AlgorithmiqueProgrammation 2A/1T Iut de Caen Département d'Informatique

Feuille 1 d'exercices Remise en marche

Exercice 1 Ecrire une méthode qui implémente la fonction C(n,p) suivante :

$$C(n,p) = \frac{n!}{p!.(n-p)!}$$

où n! est la factorielle de n.

Ecrire une méthode qui imprime à l'écran le triangle de Pascal défini à partir de la formule ci-dessus :

```
1
1 1
1 2  1
1 3  3  1
1 4  6  4  1
1 5 10 10  5 1
```

Exercice 2 - Une classe **TabTrie** qui gère un tableau trié d'entiers

Il s'agit de définir une classe représentant un tableau trié d'entiers.La classe, appelée **TabTrie**, aura un attribut privé qui sera un tableau d'entiers.Au moment de la construction d'une instance de la classe, le tableau aura une capacité qui aura une valeur par défaut donnée par le programmeur et pourra aussi être indiquée par le constructeur. Le tableau initial ne contiendra aucun élément. Les méthodes suivantes seront prévues :

void inserer(int entier) : insére un entier dans le tableau en respectant un ordre croissant sur les entiers.
void retirer(int entier) : retire un entier donné, si un tel entier est dans le tableau. Si l'entier figure plusieurs fois, une seule occurrence est retirée.
public String toString() : retourne une chaîne de caractères décrivant le tableau, et redéfinit la méthode correspondante de la classe Object.

Lorsque le tableau devient trop petit, il faut l'agrandir en définissant un tableau plus grand et en recopiant les entiers un à un. La différence entre la capacité de l'ancien tableau et celle du nouveau sera fournie par la donnée entière increment dont la valeur pourra être indiquée par le constructeur. Lorsque le nombre de données aura diminué de telle sorte que la capacité inemployée soit au moins égale au double de l'incrément, on diminuera le tableau.

On prévoira deux constructeurs : un sans paramètre, et un qui prend en paramètres la capacité initiale du tableau et la valeur de l'incrément.
Ajouter une méthode qui fusionne 2 tableaux triés.

Exercice 3 - Ecrire les méthodes suivantes :

3.1 conversion d'un nombre entier en binaire ;
3.2 calcul du pgcd de deux nombres par l'algorithme d'Euclide ;
3.3 calcul du $n^{ième}$ de la suite de Fibonacci (en récursif et en itératif) ;
3.3 calcul de x^n par la méthode "brute" et par la méthode d'exponentiation rapide.

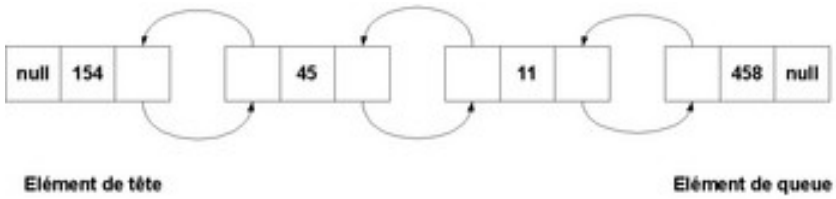
Exercice 4

Partie algorithmique

(temps conseillé : 20 à 30 minutes - barême : 5 points)

A RENDRE SUR UNE FEUILLE SEPAREE

Liste doublement chaînée



On considère la représentation d'une File Réversible par liste doublement chaînée. On utilisera les représentations suivantes :

```
type Noeud {
     val : Element
     prec : Noeud
     suiv : Noeud}
```

```
type FileReversible {
     premier : Noeud
     dernier : Noeud
}
```

La fonction nouvNoeud qui crée un Noeud est donnée :

```
fonction nouvNoeud(el : Element)
     var tmp : Noeud
     début
          tmp.val <- el
          tmp.prec <- null
          tmp.suiv <- null
          retourner tmp
     fin
```

1) Écrire les opérations suivantes:

1.1) `fonction ajouterEnQueue(var f : FileResersible, el : Element)`

1.2) `fonction retirerEnQueue(var f : FileResersible) : Element`

1.3) `fonction estPrésent(f : FileResersible, el : Element) : booléen`

qui teste si l'élément `el` appartient à la file `f`

1.4) `fonction insereKieme(var f : FileResersible, el : Element, k:entier)`
qui insére l'élément `el` à la postition`k`, `k` est un entier positif, inférieur ou égal au nombre d'éléments de la liste. Par exemple, inserekieme(f, 10, 3), où f est la file représentée par la figure ci-dessus, donne la liste [154, 45, 10 , 11, 458].

1.5) que vaut `.f.dernier.prec.suiv.val` avec f représentée par la figure ci-dessus ?

2) Quelle est l'intérêt de la représentation doublement chaînée ?