

T.P.2

Files et Piles

1 Indications/Consignes

- Les trois fichiers seront incrémentés en parallèle et au fur et à mesure. Avant de passer à une fonction numéro i , assurez vous que la fonction numéro $i - 1$ a été implémentée et testée (par un appel dans la fonction main).
- Réfléchissez bien aux prototypes des fonctions avant de les définir.

2 Les Files

Une file est une structure de données basée sur le principe "Premier arrivé, premier servi" en anglais FIFO (First In, First Out), ce qui veut dire que les premiers éléments ajoutés à une file seront les premiers à être récupérés. Les files sont utilisées le plus souvent afin de mémoriser des données en attente de traitement.

Nous nous proposons de concevoir plusieurs programmes permettant la gestion d'une file d'éléments. Il faudra créer un projet C par exercice, composé des trois fichiers suivants:

1. **file.h** : contiendra la définition de la structure cell (cellule) ainsi que les prototypes des fonctions à développer.
2. **file.c** : contiendra les définitions des fonctions à développer.
3. **main.c** : contiendra le programme principal. Il s'agit d'afficher un menu permettant à l'utilisateur de choisir une opération sur la file et d'appeler la fonction correspondante. Tant que l'utilisateur n'a pas choisi d'arrêter le programme, le menu est réaffiché invitant l'utilisateur à choisir une nouvelle opération.

Chaque implémentation proposée utilisant des files, doit comporter des fonctions permettant :

1. d'initialiser une file
2. de tester si la file est vide
3. d'afficher la file
4. d'insérer un nouvel élément dans la file (enfiler)
5. de supprimer un élément dans la file (défiler)

Exercice 1 *File d'entiers implémentée dynamiquement avec un chaînage des éléments*
Les structures de données et les prototypes suivants sont à utiliser.

```
/* Fichier file.h */
#ifndef __FILE__H
#define __FILE__H

// definition d'un element d'une file */
typedef struct cell {
```

```
    int val;
    struct cell * next;
} cell;
/* definition d'une file */
typedef struct fifo {
    cell * head;
    cell * queue;
    int length;
} fifo;

//A Completer.....
/* affichage des elements de la file */
void print_fifo(fifo * file);
#endif
```

1. Ecrire les 5 fonctions indispensables pour la manipulation de pile.

2. Le programme doit pouvoir s'exécuter avec le code suivant.

```
#include <stdio.h>
#include "file.h"
static void test1();

int main()
{
    test1();
    return 0;
}

static void test1()
{
    fifo file;
    int x;
    int i;

    init(&file);
    print_fifo(&file);
    for (i = 0; i < 5; i++)
    {
        push(i, &file);
    }
    printf("\n");
    print_fifo(&file);
    for (i = 0; i < 5; i++)
    {
        pop(&file);
    }
    printf("\n");
    pop(&file);
    print_fifo(&file);
}
```

3 Les Piles

Une pile est une structure de données basée sur le principe "Dernier arrivé, premier servi" en anglais LIFO (Last In First Out), ce qui signifie que les derniers éléments à être ajoutés à la pile seront les premiers à être récupérés. Exemple : une pile d'assiettes, lorsqu'on ajoute une assiette en haut de la pile, on retire en premier celle qui se trouve en haut de la pile, i.e. celle qui a été ajoutée en dernier, sinon tout le reste s'effondre. Les piles sont utilisées dans des algorithmes d'évaluation d'expressions mathématiques.

Nous nous proposons de concevoir un programme permettant la gestion d'une pile d'éléments. Il faudra créer un projet C par exercice, composé des trois fichiers suivants:

1. **pile.h** : contiendra la définition de la structure cellule ainsi que les prototypes des fonctions à développer.
2. **pile.c** : contiendra les définitions des fonctions à développer.
3. **main.c** : contiendra le programme principal. Il s'agit d'afficher un menu permettant à l'utilisateur de choisir une opération sur la pile et d'appeler la fonction correspondante. Tant que l'utilisateur n'a pas choisi d'arrêter le programme, le menu est réaffiché invitant l'utilisateur à choisir une nouvelle opération.

Chaque implémentation proposée utilisant des piles, doit comporter des fonctions permettant :

1. d'initialiser une pile
2. de tester si la pile est vide
3. d'afficher la pile
4. d'insérer un nouvel élément dans la pile (push)
5. de supprimer un élément dans la pile (pop)

Exercice 2 Inversement du contenu d'un tableau

On souhaite inverser un tableau d'entiers en utilisant une pile. Après inversion, le tableau d'entier {1, 2, 3} est transformé en {3, 2, 1}.

1. *Ecrire les 5 fonctions indispensables pour la manipulation de pile.*
2. *Ecrire la fonction "reverse" dont le rôle est d'inverser le contenu d'un tableau.*
3. *Testez l'inversion sur le tableau d'entier [10, 20, 30, 40, 50].*