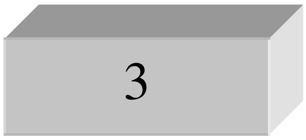
DEUG MIAS / MIAS 302

Devoir Surveillé du 29 avril 2002
Durée 1 heure 20, tout document autorisé

On a N boites (N valant au maximum MAX_BOITES) sans couvercle, du type boites à chaussures, que l'on veut ranger en les mettant les unes dans les autres. Chaque boite porte un numéro quelconque. Par exemple si on a 5 boites (N=5), elles peuvent par exemple porter les numéros 8, 3, 20, 5, 4.



Hypothèses simplificatrices :

1. Toutes les boites ont une taille différente.
2. Si une boite est moins longue qu'une autre boite, alors elle est aussi moins large et moins haute.
3. Toutes les boites ont un numéro différent.

Ces hypothèses permettent en particulier de garantir que le rangement recherché est effectivement réalisable.

L'objectif de ce devoir surveillé est de réaliser un ensemble de fonctions dont l'objectif ultime est d'aider à effectuer ce rangement.

Questions :

1. Définir le type **Boite.**

2. Définir un type **EnsembleDeBoites** permettant de stocker un ensemble d'éléments de type **Boite**.
On rappelle que l'on a au maximum MAX_BOITES.

3. Ecrire les fonctions suivantes :

a. Boite recherchePlusGrandeBoite(EnsembleDeBoites ens);

Cette fonction retourne la boite la plus grande dans l'ensemble de boites *ens*. Le contrat de cette fonction stipule qu'elle ne doit être appelée que si *ens* n'est pas vide.

b. Boite rechercherBoiteNumero(int num, EnsembleDeBoites ens);

Cette fonction retourne la boite qui porte le numéro *num* dans l'ensemble de boites *ens*. Le contrat de cette fonction stipule qu'elle ne doit être appelée que si *ens* contient effectivement une boite portant le numéro *num*.

c. Boite rechercherBoiteSuivante(Boite b, EnsembleDeBoites ens);

Cette fonction retourne la boite contenue dans l'ensemble de boites *ens* qui est de taille immédiatement supérieure à la boite *b*. Le contrat de cette fonction stipule qu'elle ne doit être appelée que si il existe effectivement dans *ens* une boite plus grande que *b*.

4. On souhaite maintenant ranger effectivement les boites. Pour cela, on va écrire un programme qui a partir d'un ensemble de boites va indiquer la suite des opérations à réaliser. Par exemple, si le programme affiche 8, 3, 20, 5, 4 ceci signifiera qu'il faut d'abord prendre la boite portant le numéro 8, y mettre la boite portant le numéro 3, etc.

a. Ecrire la fonction

```
void trouverRangement(EnsembleDeBoites ens);
```

qui affiche la succession des opérations à effectuer. Le contrat de cette fonction stipule qu'elle ne doit être appelée que si il y a au moins une boite dans *ens*.

b. On suppose donnée la fonction

```
EnsembleDeBoites initialiserEnsembleDeBoites();
```

qui en lisant des données sur son entrée standard remplit et retourne un ensemble de boites avec au plus MAX_BOITES.

Ecrire la fonction *main* du programme qui résout le problème de rangement.