Chapitre 3 : Listes, piles et files

1. Introduction

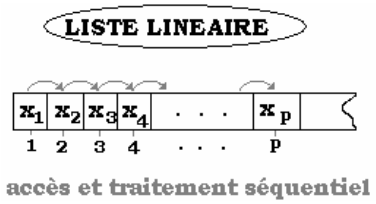
Nous allons étudier dans la suite 3 exemples complets de TAD classiques : la **liste linéaire**, la pile **LIFO**, la file **FIFO**.

2. Liste linéaire (spécifications abstraite et concrète)

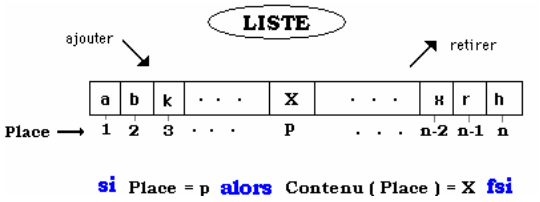
Spécification abstraite

Répertorions les fonctionnalités d'une liste en soulignant les verbes d'actions et les noms, à partir d'une description semi-formalisée:

- C'est une structure de donnée séquentielle dans laquelle les données peuvent être traitées les unes à la suite des autres.



- Il est possible dans une telle structure d'ajouter ou de retirer des éléments en n'importe quel point de la liste.
- L'ordre des éléments est primordial. Cet ordre est construit, non sur la valeur des éléments de la liste, mais sur les places (rangs) de ces éléments dans la liste.
- Chaque place a un contenu de type T_0.
- Le nombre d'éléments d'une liste l est appelé longueur de la liste. Si la liste est vide nous dirons que sa longueur est nulle (longueur = 0).
- On doit pouvoir effectuer au minimum (non exhaustif) les actions suivantes sur les éléments d'une liste l : accéder à un élément de place fixée, supprimer un élément de place fixée, insérer un nouvel élément à une place fixée, etc ....



Signification des opérations : (spécification abstraite)

- **acces(L,k)** : opération générale d'accès à la position d'un élément de rang k de la liste L.
- **supprimer(L,k)** : suppression de l'élément de rang k de la liste L.
- **inserer(L,k,e)** : insérer l'élément e de T_0 , à la place de l'élément de rang k dans la liste L.
- **kième(L,k)** : fournit l'élément de rang k de la liste.

Spécification opérationnelle concrète

- La liste est représentée en mémoire par un ***tableau*** et un attribut de ***longueur***.
- Le kème élément de la liste est le kème élément du tableau.
- Le tableau est plus grand que la liste (il y a donc dans cette interprétation une contrainte sur la longueur. Notons Longmax cette valeur maximale de longueur de liste).

Il faut donc, afin de conserver la cohérence, ajouter deux préconditions :

long(**L**) **def_ssi** long(**L**) **≤** Longmax

inserer(**L,k,e**) **def_ssi** (1 **≤ k ≤** long(**L**)+1) **et** (long(**L**) **≤** Longmax**)**

D'autre part la structure de tableau choisie permet un traitement itératif de l'opération kème (une autre spécification récursive de cet opérateur est possible dans une autre spécification opérationnelle de type dynamique).

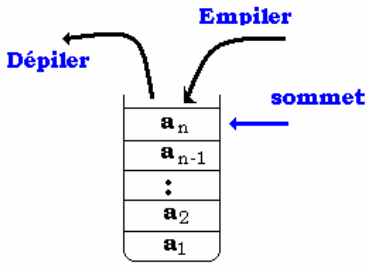
kème(L,k) = contenu(acces(L,k))

3. La pile LIFO (spécification abstraite et concrète)

Spécification abstraite

Répertorions les fonctionnalités d'une pile LIFO (Last In First Out) en soulignant les verbes d'actions et les noms, à partir d'une description semi-formalisée :

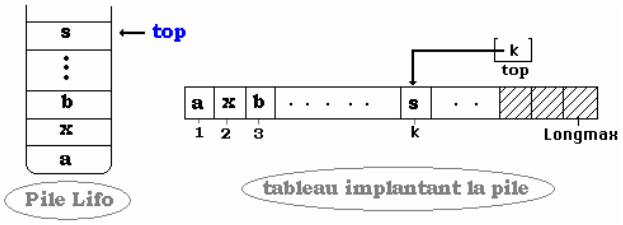
- C'est un modèle pour toute structure de donnée où l'on accumule des informations les unes après les autres, mais où l'on choisit de n'effectuer un traitement que sur le dernier élément entré. *Exemples* : pile de dossiers sur un bureau, pile d'assiettes, etc...
- Il est possible dans une telle structure d'ajouter ou de retirer des éléments uniquement au début de la pile.
- L'ordre des éléments est imposé par la pile. Cet ordre est construit non sur la valeur des éléments de la liste, mais sur les places (rangs)de ces éléments dans la liste. Cet ordre n'est pas accessible à l'utilisateur, c'est un élément privé.
- La pile possède une place spéciale dénommée sommet qui identifie son premier élément et contient toujours le dernier élément entré.
- Le nombre d'éléments d'une pile LIFO P est appelé profondeur de la pile. Si la pile est vide nous dirons que sa profondeur est nulle (profondeur = 0).
- On doit pouvoir effectuer sur une pile LIFO au minimum (non exhaustif) les actions suivantes : voir si la pile est vide, dépiler un élément, empiler un élément, observer le premier élément sans le prendre, etc...



- C'est une structure de donnée séquentielle dans laquelle les données peuvent être traitées les unes à la suite des autres à partir du sommet.

Spécification opérationnelle concrète

- La Pile est représentée en mémoire dans un *tableau*.
- Le *sommet* (noté top) de la pile est un *pointeur* sur la case du tableau contenant le début de la pile. Les variations du contenu k de top se font au gré des empilements et dépilements.
- Le tableau est plus grand que la pile (il y a donc dans cette interprétation une contrainte sur la longueur, notons *Longmax* cette valeur maximale de profondeur de la pile).



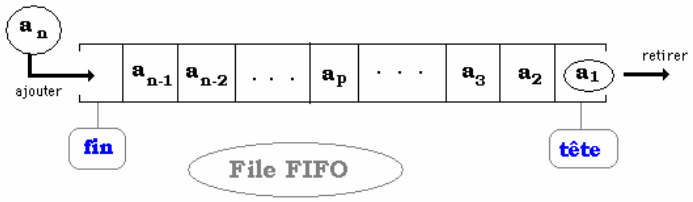
- L'opérateur *empiler* : rajoute dans le tableau dans la case pointée par top un élément et top augmente d'une unité.
- L'opérateur *depiler* : renvoie l'élément pointé par top et diminue top d'une unité.
- L'opérateur *premier* fournit une copie du sommet pointé par top (la pile reste intacte).
- L'opérateur *Est_vide* teste si la pile est vide (vrai si elle est vide, faux sinon).

4. La file FIFO (spécification abstraite seule)

Spécification abstraite

Répertorions les fonctionnalités d'une file FIFO (First In First Out) en soulignant les verbes d'actions et les noms, à partir d'une description semi-formalisée:

- C'est un modèle pour toute structure de données où l'on accumule des informations les unes après les autres, mais où l'on choisit d'effectuer un traitement selon l'ordre d'arrivée des éléments, comme dans une file d'attente.
- *Exemples* :toutes les files d'attente, supermarchés, cantines , distributeurs de pièces, etc...
- Il est possible dans une telle structure d'ajouter des éléments à la fin de la file, ou de retirer des éléments uniquement au début de la file.
- L'ordre des éléments est imposé par la file. Cet ordre est construit non sur la valeur des éléments de la liste, mais sur les places (rangs)de ces éléments dans la liste. Cet ordre n'est pas accessible à l'utilisateur, c'est un élément privé.
- La file possède deux places spéciales dénommées tête et fin qui identifient l'une son premier élément, l'autre le dernier élément entré.
- Le nombre d'éléments d'une file FIFO " F " est appelé longueur de la file ; si la file est vide nous dirons que sa longueur est nulle (longueur = 0).
- On doit pouvoir effectuer sur une file FIFO au minimum (non exhaustif) les actions suivantes : voir si la file est vide, ajouter un élément, retirer un élément, observer le premier élément sans le prendre, etc...

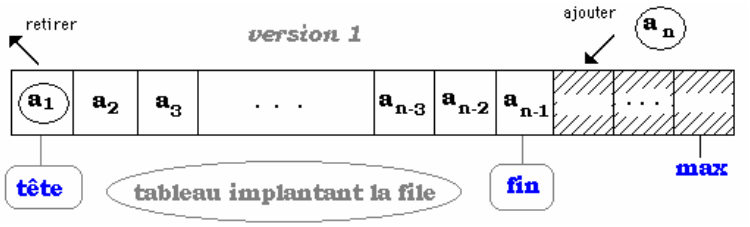


Spécification opérationnelle concrète

- La file est représentée en mémoire dans un *tableau*.
- La *tête* de la file est un pointeur sur la case du tableau contenant le début de la file. Les variations de la valeur de la tête se font au gré des ajouts et des retraits.
- La *fin* ne bouge pas, c'est le point d'entrée de la file.
- Le tableau est plus grand que la file (il y a donc dans cette interprétation une contrainte sur la longueur ; notons *max* cette valeur maximale de longueur de la file).

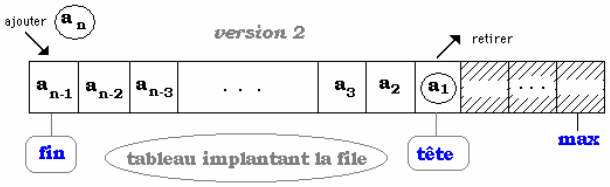
- L'opérateur *ajouter* : ajoute dans le tableau dans la case pointée par fin un élément et tête augmente d'une unité.
- L'opérateur *retirer* : renvoie l'élément pointé par tête et diminue tête d'une unité.
- L'opérateur *premier* fournit une copie de l'élément pointé par tête (la file reste intacte).
- L'opérateur *Est_vide* teste si la file est vide (vrai si elle est vide, faux sinon).

On peut ajouter après la dernière cellule pointée par l'élément fin comme le montre la figure ci-dessous :



dans ce cas retirer un élément en tête impliquera un décalage des données vers la gauche.

On peut aussi choisir d'ajouter à partir de la première cellule comme le montre la figure ci-dessous :



dans ce cas ajouter un élément en fin impliquera un décalage des données vers la droite.