

TP n°1

Introduction à C++

Note : ces exercices nécessitent la lecture des supports des deux premiers cours que l'on peut retrouver à l'adresse <http://www.liafa.univ-paris-diderot.fr/~yunes/cours/cpp/>

Exercice 1 (Hello World)

Comme la tradition le veut, vous devez écrire un programme C++ écrivant un message de bienvenue à l'écran. On rappelle que l'instruction permettant d'envoyer une chaîne de caractères à l'écran a la forme :

```
cout << chaine de caractères << endl;
```

Utilisez votre éditeur préféré pour le faire. Appelez votre fichier : `Hello.cpp`.

Compilez avec `gcc`. Compilez avec `g++`. Utilisez le `Makefile` suivant pour la compilation (attention le décalage des lignes de commandes correspond et doit correspondre à une tabulation - caractère TAB).

```
GOALS = exo1

all : $(GOALS)

exo1 : exo1.o
      g++ -o exo1 exo1.o

exo1.o : exo1.cpp
      g++ -Wall -c exo1.cpp

clean :
      -rm *.o $(GOALS)
```

Exercice 2 (Code Blocks)

Utilisez l'application `Code::blocks` (binaire codeblocks) pour faire l'Exercice 1. Pour cela il faut créer un projet en faisant attention de choisir un projet de type « Console Application » (tout autre type pouvant potentiellement poser des problèmes par la suite)...

Exercice 3 (Constantes)

Écrivez en C++ une fonction permettant de calculer le sinus d'un angle donné dans une unité parmi (degrés, radians et grades). Note : on utilisera des constantes de type `int` pour représenter les unités et pour pouvoir utiliser la fonction `sin` on utilisera la directive d'importation de fonction C dans le monde C++.

Exercice 4 (Namespaces)

Modifiez l'exercice précédent de sorte que les différentes choses (constantes, fonctions, etc) soient protégées par un espace de nom (namespace). Fabriquez avec le tout, un module (fichiers sources séparés et fichiers d'entête séparés). Profitez-en pour fabriquer un `Makefile` (en adaptant le `Makefile` donné) permettant de compiler correctement et séparément les divers modules. Testez en modifiant légèrement l'un ou l'autre des fichiers...

Exercice 5 (Valeurs par défaut)

Ajoutez au code de l'exercice précédent une valeur par défaut permettant de spécifier l'unité par défaut pour exprimer les angles... par exemple les degrés...

Exercice 6 (Surcharge)

Écrivez un nouveau module de fonctions toutes appelées `plus` permettant de calculer : pour l'une la somme de deux ints passés en paramètres et renvoyant un int, pour l'autre la somme de deux doubles et renvoyant un double. Appelez la fonction `plus` en passant : deux ints, un int et un short, deux floats, deux doubles, un int et un double. Que dit le compilateur ? Pourquoi ?

Exercice 7 (Passage d'un tableau)

Écrivez un nouveau module de fonctions toutes appelées `somme` et qui permettent de calculer la somme des éléments d'un tableau d'ints pour l'une et de double pour l'autre. Appelez la fonction `somme` en passant : un tableau d'ints, un tableau de shorts, un tableau de doubles. Que dit le compilateur ? Pourquoi ?

Exercice 8 (Arguments constants)

Ajoutez `const` aux arguments des fonctions de l'exercice 7 après les avoir clonés. Essayez dans le corps des fonctions de modifier les arguments. Que dit le compilateur ? Pourquoi ?

Exercice 9 (Encore des constantes)

Définissez les fonctions `f(char)` , `g(char &)` , `h(const char &)` et appelez les avec les arguments `'a'`, `49`, `3300`, `c`, `uc` où `c` est un `char` et `uc` un `unsigned char`. Qu'est-ce qui se passe ? Dans quels cas le compilateur doit créer une variable temporaire ?

Exercice 10 (Tableau)

Dans un module approprié, écrivez un ensemble de fonctions permettant d'accéder de façon sécurisée à un tableau (créez une structure `tableau` avec un champ `longueur`, etc) : ces fonctions vérifieront que l'on utilise le tableau correctement (débordements). Indication : on veut pouvoir (entre autres) écrire `elementAt(tableau,i) = 12;`

Exercice 11 (Une première classe)

Écrivez une classe `Compte` qui représente un compte de banque. Le constructeur prend le montant initial du compte. Ajoutez des méthodes pour retirer de l'argent, pour déposer de l'argent et pour afficher le montant actuel. Dans la fonction `main` créez un objet `c1` de type `Compte` de façon statique et un objet `c2` de type `Compte` de façon dynamique. Appelez les méthodes des objets en utilisant `.` et `->`.