Univérsité Hassan II Casablanca

Ecole normale supérieure (ENS) Casablanca

Module : Structures de données en C (Info 4)

Année universitaire 2020/2021

Parcours 2éme année LEESM **Prof. Mohamed Lahby**

TD 6

Structures de données :Pile et File

Exercice 1 : *"Apples des fonctions"*

On donne un nombre réel X et un entier n positif. On désire calculer la somme suivante :

$$S = \sum_{i=1}^{n} \frac{X^i}{i!}$$

Q1. Ecrire une fonction Pile **Empiler_Fat(int n, Pile P1)** permettant d'empiler tous les apples de la fonction factorielle jusqu'à n dans une pile P1.

Q2. Ecrire une fonction **Pile Empiler_Puiss(int n,float x,Pile P2)** permettant d'empiler tous les appels de la fonction puiss jusqu'à n dans une pile P2.

Q3. Ecrire une fonction **Pile Empiler_Frac(Pile 1 , Pile 2)** qui permet de construire la pile 3 à partir de de la pile 1 et la pile 2 en empilant la fraction $X^i/i!$

Q4. Ecrire une fonction **float Somme(int n , float x)** qui calcule la somme S.

Exercice 2 : *"Parenthésage"*

Ecrire une fonction d'entête **int PARENTHESAGE(char *S)** qui utilise les différentes fonctions primitives pour tester le bon parenthésage d'une chaîne reçue en paramètre. Utiliser pour cela le principe suivant : Lors de la lecture de S, Empiler chaque parenthèse, crochet ou accolade ouvrant rencontré et dépiler à chaque parenthèse, crochet ou accolade fermant rencontré si la pile n'est pas vide et s'il y'a correspondance avec le sommet de la pile.

Exercice 3

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**.

Q1. Définir les structures de données pour l'algorithme de simulation.

Q2. 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.

Q3. 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).

Rappel : la fonction rand() permet de générer un nombre aléatoire

Exercice 4 : *"Piles et Files"*

Écrivez les méthodes suivantes. On pourra éventuellement utiliser une ou des piles/files temporaires, on utilisera les deux primitives **Pile()** qui renvoie une pile vide et **File()** qui renvoie une file vide.

Q1. **Appartient(elt)** : cette une primitive de la gestion de la Pile renvoie vrai si l'élément appartient à la pile, faux sinon. Attention : il est important que la pile ne change pas.

Q2. **InverserFile()** : cette une primitive de la gestion de la File inverse les éléments de la file à laquelle elle est appliquée. On a le droit d'utiliser des files ou des piles temporaires.

Q2. **InverserPile()** : cette une primitive de la gestion de la Pile inverse les éléments de la pile à laquelle elle est appliquée. On interdit l'utilisation de files. Seules des piles temporaires peuvent être utilisées.

Exercice 5 : *"Implémentation d'une pile/file en utilisant un tableau"*

Pour implémenter une pile sous forme de tableau, on crée la structure de données suivante. L'implémentation est donnée pour des données de type float. Le principe est le même pour n'importe quel type de données.

```
typedef struct TPile_ {
float Elts[MAX];/* MAX est la capacité de la pile */
int NbElts;/* nombre d'éléments dans la pile */
} Tpile;
```

Q1. Ecrire toutes les primitives de la gestion de la pile

Q2. Proposer une structure tableau pour implanter une file

Q3. Ecrire toutes les primitives de la gestion de la file

Problème 1 : *" Évaluation des expressions arithmétiques"*

Le but de ce travail est d'implémenter les opérations de base sur les piles et d'évaluer des expressions arithmétiques au format préfixé.
Dans une expression arithmétique au format préfixé, on trouve d'abord l'opérateur +,-, *, /, puis les opérandes.
Exemples :

- ✓ l'expression 10 + 2 sera écrite : + 10 2.
- ✓ l'expression (10+2)/3 sera écrite : / + 10 2 3.

Les opérandes peuvent être eux mêmes des expressions.
L'objectif de ce travail est d'implémenter une fonction permettant d'évaluer la valeur d'une expression préfixée entrée sous forme de chaîne de caractère, c'est à dire de calculer sa valeur. Pour cela, nous utiliserons une pile de float, appelée pile de résultats.
Nous parcourons la chaîne contenant l'expression préfixée de droite à gauche. Lorsque nous rencontrons un nombre, nous l'empilons. Lorsque nous rencontrons un opérateur, nous dépilons deux nombres, nous effectuons l'opération, et nous empilons le résultat. A la fin, la pile ne contient plus qu'un nombre : c'est le résultat de l'évaluation à retourner.

Q1. Définir la structure de données qui permet d'implémenter une pile de réels sous forme d'une liste chaînée.

Q2. Ecrire les primitives nécessaires de gestion d'une pile sous forme d'une liste chaînée : PileVide, EstVide, Empiler, Depiler, et SommetPile .

Q3. Écrire une fonction d'entête : **int EstChiffre(char c)** qui renvoie 1 si le caractère passé en paramètre est un chiffre et 0 sinon. (Rappeler bien que les chiffre '0','1',...'9' sont aussi des caractères)
Par exemple : EstChiffre('+')=0 et EstChiffre('4')=1

Q4. Écrire une fonction d'entête : **int Convertir(char c)** qui permet de convertir un caractère qui représente un chiffre en entier.
Par exemple : Convertir('2')=2 ; et Convertir('0')=0

Q5. Écrire une fonction d'entête : **int ChaineVersEntier(char *s)** qui permet de convertir une chaîne de caractères chiffres vers en entier.
Par exemple : ChaineVersEntier("102")=102 ; et ChaineVersEntier("231")=231

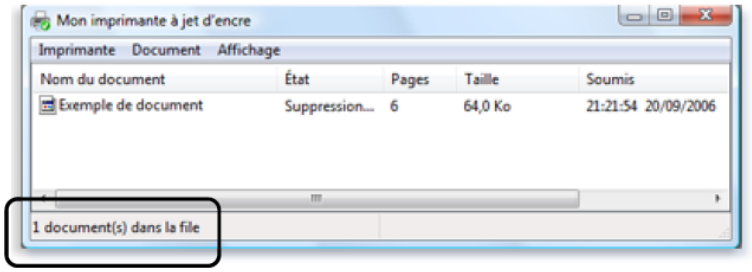
Q6. Écrire une fonction d'entête : **char * split(char *s)** qui permet d'extraire toutes les sous chaînes de caractères existantes on utilise l'epace comme séparateur. Cette fonction retourne un tableau de chaînes de caractères
Par exemple : split("/ + 10 2 3") retourne le tableau {"/", "+", "10","2","3"}

Q7. Écrire une fonction d'entête : **float Evaluer(char * expression)** qui permet d'évaluer la valeur d'une expression préfixée entrée sous forme chaîne de caractères, c'est à dire de calculer sa valeur. On suppose que tous les termes de l'expression sont séparés par des espaces. On lira l'expression de droite à gauche et on reconstituera les nombres à plusieurs chiffres.
Par exemple : Evaluer("+ 10 2")=12.000000 ; Evaluer("/ + 10 2 3")=4.00000

Q8. Écrire une fonction d'entête : **float EvaluerTexte(char * src, char *dest)** qui permet d'évaluer le contenu du fichier source src ensuite stocker le resulat de chaque calcul dans le fichier destination dest)

Problème 2 : *" Gestion de la file d'attente d'impression "*

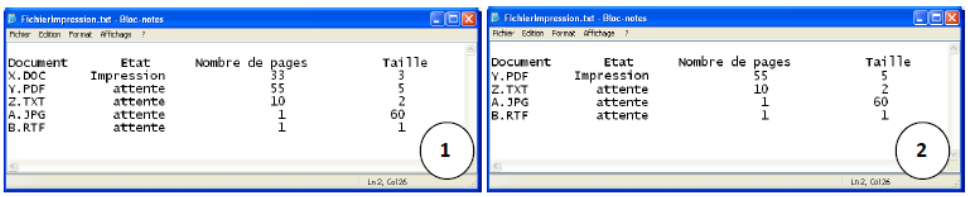
L'impression des documents nécessite une gestion de la part du logiciel d'impression pour passer cette opération dans l'ordre souhaité. Ce logiciel d'impression utilise le principe d'une file d'attente ; ainsi le premier document lancé en impression sera le premier imprimé et ainsi de suite.
La fenêtre suivante montre qu'un document dans la file attend son impression ou bien il sera supprimé au cas d'échec. Une fois qu'on imprime un document, on le supprime de la file et c'est le document suivant de la file qui devient en cours d'impression.



Dans ce travail, on va programmer une méthode pour l'impression de plusieurs documents. Chaque document lancé en impression possède les informations suivantes :

- ✓ Le nom du document
- ✓ L'état du document (impression ou attente)
- ✓ Le nombre de pages du document
- ✓ La taille du document (en Ko)

La file d'attente d'impression sera enregistrée dans un fichier texte qu'on va nommer "FichierImpression.txt". Le premier document de la file sera en cours d'impression et le reste des documents sera en attente. Une fois qu'on imprime un document, on le supprime de la file et c'est le document suivant de la file qui devient en cours d'impression.



Q1. Définir la structure DocImprime (struct DocImprime) de telle façon qu'elle puisse contenir les informations concernant un document lancé en impression.

Q2. Créer une file d'attente (struct FileImp) qui permettra de mémoriser tous les documents qui sont en attente d'impression.

Q3. Ecrire une fonction d'en-tête struct FileImp* LancerImpr(struct FileImp *L) qui permet d'ajouter un document dans la file d'attente d'impression.

Q4. Ecrire une fonction d'en-tête struct FileImp* ImprTerminee(struct FileImp *L) qui permet de supprimer de la file d'attente un document qui vient d'être imprimé.

Q5. Ecrire une fonction d'en-tête void EcritureFichier(struct FileImp *F) qui permet d'écrire les données concernant les documents en attente d'impression dans un fichier « FichierImpression.txt ».

Q6. Ecrire une fonction d'en-tête void LectureFichier(struct FileImp *F) qui permet de lire les données du fichier « FichierImpression.txt » et de les afficher sur sortie standard.