DOSSIER ALGORITHMIQUE

INSERTION EN TETE ET EN QUEUE D'UNE LISTE SIMPLEMENT CHAINEE CIRCULAIRE

INSERTION EN TETE DE LISTE

Problème : ajouter un nouvel élément au début de la liste ;

Principe : -créer une nouvelle cellule p qui contient la valeur v ;

-si la liste est **vide**, p pointe vers lui-même sinon on ajuste les pointeurs, on relie p avant l'ancien premier et après le dernier élément ;

Dictionnaire de données :

l : pointe vers la première cellule de la liste ;

p : nouvelle cellule à insérer ;

t : dernière cellule de la liste ;

val : valeur contenue dans une cellule ;

suiv : pointeur vers le suivant .

Algorithme :

```
#include<stdlib.h>
typedef struct cellule{
       int val;
       struct cellule*suiv;
}cellule;
void inserertete(cellule*l,int v){
     cellule*p=malloc(sizeof(cellule));
     p->val=v;
     cellule*t=l;
     if(l==NULL){
       p-suiv=p;
       l=p;
     }
     else{
       t->suiv=t;
       p->suiv=l;
       t->suiv=p;
       l=p;
     }
```

```
}
```

Complexité :

Temps : O(1) car on ne parcours pas la liste, on modifie uniquement quelques pointeurs ;

Espace : O(1) chaque nouvelle cellule alloue un espace fixe pour une valeur et un pointeur(suiv).

INSERTION EN QUEUE DE LISTE

Problème : ajouter un nouvel élément p à la fin de la liste ;

Principe : -créer une nouvelle cellule p qui contient la valeur v ;

- Si la liste est **vide**, p pointe vers lui-même sinon on ajuste les pointeurs, on relie p après l'ancien dernier et avant le premier ;

Dictionnaire de données :

l : pointe vers la première cellule de la liste ;

p : nouvelle cellule à insérer ;

t : dernière cellule de la liste ;

val : valeur contenue dans une cellule ;

suiv : pointeur vers le suivant ;

Algorithme :

```
void insererqueue(cellule*l,int v){
     cellule*p=malloc(sizeof(cellule));
     p->val=v;
     cellule*t=l;
     if(l==NULL){
       p-suiv=p;
       l=p;
     }
     else{
       t->suiv=t;
       t->suiv=p;
       p->suiv=l;
```

```
     }
}
int main(){
    cellule*p;
    int v;
    inserertete(p,v);
    insererqueue(p,v);
}
```

Complexité :

Temps : O(1) car on ne parcours pas la liste, on modifie uniquement quelques pointeurs ;

Espace : O(1) chaque nouvelle cellule alloue un espace fixe pour une valeur et un pointeur(suiv).

La complexité en espace de la structure est **0(n).**