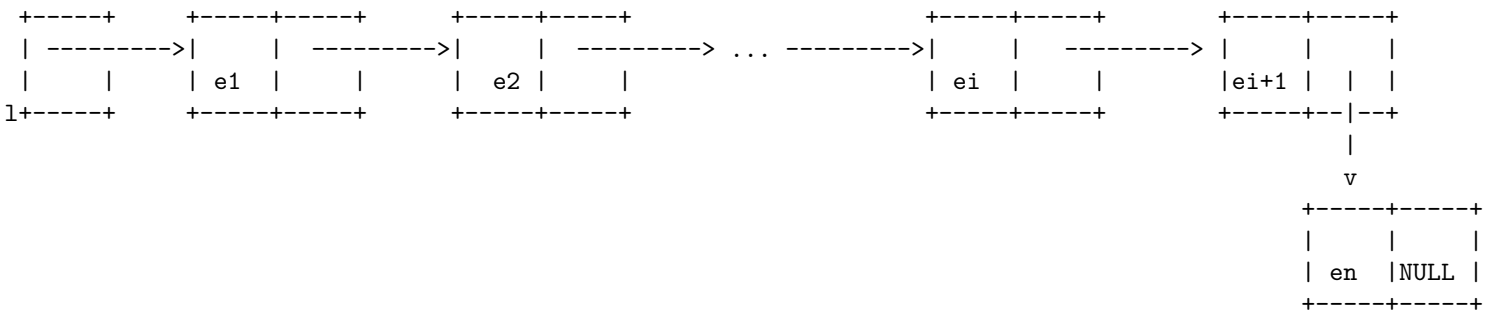
TD 7
Implémentation dynamique d'une liste

TAD
Semestre 2

1

La liste $l = (e_1, e_2 \dots e_i, e_{i+1} \dots e_n)$



2

```
1  fonction longueur (entree l <Liste[T]>)
2    retourne <Entier>
3  glossaire
4    k <Entier>; -- compteur de cellules
5    aux <Liste[T]>; -- pour parcourir les cellules
6  debut
7    k <- 0;
8    aux <- l;
9    tantque aux /= NULL faire
10     k <- k + 1;
11     aux <- aux↑.suivant;
12   fin tantque;
13   retourner (k);
14 fin
```

Listing 1 – Longueur

```
1  procedure creerListe (sortie l <Liste[T]>)
2  debut
3    l <- NULL;
4  fin
```

Listing 2 – creerListe

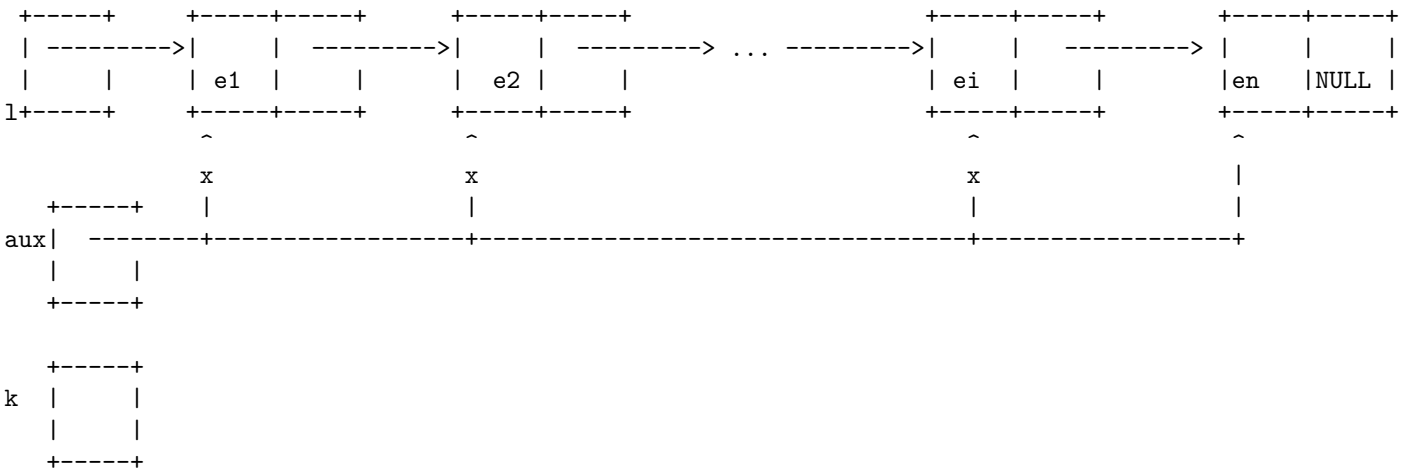
```
fonction ieme (entree l <Liste[T]>, entree i <Entier>)
retourne <Entier>
declenche listeVide, rangInvalide
glossaire
  k <Entier>; -- pour compter les cellules de 1 à i
  courant <Liste[T]>; -- pointeur sur la cellule de rang i
debut
  si l = NULL alors
    declencher listeVide;
  fin si;

  -- on ne test pas i > longueur(l) tout de suite: pour éviter un parcours
  --   séquentielle die plus
  si i < 1  alors
    declencher (rangInvalide);
  fin si;
  k <- 1;
  courant <- l;
  tantque k <= i - 1 faire
    courant <- courant↑.suivant;
    si courant = NULL alors
      declencher (rangInvalide);
    fin si;
    k <- k + 1;>i]
  fin tantque;
  retourner (courant↑.valeur);
fin
```

Listing 3 – ieme

2.1 inserer

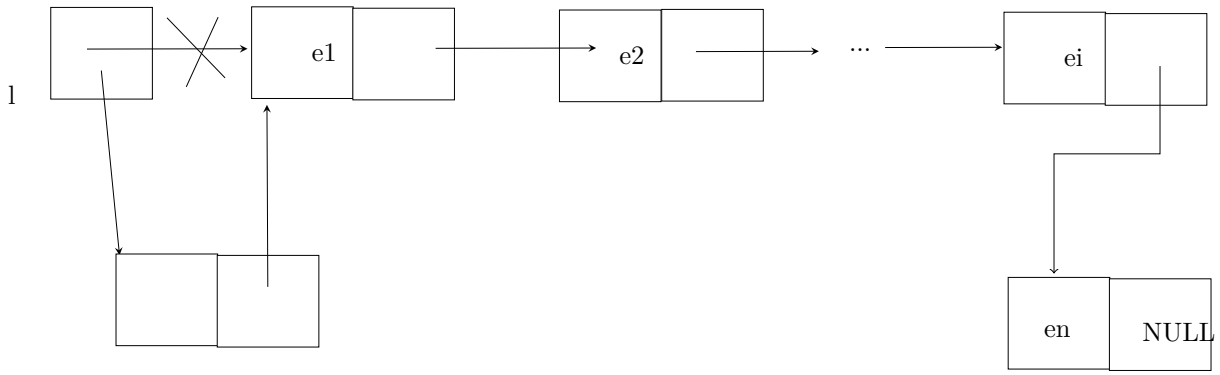
2.1.1 Cas généra




```
-- parcourir la liste l jusqu'a la cellule ei (pointée par courant)
-- et précédent pointe ei - 1
courant <- l;
k <- 1;
tantque k <= i -1 ;
  precedent <- courant;
```

```
7     courant <- courant↑.suivant;
8     k <- k+1;
9   fin tantque;
10  -- insérer e entre ei-1 et ei
11  nouveau↑.valeur <- e;
12  nouveauuparrow..suivant <- courant;
13  precedentuparrow.suivant <- nouveau;
```

2.1.2 Cas particulier i=1 pour insérer




```
1   allouer(nouveau);
2   si nouveau = NULL alors
3     declencher(listePleine);
4   fin si;
5   nouveau↑.valeur <- e;
6   nouveau↑.suivant <- l;
```

2.1.3 Cas particulier fin de liste pour insérer

Ce cas se traite de la même manière qu'un cas général.

2.1.4 Programmation de la fonction

```
1   procedure inserer (maj l <Liste[T]>, entree i <Entier>, entree e <T>)
2     declenche rangInvalide, listePleine
3   glossaire
4     k <Entier>.
5     courant <Liste[T>;
6     precedent <Liste[T]>;
7     nouveau <Liste[T]>g
8   debut
9     -- intercepter l'anomalie rangInvalide pour i < 1
10    si i < 1 alors
11      declencher (rangInvalide);
12    fin si;
13
14    --insérer l'élément e dans la liste
15    --cas début de liste
16    si i = 1 alors
17      allouer(nouveau);
18      si nouveau = NULL alors
19        declencher(listePleine);
20      fin si;
21      nouveau↑.valeur <- e;
22      nouveau↑.suivant <- l;
```

```
23    sinon
24      --rechercher la cellule de rang i
25      courant <- l;
26      k <- 1;
27      tantque k <= i -1 faire
28        precedent <- courant;
29        si precedent = NULL alors
30          declencher(rangInvalide);
31        fin si;
32        courant <- courant↑.suivant;
33        k <- k + i;
34      fin tantque;
35      -- insérer l'élément e après ei-1
36      allouer(nouveau);
37      si nouveau = NULL alors
38        declencher (listePleine);
39      fin si;
40      nouveau↑.valeur <- e;
41      nouveau↑.suivant <- courant;
42      nouveau↑.precedent <- nouveau;
43    fin si;
44  fin
```

Listing 4 – inserer