TP 2 Algorithmique Avancée : Piles, Files

Serigne A. Gueye

CERI - Licence 2 Informatique

Exercice 1 [Pile].

Nous voulons coder une structure de pile d'au maximum "max" éléments. Nous supposerons que "max" est par défaut égale à une grande constante M.

$$const\ int\ M = 1e + 10;$$

On considère la classe suivante :

```
class   Pile
{
        int  tete;
        int * tab;
        int max;
        public :
        Pile();
        ~ Pile();
}
```

1/ Coder le constructeur $Pile()$ permettant d'affecter une valeur par défaut à "max", d'affecter la première valeur de "$tete$", et d'allouer le tableau "tab".

2/ Coder un nouveau constructeur $Pile(int\ n)$ permettant de créer une pile d'au maximum n éléments.

3/ Ajouter les méthodes suivantes :

- $\sim Pile()$: le destructeur,
- $Vide()$: test de vacuité,
- $Pleine()$: retourne vraie si la pile est pleine, faux sinon,
- $Empiler(x)$: insertion de l'élément de valeur x,
- $Depiler()$: retour et suppression de l'élément en tête de la pile,
- $Afficher()$: affiche le contenu de la pile.

Pour tester, vous lirez une suite quelconque de nombres à l'écran et créerez en mémoire la pile correspondante.

Exercice 2 [File].

1/ En vous inspirant de la classe *Pile* créez une classe File. Elle comportera les mêmes attributs que ci-dessus auquel s'ajoutera un attribut "*queue*" indiquant l'emplacement où peut s'insèrer un nouvel élément.

2/ Ajouter les méthodes suivantes :

- $File()$: constructeur sans argument permettant d'affecter une valeur par défaut à "*max*", d'affecter "*tete*" et "*queue*", d'allouer le tableau "*Tab*",
- $File(int\ n)$: constructeur créant une file d'au maximum n éléments,
- $\sim File()$: destructeur,
- $Vide()$: test de vacuité,
- $Pleine()$: retourne vraie si la file est pleine, faux sinon,
- $Enfiler(x)$: insertion de l'élément de valeur x,
- $Defiler()$: retour et suppression de l'élément en tête de la file,
- $Afficher()$: affiche le contenu de la file.