

Structures de données et algorithmes

TP : Implantation contiguë et chaînée des files

Avant de commencer

On rappelle la signature de la spécification algébrique des files vue en cours.

Spéc FILE(TRIV) **etend** BASE

Sorte File

Opérations

```
filenouv :      -> File
adjq   : File S -> File
supt   : File   -> File
tete   : File   -> S
vide   : File   -> Bool
longueur : File -> Nat
```

1 Gestion contiguë

On considère des files d'entiers naturels et l'implantation contiguë avec la structure de données:

```
#define MAX_P 50
typedef struct {
    Nat tab[MAX_P];
    Nat h;      /* tête de la File */
    Nat l;      /* longueur de la File */
} File;
```

Écrivez proprement le fichier header correspondant, de nom `file_int.h` avec les structures de données et les profils des fonctions, puis le fichier contenant le source des fonctions, de nom `file_int.c`.

Écrivez dans un fichier de nom `test.c` un petit programme permettant de tester toutes les opérations sur les files d'entiers naturels. Écrivez également le fichier `Makefile` permettant de faire la compilation séparée de ces fichiers.

2 Gestion chaînée

Même exercice qu'à la section précédente mais en utilisant une implantation chaînée comme celle vue en cours. On considérera donc la structure de données :

```
typedef struct strfile {
    Nat v;
    struct strfile *s;
} Strfile;

typedef struct sfile {
    Strfile *h;
    Strfile *q; } File;
```

Les noms des fichiers seront respectivement `file_int2.h`, `file_int2.c`. Vous vous arrangerez pour que le source des programmes de test, écrits dans `test.c` reste inchangé.

3 Gestion chaînée circulaire

Même exercice qu'à la section précédente mais en utilisant une implantation chaînée circulaire et un unique pointeur sur la queue de file.

Programmez dans un fichier de nom `flavius.c` un programme permettant de résoudre le problème de Flavius Josèphe.