Programmation et Algorithmique II

Devoir 2 : Pile & File

### Exercice 1:

Implémenter en C le type abstrait Pile en utilisant le principe de liste chaînée.

1. Utiliser le fichier .h en-dessous pour implémenter les fonctions primitives de la pile.
2. Tester cette implémentation dans un programme.

Supposant que le fichier .h est le suivant :

#ifndef PILES

#define PILES

#include <stdio.h>

#include <stdlib.h>

typedef float Element;

struct SCellule {

Element info;

struct SCellule \*psuiv;

};

typedef struct SCellule \*Cellule;

struct SPile{

struct SCellule \*sommet;

int nbElements;

};

typedef struct SPile \*Pile;

Pile pileVide(); // creer une pile vide (initialiser)

Pile pileAjouter(Pile p, Element e); //ajouter un élément au sommet de la pile

Pile pileSupprimer(Pile p); //supprimer l’élément au sommet de la pile

Element pileSommet(Pile p); //renvoyer l’élément au sommet de la pile

#endif

### Exercice 2:

Un fichier texte peut contenir des parenthèses ( ), des crochets [ ], et des accolades { }. Ces éléments peuvent être imbriqués les uns dans les autres (exemple : {a(bc[d])[{e f }(g)]} ) Écrire une fonction qui parcourt le fichier texte et détermine si le fichier est correctement parenthésé, c’est-à-dire si toutes les parenthèses, crochets, etc. sont bien refermés par un caractère du même type, et si les parenthèses, crochets

et accolades sont correctement imbriqués. Exemple de fichier incorrect : ({)}. Exemple correct : {(())}

### Exercice 3:

Implémenter une fonction, qui peut convertir une expression infixe à postfixe. L’expression est entrée par une chaîne de caractère.

Implémenter une fonction, qui évalue une expression postfixe, entré par une chaine de caractère.

**Dans les exercices 4-6, il faut utiliser représenter les files par liste chainées**

### Exercice 4:

Implémenter en C le type abstrait File en utilisant le principe de liste chaînée.

### Exercice 5:

On se donne une file F1 contenant des entiers positifs.

1. Ecrire un algorithme pour déplacer les entiers de F1 dans une file F2 de façon à avoir dans F2 tous les nombres pairs avant les nombres impairs.
2. Ecrire un algorithme pour copier dans F2 les nombres pairs contenus dans F1. Le contenu de F1 après exécution de l’algorithme doit être identique à celui avant exécution. Les nombres pairs dans F2 doivent être dans l’ordre où ils apparaissent dans F1.

### Exercice 6 : File d'attente double (Dequeue)

EN cours, nous avons vu l’implémentation file d’attente double (dequeue) représentée par un tableau. Dans cet exercice, on souhaite implémenter une file d’attente double (dequeue) en utilisant les listes chainées.

1. Proposer une structure pour représenter une file d’attente double une liste doublement chainée
2. Ecrire le code en C qui permet de réaliser les opérations suivantes :
3. Initialiser : cette fonction crée une file vide.
4. EnfilerTete : cette fonction permet d’ajouter un élément à la tête de la file.
5. EnfilerQueue : cette fonction permet d’ajouter un élément à la queue de la file.
6. DefilerTete : cette fonction supprime le début de la file. L’élément supprimé est retourné par la fonction Defiler pour pouvoir être utilisé.
7. DefilerQueue : cette fonction supprime l’élément se trouvant à la fin de la file. L’élément supprimé est retourné par la fonction Defiler pour pouvoir être utilisé.

### Exercice 7 : File de priorité

Implanter en C la file de priorité avec la liste chaînée.

### Exercice 8 : implanter l’application suivante

Dans une gare, un guichet est ouvert. Les clients arrivent à des dates **aléatoires** et rentrent dans une queue. L’intervalle entre l’arrivée de deux clients successifs est un nombre **aléatoire** entre 0 et INTERVALLE\_MAX (les dates sont des entiers indiquant des secondes). Lorsque le guichetier a fini de traiter un client, il appelle le client suivant dont le traitement va avoir une durée **aléatoire** entre 0 et DUREE\_TRAITEMENT\_MAX.

### Tâches à réaliser

En utilisant les structure de donnée suivantes :

#define INTERVALLE\_MAX 180

#define DUREE\_TRAITEMENT\_MAX 600

**typedef** struct **{**

int numero**;**

int datearrivee**;**

int dureetraitement**;**

**}** TypeClient**;**

**typedef** struct Cell **{**

TypeClient donnee**;**

struct Cell **\*** suivant**;**

**}** TypeCellule**;**

**typedef** struct **{**

TypeCellule **\*** tete**,** **\*** queue**;**

**}** File**;**

1. Définir la fonction ***Enfiler*** qui permet de rajouter un client (TypeClient) dans une file (File).
2. Écrire une fonction ***CreerListeClients***, qui crée une file de clients, le nombre de clients étant saisi au clavier. Cette fonction initialise aussi la date d’arrivée et la durée d’attente de chacun des clients. On supposera que le premier client est arrivé à 8h.
3. Écrire une fonction d’affichage qui affiche le numéro de chacun des clients, sa date d’arrivée et sa date de fin de traitement en format (h min sec).