L'avantage d'un tableau est qu'il permet un accès direct aux données : accéder au $\texttt{n}^{\text{ème}}$ élément d'un tableau demande un temps constant tant que `n` est d'un type scalaire comme `int` et un temps proportionnel au nombre de chiffres de `n`, c'est-à-dire au logarithme de `n`, si on remplace ce type scalaire par un type de grands entiers — ce qui est conceptuellement nécessaire pour que l'on puisse faire tendre `n` vers l'infini. Accéder au $\texttt{n}^{\text{ème}}$ élément d'une liste, en revanche, demande un temps linéaire en `n`.

L'avantage d'une liste est que sa taille n'a pas besoin d'être connue à l'avance et les listes utilisées avec la récursivité et la recopie des arguments permettent de se restreindre au fragment fonctionnel du langage utilisé. Une donnée n'est alors jamais transformée, elle peut être utilisée un nombre arbitraire de fois et les fonctions ressemblent aux fonctions mathématiques : elles prennent un objet en argument, renvoient un autre objet et ne modifient aucun objet déjà construit.

En simplifiant quelque peu, on peut donc dire que les listes permettent une programmation plus simple, mais les tableaux une programmation plus efficace.

6.5 Les piles et les files

En mathématiques, en munissant le même ensemble d'opérations différentes on construit des structures différentes. De même, dans les langages de programmation, les listes, les piles et les files sont trois structures dont l'ensemble de base est le même, mais dont les opérations sont différentes. Les opérations sur les listes étaient

1. fabriquer la liste vide,
2. tester si une liste est vide,
3. fabriquer une liste dont la tête est l'objet `x` et la queue la liste `l`,
4. accéder à la tête d'une liste,
5. accéder à la queue d'une liste,
6. modifier la tête d'une liste,
7. modifier la queue d'une liste.

6.5.1 Les piles

Les opérations sur les piles sont

1. fabriquer la pile vide,