *fin;
int taille;
};

typedef struct liste Liste;
```

Element* CreerElement( ) ;

Liste* CreerListe( ) ;

void CreerListeOrdre(Liste*) ;

void CreerListeCroissante(Liste*) ;

void AjoutFin(Liste*, int) ;

Liste* ConcatListe (Liste*, Liste*) ;

void Affichage(Liste *) ;

int RechercheElem(Liste*, int) ;

int SuppPos(Liste*, int) ;

```
//**********contenu du fichier liste.c****************//
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include "liste.h"

Element *CreerElement()
{
        Element *m ;
        m=( Element *)malloc(sizeof(Element)) ;
        if(m==NULL) {printf("allocation non reussie");exit(-1);}
        else { m->donne=0; m->suivant=NULL; }
        return m ;
}

Liste *CreerListe()
{
        Liste *L ; L=(Liste *)malloc(sizeof(Liste)) ;
        if(L==NULL) {printf("allocation non reussie");exit(-1);}
        else {L->debut=NULL;  L->fin=NULL;  L->taille=0;}
        return L ;
}

void CreerListeOrdre (Liste *L)
{
        int nb;
        Element *m;
        do{
                printf("\n Saisir un nombre positif:\n");
                scanf("%d", &nb);
                printf("vous avez tappe %d\n", nb);
                if (nb>=0)
                {
                m=CreerElemen();     m->donnee=nb;
                 if(L->taille==0)
                  {L->debut=m; L->fin=m;} else { L->fin->suivant=m;  L->fin=m; }
                 L->taille++;
                }
        }
         while(nb>=0); // la condition d'arrêt et la saisie d'un nombre négatif
}

void CreerListeCroissante(Liste* L)
{
        int nb ;
        Element *m, *courant, *pred;
        do{
                printf("\n Saisir un nombre positif:\n");
                scanf("%d", &nb);
                printf("vous avez tappe %d:\n", nb);
                if (nb>=0) {
```

```
                m=CreerElemen(); m->donnee=nb;
                if (L->taille==0) {L->debut=m; L->fin=m;}
                else if(L->debut->donnee > nb) {m->suivant=L->debut; L->debut=m;}
                else if(L->fin->donnee < nb){ L->fin->suivant=m;  L->fin=m;}
                else{
                        courant=L->debut; pred=NULL;
                        while((courant!=NULL)&&(courant->donnee < nb))
                        { pred=courant; courant=courant->suivant; }
                        pred->suivant=p;  m->suivant=courant;
                      }
                 L->taille++;
                        }
            }
        while(nb>=0); // la condition d'arrêt et la saisie d'un nombre négatif
}

Liste * ConcatListe (Liste *L1, Liste *L2)
{
        Element *courant;
        Liste *L; L =CreerListe() ;
// La fonction va retourner la liste L qui est la concaténation des listes L1 et L2
        courant=L1->debut;
        while(courant!=NULL) { AjoutFin(L, courant->donnee); courant=courant->suivant; }
        courant=L2->debut;
        while(courant!=NULL) { AjoutFin(L, courant->donnee); courant=courant->suivant; }
return L;
}

void AjoutFin (Liste *L, int val)
{
          Element *m;
        m=CreerElemen(); m->donnee=val;
         if(L->taille==0)
          { L->debut=m; L->fin=m ;  }
          else
          {  L->fin->suivant=m; L->fin=m;}
         L->taille++;
 }

void Affichage(Liste *L)
 {
        int i ;
        Element *courant;
        printf("\n Votre liste est: \n");
        if(L->taille==0) printf("\a liste vide!\n");
else{
        courant=L->debut ;
        for (i=0 ; i<L->taille ; i++)
         {printf("->%d ",courant->donnee);courant=courant->suivant; }
    }
}
```

```
int RechercheElem(Liste* L, int val)
{
        int res=0 ;
        Element *courant;
        if (L->taille!=0){ courant=L->debut ;
        while(courant!=NULL&& (courant->donnee !=val)){ courant=courant->suivant; }
       if (courant !=NULL) res=1 ; }
return res ;
}

int SuppPos(Liste*L, int pos)
{
        int res i ;
        Element *courant;
        if (L->taille==0||pos<=0 ||pos>L->taille) res=-1 ;// recherche infructueuse
        else {
        courant=L->debut ;
        for (i=1 ;i<pos ;i++) { courant=courant->suivant; }
        supp_element = courant->suivant;
        res= supp_element ->donnee,
        free (supp_element);
        courant -> suivant = courant -> suivant -> suivant;
        if(courant -> suivant == NULL)  L->.fin = courant;
        }
        L->taille-- ;
return res ;
}
```

```
//**********contenu du fichier liste_run.c*****************//
#include "liste.c"

int main()
{
        int v , pos,res ;
Liste *L1,*L2,*L3;
        L1=CreerListe() ; CreerListeOrdre (L1); affichage(L1);
        L2 CreerListe(); CreerListeCroissante (L2);  affichage(L2);
        L3=Concatenation(L1, L2); affichage(L3);
        printf("Donner la valeur à rechercher dans la liste concaténée\n ") ;
        scanf("%d",&v) ;
        res= RechercheElem(L3,v) ;
        if(res==-1) printf("la valeur %d n'existe pas dans la liste concaténée\n",v) ;
        else printf("la valeur %d existe dans la liste concaténée\n",v) ;
        printf("Donner la position après laquelle vous voulez supprimer un élément  dans la liste
        concaténée\n ") ;
        scanf("%d", &pos) ;
        SuppPos(L3, pos)
        getch();
        return 0;
}
```