Algorithmique - TP1

IUT 1ère Année

7 septembre 2014

1 Environnement de Compilation

Nous travaillerons sous le système d'exploitation Linux. Pour la programmation C++, il sera nécessaire de :

- — créer un répertoire appelé algorithmique (dans votre répertoire documents), dans lequel vous construirez plusieurs répertoires (TP1, TP2, etc.), chacun correspondant à une feuille de TP.
- — utiliser un éditeur de code source, comme gedit ou emacs (le dernier existe dans la plupart des environnements).
- — utiliser une fenêtre terminal ouverte afin de lancer des commandes en ligne.

Lorsque vous aurez terminé une feuille de TPi, vous archiverez le répertoire TPi avec vos codes sources, et enverrez le fichier compressé sur votre compte Moodle.

Note : tous les documents (cours, TD et TP) sont accessibles à la fois sur Moodle et Dropbox. Pour ce dernier, l'adresse d'accès est :

`https://www.dropbox.com/sh/r14mpbtujaz8p38/AABOpg9riQBYjyBsGGlKTnNQa?dl=0`

1.1 Ecriture d'un programme C++

Le code source d'un programme C++ est sauvegardé dans un fichier dont l'extension est .cc ou .cpp. Pour vous familiariser avec votre éditeur, commencez par écrire le programme (un grand classique) suivant :

Listing 1 – helloWorld.cpp

```
#include <iostream>
using namespace std;

int main()
{
cout << "Hello world" << endl;
}
```

1.2 Compilation d'un programme C++

La compilation d'un programme C++ inclut plusieurs étapes

1. **prétraitement** : le pré-processeur transforme le code source (ascii) en un code prétraité (toujours ascii) selon les directives qui commencent par #. Dans le cas du fichier helloWorld.cpp, le pré-processeur remplace la commande #include <iostream> par le contenu du fichier iostream installé sur le système. Ce fichier iostream contient les déclarations des opérations d'affichage et de lecture.
2. **compilation** : le compilateur transforme le code prétraité (ascii) en code objet (binaire), en utilisant plusieurs analyses (lexicale, syntaxique et sémantique) et en terminant (pour les compilateurs modernes) par une phase d'optimisation (selon le type de processeur utilisé par la machine).

3. **édition de lien** : l'éditeur de lien intègre au code objet toutes les bibliothèques (libraries) contenant le code objet des fonctions utilisées par le programme (par exemple l'opérateur ≪) et retourne ainsi un code exécutable du programme.

Pour nos programmes, nous utiliserons le compilateur **g++**. Notre programme helloWorld sera compilé par la ligne de commande suivante :

g++ -o helloWorld helloWorld.cpp

L'exécutable final est lancé en tapant ./helloWorld.

2 Exercices du TP1

Exercice 1. Ecrire un programme produit qui saisit un flottant et l'affiche à l'écran.

Exercice 1. Ecrire un programme divisionReste qui saisit deux entiers x et y et affiche

1. la division entière de x par y,
2. le reste de la division de x par y

Exercice 3. Ecrire un programme perimetreCercle permettant d'afficher le périmètre d'un cercle dont le rayon est donné par l'utilisateur.

Exercice 4. Ecrire un programme distanceEntrePoints permettant de calculer la distance entre deux points dont l'abscisse et l'ordonnée sont entrées par l'utilisateur. Utiliser pour cela la fonction `sqrt` de la bibliothèque mathématique.

Exercice 5. Ecrire un programme heuresMinutesSecondes permettant de convertir un nombre entier (représentant une quantité de secondes) et heures, minutes et secondes.

Exercice 6. Ecrire un programme sontEnCollision permettant de déterminer si l'intersection de deux cercles sont en collision (intersection non vide). Pour chaque cercle, le rayon et l'abscisse et l'ordonnée du centre sont entrés par l'utilisateur.

Exercice 7. Ecrire un programme sontParalleles permettant de déterminer si deux segments définis par leurs extrémités sont parallèles. Pour chaque segment, l'abscisse et l'ordonnée de chaque extrémité sont entrés par l'utilisateur.

Exercice 8. Ecrire un programme qui renvoie le plus grand nombre parmi trois entiers.

Exercice 9. Ecrire un programme permettant de convertir les températures : Degré Kelvin ($°K$), Fahrenheit ($°F$) et Celcius ($°C$). Rappelons que

$$T°K = T°C + 273.15$$
$$T°F = (T°C * 1.8) + 32$$

Le programme doit saisir en entrée l'unité de départ, l'unité d'arrivée, et la température de départ, et afficher en sortie la température convertie dans l'unité d'arrivée.

Exercice 10. Ecrire un programme permettant d'afficher la conversion de devises : Euro, Dollar (US), Yen (Japonais) et Livre (Sterling). Utiliser le taux de conversion :

`http://www.boursorama.com/bourse/devises/`

Le programme doit saisir en entrée une devise et une valeur (monnaie), et doit retourner en sortie un tableau donnant la conversion de la monnaie dans les différentes devises. Par exemple, si l'utiliseur entre 1 Euro, le programme affiche :

```
Euros    1
Dollars  1.2954
Yens     136.1050
Livres   0.7932
```

Exercice 11. Ecrire un programme calculette permettant de simuler une calculette pour les nombres entiers : le programme lit en entrée deux nombres entiers et un opérateur arithmétique, et affiche en sortie le calcul de l'opération. Les opérateurs sont $+, -, *, /, \%$.

Exercice 12. Construire un programme solveur permettant de résoudre une équation du second degré :
- Données : les coefficients a, b et c de l'équation $ax^2 + bx + c = 0$
- Résultat : le nombre et la valeur des solutions réelles de l'équation

Exercice 13. Construire un programme factorielle permettant de calculer la factorielle d'un nombre entier.

Exercice 14. Construire un programme PGCD permettant de calculer le PGCD de deux nombres entiers.

Exercice 15. Ecrire un programme qui retourne les 100 premiers nombres *premiers*.

Exercice 16. Construire un programme qui saisit un entier n entré par l'utilisateur et retourne en sortie toutes les puissances de 2 qui sont inférieures ou égales à n. Par exemple, si l'utiliseur entre 9, le programme retourne 1, 2, 4, 8.

Exercice 17. Construire un programme conversionBinaire permettant de convertir un nombre entier en base 2.

Exercice 18. En généralisant l'exercice précédent, construire un programme conversions permettant de convertir un nombre entier en base 10, base 2 et hexadécimal. Le programme doit saisir en entrée le nombre et son unité, et doit retourner en sortie un tableau donnant la conversion du nombre dans les différentes unités.

Exercice 19. Ecrire un programme notationScientifique permettant d'écrire un nombre réel en notation scientifique. Le programme lit en entrée un "double" et le convertit en notation scientifique de la forme $\pm x\ 10^{\mp n}$ où x est un réel dans $[1,10)$ et n est un entier naturel. Par exemple :

$$\begin{array}{lcl} 145 & \rightarrow & 1.45\ 10^{2} \\ 0.00234 & \rightarrow & 2.34\ 10^{-3} \\ -0.40234 & \rightarrow & -4.0234\ 10^{-1} \end{array}$$

Exercice 20. En généralisant l'exercice précédent, construire un programme notations permettant de convernir un nombre réel en notation scientifique et en notation ingénieur. Rappelons qu'un nombre en notation ingénieur est de la forme $\pm x\ 10^{\mp 3n}$ où x est un réel dans $[1,1000)$ et n est un entier naturel. Le programme lit en entrée un "double" et retourne en sortie les deux conversions. Par exemple :

$$\begin{array}{ll} 123456 & \rightarrow \\ \text{Scientifique :} & 1.23456\ 10^{5} \\ \text{Ingénieur :} & 123.456\ 10^{3} \end{array}$$