IFT1166 – TRAVAIL PRATIQUE #4 – 22 mars 2004

Statistiques et Histogrammes au menu du C++!

Mohamed Lokbani

Équipes: Le travail est à faire en monôme ou en binôme, mais pas plus de deux. Vous ne remettez qu'un seul travail par équipe.
Remise : Deux remises à effectuer : électronique et papier le **mercredi 13 avril, 20h00 au plus tard, sans possibilité de retard**.
Conseil: N'attendez pas le dernier jour avant la remise pour commencer, vous n'aurez pas le temps.

But : Ce TP Ce TP a pour but de vous faire pratiquer l'amitié en C++ ; la surcharge des opérateurs ; les patrons de fonctions et de classes et finalement l'héritage simple.

Énoncé : Dans ce travail pratique, vous allez concevoir un patron de classe permettant de stocker des données de types prédéfinis, de calculer des fréquences sur ces données, et finalement, de simuler une représentation graphique de ces données.

Calcul des fréquences: Le calcul de fréquence consiste à mesurer le nombre de fois qu'un élément apparaît dans un ensemble de données. Si nous représentons cet ensemble de données par le tableau « data », de taille « taille » et contenant des données de type « int », par: data[8] = {1,2,1,3,4,2,6,2}. Les fréquences d'apparition des éléments dans le tableau « data » sont comme suit:

élément du tableau data	fréquence
1	2
2	3
3	1
4	1
6	1

Caractéristiques techniques : Dans ce travail pratique, vous allez concevoir les classes suivantes :

Patron de la classe Enregistrement Les statistiques sont réalisées sur un ensemble de données préalablement stockées dans le tableau data de type « T » et de taille « taille ». Vous avez à écrire, en langage C++, le patron de la classe « Enregistrement », contenant au minimum les membres décrits dans le fichier « enr.h ».

En plus de cela, vous devez calculer la fréquence d'apparition des éléments du type « T », contenus dans le tableau « data ». Pour ce faire, vous pouvez mémoriser le résultat du calcul de fréquence dans un autre tableau que vous déclarez comme membre privé de la classe « Enregistrement ». Sinon vous pouvez choisir une autre approche pour stocker les données traitées.

Le paramètre « T » peut prendre la place des trois types suivants: **int**, **char**, **string**.

Représentation graphique Pour ce travail pratique, nous allons simuler la sortie graphique sous les deux formes suivantes (pour plus de détails, voir annexe -1-):

- **Histogramme**: Pour chaque élément, nous représentons, sa fréquence relative en pour-cent.

- **Tarte** (i.e. diagramme sectoriel ou camembert): Pour chaque élément, nous représentons, sa fréquence relative en degrés.

Pour ce faire, vous devez écrire en C++, ce qui suit:

- Une classe de base abstraite **Graphe** ayant les 3 fonctions membres:

1) Constructeur: Graphe()
2) Destructeur: ~Graphe()
3) Une fonction virtuelle pure « Dessine(const Enregistrement&) », permettant de représenter les statistiques sous la forme de Histogramme ou Tarte:

- Une classe dérivée **Histogramme**, qui hérite avec le mode public, de la classe Graphe et contient les méthodes :

1) Constructeur: Histogramme()
2) Destructeur: ~Histogramme()
3) Une fonction virtuelle pure « Dessine(const Enregistrement&) », permettant de représenter les statistiques mémorisées dans la classe Enregistrement<T>, sous la forme de Histogramme:

- Une classe dérivée **Tarte**, qui hérite avec le mode public, de la classe Graphe et contient les méthodes :

1) Constructeur: Tarte()
2) Destructeur: ~Tarte()
3) Une fonction virtuelle pure « Dessine(const Enregistrement&) », permettant de représenter les statistiques mémorisées dans la classe Enregistrement<T>, sous la forme de Tarte:

Les fonctions Dessine des classes Histogramme et Tarte diffèrent par leur mode de calcul des fréquences relatives (l'un est en pour-cent, l'autre est en degrés).

Fichiers fournis :

enr.h : Un canevas de votre code qu'il faudra compléter.

graphe.h : Un canevas de votre code qu'il faudra compléter.

tp4H05.cpp : Le programme principal.

SortieEnr.txt : La sortie obtenue avec option de compilation Enr active et ceci afin de tester la classe Enregistrement.

SortieAll.txt : La sortie obtenue pour tester tout le programme.

Contraintes algorithmiques : Nous n'imposons aucune approche pour résoudre ce problème.

Contraintes Techniques :

- Assurer vous de respecter les noms des fichiers demandés, les noms des méthodes ainsi que le format de l'affichage en sortie.

- Des fichiers d'entrée/sortie sont fournis. On ne vous demande pas d'écrire du code pour lire/écrire à partir de fichiers, vous pouvez tester vos programmes à la main (en insérant manuellement les informations) comme vous pouvez utiliser les redirections (plus facile à manier et évite les erreurs de frappe). Les redirections sont au nombre de deux : < pour

l'entrée et > pour la sortie. Exemple : [**tp4.exe > masortie.txt**]. Dans cet exemple l'écriture se fera dans le fichier masortie.txt.

- Les fichiers entrée/sortie ont été générés sous **MinGW/Msys,** ils sont donc au format DOS étendu. Si vous travaillez sur LINUX assurez-vous de les convertir au format LINUX en utilisant pour cela la commande **dos2unix**. Pour convertir de LINUX vers le format dos utiliser plutôt la commande **unix2dos**. Ces commandes sont disponibles dans un xterm d'une plateforme LINUX.

- À signaler que nous utiliserons d'autres fichiers pour la correction. Si vous avez respecté les contraintes imposées, peu importe le jeu de test utilisé, il devra fournir normalement un résultat correct.

Automatisation de la correction : C'est « l'ordinateur » qui va corriger en quelque sorte vos travaux. Ce dernier va utiliser des scripts qui vont tenir compte au détail près des contraintes imposées.
Rediriger la sortie dans un fichier avec le symbole > et la comparer au caractère près avec les sorties fournies en exemple, aider vous dans cette tâche de la commande diff (sous MinGW/MSys sinon la commande fc sous dos, ajuster juste les options) comme suit :

```
diff –b –w –i –B out1.txt votresortie.txt
```

Par ailleurs, nous utiliserons d'autres jeux de test pour la correction.

Si les fichiers sont identiques, vous n'obtenez rien en sortie. Sinon, vous obtiendrez les différences entre les deux fichiers. Pour plus de détails sur cette commande unix, sur votre console xterm, exécutez la commande suivante: **man diff**.

Compilation : Pour générer l'exécutable, tp4.exe, vous pouvez utiliser par exemple une des solutions suivantes :

-1- Faire une compilation séparée à la main :

a) Pour la classe « Enregistrement » à part :

```
g++ -Wall -pedantic -Os -o tp4.exe -DEnr tp4H05.cpp
```

b) Pour tester tout le programme i.e les classes « Enregistrement » et « Graphe » :

```
g++ -Wall -pedantic -Os -o tp4.exe tp4H05.cpp
```

tp4.exe sera le programme exécutable. Ne pas oublier de mettre dans le même répertoire les fichiers *.h et *.cpp pour que la compilation puisse avoir lieu correctement.

Remise : Il est important de noter que votre TP sera compilé avec gcc3.2. Si par choix vous décidez d'utiliser un autre compilateur, vérifiez que le code que vous avez produit (qui normalement fonctionne correctement chez vous) fonctionne bien sur les ordinateurs du DESI. Avant de faire cela, assurez-vous d'abord que vous avez la bonne version de gcc, en utilisant pour cela la commande : "**gcc –v**" devra donnée le numéro de version "**3.2**".

La remise comprend **3** (**TROIS**) choses (avant le mercredi 13 avril à 20h00) :

1. Envoyez vos fichiers « enr.h », « graphe.h » et le rapport par la procédure de remise électronique habituelle (Pour obtenir de l'aide sur cette commande, tapez dans un xterm : man remise). Respecter les noms des fichiers.

   remise pift1166 travail04 enr.h graphe.cpp rapport

2. Remettez une **copie papier** d'un rapport qui devrait décrire votre programme (pour le contenu d'un rapport voir la FAQ sur la page web du cours).

3. N'oubliez pas de remettre avec votre rapport une **copie papier** de votre programme C++.

Barème : Ce TP1 est noté sur 12 points.

Compilation et respect des spécifications	**1**
Algorithmique, codage, commentaires etc.	**4**
Rapport	**3**
Tests (fournis et non fournis)	**4**

En plus du précédent barème, vous risquez de perdre des points dans les cas suivants ….

- La non remise électronique (volontaire ou par erreur) est sanctionnée par la note 0.
- La non remise papier vous pénalise de 1 point.
- Les programmes ne contenant pas d'en-tête, -1 point.
- Un programme qui ne compile pas : 0/12.
- Un programme qui compile mais ne fait les choses prévues dans la spécification : 0/12.
- Les avertissements (warnings) non corrigés : cela dépend de la quantité! À partir de -0.5 et plus.
- Le nom respect du nom de fichier, donc forcement il ne va pas compiler et un des points de la spécification n'a pas été respecté : 0/12.
- Aberration dans le codage : Même si tous les chemins mènent à Rome, faites l'effort nécessaire pour éviter de prendre le plus long! -0.5 et plus, en fonction des dégâts constatés !

Des questions à propos de ce TP?

Une seule adresse : pift1166@iro.umontreal.ca

Pour faciliter le traitement de votre requête, inclure dans le sujet de votre email, au moins la chaîne: [IFT1166] et une référence au **travail04**.

Mise à jour

22-03-2005 diffusion

Annexe

Exemple d'un calcul de fréquence

Si le tableau `data` est un ensemble de données du type `int`:

```
data[8] = {1,2,1,3,4,2,6,2}
```

Nous obtenons les résultats suivants:

éléments distincts du tableau data	fréquence (F)	Histogramme (H en %)	Tarte (T en degrés)
1	2	25	90
2	3	37.5	135
3	1	12.5	45
4	1	12.5	45
6	1	12.5	45

Où:

*** fréquence (F):** indique le nombre de fois qu'un élément apparaît dans `data`. Par exemple, dans `data`, le "**`1`**" apparaît **`deux`** fois.

*** fréquence totale (FT):** est la somme des fréquences de chaque élément distinct de data (i.e. la somme des FQ). Par exemple, `FT = 2+3+1+1+1+1 = 8`

*** représentation en Histogramme (H):** une règle de trois nous permet de calculer pour chaque élément, la fréquence relative, en pour-cent `(H)`, par rapport à la fréquence totale. Par exemple, pour l'élément "**`1`**", cette valeur est calculée comme suit:

```
F = 2, FT = 8 => H = 2*100/8 = 25%
```

*** représentation en Tarte (T):** une règle de trois nous permet de calculer pour chaque élément, la fréquence relative, en degrés `(T)`, par rapport à la fréquence totale. Par exemple, pour l'élément "**`1`**", cette valeur est calculée comme suit:

```
F = 2, FT = 8 => H = 2*360/8 = 90°
```

Questions et Réponses

Questions Pratiques

-Q : En quoi consiste notre travail au juste?
-R : Écrire les classes Enregistrement et Graphe.

-Q : Où doit-on écrire nos programmes?
-R : Les fichiers enr.h et graphe.cpp

-Q : Par quoi dois-je commencer pour m'assurer que je suis sur la bonne voix ?
-R : Faire d'abord la classe template Enregistrement, puis la tester ; avant d'aller faire l'héritage et la classe « Graphe ».

-Q : Est-ce que je dois écrire une fonction main ?
-R : Non, puisqu'on fournit une. Voir le fichier tp4H05.cpp.

-Q : Est-ce que je peux modifier alors le contenu du fichier tp4H05.cpp ?
-R : Non vous ne pouvez pas modifier le fichier tp4H05.cpp. Ceci écrit, vous pouvez quand même écrire un pseudo fichier montp4.cpp qui va contenir votre fonction main test et ceci dans le processus de développement uniquement. Avant de remettre votre travail assurez-vous de compiler avec la version originale fournie, i.e. tp4H05.cpp. C'est cette version et rien d'autre que nous allons utiliser lors de la correction.

-Q : L'énoncé n'est pas clair et j'ai besoin d'aide ! Comment procéder ?
-R : Nous sommes disponibles suivant l'horaire affiché sur la page web du cours, rubrique « Professeurs et démonstrateurs ». Vous pouvez aussi nous envoyer vos questions par courriel.

-Q : Mais je ne pourrai pas être présent aux horaires indiqués ?
-R : Envoyer nous un courriel pour voir s'il nous est possible de vous rencontrer à un autre moment.

Q : Pouvez-vous m'expliquer ce qui ne va pas? Je vous envoie pour cela mes fichiers enr.h et graphe.h !
R : N'envoyer rien par courriel. Privilégier plutôt les séances de disponibilité offertes à vous.

-Q : Où se trouvent les fichiers du « tp#4 » ?
-R : Sur la page web du cours, rubrique « Démonstrations et devoirs », puis « tp4 ».

<u>Questions Techniques générales</u>

-Q : A propos de l'opérateur surcharges, (+=): lorsqu'on demande d'ajouter les éléments de tab dans « data_ », est-ce que ça signifie par exemple:
si data_ = [1, 4, 5, 2] et tab = [1, 0, 0]
alors data_ += tab
va donner data_ = [1, 4, 5, 2, 1, 0, 0] ?
-R : Oui.

-Q : Pour l'opérateur (-=):
si data_ = [1, 4, 5, 2, 1, 0, 0] et tab = [1, 0]
alors data_ -= tab va donner data_ = [4, 5, 2] ou data_ = [4, 5, 2, 1, 0] ?
-R : data_ = [4, 5, 2]

-Q : Pour l'opérateur (-=), la version où on enlève un élément en particulier,
si data_ = [1, 1, 1, 2] et tab[i] = 1
alors data_ -= tab[i] va donner data_ = [2] ou data_ = [1, 1, 2]?

-R : tab[i] ou "tab" il faut juste qu'il soit un "T" ...
data_ = [2]
Pour gagner du temps essayes de voir si tu ne peux pas dériver "l'un" (un des opérateurs) à partir de "l'autre". Histoire de ne pas réinventer la roue ...

-Q : Dans l'énoncé du TP4, vous dites que l'on doit créer une classe abstraite Graphe ayant les trois méthodes « Graphe(); », « ~Graphe(); » et « void Dessine(); ». Est-ce que l'on peut aussi créer nos propres fonctions au besoin (En private ou protected.)
-R : Oui si c'est nécessaire. Puisque il y a la notion d'héritage qui entre en jeu, essayes de voir si par exemple, il te faut introduire ces fonctions dans la classe de base, ou bien plutôt dans les classes dérivées.

-Q : Comment dois-je gérer les erreurs générées dans les fichiers d'entêtes ?
-R : Vous allez coder les templates dans ce tp#4. Les templates doivent être codées dans les fichiers *.h. De ce fait la plus part des erreurs de "compilation" que vous pourriez avoir seront en rapport avec des fichiers *.h de la bibliothèque C++. Sachez dès à présent, qu'il n'y a pas d'erreurs ("connues") dans les fichiers de la bibliothèque C++ fournie avec g++ ni dans le fichier de test tp4H05.cpp. De ce fait, si erreur de compilation il y a, elle est belle et bien dans votre programme, i.e. dans vos fichiers *.h

Reste à savoir comment la trouver? D'où l'intérêt de ce tp. Quelques conseils:

-1- Examiner très bien le message d'erreur, et vérifier très bien que votre code est bien conforme aux règles d'écriture des classes templates et des fonctions amies.

-2- Faites d'abord la classe template enregistrement avant d'aller faire l'héritage. Vous serez au moins "rodés" avec les messages d'erreur associés aux templates.