IFT1169 – TRAVAIL PRATIQUE #2 – 07 février 2010

« Héritage sous toutes les sauces »

Mohamed Lokbani

Équipes : le travail est à faire en monôme (**une seule personne**).
Remise : une seule remise est à effectuer par voie électronique **le jeudi 25 février 2010, 23h59 au plus tard, sans possibilités de prorogation**.
Conseils : n'attendez pas le dernier jour avant la remise pour engager votre travail. Vous n'aurez pas le temps nécessaire pour le réaliser.

But : ce TP a pour but de vous faire pratiquer l'héritage, le polymorphisme et l'utilisation des « makefiles ».

Énoncé : il vous est demandé d'écrire un programme C++ permettant de traiter les relations d'héritage décrites dans ce qui suit.
Une figure géométrique est un ensemble de points permettant de représenter un objet géométrique comme le point, une ligne, un cercle, un cube etc. Pour chaque figure géométrique nous avons le type de la figure et la méthode « getAire» qui permet de calculer l'aire de la figure. Pour une figure à 3 dimensions, « getAire » calcule la surface de cette figure.

Chaque figure géométrique peut-être représentée dans un espace à 2 dimensions ou 3 dimensions.

- À 2 dimensions : rectangle, cercle, triangle, ellipse, carré etc. Notons que si les 4 côtés d'un rectangle sont identiques, nous avons affaire à un carré. De même, un cercle est un cas particulier de l'ellipse.
- À 3 dimensions : sphère, cube, parallélépipède, cylindre etc. À signaler qu'un cube n'est qu'un parallélépipède bien particulier.

Pour chaque figure géométrique à 2 dimensions, le périmètre de la figure est calculé à l'aide de la méthode « getPerimetre ». Alors que pour une figure à 3 dimensions, le volume de la figure est calculé à l'aide de la méthode « getVolume ».

1ere partie

-1- Définir le schéma d'héritage pour l'ensemble des classes décrites dans le précédent énoncé: Figure, F2D, F3D, Ellipse, Sphère, Carré, Cube, Triangle, Cylindre, Parallélépipède, Cercle et Rectangle.

2- Définir les différentes classes et les méthodes associées « getAire », « getPerimetre » et « getVolume ».

2e partie

Dans cette partie nous allons appliquer les notions de polymorphisme apprises dans le cours.

-3- Définir un tableau avec une taille par défaut « T » égale à « 5 », si le paramètre « -t » n'a pas été spécifié sur la ligne de commande. Sinon « T » prend cette nouvelle valeur. Ce tableau peut contenir des éléments du type « Figure ».

-4- Générer aléatoirement « T » types de figures et les insérer dans le tableau. Pour ce cas, passer la valeur « 1 » à la fonction « time » présentée dans l'annexe « 2 ». Les figures seront initialisées avec des valeurs elles aussi aléatoires.

-5- Si l'option « -p » n'est pas spécifiée sur la ligne de commande, afficher en sortie le contenu du tableau. Dans le cas contraire, l'option « -p » sera toujours suivie par le nom de la figure recherchée. Dans ce cas, valider par deux méthodes (une directe, l'autre utilisant la ligature dynamique), si la figure passée en argument est présente ou non dans le tableau. Si c'est le cas, afficher ses paramètres. Dans le cas contraire, mentionner qu'elle n'est pas présente.

Fichiers à remettre : vous devez remettre les fichiers nécessaires pour une compilation et une exécution correctes de votre programme. Le fichier principal qui contient la fonction « main » doit porter le nom de « tp2H10.cpp ». Vous devez inclure aussi un fichier « makefile » permettant d'automatiser la compilation de vos fichiers. Par ailleurs, vous devez inclure dans votre remise un rapport décrivant le travail effectué. Ce rapport doit porter le nom « tp2H10.pdf ».

Remise : il est important de noter que votre TP sera compilé avec « gcc4.4.0 ». Si par choix vous décidez d'utiliser un autre compilateur, vérifiez que le code que vous avez produit (devant normalement fonctionner correctement chez vous) fonctionne aussi bien sur les ordinateurs de la DESI. Pour avoir la version du compilateur, utilisez la commande "**gcc –v**", qui devra donner le numéro de version « **4.4.0** ».

Par ailleurs, assurez-vous de la présence de l'option « pedantic » sur la ligne de compilation.

Pour éviter d'omettre un des fichiers associés à ce travail lors de la remise, nous vous conseillons de regrouper l'ensemble des fichiers dans un seul fichier compressé **tp2h10.zip** et de remettre ce dernier comme suit :

1. commencez d'abord par vous connecter sur la machine « **remise** » comme cela a été pratiqué dans la démo #01.
2. envoyez l'ensemble de vos fichiers par la procédure de remise électronique habituelle (Pour obtenir de l'aide sur cette commande, tapez dans un Xterm : man remise). Respectez les noms des fichiers.

   remise ift1169 tp2 tp2h10.zip

3. vérifiez que la remise s'est effectuée correctement.

   remise –v ift1169 tp2

Barème : ce TP2 est noté sur **12 points**. Les points sont répartis comme suit :

Compilation et respect des spécifications	**1**
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.
- Les programmes ne contenant pas d'en-tête, -1 point.
- Un programme qui ne compile pas : 0.
- Un programme qui compile mais ne réalise pas les choses prévues dans la spécification : 0.
- Les avertissements (warnings) non corrigés : cela dépend de la quantité! À partir de -0.25 et plus.
- Le non respect du nom du fichier va générer une erreur de compilation donc un des points de la spécification n'a pas été respecté : 0.
- Aberration dans le codage : même si tous les chemins mènent à Rome, faites l'effort nécessaire pour éviter de prendre le plus long!

Des questions à propos de ce TP?

Une seule adresse : dift1169@iro.umontreal.ca

Pour faciliter le traitement de votre requête, inclure dans le sujet de votre email, au moins la chaîne: [IFT1169] et une référence au **tp02**.

Mise à jour

07-02-2010 diffusion de l'énoncé.

Annexe -1-

Calcul des périmètres, surfaces et volumes

Si **π = 3.1415**

Carré

Le périmètre d'un carré de côté « c » est : P = 4c
La surface d'un carré de côté « c » est : $S = c^2$

Cercle

Le cercle est un cas particulier de l'ellipse où (a = b = r) où « r » est le rayon du cercle.
Le périmètre d'un cercle de rayon « r » : $P = 2 \pi r$
La surface d'un carré de rayon « r » : $S = \pi r^2$

Cube

La surface d'un cube d'arête « c » est : $S = 6c^2$
Le volume d'un cube d'arête « c » est : $V = c^3$

Cylindre (droit)

La surface d'un cylindre de rayon « r » et de hauteur « H » est : $S = 2 \pi r (H + r)$
Le volume d'un cylindre de rayon « r » et de hauteur « H » est : $V = \pi r^2 H$

Ellipse

Le périmètre d'une ellipse de demi-grand axe « a » et de demi-petit axe « b », d'après une estimation de Ramanujan, est : $P = \pi \times [3(a + b) - \sqrt[2]{(3a + b)(a + 3b)}]$

La surface d'une ellipse de demi-grand axe « a » et de demi-petit axe « b » est : $S = \pi a b$
(Si l'on prend le grand axe et le petit axe au complet : $S = \pi a b/4$)

Parallélépipède

La surface d'un parallélépipède de longueur « L », largeur « l » et de hauteur « H » est : S = 2 (L + l) H + 2 (L x l)
Le volume d'un parallélépipède de longueur « L », largeur « l » et de hauteur « H » est : V = L x l x H

Rectangle

Le périmètre d'un rectangle de longueur « L », largeur « l » est : P = 2 (L + l)
La surface d'un rectangle de longueur « L », largeur « l » est : S = L x l

Sphère

La surface d'une sphère de rayon « r » est : $S = 4\pi r^2$
Le volume d'une sphère de rayon « r » est : $V = (4 \pi r^3)/ 3$

Triangle

Le périmètre d'un triangle ayant les côtés « a », « b » et « c » : P = a+b+c
La surface d'un triangle de base « B » et de hauteur « h » : S = B x h/2

Annexe -2-

Génération des nombres aléatoires

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main() {

    const int M = 100; // Pour générer aléatoirement un nombre [0,100[
    const int P = 52;  // Pour générer aléatoirement une lettre de l'alphabet [azAZ]

    // Pour générer des nombres différents à chaque fois
    // Si on veut générer toujours la même valeur, il suffit d'effectuer l'appel
    // comme suit : srand(1)
    srand(static_cast<unsigned>(time(NULL)));

    // Nombre réel entre [0,1[
    double a = (((double) rand())/((double) (RAND_MAX) + (double) (1)));

    cout << "Nombre réel entre [0,1[ : " << a << endl;

    // Nombre réel entre [0,100[
    double b = (((double) rand())/((double) (RAND_MAX) + (double) (1)))*M;

    cout << "Nombre réel entre [0,100[ : " << b << endl;

    // Nombre entier entre [0,1[ donc forcement 0
    int c = (int) (((double) rand())/(double) (RAND_MAX) + (double) (1)));

    cout << "Nombre entier entre [0,1[ : " << c << endl;

    // Nombre entier entre [0,100[
    int d = (int) ((((double) rand())/((double) (RAND_MAX) + (double) (1)))*M);

    cout << "Nombre entier entre [0,100[ : " << d << endl;

    // Nombre entier entre [1,101[
    int e = (int) ((((double) rand())/((double) (RAND_MAX) + (double) (1)))*M)+1;

    cout << "Nombre entier entre [1,101[ : " << e << endl;


    char tab[] = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

    // Retourne une valeur aléatoire entre 0 et 51 qui va servir à récupérer
    // la lettre du tableau « tab »


    char f = tab[(int) ((((double) rand())/((double) (RAND_MAX) + (double) (1)))*P)];
    cout << "Une des lettres de l'alphabet [azAZ]: " << f << endl;



    return 0;

}
```