Université Paris Diderot
Master 1

Langages à Objets Avancés
Année 2015-2016

TP n°1

Introduction à C++

Exercice 1 (Calculette)

1. Créez un repertoire `ObjetsAvances/TP1/Calculette`, puis écrivez dans un éditeur de votre choix un programme `calculette.cpp`. Il consistera à demander à l'utilisateur 2 nombres entiers puis affichera la somme, la différence, le produit, le quotient, le reste de la division entière entre ces 2 nombres. Compilez le avec `c++`.
2. Modifier ce programme en demandant à l'utilisateur quelle opération il veut réaliser. (On continuera à demander tant que l'entrée ne correspondra pas à l'une des opérations citées.) Voici un exemple de ce que pourra donner l'exécution :

```
Entrer le premier nombre : 12
Entrer le deuxième nombre : 2
Opération à effectuer? ( +,  - , *,  /, % ) : *
Le résultat de la multiplication de 12 par 2 est 24
```

Exercice 2 (Environnement de travail)
Pour que vous puissiez travailler chez vous comme à l'ufr, voici différentes façons de procéder.

1. (à la main) Vous venez de le faire au premier exercice (avec `c++` ou `g++`) dans une console.
2. (avec Makefile) La paire (commande make / fichier Makefile) permet de faire une séries de compilations à l'aide d'une seule commande. Le fichier Makefile contient des règles de la forme :

```
cible : dependances
        commandes
```

   les dépendances étant d'autres cibles, séparées par des espaces, ou un fichier. (Attention à la syntaxe : avant "commandes" il doit y avoir une tabulation).

   La commande `make cible` se chargera d'exécuter les commandes de votre cible, après avoir rafraîchit les dépendances si c'est nécessaire.

   Dans notre cas, créez un repertoire TestMakefile où vous ferez une copie de votre calculette, puis créez le fichier `Makefile` suivant :

```
GOALS = calculette
HEADERS_DIR = .
all : $(GOALS)
```

```
calculette : calculette.o
    g++ -o calculette calculette.o
calculette.o : calculette.cpp
    g++ -Wall -I $(HEADERS_DIR) -c calculette.cpp
clean :
    rm *.o $(GOALS)
```

puis à la console faites `make all` (ou simplement `make`).

Vous pourrez alors executez `./calculette`.

Pour terminez faites `make clean`

3. (avec Makefile et emacs) Dans le repertoire précédent, en éditant `calculette.cpp` avec `emacs` appuyez simultanément sur Alt et x puis tapez `compile` dans le mini-menu, qui vous permettra de lancer une commande `make`
4. (avec codeblocks) lancez `codeblocs calculette.cpp &` puis appuyez sur l'icone de compilation/exécution (flèche verte avec un engrenage). Passez quelques minutes à observez le menu de codeblocks qui vous permettra par la suite de créer des projets et des classes.
5. (avec netbeans) Lancer netbeans et acceptez le repertoire de travail proposé, puis créez un nouveau projet `c++` editez le main.cpp par defaut et copier-y le code de votre calculette. Lancer la compilation.

   Passez également quelques minutes à lire le menu qui apparait lorsque vous cliquez avec le bouton droit sur le nom de votre projet, il vous permettra par la suite de créer des classes.

Exercice 3 (Classes)
On cherche à écrire une classe qui modélise les points du plan par ses coordonnées. Elle se réduit pour le moment à :

— des accesseurs,
— des modifieurs,
— une méthode d'affichage,

1. (Compilation séparée) Ecrivez trois fichiers `Point.hpp` de description, `Point.cpp` de réalisation, et `Test.cpp` où vous aurez un `main` pour tester la création d'un objet et son affichage.
2. (Utilisation de this) Ajoutez une méthode `distance(Point)` qui retourne la distance entre deux points (indication : utilisez `#include <math.h>`)
3. (Encapsulation) Ecrivez une classe triangleOrienté dont les 3 sommets s'appelleront respectivement A,B,C selon l'ordre dans lequel ils apparaissent au moment de la construction.
4. Testez son affichage, et écrivez une méthode qui calcule son périmètre.
5. On considère le point P de coordonnées (1,0) et un triangle construit entre autres avec le point P. Qu'arrive t'il au triangle après l'exécution de `P.setX(2);` ? Que pouvez vous dire de la nature du passage des paramètres aux méthodes ?
6. Ecrivez une méthode `void rotateHor()` qui échange les places de sommets d'un triangle de sorte que A se retrouve en B, B en C, et C en A.

7. Combien d'objets sont créés durant cette opération ?
8. Lorsqu'un triangle a un périmètre trop grand, on souhaite le rapetisser de la facon suivante : soit XY le coté plus long d'un triangle de sommets X, Y, Z. On remplace X par le milieu M de XY, de sorte que le triangle soit à présent constitué des points M,Y et Z. Ecrivez une méthode `void rapetisse()` qui effectue cette manipulation, ainsi que la méthode intermédiaire `milieu`.