



TP2 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.

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 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.

Indications:

- 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.

Les 2 implémentations proposées devront 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

a) Les structures de données et les prototypes suivants sont à utiliser.

```
/* Fichier file.h */

#ifndef __FILE__H
#define __FILE__H

// définition d'un élément d'une file */
typedef struct Cellule {
    int valeur;
    struct Cellule * suivant;
} Cellule;

/* définition d'une file */
typedef struct Fifo {
    Cellule * entree;
    Cellule * sortie;
} Fifo;

/* initialise une file existante et allouée statiquement en mettant ses pointeurs à NULL */
void initialiser(Fifo * fifo);

/* alloue une file dynamiquement et l'initialise en mettant ses pointeurs à NULL */
void initialiserDynamique(Fifo ** fifo);

/* retourne vrai si la file est vide, faux sinon */
int estVide(const Fifo * fifo);

/* Crée une nouvelle cellule, y affecte la valeur, et l'insère après la dernière cellule. Si file était vide, elle
contient maintenant l'unique cellule qui vient d'être créée.*/
void enfiler(int valeur, Fifo * fifo);

/* Supprime la première cellule de la file et retourne la valeur qu'elle contient. Si la file ne contenait
qu'une seule cellule, la file devient vide. Erreur si la file est vide. */
void defiler(int * valeur, Fifo * fifo);

/* affichage des éléments de la file */
void afficher(const Fifo * fifo);

#endif
```

b) L'implémentation doit comporter dans le fichier File.c, les définitions de fonctions déclarées dans le fichier File.h.

Il faut également définir une fonction locale au fichier File.c, qui crée dynamiquement une cellule avec la valeur passée en paramètre et de prototype:

static Cellule * creerCellule (int valeur).

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

```
#include <stdio.h>
#include "File.h"

static void test1();
static void test2();

int main()
{
    test1();
    test2();
    return 0;
}

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

    printf("tests avec une file allouée statiquement \n");
    initialiser(&fifo);
    afficher(&fifo);

    for (i = 0; i < 5; i++)
    {
        enfiler(i, &fifo);
    }
    printf("\n");
    afficher(&fifo);

    for (i = 0; i < 5; i++)
    {
        defiler(&x, &fifo);
    }
    printf("\n");

    defiler(&x, &fifo);

    afficher(&fifo);
}

static void test2()
{
    Fifo * fifo;
    int x;
```

L2 TP SDD 2012-2013

```
int i;

printf("tests avec une file allouée dynamiquement \n");
initialiserDynamique(&fifo);
afficher(fifo);

for (i = 0; i < 5; i++)
{
    enfiler(i, fifo);
}
printf("\n");
afficher(fifo);

for (i = 0; i < 5; i++)
{
    defiler(&x, fifo);
}
printf("\n");

defiler(&x, fifo);

afficher(fifo);
}
```

Exercice 2: File d'entiers implémentée statiquement avec un tableau circulaire

Indications:

/* taille maximum de la file */

#define NMAX 10

/* définition d'une file */

```
typedef struct Fifo{
    int tab[NMAX] ;
    int entree;
    int sortie;
    int taille;
} Fifo;
```

Elaborer un test qui démontre que la capacité maximum de la file est NMAX - 1 (mettre en évidence un dysfonctionnement avec une file qui contiendrait NMAX éléments) .