TD Langage C+
Séances 1 et 2 /8

TD 1 et 2 – Prise en main

Consignes générales :

Pas deux IUT ensembles.

Point de vue du cours : il y a 6 heures de cours où sont détaillés :

- la syntaxe du C++
- les tableaux statiques
- les pointeurs (rapidement…)
- les fonctions
- les fichiers
- débogage

Prise en main : Développer en C sous linux

[1ière séance]

Se logger sous linux. Par défaut, le login est formé de la première lettre du prénom suivi du nom (ex. : ***tgrenier***). Le mot de passe est celui donnée par la DSI.

1-a Présentation rapide de l'environnement linux :

- le terminal permet de taper et exécuter des commandes ou programmes.
- ***gedit***: un éditeur de texte graphique (convivial). Il en existe d'autres en mode graphique (par exemple ***emacs***,…). Il existe aussi des éditeurs de texte en console (***vi*** étant le plus répandu).
- quelques commandes linux/unix :

ls : lister le contenu d'un répertoire, **ls –l** pour une liste détaillée,

cd repertoire: pour changer de répertoire ; ex. : cd /home/tgrenier, **cd ..**

pwd : donne le répertoire courant

man nom_commande : pages de manuel (aide) de la commande. Utiliser les flèches pour se déplacer dans le texte. La touche '**q**' permet de quitter le manuel. Pour chercher du texte faire : « /text_a_chercher ». On passe à l'occurrence suivante (précédente) avec la touche **n** (respectivement : **p**).

1-b Premier programme

Edition dans un éditeur de texte classique. Compilation avec ligne de commande.

Ecrire un programme qui affiche « Hello World » sur 2 lignes.

Correction (fichier **main.cpp**):

```
#include <iostream> // pour cout
using namespace std;

int main(void)
{ cout << "Hello \n";   // \n permet de passer à la ligne suivante
  cout << "   World !!" << endl ; // comme endl !
  return 0;  // 0 : valeur par defaut quand
             // un programme se termine normalement
}
```

Compiler le programme avec la commande (le binaire est l'exécutable) :

```
g++ fichier.cpp –o nom_du_binaire
```

En cas d'erreur toujours remonter à la première erreur détectée et lire tout le descriptif de l'erreur (souvent **3** lignes !).

Quand la compilation réussie, exécuter le programme compilé (le binaire) avec la commande :

```
./nom_du_binaire
```

1-c Lancement de l'environnement de développement *Qt Creator*.
Deux solutions : par le menu « Fedora » ou par commande.

a) Lancement par menu :
   Menu Fedora (en haut à gauche)→ chercher Qt Creator
b) Lancement par la commande (dans un terminal) :
   qtcreator

Cet environnement de développement existe aussi pour Mac et Windows.
Pour faire vos premiers pas avec ce logiciel, il est indispensable de lire – au moins- la partie « Environnement de développement » du document *NoteCpp.pdf* présent sous Moodle (GE → Enseignement → Informatique → IF1 → « Compléments C++ et QtCreator »), notamment pour le démarrage et la création de projet.

1-d Sous *QtCreator*, faire un programme interactif permettant de convertir les degrés Celsius en degrés Fahrenheit. Attention au type des variables (`float` ou `double` et non `int`). On demandera aussi de formater l'affichage des nombres flottants. L'exercice peut ensuite se compliquer pour générer une table de conversion (mise en place d'un `while`).

Prise en main : Développements sous QtCreator

[2ième séance]

2-a Débogage : Télécharger l'archive Exo2a.zip (moodle) et décompresser le fichier source (main.cpp) et le fichier 'projet' (Ex2a.pro). Pour une fois, on vous donne le programme… N'oublier pas de lire la partie consacrée au débogage du fichier NoteCpp.pdf.

Phase 1 : les bases du débogage

1) Ouvrir le projet Ex2a.pro sous QtCreator.
2) Ajouter un *breakpoint* sur la ligne : *tab[9] = 3;*
3) Passer la construction (*Build*) en mode « debug » et lancer le débogage du programme. L'exécution du programme s'arrête au *breakpoint*.
4) Afficher le contenu des variables mémoires (*Windows→Views→Locals and Expression*) et observer le contenu de ***tab.*** Quelle est la valeur de tab[9] ?
5) Exécuter la ligne *tab[9] = 3;* via *Debug → Step Over*. Quelle est la valeur de tab[9] ?

Phase 2 : la fonction

1) Dans la fonction 'functionX', placer un *breakpoint* sur la ligne '*i++ ;*'. Elle correspond à la dernière instruction de la boucle *while*.
2) Observer le contenu du dernier élément du tableau (il faudra double cliquer dans l'espace « locals and Expression » et ajouter le nom -et l'indice- de la variable à observer).
3) A la fin de la première exécution de la boucle *while*, quelle est la valeur du dernier élément du tableau ?
4) Observer maintenant pour chaque itération de la boucle *while*, la valeur de l'élément in[size-i-1]. Donner ces valeurs en fonction de la valeur de *'i'*.

Phase 3 : retour dans le *main*

5) Que fait la fonction 'functionX' ? Observer le contenu de ***tab*** après l'exécution de la fonction.

2-b Premier projet : Faire un programme permettant de trouver les racines des polynômes du second degré :

$$ax^2 + bx + c = 0$$

1) Préparation : Déterminer 3 triplets de valeurs a , b et c et les racines correspondantes vous permettant d'explorer les 3 cas de figures et de valider votre programme.
2) Réalisation :
   - Sous *QtCreator*, créer un nouveau projet C++ non Qt.
   - Ajouter à votre projet un fichier source et un fichier d'entête, nommés ***Solve***. Pour cela, dans *QtCreator*, cliquer avec le bouton droit sur votre projet puis faire « ajouter un fichier », choisir un fichier source C++ (extension .cpp). Comme nom donner ***Solve***. Faire de même pour le fichier d'entête (le .h).
   - Ces deux fichiers sont maintenant créés et ajoutés à votre projet. Vos fonctions de calcul `Determinant` et `Solution` sont à écrire dans ces deux fichiers.
   - Dans le **main.cpp**, après avoir demandé à l'utilisateur les valeurs de *a*, *b* et *c*, via une nouvelle fonction, le programme résoudra l'équation grâce aux fonctions `Determinant` et `Solution` . Une autre fonction se dédiée à l'affichage.

Fonctions à écrire dans les fichiers ***Solve.h*** et ***Solve.cpp*** :

- Ecrire la fonction `Determinant` qui calcul et retourne le déterminant (discriminant) du trinôme. Cette fonction ne doit pas afficher le résultat (le programme principal s'en chargera). Cette fonction sera compatible avec l'appel suivant :

  ```
  double d = Determinant(a,b,c) ;
  ```

- Ecrire la fonction `Solution` qui calcule les racines quelque soit la valeur du déterminant. Cette fonction ne doit pas faire d'afficher (le programme principal s'en chargera). **Cette fonction retournera les parties réelles et imaginaires de chaque racine** x_1 **et** x_2**. L'ordre des paramètres sera le suivant** :

  a, b, c, partie réelle x_1, partie imaginaire x_1, partie réelle x_2, partie imaginaire x_2

Attention : Ajouter la librairie « math » dans le Solve.cpp : ***#include <math.h>, ou #include <cmath>***

Inclure aussi *Solve.h* au début de Solve.cpp et de main.cpp

Rappels :

Calcul du déterminant : $\Delta = b^2 - 4.a.c$

Si $\Delta > 0$, les deux racines x_1 et x_2 sont : $x_1 = \dfrac{-b + \sqrt{\Delta}}{2.a}$; $x_2 = \dfrac{-b - \sqrt{\Delta}}{2.a}$

Si $\Delta = 0$, les deux racines x_1 et x_2 sont égales. $x_1 = x_2 = \dfrac{-b}{2.a}$

Si $\Delta < 0$, les racines x_1 et x_2 sont complexes : $x_1 = \dfrac{-b}{2.a} + \dfrac{\sqrt{-\Delta}}{2.a}\mathbf{i}$; $x_2 = \dfrac{-b}{2.a} - \dfrac{\sqrt{-\Delta}}{2.a}\mathbf{i}$

3) Réalisation avec IHM. Télécharger le projet PolynomeIHM.zip. Ouvrir le projet sous QtCreator et lui ajouter vos fichiers ***Solve.h*** et ***Solve.cpp*** (bouton droit sur le projet, « ajouter fichier existant », on peut préalablement avoir copié les 2 fichiers dans le répertoire du nouveau projet). Exécuter le projet et valider les résultats.