Les structures de données dynamiques

I. Présentation

Les structures de données permettent de relier entre elles des données. Nous avons déjà étudié les tableaux et les enregistrements.

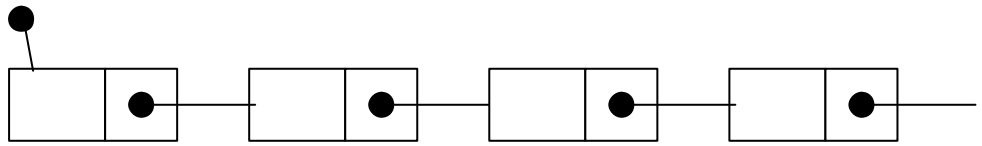
On peut classer les structures de données en deux grandes catégories

- Les structures de données linéaires, qui permettent de relier des données en séquence (on peut numéroter les éléments)
- Les structures de données non linéaires, qui permettent de relier un élément à plusieurs autres éléments

Principales structures de données **LINEAIRES**

- Tableaux
- Listes chaînées
  - à simple chaînage
    - piles
    - files
  - à double chaînage

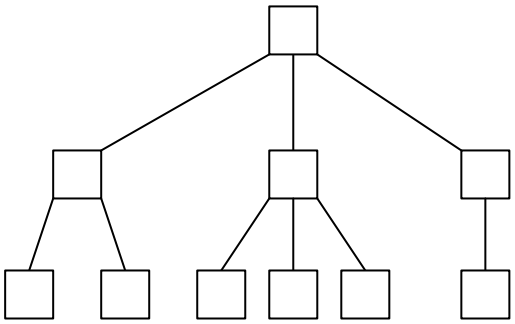
Dans une liste chaînée, les éléments sont reliés grâce à un pointeur. Le pointeur indique l'adresse de l'élément suivant. Les éléments ne sont pas forcément contigus en mémoire, contrairement aux éléments des tableaux.



Principales structures de données **NON LINEAIRES**

- Arbres
  - binaire
  - n-aire
- Graphes
- Enregistrements

Un arbre permet de représenter des structures hiérarchiques. Chaque élément (appelé nœud) a un seul « père » et plusieurs « enfants ».



Un graphe permet de relier des éléments deux à deux de façon quelconque. Cela permet, par exemple, de réaliser des calculs d'optimisation tels que le PERT.

Les structures de données **dynamiques** sont celles dont **les éléments sont mémorisés dans le tas et non dans la pile**. Cela permet de réserver des emplacements dynamiquement à l'exécution et non statiquement à la compilation.
Les structures de données dynamiques sont :

- Les listes chaînées
- Les arbres
- Les graphes

II. Les listes chaînées particulières

A. Les piles [stack]

Présentation

Les piles sont des structures de données, où **l'ajout et le retrait d'un élément se fait toujours au sommet** (en tête de liste).

Le plus souvent, les piles sont implantées sous forme de liste chaînée[1],
Une pile suit la règle **LIFO** (Last In, First Out): dernier entré, premier sorti.

Terminologie particulière:

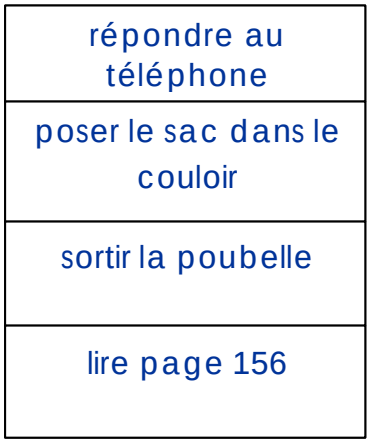
- l'extrémité où s'effectue l'ajout et le retrait s'appelle **sommet**
- ajouter un élément à une pile se dit **empiler** ou **push**
- le fait de retirer un élément d'une pile et de récupérer son information s'appelle **dépiler** ou **pop**

[1] Parfois, une pile est implémentée à partir d'un tableau

Application

Vous lisez tranquillement un livre chez vous quand votre père vous demande de sortir la poubelle. Vous marquez la page de votre livre et vous vous dirigez vers la poubelle. Pendant que votre marchez vers la porte avec le sac, un coup de téléphone vous interrompt. Vous posez le sac dans le couloir, puis vous allez répondre au téléphone. Quand le coup de fil est terminé, vous vous rappelez où il faut que vous récupériez le sac pour le sortir. Vous le sortez et revenez à votre livre, à la page que vous avez marquée et continuez votre lecture.

On peut représenter cette scène par une pile qui sera empilée puis dépilée au cours du déroulement des événements.



C'est sur le même principe que fonctionne un programme.

A chaque fois qu'on appelle une fonction ou procédure, on impose au programme d'arrêter l'exécution du code en cours et de se brancher sur le sous-programme. Le programme doit enregistrer l'endroit où il se trouve dans le programme appelant et la valeur des variables à ce moment avant de se brancher vers le sous-programme appelant. Il enregistre ces informations dans une pile. Au retour d'appel du sous-programme, la pile est dépilée, ainsi les informations de l'état avant appel sont récupérées.

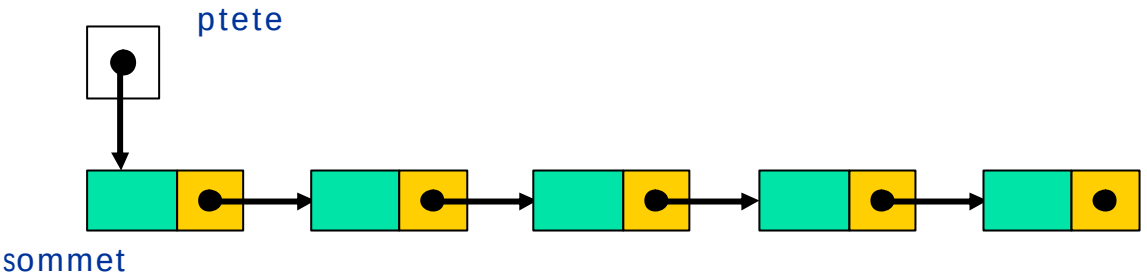
C'est donc grâce à une pile que le programme gère l'appel en cascade de plusieurs sous-programmes. Cette pile est gérée automatiquement par le système d'exploitation, le programmeur n'a pas à s'en préoccuper.

Exemple:
Un programme A appelle un sous-programme B qui appelle lui-même un sous-programme C qui appelle D. La pile du programme va évoluer comme suit:

lancement de A	pile vide
appel de B par A	l'état de A est empilé
appel de C par B	l'état de B est empilé sur A
appel de D par C	l'état de C est empilé sur B
Fin de D	dépilage de l'état de C → poursuite de C
Fin de C	dépilage de l'état de B → poursuite de B
Fin de B	dépilage de l'état de A → poursuite de A

Représentation sous forme de liste chaînée

On ne peut manipuler une pile qu'à travers son sommet. Le sommet d'une pile représentée sous forme de liste chaînée est la tête de cette liste. Le seul pointeur sur la liste est donc le pointeur de tête. La création d'un pile se fait "à l'envers": le dernier élément créé se retrouve en tête (au sommet).



B. Les files ou queues [queue]

Présentation

Une file ou queue est une structure de données où l'insertion d'un élément se fait à une extrémité appelée queue et le retrait d'un élément se fait à une autre extrémité appelée tête.

Une file est le plus souvent représentée par une liste chaînée.
Une file suit la règle **FIFO** (First In, First Out): premier entré, premier sorti

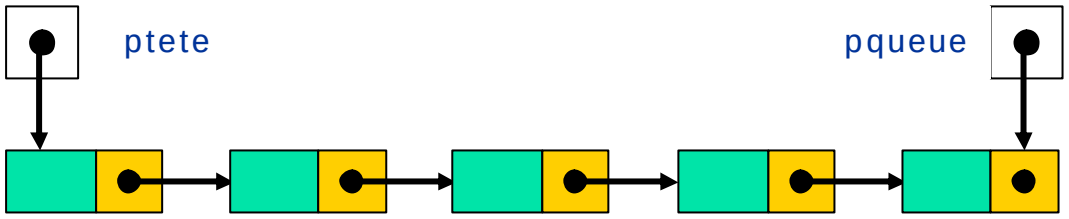
Application

Les queues servent à traiter des données dans l'ordre où elles arrivent, comme dans une file d'attente à un guichet où le premier arrivé est le premier servi.

Représentation sous forme de liste chaînée

Une file doit être accessible à la fois par la tête pour retirer un élément et par la queue pour ajouter un nouvel élément. La tête et la queue d'une file correspondent à la tête et la queue de la liste chaînée qui l'implémente.

Dans une file, le premier élément créé doit être en tête et le dernier élément créé doit être en queue. Or, si l'on n'utilise qu'un seul pointeur ptete pour créer la liste, le premier élément créé se retrouve en queue.
Comment faire pour résoudre ce problème? On va utiliser deux pointeurs sur la liste: un pointeur sur la tête et un pointeur sur la queue. **La création se fera par ajout successif en queue** (alors qu'une pile est créée par insertion successive en tête).



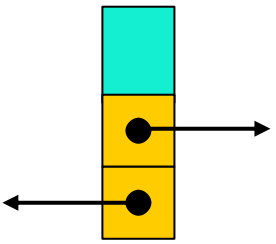
Les variantes de files

Il existe plusieurs variantes de files, notamment les files à deux queues. Dans les files à deux queues, l'entrée peut se faire en queue mais aussi en tête de file. Cela permet de gérer des données exceptionnelles qui doivent être traitées avant toutes les autres.

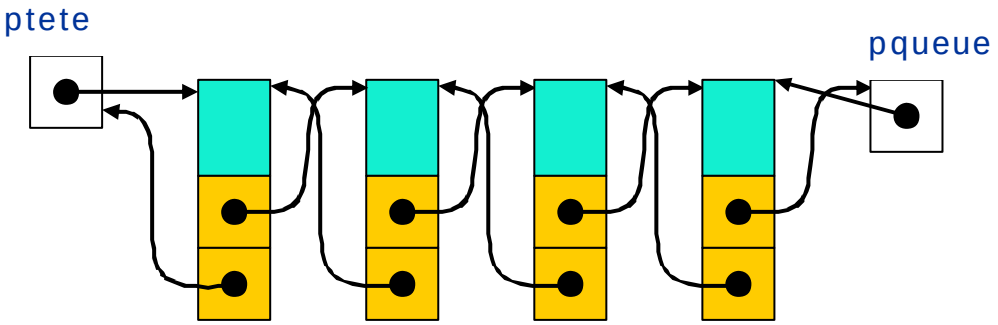
C. Les listes doublement chaînées

Les listes doublement chaînées sont des listes où chaque élément pointe sur son suivant et sur son précédent. Le parcours d'une telle liste peut se faire dans les deux sens. Ce type de liste facilite en outre l'ajout d'un élément au milieu de la liste de sorte que celle-ci reste triée. Le premier élément possède un pointeur qui pointe sur la tête. Le dernier élément possède un pointeur qui pointe sur la queue.

Telem : **enregsitrement**
    info : type de l'info
    psuiv : pointeur sur Telem
    pprec : pointeur sur Telem
Fin **enregistrement**



Remarque1 : Les pointeurs doivent toujours être les derniers champs déclarés.



Remarque 2: ptête est pqueue sont déclarés dans la pile (et non dans le tas)