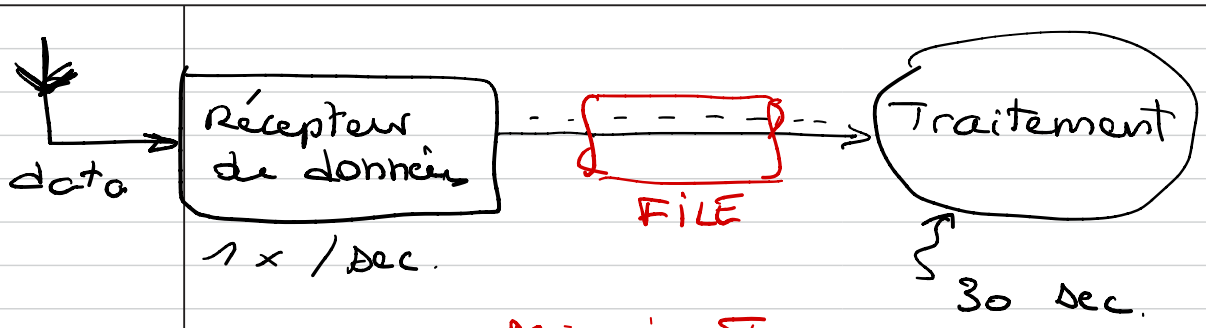
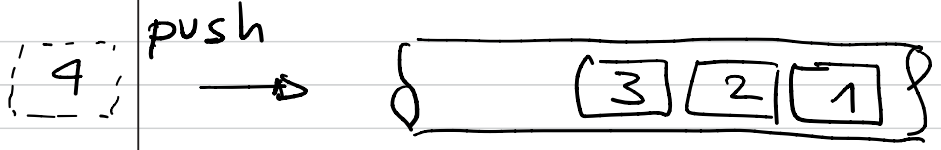
INFO2 TIN - FILE 2.VI.2020



Mémoire Tampo
l'ordre des données est conservé.

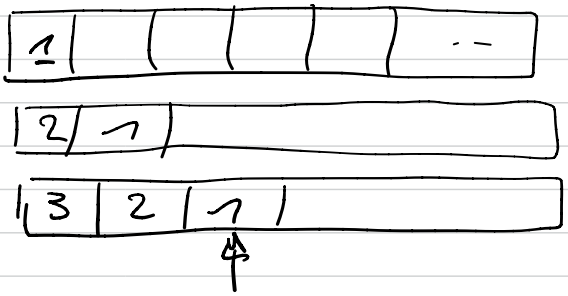
Entrée	1	puis 2	puis	3	Si Pile 1, 2, 3
Sortie	1	puis 2	puis	3	3, 2, 1

"FIFO" First IN First OUT



push →

- insertion (pos = ∅)



pop

- getElem (pos = fin)
- removeElem (pos = fin)

TD20200602

- list.c
- list.h
- listType.h
- fifo.c
- fifo.h

) TDA "List"

- initFifo
- push
- pop

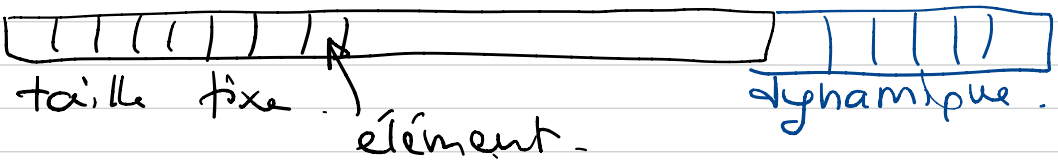
Structure du TD
+ prototype des 3 fonctions

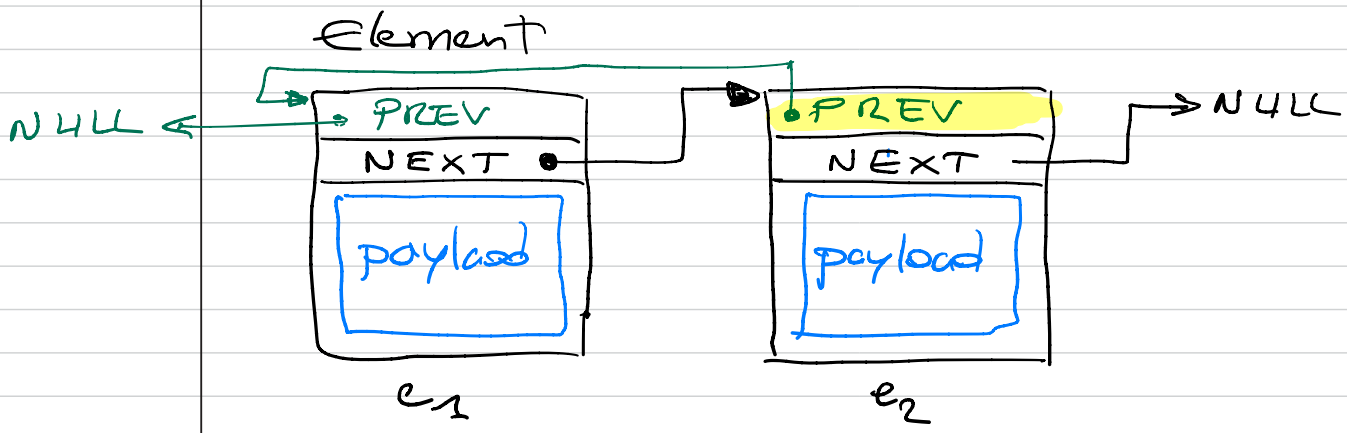
impl. fifo.c 15h40 ~~15h20~~

LISTE CHAÎNEE

02.VI.2020.

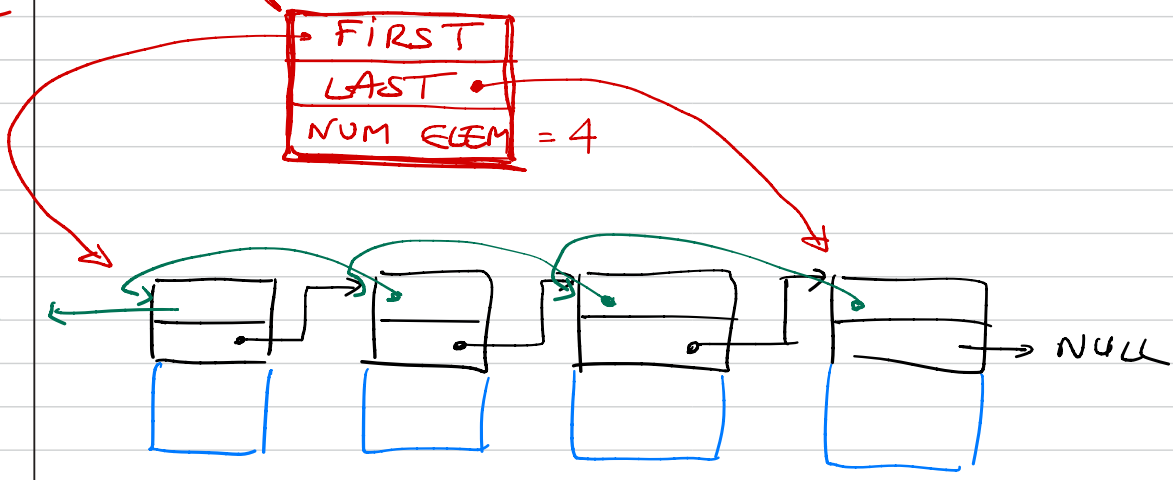
Liste Tableau





alloué dynamiquement

Liste chainée



Element structure de donnée "elem"

1) champs prev pointeur sur un elem
2) champs next pointeur sur un elem
3) champs payload structure des données dans la liste

TD 2020 0602b

TP2020602.c / .h
LinkedList.c / .h.

- .h → struct payload (Point 2D - listType.h)
  → struct elem (3 champs)

compilation

16h05

Structure payload : typedef struct {
int32_t row;
int32_t col;
} payload;

Elem:

pointeur sur elem prev
pointeur sur elem next
payload.

"elem"

```
typedef struct {
    elem * prev;
    elem * next;
    payload p;
} elem;
```

type utilisé AVANT sa déclaration

Modèle d'écriture

```
typedef struct elem {
    struct elem * prev;
    struct elem * next;
    payload p;
} elem;
```

structure "récursive" de données

linkedList

```
typedef struct {
    elem * first;   //struct elem * first;
    elem * last;
    uint32_t numElem;
} linkedList;
```

```
// functions prototypes
void initList(linkedList *l);
elem *createElem(payload p);
```

- initList => (linkedList) vide ≡ numElem = 0, first = NULL; last = NULL
- createElem => (alloc. dynamique), payload, prev et next = NULL
- insertElem => position, elem

16h30