TP Soutien : Bases de Programmation

En info 121, vous allez écrire vos programmes dans un éditeur de texte, et les compiler et les exécuter en utilisant le terminal (comme cela a été vu en Info 111). L'objectif du premier exercice est de revoir la compilation et l'exécution depuis le terminal.

Exercice 1 (Premier programme compilé en `C++`)**.**

Le *terminal* permet d'interagir avec le système en tapant une à une des commandes. Le programme avec lequel vous interagissez pour exécuter les commandes s'appelle le *shell*. À chaque fois, il vous indique qu'il est prêt en affichant en début de ligne le caractère dollar ($). Vous lui indiquez que vous avez fini de taper votre commande en appuyant sur la touche `Entrée` (`Enter`).

En général, si le terminal n'indique aucun message une fois qu'une commande a été exécutée, c'est que celle-ci s'est bien déroulée. Par contre si le terminal affiche un message, il s'agit probablement d'un message d'erreur. Dans ce cas, lire le message et essayer de comprendre comment résoudre le problème. Si vous ne voyez pas comment faire, demandez à l'enseignant.

(1) Lancer un terminal. Pour cela vous pouvez utiliser le menu principal (bouton en bas à gauche) puis cliquer sur « terminal ».

(2) Créer un répertoire `Soutien` en tapant la commande `mkdir Soutien` dans le terminal (n'oubliez pas d'appuyer sur entrée une fois chaque commande tapée).

(3) Se placer dans ce répertoire `Soutien` en tapant `cd Soutien`

(4) Charger l'archive contenant les fichiers utiles pour faire le TP en tapant `wget https://www.lri.fr/~hivert/COURS/Info121/archive.zip`

(5) Décompresser l'archive en tapant `unzip archive.zip`

(6) Aller dans le répertoire `Soutien1` créé par l'archive en tapant `cd Soutien1`

(7) Taper la commande `ls` pour vérifier que vous avez bien dans votre dossier un fichier nommé `bonjour.cpp`

(8) Ouvrir le fichier `bonjour.cpp` grâce à l'éditeur de texte `jedit` en tapant `jedit bonjour.cpp &`

(9) Lire le contenu du fichier `bonjour.cpp`. Il s'agit d'un programme écrit en `C++`. Essayer de deviner ce que ce programme va faire si on l'exécute, puis passer à la question suivante pour vérifier.

(10) Avant d'exécuter un programme, il faut le compiler. Compiler le programme `bonjour.cpp` en tapant la commande suivante sur votre terminal `g++ bonjour.cpp -o bonjour`
Si tout se passe bien, vous n'avez aucun message d'erreur.

(11) Taper la commande `ls` pour vérifier que vous avez maintenant dans votre dossier un fichier nommé `bonjour` (sans extension) en plus du fichier `bonjour.cpp`. Ce fichier `bonjour` s'appelle un exécutable.

(12) Exécuter le programme `bonjour` en tapant `./bonjour`
Lire ce qu'affiche alors le terminal. Est-ce que cela correspond au résultat que vous attendiez ? Vous pouvez relire le contenu du fichier `bonjour.cpp` pour être sûr d'avoir bien compris.

(13) Modifier le fichier `bonjour.cpp` en remplaçant "Bonjour !" par le message de votre choix. **Enregistrer** le fichier, puis l'exécuter à nouveau avec la commande `./bonjour` Que remarquez-vous ?

(14) Pour que les modifications soient prises en compte, il faut re-compiler le fichier en utilisant comme avant la commande `g++ bonjour.cpp -o bonjour` puis l'exécuter avec `./bonjour` Faites-le et vérifier que vous avez bien les modifications attendues.

Exercice 2 (affichages (`cout`))**.**

(1) Ouvrir le fichier `affichage.cpp` grâce à l'éditeur de texte `jedit` en tapant `jedit affichage.cpp &`

(2) Lire le contenu du fichier `affichage.cpp` et essayer de deviner ce que ce programme va faire si on l'exécute, puis passer à la question suivante pour vérifier.

(3) Compiler le programme `affichage.cpp` en tapant `g++ affichage.cpp -o affichage` puis l'exécuter en tapant `./affichage`
Lire ce qu'affiche alors le terminal. Est-ce que cela correspond au résultat que vous attendiez ? Vous pouvez relire le contenu du fichier `affichage.cpp` pour être sûr d'avoir bien compris.

(4) Modifier le fichier `affichage.cpp` en remplaçant la ligne `cout << x << endl;` par `cout << "x" << endl;` (c'est-à-dire ajouter des guillemets). Essayer de deviner ce que ce programme va faire si on l'exécute, puis passer à la question suivante pour vérifier.

(5) **Enregistrer** le fichier, puis le **compiler** et l'**exécuter** (comme avant). Vérifier que l'affichage obtenu est différent (cela ne doit pas donner la même chose qu'avant la modification). Est-ce que cela correspond au résultat que vous attendiez ? Vous pouvez relire le contenu du fichier `affichage.cpp` pour être sûr d'avoir bien compris.

Exercice 3 (lecture au clavier (`cin`))**.**

(1) Ouvrir le fichier `lecture.cpp` grâce à l'éditeur de texte `jedit` en tapant `jedit lecture.cpp &`

(2) Lire le contenu du fichier `lecture.cpp` et essayer de deviner ce que ce programme va faire si on l'exécute, puis passer à la question suivante pour vérifier.

(3) Compiler le programme `lecture.cpp` en tapant `g++ lecture.cpp -o lecture` puis l'exécuter en tapant `./lecture`
Lire ce qu'affiche alors le terminal et suivre ses instructions. Est-ce que cela correspond au résultat que vous attendiez ? Vous pouvez relire le contenu du fichier `lecture.cpp` pour être sûr d'avoir bien compris.

(4) Reprendre les trois questions précédentes en remplaçant `lecture.cpp` par `lecture2.cpp`

Exercice 4 (conditions (`if`))**.**

(1) Ouvrir le fichier `condition.cpp` grâce à l'éditeur de texte `jedit` en tapant `jedit condition.cpp &`

(2) Lire le contenu du fichier `condition.cpp` et essayer de deviner ce que ce programme va faire si on l'exécute, puis passer à la question suivante pour vérifier.

(3) Compiler le programme `condition.cpp` en tapant `g++ condition.cpp -o condition` puis l'exécuter en tapant `./condition`
Lire ce qu'affiche alors le terminal et suivre ses instructions. Est-ce que cela correspond au résultat que vous attendiez ?

(4) Exécuter une seconde fois le programme en tapant `./condition` mais cette fois arrangez-vous pour que l'affichage final fait par le programme ne soit pas le même que lors de l'exécution précédente (sans changer le programme, simplement en ne rentrant pas le même nombre dans le terminal lors de l'exécution du programme).

(5) Reprendre les trois questions précédentes en remplaçant `condition.cpp` par `condition2.cpp` et cette fois exécutez le trois fois en vous arrangeant pour avoir trois affichages finaux différents lors de ces exécutions.

(6) Ecrivez un programme qui demande à l'utilisateur son âge et qui affiche "adulte" si la personne a plus de 18 ans, "enfant" si elle a entre 2 et 18 ans, "bébé" si elle a entre 0 et 2 ans, et "erreur" si l'âge entré est négatif. Testez votre programme sur plusieurs valeurs pour le vérifier.

Exercice 5 (boucles `for`)**.**

(1) On s'intéresse au fichier `for.cpp`. Comme avec les autres programmes, l'ouvrir avec jedit, le lire, deviner ce qu'il va faire, le compiler, l'exécuter, observer le résultat et vérifier que vous avez bien compris.

(2) Même chose avec `for2.cpp`

(3) Ecrivez un programme qui affiche tous les multiples de 10 compris entre 33 et 88. Testez votre programme pour le vérifier.

Exercice 6 (boucles `while` et `do while`)**.**

(1) On s'intéresse au fichier `while.cpp`. Comme avec les autres programmes, l'ouvrir avec jedit, le lire, deviner ce qu'il va faire, le compiler, l'exécuter, observer le résultat et vérifier que vous avez bien compris.

(2) Même chose avec `do-while.cpp`, mais lorsque vous exécutez ce programme, arrangez vous pour que le programme vous pose plusieurs fois la même question puis s'arrête (sans changer le programme, simplement en tapant des choses adaptées dans le terminal lors de l'exécution du programme).

Exercice 7 (fonctions)**.**

(1) On s'intéresse au fichier `fonction.cpp`. Comme avec les autres programmes, l'ouvrir avec jedit, le lire, deviner ce qu'il va faire, le compiler, l'exécuter, observer le résultat et vérifier que vous avez bien compris.

(2) Enregistrez ce fichier sous un nouveau nom (par exemple `fonction2.cpp`) puis modifiez-le pour faire une fonction qui calcule le minimum et un programme principal qui affiche la moitié du minimum des nombres entrés par l'utilisateur.