Ministère De L'enseignement Supérieur Et De La Recherche Scientifique
Université Hassiba Benbouali de Chlef
Faculté des Sciences Exactes et Informatique
1ère année Mathématique 2022-2023
Module : Algorithmique et structure de données

Fiche TD N° 09 : Les Listes Linéaires Chaînées (Partie 1)

Exercice 1 :

Ecrire un algorithme qui calcule la somme de deux entiers a et b en utilisant les pointeurs. Le traduire par la suite en langage C.

```
Algorithme Exo1
Var p, q : pointeur sur entier ; /*on peut créer une 3ème variable où mettre le résultat*/
      a, b:entier ;
Debut
p← Adr(a) ;
q← Adr(b) ;
Ecrire(« donner deux entiers ») ;
Lire(Valeur(p),Valeur(q) ) ;
Ecrire(« La somme = », Valeur(p)+Valeur(q) ) ;
Fin.
```

```
#include <stdio.h>
int main(){
int *p,*q;
int a, b;
p=&a;
q=&b;
printf("donner 1er entier\n");
scanf("%d", p);
printf("donner 2eme entier\n");
scanf("%d", q);
printf("la somme  = %d ",*p+*q); /*on peut réinitialiser p et q à NULL s'ils ne vont plus pointer sur a et b*/
return 0;
}
```

Exercice 2 :

Ecrire un algorithme qui calcule la somme de deux entiers en utilisant que les pointeurs (sans déclarer les variables a et b). Le traduire par la suite en langage C.

```
Algorithme Exo2
Var p, q, s : pointeur sur entier ;
Debut
Ecrire(« donner le 1er entier ») ;
Allouer(p) ;
Lire(Valeur(p)) ;
Ecrire(« donner le 2eme entier ») ;
Allouer(q) ;
Lire(Valeur(q)) ;
Ecrire(« la somme de », Valeur(p), « et », Valeur(q),  « = ») ;
Allouer(s) ;
Valeur(s)← Valeur(p)+Valeur(q) ;
Ecrire(Valeur(s)) ;
Fin.
```


Mme A. Berrachedi-Meknassi

```c
#include <stdio.h>
#include <stdlib.h>  // pour malloc
int main(){
int *p,*q,*s;
printf("donner le 1er entier\n");
p=malloc(sizeof(int));
scanf("%d",p);
printf("donner le 2eme entier\n");
q=malloc(sizeof(int));
scanf("%d",q);
printf("la somme de %d et %d = ",*p, *q);
s=malloc(sizeof(int));
*s=*p+*q;
printf("%d",*s);
free(p);
free(q);
free(s);
return 0;
}
```

Exercice 3 :

Déclarer en algorithmique et puis en langage C, une liste linéaire chaînée qui permet de stocker les informations d'un groupe d'étudiants. Chaque étudiant possède un nom, l'âge ainsi que la moyenne de l'année universitaire achevée.

```
Type etudiant=enregistrement ;
        nomEt : chaîne de caractères ;
        ageET : entier ;
        moyEt : réel ;
        adrEtSuiv : pointeur sur etudiant ; //on peut mettre adrEtSuiv : *etudiant ; pour ne pas trop écrire
Fin etudiant ;
Var tete :*etudiant ;
Début
tete←NULL ;
```

```c
struct etudiant{
        char nomEt[30];
        int ageEt;
        float moyEt;
        struct etudiant *adrEtSuiv;
};
int main(){
        struct etudiant *tete=NULL;
```

Exercice 4 :

Avec le même énoncé de l'exercice 3, veuillez remplir les informations des étudiants, sachant que l'on ne connait pas leur nombre.

```
Type etudiant=enregistrement ;
        nomEt : chaîne de caractères ;
        ageET : entier ;
        moyEt : réel ;
        adrEtSuiv : *etudiant ;
Fin etudiant ;
```

```
Var     tete, p, q :*etudiant ;
        reponse : caractère ;
Début
tete←NULL ;
Allouer(tete) ;
p← tete ;
Répéter
        Ecrire("Nom de l'etudiant: ");
        Lire (Valeur(p). nomEt) ;
        // Dorénavant, on peut mettre Lire((*p).nomEt), ou bien, Lire(p->nomEt), pour ne pas trop écrire
        Ecrire("Age de l'etudiant: ");
        Lire (p->ageEt) ;
        Ecrire("Moyenne de l'etudiant: ");
        Lire (p->moyEt) ;
        Ecrire("Voulez-vous ajouter un autre etudiant? (o/n)");
        Lire(reponse);
        Si  (reponse = 'o'  ou  reponse ='O') alors
        |       Allouer(q) ;
        |       p->adrEtSuiv ←q;
        |       p ← p->adrEtSuiv;
        |        q← Null ;
        Sinon   p->adrEtSuiv ← NULL;
        FinSi ;
 Jusqu'à (reponse<>'o' et reponse<>'O');
Fin.
```

```
#include <stdio.h>
#include <stdlib.h>
struct etudiant{
        char nomEt[30];
        int ageEt;
        float moyEt;
        struct etudiant *adrEtSuiv;
};
int main(){
        struct etudiant *tete=NULL;
        tete = malloc(sizeof(struct etudiant));
        struct etudiant *p=tete;  //dans langage C, on peut ne pas utiliser le pointeur q
        char reponse;
        do{
                printf("Nom de l'etudiant:\n");
                scanf("%s",&p->nomEt);  // ou bien: scanf("%s",&(*p).nomEt);
                printf("Age de l'etudiant:\n");
                scanf("%d",&p->ageEt);
                printf("Moyenne de l'etudiant:\n");
                scanf("%f",&p->moyEt);
                printf("Voulez-vous ajouter un autre etudiant? (o/n)");
                scanf("%c", &reponse);
                if(reponse == 'o' || reponse =='O'){
                        p->adrEtSuiv = malloc(sizeof(struct etudiant));
                        p = p->adrEtSuiv;
                }
                else      p->adrEtSuiv = NULL;
        }while (reponse=='o' || reponse=='O');
        return 0;
}
```

Exercice 5 : ***(Du cours)***

Ecrire une procédure qui permet d'insérer une valeur entière 'v' dans une liste ordonnée 'L'

```
Procédure ins_ord( v : entire , var L : *maillon );
/* insère v dans L, le résultat (éventuellement la nouvelle tête) est dans L */
Var
p, q, n : *maillon;
trouv : booléen;
Début
/* on commence par rechercher l'endroit où doit être insérée v pour que la liste reste ordonnée (parcours séquentiel jusqu'à trouver une valeur >= v) */
trouv ← FAUX; /* booléen indiquant la fin du parcours */
p ← L; /* pointeur pour parcourir les maillons de la liste L */
q ← NULL; /* pointeur pour garder le précédent de p */
TantQue ((p <> NULL) ET (NON trouv))
    Si ( v <= Valeur(p) ) alors trouv ← VRAI
    Sinon
        q ← p; /* on sauvegarde dans q l'adr du maillon, */
        p ← Suivant(p); /* avant de passer au prochain avec p. */
    FinSi ;
FinTQ;
/* quand on sort de cette boucle TQ, on pourra insérer v entre les maillons pointés par q et p, avec les cas particuliers suivants :
si on sort avec q=NIL, alors v sera insérée au début de la liste
si on sort avec p=NIL, alors v sera insérée à la fin de la liste */
Allouer( n ) ;  /* n pointe maintenant un nouveau maillon */
Aff-val( n, v ) ; /* on y affecte la valeur v */
Aff-adr( n, p ) ;  /* le suivant de n est maintenant p (même si c'est NIL) */
Si(q <> NIL) alors /* s'il existe un maillon qui précède p :*/
    Aff-adr( q, n ) ;  /* le suivant de q devient alors n */
Sinon /* sinon (q=NIL) :*/
    L ← n ; /* n devient la nouvelle tête de la liste */
FinSi ;
Fin.
```