

TP1 C++ – M2 ISIFAR

Utilisation d'Eclipse et remise à niveau

M. de Falco [defalco@pps.jussieu.fr] – Site de Chevaleret, Bureau 5A16

29 septembre 2009

Dans tous les TPs on suivra les indications typographiques suivantes :

- code source; par exemple :
`cout << "Test" << endl;`
- Indication dans les menus; par exemple : Fichier->Ouvrir
- <Touche du clavier ou de la souris>; par exemple : <F3>, <Ctrl-C>, <Clic-Droit>
- Pour bien distinguer les parties qui ne sont que de purs commentaires de celles vous indiquant d'effectuer des actions, ces dernières sont préfixées du symbole •.
- Les parties correspondant à des exercices sont préfixées du symbole *.

1 Premiers pas

1.1 Lancement d'Eclipse

- Pour lancer Eclipse, comme pour tout autre programme sous Windows, double-cliquez sur l'icône sur le Bureau s'il en existe une, sinon cherchez le programme dans le menu Démarrer.
- Au lancement, Eclipse vous propose de choisir un espace de travail : une fenêtre intitulée *Select a Workspace* s'affiche. Choisissez celui par défaut en appuyant sur OK.
- Pour le premier lancement Eclipse peut vous accueillir avec une fenêtre intitulée *Welcome* remplie d'icônes. Cliquez sur celle la plus à droite de la fenêtre et représentant une flèche : *Workbench, Go to Workbench*.

1.2 Création d'un nouveau projet C++

- File->New->C++ Project
- Dans la fenêtre *C++ Project*, écrivez `TP1-1` dans le champ *Project Name*.
- Dans *Project Type*, sélectionnez *Executable* puis *Empty Project*.
- Appuyez sur le bouton *Finish*.

- File->New->Source file

- Entrez `main.cpp` dans le champ *Source File* puis appuyez sur *Finish*.

- Le fichier `main.cpp` a été rajouté à votre projet, et il est ouvert dans l'éditeur. Il contient les lignes :

```
/*  
 * main.cpp  
 *  
 * Created on: 29 sept. 2009  
 * Author: <votre compte>  
 */
```

Ces lignes comprises entre `/*` et `*/` sont des commentaires, elles ne sont pas prises en compte par le compilateur. Il est également possible de rajouter un commentaire à la fin d'une ligne en utilisant `//`, voir exemple ci-dessous.

- Rajoutez les lignes suivantes au fichier :

```
#include <iostream>  
using namespace std;  
  
int main(int argc, char **argv) {  
    cout << "Bonjour!" << endl;  
}
```

- Run->Run ou <Ctrl-F11> vous propose de sauver votre fichier et va ensuite compiler celui-ci, produire un programme exécutable, le lancer et vous afficher le résultat dans le cadre en bas de l'écran intitulé *Console*. Vous devriez voir s'afficher :

```
<terminated> TP1-1.exe [...]  
Bonjour !
```

- Dans la suite cette dernière étape sera abrégée en : Testez votre programme.

1.3 Sortie sur la console avec cout

Dans l'exemple précédent on a utilisé la ligne

```
cout << "Bonjour!" << endl;
```

pour afficher **Bonjour!** sur la console. On va décomposer cette ligne :

1. `cout` est une sorte de conduit vers la console, tout ce que l'on passe dans ce conduit est affiché sur la console, quitte à effectuer des transformations pour le mettre sous une forme affichable.
2. On utilise `<< valeur` pour rajouter la valeur dans le conduit.
3. `endl` est une valeur particulière qui représente un retour à la ligne.
4. Comme toute instruction en C++, celle-ci se termine par le délimiteur `;`.
5. • Remplacez la ligne précédente par

```
cout << "J'affiche le nombre" << 5
<< " et le caractère"
<< 'A' << endl;
```

et testez le programme.

6. Les valeurs en C++ sont de différents types : caractère, entiers, flottant. Le langage utilise en interne des noms pour ces types de données. Pour le moment nous avons vu les types suivants :
 - Les entiers relatifs, de type `int`, comme 5. La valeur du plus grand entier représentable dépend de votre machine, considérez en général qu'il s'agit de $2^{31} - 1 = 2147483647$.
 - Les caractères, de type `char`, comme 'A'. Ce sont en fait des petits entiers relatifs allant de -128 à $+127$.
 - Les chaînes de caractères, de type `char *`, comme "Bonjour!".
7. ★ Écrivez une instruction affichant sur la console :

```
3 + 2 = 5
```

en n'utilisant aucune chaîne de caractère et seulement les caractères '+' et '='. Testez votre programme. Remplacez la valeur 5 par (2+3) et re-testez. Moralité : les valeurs peuvent provenir de calculs.

1.4 Utilisation de variables

1. • Remplacez le corps de la fonction `main`, c'est-à-dire tout ce qui est compris entre `{` et `}` par :

```
int n;
n = 2;
cout << "La valeur de n est : "
<< n << endl;
```

et testez le programme.

2. L'instruction `int n;` déclare une nouvelle variable appelée `n` et pouvant contenir une valeur de type `int`.
3. L'instruction `n = 2;` attribue la valeur 2 à la variable `n`. On peut fusionner ces deux instructions, déclaration et attribution, avec l'instruction suivante : `int n=2;`.
4. Pour utiliser la valeur de `n` on utilise juste `<< n`. Le compilateur se sert du type de `n` pour savoir comment l'afficher.

1.5 Définition et appel de fonction

1. Dans le programme précédent se trouve la définition d'une fonction `main`. Celle-ci est essentielle car c'est la fonction appelée quand votre programme se lance. On reviendra en cours sur la définition complète de cette fonction.

2. Une définition de fonction a toujours la forme suivante :

```
type_de_la_valeur_calculée nom_de_la_fonction
(type_du_premier_argument premier_argument,
...,
type_du_dernier_argument dernier_argument) {
[corps de la fonction]
return [valeur de retour];
}
```

3. • Rajoutez avant la définition de `main` la définition suivante :

```
int somme(int n, int m) {
    return (n+m);
}
```

4. Les arguments peuvent être utilisées comme des variables, ici `n` et `m`, dans le corps de la fonction.
5. • remplacez dans le corps de `main` l'affichage du calcul (2+3) par un appel à votre fonction avec `<< somme(2,3)`.
6. Une fonction peut ne pas renvoyer de valeur, on utilise alors le type de donnée spécial `void` comme type de retour et on omet la ligne finale `return`.
7. ★ Définissez une fonction `afficher_somme` prenant comme arguments deux entiers et affichant la somme de ces entiers sur la console ainsi :

```
La somme de 2 et 3 est 5.
```

Rajoutez dans le corps de `main` un appel à cette fonction avec l'instruction `afficher_somme(2,3);` et testez votre programme.

1.6 Flot de contrôle

1. Jusqu'ici, une fonction est constituée d'une suite d'instructions et son exécution consiste à parcourir séquentiellement cette liste. Le parcours des instructions est appelé le *flot de contrôle* et il existe des instructions permettant de le complexifier.

2. • Remplacez le corps de `main` par

```
int i;
for (i = 0; i < 10; i++) {
    cout << '(' << i << ')' << ', ';
}
cout << endl;
```

et testez le programme.

3. L'instruction `for` répète un bloc d'instructions tant qu'une certaine condition est remplie. Elle est structurée ainsi :

```
for(<instruction d'initialisation> ;
    <condition> ;
    <instruction à itération>) {
    <instructions à répéter>
}
```

4. La plupart du temps on utilisera une boucle `for` pour dire : *pour i allant de m à M faire ...*, et la syntaxe sera alors

```
for (i = m; i < M; i++) { ... }
```

L'instruction `i++` étant synonyme de `i=i+1`. On peut même définir la variable `i` dans l'instruction d'initialisation : `int i = m`.

5. ★ Écrivez une fonction `void boite(int l, int c, char e)` qui va afficher `l` lignes de `c` caractères `e`. Par exemple, `boite(2,3,'#')`; affichera :

```
###
###
```

Testez votre programme en l'appelant depuis `main` avec différents arguments.

6. Une autre instruction très importante est l'instruction `if` permettant d'effectuer des instructions de manière conditionnelle. Si l'on remplace l'exemple du 2 par :

```
int i;
for (i = 0; i < 10; i++) {
    if (i % 2 == 0) {
        cout << '(' << i << ')' << ', ';
    } else {
        cout << '[' << i << ']' << ', ';
    }
}
cout << endl;
```

On va obtenir le résultat suivant sur la console :

```
(0), [1], (2), [3], (4), [5], (6), [7], (8), [9],
```

En effet, l'opérateur `a % b` renvoie le reste dans la division euclidienne de `a` par `b`, le plus petit entier `r` tel que $a = qb + r$. Donc `i % 2` vaut 0 quand `i` est pair et 1 sinon. L'opérateur `==` effectue une comparaison, et la condition correspondante est vraie si les deux membres sont égaux, fausse sinon.

La syntaxe générale d'une instruction `if` est :

```
if (<condition>) {
    <instructions à effectuer
        si la condition est vraie>
} else {
    <instructions à effectuer
        si la condition est fausse>
}
```

La partie `else { ... }` est optionnelle et peut être omise.

7. ★ Inspirez vous de votre fonction `boite` pour réaliser une fonction `echiquier` telle que

```
echiquier(2,3,'#','*');
```

affiche

```
###
###
```

2 ★ Exercices

2.1 Encore des boîtes...

Écrivez des fonctions produisant, entre autres, les affichages suivant :

```
- boite_bordure(3,3,'#*');
***
* *
***
- boite_cadre(3,4,'#','|','-' );
*---*
| |
*---*
- diagonales(3,3,'#*');
* *
 *
* *
```

2.2 Équations du second degré

On introduit un nouveau type de donnée que vous serez amenés à utiliser très souvent : les flottants en précision double, `double`. Ils représentent des nombres rationnels avec une précision importante sur 64 bits.

1. Écrivez une fonction `double discriminant(double a, double b, double c)` qui renvoie le discriminant de l'équation $ax^2 + bx + c = 0$.

2. Écrivez une fonction `void resoudre_eq2(double a, double b, double c)` qui résout l'équation $ax^2 + bx + c = 0$ en affichant les solutions.

Pour calculer une racine carrée il faut ajouter en haut de votre document, après

```
#include <iostream>
```

la ligne

```
#include <math.h>
```

Ensuite vous pourrez utiliser la fonction `double sqrt(double x)` qui renvoie \sqrt{x} quand x est positif ou nul.

3. Testez votre programme. Les appels suivant :

```
resoudre_eq2(5,3,0);  
resoudre_eq2(1,0,0);  
resoudre_eq2(0.5,0,1);
```

produisent sur la console :

```
Résolution de l'équation 5x2 + 3x + 0  
Solution 1 : 0  
Solution 2 : -0.6  
Résolution de l'équation 1x2 + 0x + 0  
Solution double : -0  
Résolution de l'équation 0.5x2 + 0x + 1  
Solution 1 : -0+1.41421i  
Solution 2 : -0+-1.41421i
```

2.3 Temperatures

Ecrivez les fonctions suivantes :

- `int celsius(int t)` prenant une température t en degré Fahrenheit et renvoyant l'équivalent en degré Celsius.
- `int fahrenheit(int t)` prenant une température t en degré Celsius et renvoyant l'équivalent en degré Fahrenheit.

Utilisez ces fonctions pour écrire un programme qui affiche des tableaux illustrant les équivalents en Fahrenheit de toutes les températures en Celsius de 0 à 100 degrés, ainsi que les équivalents en Celsius de toutes les températures Fahrenheit de 32 à 212 degrés.

On rappelle d'ailleurs que 32 degrés F = 0 degrés C et 212 degrés F = 100 degrés C.