TD5 : structures de données linéaires

Exercice 1 : Files et piles

Q 1.1 Montrer que l'on peut implanter une pile avec deux files.

Q 1.2 Donner la complexité des opérations empiler et dépiler dans ce cas.

Q 1.3 Montrer que l'on peut implanter une file avec deux piles.

Q 1.4 Donner la complexité des opérations enfiler et défiler dans ce cas.

Exercice 2 : Liste avec tableaux

On se propose dans cet exercice d'implanter une structure de liste grâce à des tableaux. On sait que cette implantation possède l'inconvénient majeur de ne pas être extensible sans recopie d'éléments mais l'avantage de pouvoir accéder au i-ème élément en temps constant. Nous allons proposer une structure de données pour tenter de remédier au premier problème tout en conservant de bonnes propriétés pour l'accès au i-ème élément.

L'idée proposée est la suivante : lorsque le tableaux d'éléments sera rempli, nous allons créer un second tableau d'éléments, puis lorsque le second tableau d'éléments sera rempli nous créerons un troisième tableau, etc. Nous préservons ainsi l'extensibilité de la structure de données. Pour accéder au i-ème élément, il suffira de trouver le bon tableau. En fait, on a une liste de tableaux d'éléments au lieu d'une liste d'éléments.

On suppose disposer d'un type `TABLISTE` déclaré comme `array[1..MAX] of ELEMENT`.

Q 2.1 Supposons que la liste ne contienne qu'un seul tableau, non rempli. Quelle astuce mettre en place pour éviter d'avoir à recopier tous les éléments de la liste lorsqu'on insère en tête ?

Q 2.2 Dessiner la structure de données si `MAX` vaut 5 et la liste est composées des éléments 1 à 12 insérés en tête dans l'ordre croissant.

Q 2.3 Quels vont être les informations nécessaires à la gestion du type `LISTE` que nous cherchons à mettre en place ?

Q 2.4 Donner la déclaration en PASCAL du type `LISTE`.

Q 2.5 Ecrire le code permettant l'ajout d'un élément en tête de la liste.

```
procedure insererEnTete(var l : LISTE; const e : ELEMENT);
```

Q 2.6 Ecrire le code permettant d'obtenir la valeur de l'élément en i-ème position de la liste (on suppose la liste indicée à partir de 1).

```
function ieme(l : LISTE; i : CARDINAL) : ELEMENT);
```

Q 2.7 Discuter des complexités de recherche d'un élément et d'accès au i-ème élément.