Atelier 3 : Liste chainée : Structure – Mémoire – Pointeurs – Fonctions

Dans cet atelier, nous allons utiliser la structure suivante :

```
typedef struct Element{
 int data ;
 struct Element *next ;
}
```

Le test des fonctions à implémenter se fait dans la fonction main() dont le code est le suivant :

```
int main(){
Element *L ; // &L contient le représentant « L » de note liste chainée
L=NULL ; // la liste est initialisée à NULL

}
```

Exercice 1 : insérer un nouveau élément au début de la liste chainée

1. Soit le code source suivant :

```
void insertBeginning1( ?, int c){

}
```

Modifier le code source ci-dessus pour insérer un nouveau élément au début de la liste sans changer le type de retour

2. Soit le code source suivant :

```
Element* insertBeginning2(Element *L, int c){

}
```

Modifier le code source ci-dessus pour insérer un nouveau élément au début de la liste sans changer les paramètres

Exercice 2 : trier une liste chainée

L'objectif de cet exercice est de trier une liste chainée

1. Écrit une fonction qui permet de trier le contenu d'une liste chainée en se basant sur l'algorithme tri par sélection (échanger le contenu des éléments à permuter)
2. Écrit une fonction qui permet de trier le contenu d'une liste chainée en se basant sur l'algorithme tri par sélection (échanger les adresses des éléments à permuter)

Exercice 3 : insérer un nouveau élément à la bonne position

Étant donnée une Liste chainée « L » d'entiers. Le contenu de cette liste est supposé trié.
L'objectif est d'écrire une fonction **InsertPosition** qui insère un nouveau élément tout en gardant la liste triée.

3. Modélisation du problème
   - Proposer un schéma qui met en évidence l'insertion d'un nouveau élément à la bonne position. Ce schéma doit mettre en évidence aussi la communication entre le **main** et InsertPosition
   - Quels sont les cas qu'il faut prendre en compte
   - Mentionner les étapes à suivre pour chaque cas pour insérer le nouveau élément
4. En fonction de la modélisation, écrire le code qui permet d'insérer un nouveau entier (c) à la bonne position.

Exercice 3 :

1) Donner le résultat du code source suivant :

```
typedef struct Element{
int data;
struct Element *next;
}Element;
Element* createElement(int a){
Element *newElement;
 ewElement=(Element*)
malloc(sizeof(Element));
newElement->next=NULL;
newElement->data=a;
return newElement;
}
Element *inserElement(Element
*L,int c){
Element *elt;
elt=createElement(c);
elt->next=L;
return elt;
}
```

```
Element *findLastAdresse(Element *L){
   Element *ptr;
   if(L==NULL) return NULL;
   ptr=L;
   while(ptr->next!=NULL)
      ptr=ptr->next;
   return ptr;
}
void displayList(Element *L){
   Element *ptr;
   for(ptr=L;ptr!=NULL;ptr=ptr->next)
      printf("%d - ",ptr->data);
}
```

```
int main(){
Element *L;
L=NULL;
L=inserElement(L,1);
L=inserElement(L,2);
L=inserElement(L,3);
L=inserElement(L,4);
displayList(L);
getch();
Element *adr;
adr=findLastAdresse(L);
adr->next=L;
displayList(L);
return 0;
}
```