TP 2 : premiers programmes

Tout exercice non marqué d'un ♣ est à terminer pour la semaine prochaine.

Exercice 1 (Premier programme `C++` en ligne de commande)**.**

(1) Ouvrez l'éditeur de texte `gedit`. Créez un nouveau fichier texte `bonjour.cpp` que vous enregistrerez dans le dossier `Info111/Semaine2` créé la semaine dernière.

(2) Écrivez dans le ficher le programme `bonjour` vu en cours. **Attention**, le programme que vous écrivez doit respecter la même mise en page que celui du cours (retour à la ligne, espaces, majuscules).

(3) Ouvrez un terminal et utilisez la commande `cd` pour vous rendre dans le répertoire `Info111/Semaine2` où est enregistré votre fichier.

(4) Vérfiez avec `ls` que votre répertoire contient bien le fichier `bonjour.cpp`.

(5) **Compilez le programme** `bonjour.cpp` en tapant la commande suivante sur votre terminal : `g++ bonjour.cpp -o bonjour`. **Attention**, cette commande ne peut fonctionner que si vous êtes dans le bon répertoire !

(6) Si une erreur s'affiche, lisez-la et essayez d'identifier le problème en trouvant le numéro de ligne : avez-vous bien respecté les minuscules et majuscules ? Avez-vous bien suivi les retours à la ligne et espaces tels qu'ils étaient dans le cours ? Avez-vous bien terminé les deux lignes d'instructions par des points-virgules ? Après chaque correction, enregistrez le fichier et relancez la **compilation** en retapant la commande. Rappel : on peut retrouver les dernières commandes tapées en utilisant la touche ↑.

(7) Lancez la commande `ls` ; quels fichiers se trouvent maintenant dans votre répertoire ?

> `g++` : La commande `g++` ⟨`fichierX`⟩ `-o` ⟨`fichierY`⟩ lance le compilateur `C++ gcc` sur `fichierX` et crée un fichier exécutable `fichierY`.

(8) Votre répertoire doit contenir un fichier `bonjour` (sans extension). Exécutez-le en tapant `./bonjour` . Le texte "Bonjour !" doit s'afficher : bravo, vous avez lancé votre premier programme `C++` !

(9) Modifiez le fichier `bonjour.cpp` en remplaçant "Bonjour !" par le message de votre choix. Enregistrez puis lancez la commande `./bonjour` . Que remarquez-vous ?

(10) Comment faire pour que l'exécution du programme reflète le changement effectué dans `bonjour.cpp` ?

Exercice 2 (Un deuxième programme en utilisant un environnement de développement)**.**

Lorsque les programmes deviennent plus importants ou que l'on souhaite configurer le compilateur `g++` avec de nombreuses options, le passage constant par le terminal peut être jugé fastidieux. Ainsi, de nombreux logiciels fournissant des `environnements de développement` (ou `IDE` en anglais) sont proposés aux développeurs.

Cette année, nous utiliserons `Code::Blocks` : c'est un logiciel **libre**, **gratuit** et multiplateforme (`GNU/Linux`, `Windows` et `Mac OS X`).

(1) Lancez l'environnement de développement `Code::Blocks` (par exemple en tapant `codeblocks` dans le terminal).
(2) Ouvrez un nouveau fichier (Menu `File -> New -> Empty file`).
(3) Sauvegardez le fichier dans le répertoire `Semaine2`, sous le nom `mon_programme.cpp` (Menu `File -> Save file as`, puis aller dans le bon répertoire).
(4) Tapez dans ce fichier un programme similaire à `bonjour.cpp` en modifiant la chaîne de caractères à afficher.
(5) Sauvegarder au fur et à mesure, en utilisant le raccourci clavier (`Ctrl S`).
(6) Compilez le programme (Menu `Build -> Build`).
(7) Quel est le raccourci clavier pour compiler ?
(8) Corrigez les éventuelles erreurs de frappe (et dans ce cas re-compilez).
(9) Exécutez le programme (Menu `Build -> Run`).
(10) Quel est le raccourci clavier pour exécuter ?
(11) Avec le terminal, déplacez-vous dans le dossier `Semaine2` et observez les fichiers créés par la compilation avec `Code::Blocks`.
(12) Exécutez depuis le terminal le programme compilé avec `Code::Blocks`.

Exercice 3 (Notes de cours)**.**

(1) Téléchargez les notes de cours dans votre répertoire `Info111`.
(2) Quelle est la syntaxe et la sémantique de l'affectation de variables ?
(3) Que deviennent les blancs placés en tête lors de la lecture d'une variable de type `int` ?

Exercice 4 (À faire « à la maison » pour la semaine prochaine)**.**
Vous aurez besoin de programmer quelques heures par semaine en dehors des séances de TP. Pour cela, vous aurez besoin d'un ordinateur à votre disposition avec tous les outils appropriés, que ce soit un portable, un fixe chez vous, ou tout simplement une des machines en libre service de l'Université.

(1) Installer `Code::Blocks` (`www.codeblocks.org`) sur cet ordinateur s'il n'y est pas déjà installé. Vous trouverez sur la page du cours quelques indications pour l'installation (`http://nicolas.thiery.name/Enseignement/Info111/install.html`). Ne pas hésiter à demander de l'aide à vos camarades.
(2) Refaire l'exercice 2 « à la maison ».
(3) Terminer « à la maison » tous les exercices de cette fiche non marqués d'un ♣.

Exercice 5 (Entrées sorties, variables)**.**

(1) Modifier le programme `mon_programme.cpp` pour qu'il affiche :

```
Bonjour, mon nom est Alice
```

(2) Modifier le programme pour qu'il affiche :

```
Bonjour, mon nom est Alice
Alice est dans la classe
Sans le moindre doute, Alice travaille sur cet exercice
```

(3) Changer le prénom « Alice » en « Gertrude ».

(4) On veut changer le prénom encore une fois. Le problème est que le prénom apparaît à plusieurs endroits dans le programme. Il est pénible de devoir à chaque fois modifier chacun de ces endroits, non ? Imaginez ce que ce serait si le programme faisait 10000 lignes.

Pour éviter cela, introduire au début du programme principal (`main`) une variable « prenom » de type `string` (`string` signifie « *chaîne de caractères* », c'est-à-dire « texte ») avec :

```
string prenom;
prenom = "Claude";
```

Adapter la suite du programme pour que « Claude » n'y apparaisse plus.

(5) Essayer votre programme avec plusieurs prénoms.

(6) Modifier le programme pour qu'il demande à l'utilisateur de saisir le prénom au clavier.

Exercice 6 (Affectations, conditionnelles)**.**

(1) Reprendre la fiche de TD 2, et taper dans un fichier `permutation.cpp` le corps de programme de l'exercice 1, en le complétant en un programme complet (entête, ...). Le compiler et l'exécuter.

(2) Même chose pour l'exercice 2 du TD, dans un fichier `pairOuImpair.cpp`.

Exercice ♣ 7.
Implanter (en `C++`) les algorithmes des exercices 3 et 4 du TD.

Exercice ♣ 8.
Réimplanter les programmes ci-dessus en `Python`. Vous trouverez un cours complet sur `Python` à cette adresse :

`http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-python`

Il est intéressant de le lire et de comparer avec ce que vous apprenez en C++ ce semestre.

Pour utiliser `Python` : depuis un terminal, lancez `ipython`, vous pouvez alors directement écrire vos programmes dans le terminal ou les copier-coller depuis un fichier. Vous pouvez aussi lancer `sage` pour avoir en sus une large bibliothèque d'outils de calcul mathématique.

Un peu de syntaxe `Python` :
Une fonction :

```
def maFonction(argument):
    instructions
```

Une conditionnelle :

```
if a == 1:
    ...
else:
    ...
```

Une boucle :

```
while a <= 10:
    ...
    a += 1
```

Exercice ♣ 9 (Euler forever !)**.**
Vous trouverez sur le site du projet Euler `projecteuler.net` une série de problèmes mathématiques qui nécessitent chacun une combinaison de réflexion sur feuille et de programmation (voir `http://submoon.freeshell.org/fr/sphinx/euler.html` pour avoir les énoncés en français).

Essayez de résoudre les problèmes 1, 2, 5 et tous ceux qui vous plairont !