Listes, Piles et Files

En informatique, un type abstrait (en anglais, abstract data type ou ADT) est une spécification mathématique d'un ensemble de données et des opérations qu'on peut effectuer sur elles.

Les **listes, pile** et **file** sont des types abstraits.

Les Listes

Les listes mémorisent les données en respectant un certain ordre : on peut ajouter, enlever ou consulter un élément en début ou en fin de liste, vider une liste ou savoir si elle contient un ou plusieurs éléments.

Opérations usuelles sur les listes :

- modifier un élément
- supprimer un élément
- insérer un nouvel élément
- fusionner 2 listes
- rechercher un élément d'une valeur particulière
- trier
- …

Une liste L est composée de 2 parties : sa **tête** (head), qui correspond au dernier élément, et sa **queue** (tail)) qui correspond au reste de la liste.

Le langage de programmation **Lisp** (inventé par John McCarthy en **1958**) a été un des premiers langages de programmation à introduire cette notion de liste (Lisp signifie "list processing").

Piles

Dans les piles, on ne peut manipuler que le dernier élément introduit, au sommet de la pile (comme avec une pile d'assiettes). Le principe de la pile est appelé **LIFO** (**L**ast **I**n **F**irst **O**ut : le dernier rentré sera le premier à sortir).

Une pile est une liste sur laquelle on autorise 4 opérations seulement :

- consulter le dernier élément de la pile ==> SOMMET(P)
- tester si la pile est vide ==> PILEVIDE(P) booleen
- empiler un élément, le mettre au sommet de la pile ==> PUSH(P,element)
- dépiler un élément (par le sommet) ==> POP(P)

Files

Dans les files on insère les nouveaux éléments à la fin et où on enlève des éléments au début (comme dans une file d'attente). Le principe de la file est appelé **FIFO** (**F**irst **I**n **F**irst **O**ut) : le premier élément inséré est aussi le premier retiré.

Opérations autorisées sur une file :

- Tester si la file est vide
- ajouter un nouvel élément
- récupérer l'élément situé en bout de file tout en le supprimant
- accéder à l'élément situé en bout de file sans le supprimer
- connaitre le nombre d'éléments présents

Types abstraits et implémentation

Lorsqu'on traduit un algorithmes dans un langage compréhensible pour un ordinateur (Python, Java, C,...). On dit alors que l'on implémente l'algorithme. Il faut donc implémenter les types de données comme les listes, les piles ou les files.

L'implémentation de ces types abstrait est loin d'être triviale et dépend du langage de programmation.

Pour implémenter les listes (ou les piles et les files), la plupart des langages de programmation utilisent 2 structures : les **tableaux** et les **listes chaînées**.

Tableaux

Un tableau est une suite contiguë de cases mémoires (les adresses des cases mémoire se suivent). Le système réserve une plage d'adresse mémoire afin de stocker des éléments. La taille d'un tableau est fixe : une fois le nombre d'éléments défini il n'est pas possible modifier sa taille. Si l'on veut insérer une donnée, on doit créer un nouveau tableau.

Dans certains langages de programmation, on trouve une version "évoluée" des tableaux : les tableaux dynamiques (les listes Python sont des tableaux dynamiques). Les tableaux dynamiques ont une taille qui peut varier. Ce type de tableaux permet d'implémenter facilement les types abstraits.

Listes chaînées

Dans une liste chaînée, à chaque élément de la liste on associe 2 cases mémoire : la première case contient l'élément et la deuxième contient l'adresse mémoire de l'élément suivant.

Il est aussi possible d'implémenter les types abstraits en utilisant des structures plus complexes que les tableaux et les listes chaînées. Par exemple, en Python, il est possible d'utiliser les tuples pour implémenter le type abstrait liste.

Lien video

https://youtu.be/v_g1yizlUxc