© *Jean-Baptiste APOUNG KAMGA <jean-baptiste.apoung@math.u-psud.fr>*

Fiche de TP1 : Notion de base de C++

Le but de ce TP est de passer en relief les notions de base du C++ . Pour gagner du temps dans l'écriture des bouts de programmes fournis dans les exercices, nous rappelons que les fichiers sources de ces programmes sont téléchargeables à l'adresse **/doc/apoung/M2_IM**

Thème - 1 *Rappels sur la compilation*

Comme nous l'avons vu en cours, la compilation se déroule en plusieurs étapes. Cependant, dans la pratique, l'utilisateur n'en perçoit que deux. La compilation proprement dite et l'édition des liens. Supposons donc qu'on veuille compiler un programme contenu dans le fichier (**fichier.cpp**). Une commande de compilation s'écrit simplement

```
g++ -c fichier.cpp
```

Si la compilation se déroule correctement, le fichier objet **fichier.o** est créé. Reste maintenant à éditer les liens. C'est cette fois la commande

```
g++ -o nom-d-executable fichier.o
```

qui s'en charge.

Notons que ces commandes sont des versions minimales au sens où il est possible d'ajouter des options. Par exemple, pour compiler une version optimisée (de niveau 2) du code, on peut utiliser

```
g++ -c -O2 fichier.cpp
```

Pour générer une version *debug* du programme, on utilise

```
g++ -c -g fichier.cpp
```

De même, pour l'édition des liens il est nécessaire de spécifier les bibliothèques qui ont été utilisées. Par exemple, si on a utilisé la bibliothèque mathématique (utilisation d'un sinus par exemple), il faut utiliser la commande

```
g++ -o nom-d-executable fichier.o -lm
```

Thème - 2 *Entrée-sortie, sélections, boucles*

Exercice-1 **:** Dans un fichier appelé par exemple **hello.cpp**, écrire un programme qui affiche

```
Hello world!
```

à la console.

Exercice-2 : Écrire un programme qui affiche les différents arguments passés à la fonction **main**. Par exemple, si l'exécutable s'appelle **execname**, la commande

```
execname 1 deux 3
```

affichera le résultat suivant à la console

```
execname
1
deux
3
```

Pour cet exercice, on rappelle les arguments de la fonction **`main`** :

```
in main(int argc,char** argv)
```

- **`argc`** est le nombre d'arguments passés en ligne de commande et,
- **`argv`** est le tableau des chaînes de caractères.

Exercice-3 : Sur les mêmes bases, écrire un programme qui écrit les arguments à l'envers. Cette fois, l'exécution de

```
execname 1 deux 3
```

affichera à la console,

```
3
deux
1
execname
```

Exercice-4 : En suivant les mêmes principes et ceci afin de manipuler encore les chaînes de caractères, écrire cette fois un programme qui écrit chaque argument à l'envers, de sorte à obtenir

```
emancexe
1
xued
3
```

Proposer deux versions. Une première comptant les caractères de chaque mot et une seconde basée sur la récursivité.

Exercice-5 : Afin d'assimiler le lien qui existe entre les **`chars`** et les nombres entiers, on propose maintenant d'écrire un programme qui affiche les arguments en mettant la première lettre de chaque argument en majuscule. Pour cela, on se servira du fait que dans la table des caractères ASCII la distance entre les majuscules et les minuscules est constante. De ce fait, on a

$'A' - 'a' == 'Z' - 'z'$

Si on écrit

```
execname un deux trois
```

On obtiendra

```
Execname
Un
Deux
Trois
```

Modifiez votre programme pour que

```
execname 1 Deux trois
```

affiche

```
Execname
1
Deux
Trois
```

Thème - 3 *Formater en sortie*

Exercice-1 **:** La suite des nombres de Fibonacci est définie de manière récursive par

$$u_0 = 0, \quad u_1 = 1, \quad u_{n+1} = u_n + u_{n-1} \quad \text{pour} \quad n = 1, 2, 3, \cdots$$

Par exemple, $u_2 = u_1 + u_0 = 1 + 0 = 1$ et $u_3 = u_2 + u_1 = 1 + 1 = 2$. Les nombres de Fibonacci ont de nombreuses applications et de propriétés. L'une d'elles est que la suite des quotients

$$q_n = u_n/u_{n-1}, \quad n = 2, 3, 4, \ldots$$

converge vers le nombre d'or, à savoir $(2 + \sqrt{5})/2 \approx 1.618$.

En codant les nombres de fibonacci et leur quotient respectivement en *long int* et *long double*, écrire un programme qui génère l'affichage suivant.
(Utiliser les manipulateurs **`width,setw,fill,precision`** accessibles via **`#include <iomanip>`**)

Listing 1 – Sortie attendue

```
 n         Nombre de Fibonacci          Quotient de Fibonacci
------------------------------------------------------------
10                          55          1.6176470588235294118
20                        6765          1.6180339631667065295
30                      832040          1.6180339887482036213
40                   102334155          1.6180339887498947364
50                 12586269025          1.6180339887498948482
60               1548008755920          1.6180339887498948482
70             190392490709135          1.6180339887498948482
80           23416728348467685          1.6180339887498948482
90         2880067194370816120          1.6180339887498948482
```

Thème - 4 *Entrée/Sortie dans un fichiers*

Exercice-1 **:** Écrire un programme qui clone le fonctionnement de la commande Unix **cat**. C'est à dire qui affiche à l'écran le contenu d'un fichier texte. Le nom du fichier texte à afficher est à passer en argument de la ligne de commande.

Pour cela utiliser la classe **ifstream** et l'objet **cout** de la bibliothèque standard C++ . Attention, par défaut les espaces sont ignorés. Pour cela utiliser la fonction membre **unsetf** de la manière suivante :

```
std::ifstream fin(''nom-du-fichier'');
fin.unsetf(std::ios::skipws);
```

qui obligera **fin** à lire les espaces.

Exercice-2 **:**

Écrire maintenant un programme qui écrira dans un fichier des points du graphe de

$$x \mapsto \sin(x)$$

pour x compris entre $-\pi$ et π. Formater le fichier sous la forme

```
abscisse0 ordonnées0
abscisse1 ordonnées1
abscisse2 ordonnées2
...
```

Visualiser ensuite le résultat à l'aide de **gnuplot** :
- entrer la commande **gnuplot**
- dans **gnuplot** taper
  plot 'nom-du-fichier' w l

Exercice-3 **:**

On considère le fichier généré à l'exercice précédent. Dont le contenu est sous forme d'une matrice à N lignes et M colonnes. Les valeurs N et M étant inconnues, on voudrait le renseigner dans ce fichier (afin de l'ouvrir par exemple sous Matlab). On convient alors de mettre à l'entête du fichier l'information

```
# ce fichier contient une matrice de taille N, M
```

Pour cela sous l'éditeur de texte, on ajoute une ligne vide dans l'entête du fichier puis on le sauvegarde.

Écrire un programme C++ , qui ouvre ce fichier sauvegardé, le lit, et le modifie comme souhaité.

(On se renseignera sur la fonction **istream**::**seekg**).