UNIVERSITÉ PARIS DESCARTES

DUT Informatique - 1ère année
SDA - Structures de Données et Algorithmes

TRAVAUX DIRIGES – Semaine n°2

Thèmes

- Structuration et généralisation
- Définition et documentation de fonctions

Un conteneur permet de stocker des éléments a priori d'un type quelconque. Des fonctions d'accès en lecture et en écriture au conteneur doivent alors être codées de manière à pouvoir manipuler les éléments.

L'objectif est ici d'étudier un conteneur généralisé au stockage et traitement de tout type d'items fondé sur un tableau statique d'items (conteneur primaire) caractérisé par : (1) une taille donnée (`capacite`) correspondant au nombre maximum d'items pouvant être stockés dans le conteneur et (2) le nombre d'items (`nbItems`) effectivement stockés dans le conteneur de la position `0` à (`nbItems-1`) avec `nbItems ≤ capacite`. De plus, (3) le mode d'accès en écriture (ajout ou modification) se fera au moyen de la position (`i`) où écrire l'item (`it`) dans le conteneur tel que tout item écrit pourra être lu, l'ajout d'un item se fera à la position courante `nbItems`, (4) le mode d'accès en lecture se fera au moyen d'une position (i) tel que tout item lu doit avoir été écrit.

Exercice 1. Structuration des données

En supposant le type `Item` des items déjà défini, définissez le type (type-utilisateur) `ConteneurTS` permettant de stocker les éléments dans un conteneur, de capacité `15`, tel qu'il est décrit précédemment.

Exercice 2. Spécification et codage des fonctions

Analysez, prototypez, documentez puis codez les fonctions suivantes :

2.1. **`initialiser`**, initialise un conteneur d'items vide (aucun élément n'est stocké).

2.2. à la position `i`. On donnera la précondition correspondant au mode d'accès en écriture spécifié en introduction.

2.3. **`ecrire`**, écrit (ajoute ou modifie) l'item (it) à la position i. On donnera la précondition correspondant au mode d'accès en écriture spécifié en introduction.

2.4. **`afficher`**, affiche les éléments stockés dans le conteneur.

Spécification : afficher le conteneur des éléments it1, it2 et it3 suivant le format : [it1 it2 it3]

Indication pour le codage de **`afficher`** : on utilisera (sans la coder) la fonction **`afficher`** à un argument (`arg`) de type `Item` qui affiche `arg`. Avant de l'utiliser, prototypez la fonction `afficher`.

2.5. Test du conteneur et de ses fonctionnalités
Stockez cinq entiers (de 1 à 5) dans le conteneur et vérifier son affichage. vérifiez par assertion sa fonctionnalité d'écriture (en modification et en ajout d'élément et de lecture.

Préparation du TP n°2

2.6. **`extrema`**, indique les extrema du conteneur d'items par deux positions dans le conteneur : l'une correspondant à la plus petite valeur des éléments du conteneur, l'autre à la plus grande valeur. Plusieurs éléments du conteneur peuvent avoir la même valeur, la position d'un extremum correspondra à l'indice le plus petit dans le conteneur où a été détecté l'extremum. Définissez et donnez la précondition de cette fonction.

Indication pour le codage de **`extrema`** : on utilisera (sans la coder) la fonction **`comparer`** à deux arguments (`arg1` et `arg2`) de type `Item` qui renvoie (1) un nombre négatif si `arg1<arg2`, (2) 0 si `arg1=arg2` et (3) un nombre strictement positif si `arg1>arg2`. Avant de l'utiliser, prototypez la fonction `comparer`.

Remarque : Dans une application utilisant un conteneur de type `ConteneurTS` et ses fonctionnalités pour un type `Item` donné (`float`, `Date` , etc.), il faudra définir la fonction `afficher` qui affiche un argument du type Item donné et la fonction `comparer` qui compare deux arguments du type Item donné.