

```

eErrorCode insertElemAt(pList l, int32_t pos, Element e) {
    eErrorCode returnCode = E_NO_ERROR;           ↓
    int32_t k = 0;
    int32_t n = 0;

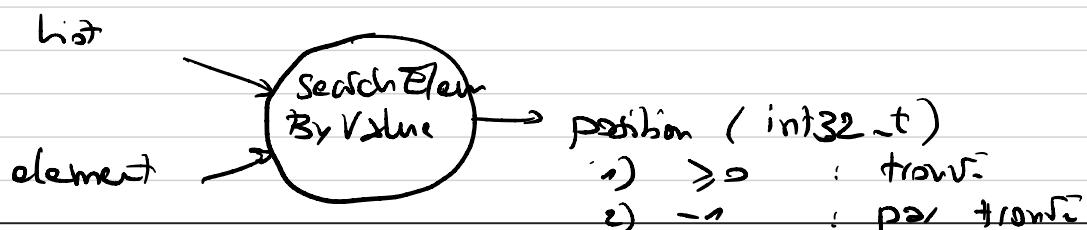
    if (l != NULL)
    {
        if (!isListFull(*l)) { ← liste non pleine
            n = l->numElem;
            if (pos == -1) ] -1 => à la fin (dernier pos.)
            | pos = n;
            for (k = n - 1; k >= pos; k--) ] décale →
            {
                l->e[k + 1] = l->e[k];
            }
            l->e[pos] = e;   ← insertion new élément
            l->numElem++;  ← M.A.J. num elem.
        }
        else {
            returnCode = E_LIST_FULL;
        }
    }
    else {
        returnCode = E_BAD_LIST;
    }
    return returnCode;
}

```

TD20210520 → copier en TD20210523)

① list.c / .h : **displayList** affiche le contenu de la liste passée en param.

② Recherche d'un élément par sa valeur.



Continuité:

Si l'élément est présent plusieurs fois  
on renvoie la position du premier  
rencontré.

"4"

↓  
1 | 4 | 3 | 4 | 2 | 4 | 5

13 "4"

## Recherche d'un élément par valeur

```
int32_t searchElemByValue(pList l, Element e)
{
    int32_t pos=-1;
    int32_t k = 0;
    bool finished = false;

    if (l != NULL) ← test si liste valide
    { ajouté si liste non vide
        boucle de recherche
        while(!finished) {
            if(e==l->e[k]) { ← test si élément
                trouvé
                pos = k; ← position de l'élément
                finished = true; → trouvée
            }
            k++; ← élément suivant
            if (k==l->numElem) {
                finished = true; } test si fin de
            } liste
        }
    }
    else { * : finished = isEmpty(l); }
    pos = -1;
}
return pos;
}
```

### Application :

3	1	1	4	1	1	5	9	1	1	1
---	---	---	---	---	---	---	---	---	---	---

Recherche la position du **4** dans la liste → **2**

~~43<sup>h</sup>60~~

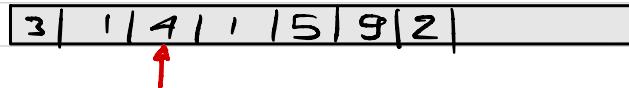
```
initList(&l);
insertElemAt(&l, END_OF_LIST, 3.0); ← 1
insertElemAt(&l, END_OF_LIST, 1.0);
insertElemAt(&l, END_OF_LIST, 4.0);
insertElemAt(&l, END_OF_LIST, 1.0);
insertElemAt(&l, END_OF_LIST, 5.0);
insertElemAt(&l, END_OF_LIST, 9.0);
displayList(l);

pos = searchElemByValue(&l, 4.0);
printf("search result : pos=%d\n", pos);
```

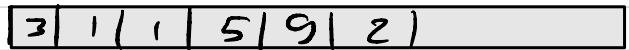
```
→ TD20210531 git:(main) ✘ ./app
3.000 1.000 4.000 1.000 5.000 9.000
search result : pos=2
```

position de l'élément trouvée

## Suppression d'un élément à partir de sa position



$n = 7$



$n = 6$

Recopie d'une position vers la gauche de tous les éléments situés après l'a position initialisée.

Ex:  $\begin{matrix} pos = 2 \\ n = 7 \end{matrix} \Rightarrow$  les éléments [3] à [6] sont déplacés [2] à [5]

M.A.J. nbre éléments

⚠ Liste non vide  
 $pos < n$

List



```
eErrorCode deleteElemAt(pList l, int32_t pos) {
    eErrorCode returnCode = E_NO_ERROR;
    int32_t k = 0;
    if (l != NULL) — test si liste valide
    {
        if (!isEmpty(*l)) {
            for (k = pos; k < l->numElem; k++) {
                l->e[k] = l->e[k + 1];
            }
            l->numElem--;
        } — MAJ nb elem.
    }
    else {
        returnCode = E_BAD_LIST;
    }
    return returnCode;
}
```

boucle de déplacement  
d'une position vers la gauche

Application: (Suite de l'exercice précédent)  
Supposons l'élément "4" ( $pos=2$ )

(17h15)

# Liste Tableau à taille dynamique

31.5.2021

copie

TD 20210531 →

TD 20210531b

La liste tableau dynamique

liste tableau taille fixe

```
typedef struct {
    uint32_t numElem;
    Element e[LIST_CAPACITY];
} sList, *pList;
```

→ Element \*e;  
ajout d'un champ  
uint32\_t maxNumElem;

1

```
eErrorCode initList(pList l) {
    eErrorCode returnCode = E_NO_ERROR;

    if(l!=NULL) {
        l->numElem=0;
    }
    else {
        returnCode = E_BAD_LIST;
    }
    return returnCode;
}
```

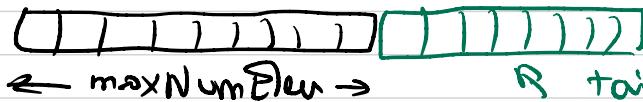
création d'un tableau dynamique  
de LIST\_CAPACITY  
maxNumElem = LIST\_CAPACITY.

malloc

Point Nécessaire: en cas d'insertion sur liste pleine

Avertis: erreur LIST-FULL

Maintenant: on pourra agrandir le tableau  
de la liste.



taille d'agrandissement  
facteur d'agrandissement

ex: + 10 cellules

#define

agrandir la liste

realloc

```
eErrorCode insertElemAt(pList l, int32_t pos, Element e) {
    eErrorCode returnCode = E_NO_ERROR;
    int32_t k = 0;
    int32_t n = 0;

    if (l != NULL)
    {
        if(!isListFull(*l)) {
            n = l->numElem;
            if(pos== -1)
                pos = n;
            for (k = n - 1; k >= pos; k--)
            {
                l->e[k + 1] = l->e[k];
            }
            l->e[pos] = e;
            l->numElem++;
        }
        else {
            returnCode = E_LIST_FULL;
        }
    }
    else {
        returnCode = E_BAD_LIST;
    }
    return returnCode;
}
```

3

à terminer pour le prochain cours  
du lundi 7.6.

17/3