IFT 339: Structures de données
Travail 1
Daniel Lemire, Ph.D.

Objectifs

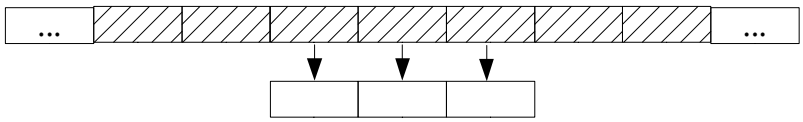
Se familiariser avec les listes et les piles tout en développant une application qui n'est pas triviale (utilise plusieurs structures de données). Une bonne modélisation orientée-objet ainsi qu'une bonne compréhension des entrées-sorties en C++ est nécessaire pour faire correctement ce travail.

Introduction

Il arrive souvent des situations en informatique où il n'est pas possible de stocker la totalité des données en mémoire. Il arrive aussi des situations où l'on ne connaît pas d'avance la mémoire qui sera nécessaire.

En particulier, lorsqu'on lit un fichier de taille importante, il peut être nécessaire de ne garder en mémoire qu'une « fenêtre » sur le contenu du fichier (voir Fig. 1) ayant une certaine taille maximale.

**Figure 1.** Lecture par fenêtre dans un fichier




On peut ensuite effectuer certaines opérations ou calculs sur la section lue. En d'autres termes, on doit mettre en oeuvre une mémoire tampon. Au besoin, on lit des données qui sont déposées dans notre mémoire tampon. Évidemment, il faut s'assurer que notre mémoire tampon n'excède pas une certaine taille fixée d'avance et on doit donc libérer de l'espace mémoire lors de la lecture. On retrouve donc le modèle First In, Last Out (FILO).

Application

Nous allons lire des « points » dans un fichier binaire. Notre mission est de trouver certaines configurations précises de points et de les répertorier. Mathématiquement, nous définissions un « point » comme étant un élément de $\mathbb{Z}\times\mathbb{Z}$ comme (-2,3) ou (4,2). La distance entre deux points est définie par $d(x,y)=\|x-y\|$ .

Nous allons utiliser une « mémoire tampon » ou « fenêtre » sur le fichier qui comportera *N* points. Nous dirons que *N* points $\{x_1, x_2, ..., x_n\}$ forment une figure régulière s'il existe $r\in\mathbb{Z}\times\mathbb{Z}$ tel que $d(r,x_j)=d(r,x_i)\,\forall\, i,j\in\{1,2,...n\}$ .

On voudrait que chaque fois qu'une figure régulière est découverte, elle se retrouve stockée dans une structure de données appropriée. En ce qui nous concerne, une pile fera très bien l'affaire.

Partie 1. Description du travail à effectuer

Vous devez créer une classe « Point » munie de la soustraction et d'une fonction « norme » (la norme d'un « Point » $(x_1, x_2)$ étant définie par $\sqrt{x_1^2+x_2^2}$ . Les points s'écrivent dans un flot sous la forme

« entier,entier » ou « 5,2 » par exemple. Si nous avons plusieurs points, nous les écirvont « 5,5 2,2  3,2 ... ». Pour simplifier le travail, nous allons supposer $N = 3$ ce qui signifie donc nous cherchons des triangles « réguliers » (isocèles). Afin de pouvoir stocker de façon pratique nos points formant des triangles, il faudra créer une classe « Triangle » avec un constructeur admettant 3 points. La classe « Triangle » devrait être muni de trois sélecteurs : « point1 », « point2 » et « point3 » nous permettant de retrouver les « Point » constituant le triangle. La classe « Triangle » doit aussi être munie d'une classe « aire » qui calcule l'aire du « Triangle » en question à l'entier le plus proche. On stocke les triangles trouvés dans une Pile (First In, First Out ou FIFO).

Une fois terminé, le programme devrait écrire dans un fichier texte l'aire (à l'entier le plus près) de tous les triangles isocèles rencontrés en utilisant une ligne par triangle. Il suffit d'écrire un entier (en texte) suivant d'un blanc et ainsi de suite. Comme les triangles doivent être dans une Pile, le dernier triangle trouvé devrait être au début du fichier et le premier à la fin (FIFO).

Exemple de syntaxe de l'exécutable:

    Aires points.txt aires.txt

Remarque: vous êtes requis d'utiliser les fichiers .h et .cpp disponibles sur le site Web du cours (http://www.ondelette.com/UdeS/) et discutés en classe. En particulier, vous êtes requis d'utiliser les mécanismes FIFO et FILO. Prenez note que cela vous sera très utile lors des examens!