Feuille de TP7 (ou devoir) - Cours C. Recanati
AIR1, Programmation Impérative

Nous reprenons le travail fait en TP sur les ensembles, mais on va ici utiliser des listes chaînées d'entiers ordonnés pour représentés des ensembles. Les ensembles seront des ensembles de points dans l'espace métrique R^3. Les points utilisés dans le programme seront rangés dans un tableau de points appelé univers. Un ensemble (ou un sous-ensemble) de points sera alors représenté par une liste ordonnée d'indices dans le tableau univers. Les opérations ensemblistes, tels que l'union, l'intersection, etc. seront donc effectuées sur des listes ordonnées d'entiers.

Le propos de ce TP est d'écrire un ensemble de fonctionnalités sur les ensembles de points, rangés dans des fichiers sources `ensembleDePoints.c` et `ensembleDePoints.h` traitant des opérations ensemblistes. Dans le cas présent, on implémentera aussi deux fichiers `geo.c` et `geo.h` traitant de points et d'opérations sur des points de R^3 (dans l'idée d'avoir ensuite des objets géométriques comme des segments ou des droites, etc. définis par des structures utilisant des points). On écrira ensuite un programme utilisant ces fonctionnalités (voir. Exercice 3).

Exercice 1. Les points de R^3
Ecrire des sources geo.c et geo.h , incluant notamment les entrées sorties (lecture, écriture d'un point).

Exercice 2. Les ensembles de points et opérations ensemblistes.
Ecrire des sources ensembleDePoints.c et ensembleDePoints.h. En plus des fonctionnalités sur les ensembles représentés par des listes chaînées d'indices, on aura aussi les fonctionnalités liant les points à leur indices dans l'univers. Celui-ci sera représenté par une structure associant un tableau alloué dynamiquement et sa taille. La fonction d'ajout d'un point à l'univers nécessitera alors de faire une réallocation dynamique. On écrira en particulier les fonctions d'écriture (affichage) d'un point mais aussi d'un ensemble de points (ce qui nécessite d'avoir accès au tableau univers). On aura besoin d'une fonction permettant d'ajouter un point dans l'univers, ce qui augmentera sa taille. On ajoutera aussi une fonction centre qui calculera le centre d'un ensemble (i.e. le barycentre donnant un poids de 1 à tous les points).

Exercice 3. Programme utilisant les fonctionnalités précédentes.
Ecrire un programme manipulant ces fonctionnalités et qui demande d'abord en entrée à lire un univers et trois de ses sous-ensembles.
a- On calculera ensuite le centre de chaque sous-ensemble, et on l'ajoutera au sous-ensemble concerné. (Il faudra alors ajouter ce nouveau point à l'univers s'il n'y est pas déjà. Pour ce faire, on implémentera en sus des opérations binaires d'union et d'intersection de deux ensembles, l'opération unaire ajouteCentre (A) qui ajoute son centre à un ensemble A.
b- On calculera ensuite à partir des trois sous-ensembles de l'univers entrés en début de programme, l'intersection de A avec l'union de B et de C. Mais on ajoutera d'abord les centres des trois sous-ensembles à eux-mêmes en utilisant ajouteCentre.

c- Demander à l'utilisateur d'entrer une expression ensembliste en notation préfixe concernant les 3 sous-ensembles, et calculer alors l'ensemble résultant, puis l'afficher. En notation préfixe le symbole d'opération (binaire) vient avant ses 2 arguments. Ainsi 3+((12-x)*2) est notée +3*-12x2.
Mais on n'aura pas le temps de traiter ce cas général (qu'on peut faire en devoir) et vous afficherez simplement l'ensemble de la question précédente, et celui résultant de l'expression expression ensembliste de votre choix.

Exemple d'exécution : ici on entre un univers de 6 points, et 3 sous-ensembles de 4 points (donc avec des indices entre 0 et 5 dans l'univers).

$> univers ?
6
1 1 1
3 3 3
-1 -1 -1
-3 -3 -3
-1 0 1
1 0 -1
$> trois ensembles de 4 points (un point a un indice entre 0 et 5) ?
0 1 2 3
2 3 4 5
0 1 4 5
$> les centres ajoutés à l'univers
0 0 0 (indice 6)
-1 -1 -1 (indice 2 : il existe déjà dans l'univers)
1 1 1 (indice 0 : il existe déjà dans l'univers)
$> les nouveaux ensembles
0 1 2 3 6
2 3 4 5
0 1 4 5
$> Expression ?
"aiui01i122"
$> Résultat (indices puis points)
4 5 6 =>
-1 0 1
1 0 -1
0 0 0