LEÇON 6 : Les piles et les files

Table des matières

I - **1- Les Piles** 3

II - **Application 1 :** 5

III - **2- Files** 7

IV - **Application 2 :** 10

1- Les Piles

I

Définition : 1.1- Définition

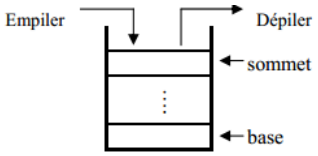
Une pile est une liste chaînée d'informations dans laquelle :

- *Un élément ne peut être ajouté qu'au sommet de la pile,*
- *Un élément ne peut être retiré que du sommet de la pile.*

Il s'agit donc d'une structure de type LIFO (Last In First Out) Traduction : « Le dernier élément qui a été ajouté est le premier à sortir ».

On ne travaille que sur le sommet de la pile. Les éléments de la pile sont reliés entre eux à la manière d'une liste chaînée. Ils possèdent un pointeur vers l'élément suivant.

Le dernier élément (tout en bas de la pile) doit pointer vers nil.



Quelque champ d'applications des piles :

- *Dans un navigateur web, une pile sert à mémoriser les pages Web visitées. L'adresse de chaque nouvelle page visitée est empilée et l'utilisateur dépile l'adresse de la page précédente en cliquant le bouton « Afficher la page précédente ».*
- *L'évaluation des expressions mathématiques en notation post fixée (ou polonaise inverse) utilise une pile.*
- *La fonction « Annuler la frappe » (en anglais « Undo ») d'un traitement de texte mémorise les modifications apportées au texte dans une pile.*
- *Un algorithme de recherche en profondeur utilise une pile pour mémoriser les nœuds visités.*
- *Les algorithmes récursifs admis par certains langages (LISP, Algol, Pascal, C, etc.) utilisent implicitement un pile d'appel. Dans un langage non récursif (FORTRAN par exemple), on peut donc toujours simuler la récursion en créant les primitives de gestion d'une pile.*

1.2- Opérations autorisées

Les opérations par défaut autorisées avec une pile sont :

- Empiler : ajoute un élément sur la pile.
- Depiler : enlève un élément de la pile et le renvoie.

La déclaration est identique à celle d'une liste chaînée,

par exemple pour une pile d'entier :

Type categorie = Structure

age : entier

Suivant : Pile

FinStructure

Type Pile = ^categorie

1.2.1- Empiler

Empiler un élément revient à faire une insertion en tête dans la liste.

Procedure Empiler ({E/S} T : Pile, {E} Valeur : entier) *//********** Ajout d'un élément dans une pile passée en paramètre*

Var

P : Pile *//********** pointeur local*

Début

Allouer(P) *//********** Réserve un espace mémoire pour le nouvel élément*

P^.age ← Valeur *//********** stocke dans l'Info de l'élément pointé par P la valeur passée en paramètre*

P^.Suivant ← T *//**********stocke dans l'adresse du Suivant l'adresse de Tête*

T ← P *//********** Tête pointe maintenant sur le nouvel élément*

Fin

1.2.2- Dépiler

Dépiler revient à faire une suppression en tête.

Procedure Dépiler ({E/S} T : Pile) *//********** Suppression de l'élément au sommet de la pile passée en paramètre*

var

P : Pile *//********** Pointeur nécessaire pour libérer la place occupée par l'élément dépilé*

Début

*//********** Vérifier si la pile est vide*

SI T <> nil alors *//********** la pile n'est pas vide donc on peut dépiler*

P ← T *//********** on garde l'adresse du sommet pour désallouer*

T ← T^.Suivant *//********** P va pointer sur le 2ème élément de la pile qui devient le sommet*

Désallouer(P)

Finsi

Fin

Application 1 :

II

Exercice

Une pile est :

- une structure de pointeur.
- une structure linéaire qui conserve que les adresses.
- une structure de liste chaînée.

Exercice

Énoncé :

Réalise une pile etudiant composée des champs matricule,nom et prenoms en faisant appelle à une procédure empiler. Avant tout créer d'abord la pile etudiant avec les champs ci-dessus.

Solution :

etudiant =

:

:

:

:

FinStructure

Type Pile = ^etudiant

Procedure Empiler (Te : , mat,nom,pre : chaîne)

Var

Pil :

Début

Allouer(Pil)

← mat

← nom

← pre

← Te

Te ← Pe

Fin

2- Files

III

2.1- Définition

Une file est une liste chaînée d'informations qui est basée sur une structure de données basée sur le principe *« Premier entré, premier sorti »*, en anglais *FIFO (First In, First Out)*, ce qui veut dire que les premiers éléments ajoutés à la file seront les premiers à être récupérés.

Le fonctionnement ressemble à une file d'attente : les premières personnes à arriver sont les premières personnes à sortir de la file.

Une file est comparable à une queue de clients à la caisse d'un magasin.

Les files servent à traiter les données dans l'ordre où on les a reçues et permettent de :

- *gérer des processus en attente d'une ressource système (par exemple la liste des travaux à éditer sur une imprimante)*
- *construire des systèmes de réservation*
- *certains moteurs multitâches, dans un système d'exploitation, qui doivent accorder du temps-machine à chaque tâche, sans en privilégier aucune.*
- *un algorithme de parcours en largeur utilise une file pour mémoriser les nœuds visités.*
- *on utilise aussi des files pour créer toutes sortes de mémoires tampons (en anglais buffers).*
- *etc.*

Pour ne pas avoir à parcourir toute la liste au moment d'ajouter un élément en queue, on maintient un pointeur de queue. Attention une file peut très bien être simplement chaînée même s'il y a un pointeur de queue.

2.2- Opérations autorisées

Les opérations par défaut autorisées avec une file sont :

- Enfiler toujours à la queue et jusqu'à la limite de la mémoire,
- Défiler toujours à la tête si la file n'est pas vide,

Le pointeur de tête pointe sur le premier élément de la file, et le pointeur de queue sur le dernier. Il faut commencer par définir un type de variable pour chaque élément de la file. La déclaration est identique à celle d'une liste chaînée.

Par exemple pour une file de chaînes de caractères :

Type categorie = Structure

age : entier

Suivant : File

FinStructure

Type File = ^categorie

2.2.1- Enfiler

Enfiler un élément consiste à l'ajouter en queue de liste. Il faut envisager le cas particulier où la file était vide. En effet, dans ce cas, le pointeur de tête doit être modifié.

Procedure Enfiler ({E/S} T,Q : File, {E} Valeur : entier) *//********** Ajout d'un élément dans une file(T pour la tête de la file et Q pour la queue de la file)*

var

P : File *//********** Pointeur nécessaire pour allouer la place au nouvel élément*

Début

Allouer(P) *//********** Réserve un espace mémoire pour le nouvel élément*

P^.age ← Valeur *//********** stocke la valeur dans l'Info de l'élément pointé par P*

P^.Suivant ← nil *//**********stocke nil (ce sera le dernier de la file) dans Suivant*

Si T = nil alors *//********** file vide*

T ← P *//********** T pointe maintenant sur l'élément unique*

Sinon *//********** il y a au moins un élément dans la file*

Q^.Suivant ← P *//********** le nouvel élément est ajouté au dernier*

Finsi

Q ← P *//********** Q pointe maintenant sur l'élément ajouté*

Fin

2.2.2- Défiler

Défiler est équivalent à dépiler et consiste à supprimer l'élément de tête si la file n'est pas vide. Si la file a un seul élément, il faut mettre à jour le pointeur de queue car on vide la file. Il faut conserver l'adresse de l'élément qu'on supprime pour libérer sa place.

Procedure Defiler ({E/S} T,Q : File, {S} Valeur : entier) *//********** Suppression de l'élément de tête de la file passée en paramètre*

var

P : File *//********** Pointeur nécessaire pour libérer la place de l'élément supprimé*

Début

Si T<> nil alors *//********** la liste n'est pas vide donc on peut défiler*

Valeur ← T^.age *//********** on récupère l'élément de tête*

P ←T *//********** on garde l'adresse du sommet pour désallouer*

T ← T^.Suivant *//********** P va pointer sur le 2ème élément de la pile qui devient le sommet*

Desallouer(P)

Si T = nil alors

Q ← nil //********** *la file a été vidée*

Finsi

Finsi

Fin

Application 2 :

IV

Exercice

NB : Toutes les réponses doivent être en minuscule et sans espace.

Énoncé :

Réalise une FILE etudiant composée des champs matricule,nom et prenoms en faisant appelle à une procédure enfiler. Avant tout créer d'abord la pile etudiant avec les champs ci-dessus.

Solution

etudiant =

:

:

:

:

FinStructure

Type File =

Procedure Enfiler (Te,Qe : , mat,nom,pre : chaîne)

var

:

Début

Allouer(F)

← mat

← nom

← pre

← nil

Si Te = nil alors

←

Sinon

← F

Finsi

← F

Fin