Langace C - Travaux Dirigés n° 10

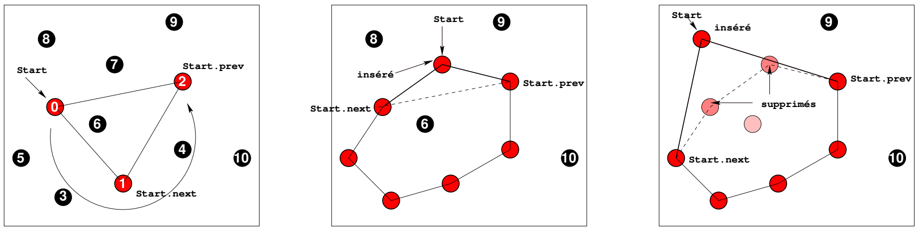
— *Licence Informatique* —

Résumé

Dans ce TD nous commencerons à utiliser certaines notions propres au C++ comme les `class`, les constructeurs, les méthodes...
L'objectif est de vous faire implémenter un algorithme de détermination d'enveloppe convexe dans le plan en utilisant une structure de type *Liste doublement chainée*.

On donne un tableau de `n` points de $\mathbb{R}^2$ dont on veut déterminer l'enveloppe convexe, c'est-à-dire le plus petit polygône convexe contenant tous les points. L'algorithme est le suivant :

- On initialise l'enveloppe par un triangle (A, B, C) formé des 3 premiers points du tableau ordonnés de telle sorte que le triangle soit direct c'est-à-dire que la composante en z du produit vectoriel $\overrightarrow{AB} \wedge \overrightarrow{AC}$ soit positive.
- Il faut ensuite tester tous les autres points et vérifier leur position par rapport à cette enveloppe partielle. Si le point est à l'intérieur, on ne fait rien et on passe au suivant. Sinon, il modifie localement l'enveloppe : soit il s'ajoute directement à la liste des sommets (il est simplement intégré à la chaîne), soit il remplace un ou plusieurs autres sommets consécutifs (il faut alors les supprimer un morceau de la chaîne et les remplacer par le nouveau maillon).



Initialisation — Etape 5 — Etape 7

On utilisera les classes suivantes pour représenter des points et des listes de points :

```
class Point
{
 private :
  double x;
  double y;
 public :
  // méthodes

};
```

```
class Cell
{
  private :
    Point* pos ; // coordonnées
    Cell*  prev; // precedant sur l'enveloppe
    Cell*  next; // suivant sur l'enveloppe
  public :
     //methodes
};
typedef Cell* LIST;
```

Les champs `prev` et `next` permettront de parcourir la liste (doublement chaînée et bouclée) des points sélectionnés comme appartenant à l'enveloppe convexe.

▶ Exercice 1 (Mise en place et sauvegarde des points)

Ecrire une fonction `Point* InitPoints(int nbpts,double range)` permettant de disposer aléatoirement `nbpts` points dans un disque de centre C(`range`,`range`) et de rayon `range` (Pour avoir un maximum de points sur l'enveloppe) et renvoyant le tableau de ces points.

▶ Exercice 2 Classe `Point` : (Triangle direct)

Écrire une méthode `double Position(Pnt* A, Pnt* B)` pour la classe `Point` calculant la composante en z du produit vectoriel $\overrightarrow{PA} \wedge \overrightarrow{PB}$, P étant le point appelant la méthode (`this`). On ajoutera les méthodes de base (constructeurs, destructeurs, accesseurs...) indispensables.

▶ **Exercice 3 Classe `Cell` : (Insertion et suppression d'une cellule)**
Ecrire deux méthodes `void Insert(Cell* C)` et `void Remove(void)` pour la classe `Cell` permettant d'insérer la cellule appelante dans une liste entre les cellules `C` et `C.next`, et de supprimer la cellule appelante de la liste.
On ajoutera les méthodes de base (constructeurs, destructeurs, accesseurs...) indispensables.

▶ **Exercice 4 (Initialisation de l'enveloppe)**
Écrire une fonction `LIST InitConvex(Point *TAB)` permettant de démarrer la chaîne à partir du tableau des points. Cette fonction renverra une liste doublement chaînée et bouclée formée des trois premiers points du tableau, ordonnés de telle sorte que le triangle soit direct (cf. figure cadre 1) - On pourra redéfinir une méthode `Position` pour la classe `Cell`.

▶ **Exercice 5 (Traitement d'un point)**

- Pour déterminer si un point P est à l'intérieur ou à l'extérieur d'un polygône convexe, il faut parcourir les arêtes $[C_i, C_{i+1}]$ (supposées orientées) de ce polygône (`LIST`) et tester si le triangle ainsi formé (P, C_i, C_{i+1}) est direct ou non. Si il l'est, cela signifie que P est à l'extérieur. Il faut alors créer une nouvelle cellule pointant sur P et l'insérer dans la liste enveloppe entre C_i et C_{i+1}. Cette nouvelle cellule devient la nouvelle tête de la liste `Start` (cf. figure cadre 2).
- Il peut être ensuite nécessaire de supprimer certaines cellules de l'enveloppe autour de cette nouvelle cellule (cf. figure cadre 3). Pour celà il suffit de reparcourrir la liste dans les deux sens (`C = Start->next` et `C = Start->prev`) et de tester la position de la cellule courante `C` par rapport au segment `[C->prev,C->next]` grâce à la méthode de position. Si la valeur reçue est négative, cela signifie que le point pointé par `C` est passé à l'intérieur ; il faut donc le supprimer et tester le suivant. Sinon on s'arrête.

Ecrire une fonction `PointEnveloppe` prenant en argument un point P et l'enveloppe convexe courante permettant de traiter le point selon l'algorithme ci-dessus.

Par ailleurs, cette fonction renverra le nombre de cellules insérées et/ou supprimées (max=+1, mini= ? ? ?, 0 si le point est intérieur).

▶ **Exercice 6 (Construction de l'enveloppe)**
Ecrire une fonction `Enveloppe` prenant en paramètre le tableau de tous les points et la liste enveloppe initiale (à trois cellules) et qui construit l'enveloppe convexe.
En outre, on souhaite que cette fonction stocke sur un fichier `Cell.dat` la progression du nombre de cellules de l'enveloppe en fonction du nombre de points traités. Pour visualiser le résultat, utiliser `gnuplot` avec la commande `gnuplot> plot "Cell.dat" with lines` .

▶ **Exercice 7 (Sauvegardes et tracés)**
Ecrire deux fonctions `SavePoints` et `SaveConvex` permettant d'écrire sur deux fichiers différents `Points.dat` et `Convex.dat` l'ensemble de tous les points et l'ensemble des points appartenant à l'enveloppe.
Pour visualiser le résultat, utiliser `gnuplot` avec la commande :

```
gnuplot> plot "Points.dat" with points, "Convex.dat" with lines
```