université PARIS-SACLAY

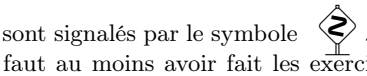
Travaux Pratiques de programmation n°1

Cours de programmation impérative

—Licence MPI L1 S2 - Info 121—

Prise en main / Conditions / Boucles

La présence aux séances de TPs est obligatoire, et l'assiduité sera prise en compte dans l'évaluation pour le contrôle continu. Par ailleurs, deux séances (choisies par votre enseignant) feront l'objet d'une évaluation : nous vous demanderons de remettre à la fin des deux heures un listing contenant votre code, avec des commentaires détaillés.

Enfin, les exercices un peu plus difficiles sont signalés par le symbole  . Pour bien suivre la progression des travaux pratiques, il faut au moins avoir fait les exercices non marqués comme difficiles. Si vous n'avez pas eu le temps de les finir en TP, vous devez les finir par vous-même avant le prochain TP.

Environnement de travail

Pour démarrer le TP, il vous faut télécharger l'archive fournie. Voici la marche à suivre :

- Ouvrir un terminal
- Créer un répertoire `Info121` ☞ `mkdir Info121`
- Se placer dans ce répertoire ☞ `cd Info121`
- Charger l'archive ☞ `wget https://www.lri.fr/~hivert/COURS/Info121/archive1.zip`
  Cette commande étant assez longue, pour éviter les erreurs de recopie vous pouver la sélectionner à la souris, faire un clic droit copier, puis aller dans le terminal et faire un clic droit coller.
- Décompresser l'archive ☞ `unzip archive1.zip`
- Aller dans le répertoire `TP1` créé par l'archive ☞ `cd TP1`
- Vous pourrez maintenant commencer à travailler en éditant le fichier `forme.cpp` tout en gardant la main sur le terminal ☞ `jedit forme.cpp &`

Editeur de texte Vous pouvez utiliser un éditeur de texte de votre choix. Selon votre système d'exploitation, vous pouvez choisir : sous Linux, `jedit, gedit, emacs` et sous Windows, `Notepad++`. Nous conseillons sur les postes à l'université de se connecter sous Linux et d'utiliser l'éditeur `jedit`.

Compilateur Un compilateur est nécessaire pour créer un fichier exécutable à partir d'un code source. Puisque nous développons en `C++`, nous allons utiliser le compilateur `g++`. Il est accessible en ligne de commande (terminal). Normalement, ce compilateur est déjà installé sur votre machine.

La syntaxe à utiliser pour compiler un fichier est la suivante

☞ `g++ -std=c++11 -Wall fichier.cpp -o nom_programme`

- `-std=c++11` : Pour obliger le compilateur à suivre la norme `ISO C++11`
- `-Wall` : Pour afficher les messages d'avertissement (Warnings) durant la compilation
- `fichier.cpp` : Le code source c++ à compiler
- `-o nom_programme` : Pour spécifier un nom à l'exécutable

Exécution du programme Pour lancer un fichier exécutable, il suffit d'exécuter la commande ☞ `./nom_programme`

▸ **Exercice 1. (Êtes-vous en forme ?)**

Voici une manière de déterminer la condition physique d'un individu :
— Après avoir observé un moment de repos, que ce soit assis ou couché, déterminer la fréquence cardiaque appelée F_0.
— Faire ensuite 30 flexions sur les jambes en 45 secondes. Faire attention à ce que les pieds soient à plat sur le sol. S'assoir ou se coucher rapidement puis déterminer la fréquence cardiaque que nous appellerons F_1.
— Une minute après l'effort, déterminer à nouveau la fréquence cardiaque que nous appellerons F_2.
— Effectuer l'opération suivante : $0{,}1 * (F_0 + F_1 + F_2 - 200)$.

Si le résultat est
— inférieur ou égal à 0, la condition est excellente ;
— supérieur à 0 et strictement inférieur à 5, la condition est très bonne ;
— supérieur à 5 et strictement inférieur 10, la condition est bonne ;
— supérieur à 10 et strictement inférieur 15, la condition est moyenne ;
— pour toutes les autres valeurs, la condition est faible.

1. Si ce n'est pas déjà fait, ouvrir le fichier `forme.cpp` avec un éditeur de texte (après avoir extrait l'archive fournie).
2. Compléter la fonction `forme` afin qu'elle évalue la condition physique d'un individu.
   Consigne : On évitera d'écrire deux conditions opposées à la suite comme dans
   ```
   if (r < 0) { ... }
   if (r >= 0 && ...) {  }
   ```
3. Écrire un programme qui demande à l'utilisateur les fréquences F_0, F_1 et F_2 et qui affiche la condition physique de l'utilisateur.
4. On ne le rappellera pas à chaque fois, mais lorsque vous avez écrit un programme, vous devez ensuite le compiler et l'exécuter, et vérifier que son comportement est bien celui attendu. Si ce n'est pas le cas, corrigez votre programme.

▸ **Exercice 2. (Conversion Heures/Minutes/Secondes d'une durée)**

On représente une durée heure, minute, seconde sous la forme d'un vecteur C++ d'entiers de taille 3. Par exemple : 5h 20min et 10s sera représenté par le vecteur `{5, 20, 10}`. Dans cet exercice, on respectera la contrainte suivante : on ne doit utiliser que des calculs sur les entiers.

1. Ouvrir le fichier `convertHMS.cpp` fourni dans l'archive.
2. Écrire une fonction
   ```
   int convertHMS2S(vector<int> hms)
   ```
   qui pour une durée exprimée sous la forme « heures, minutes, secondes » retourne la durée correspondante exprimée en secondes. Ajouter 3 autres tests pour cette fonction.
3. Écrire une fonction
   ```
   vector<int> convertS2HMS(int d)
   ```
   qui pour une durée exprimée en secondes retourne la durée correspondante exprimée sous la forme « heures, minutes, secondes ». Ajouter 3 autres tests pour cette fonction. On utilisera des boucles et des soustractions (pas de division).
4. Écrire une fonction
   ```
   void testHMS(vector<int> hms)
   ```
   qui vérifie qu'une durée exprimée par un vecteur d'entiers est bien une durée sous la forme « heures, minutes, secondes » correcte. Un vecteur d'entiers `hms` contient une heure correcte si il a trois cases, $0 \leq$ `hms[0]` < 24, $0 \leq$ `hms[1]` < 60 et $0 \leq$ `hms[2]` < 60. On écrira les vérifications en utilisant la commande `ASSERT` fournie.
   Note : `ASSERT(condition)` permet de tester n'importe quelle condition. C'est juste un raccourci pour `if (not (condition)) cout << "Test failed ...";`
5. On va tester à grande échelle nos deux fonctions de conversion. Pour ceci, écrire un programme qui pour toutes les durées de 0 à 80000 secondes
   — convertit la durée sous la forme « heures, minutes, secondes » ;
   — vérifie grâce à la fonction `testHMS` que le résultat obtenu est bien correct ;
   — re-convertit la durée en seconde et vérifie (avec `ASSERT`) que l'on retrouve bien la durée initiale.

▸ **Exercice 3. Nombre Médian**

Écrire une fonction qui prend en entrée trois nombres entiers et qui renvoie le nombre médian, c'est-à-dire le nombre qui reste si l'on enlève le plus grand et le plus petit. Par exemple, si les nombres sont `5 3 12` la fonction doit renvoyer `5`. Si les nombres sont `5 12 5`, elle doit renvoyer `5`.

Écrire des tests automatisés tels que ceux des exercices précédents pour votre fonction ainsi qu'un programme de test manuel demandant les trois nombres à l'utilisateur.

▸ Exercice 4. Calendrier

Le but de cet exercice est d'afficher un calendrier en affichant mois par mois les jours de la semaine. Voici par exemple l'affichage du mois de février 2013 qui commence un vendredi :

```
       Fevrier
lu ma me je ve sa di
 .  .  .  .  1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28
```

Pour ceci, on déclare le nom des mois et leur longueur (dans une année usuelle) en C++ de la manière suivante :

```
vector<string> nom_mois = {
    "Janvier", "Fevrier", "Mars", "Avril", "Mai", "Juin", "Juillet",
    "Aout", "Septembre", "Octobre", "Novembre", "Decembre"
  };

vector<int> long_mois = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

Attention : le mois de janvier porte le numéro 0.

1. Dans un premier temps, on suppose que le premier janvier est un lundi. On affichera les jours du mois en revenant à la ligne chaque dimanche sans se soucier des alignements. On sautera une ligne à la fin de chaque mois. On pourra utiliser trois compteurs :
   — `mois` pour stocker le numéro du mois en cours d'affichage ;
   — `j_mois` pour stocker le numéro dans le mois du jour en cours d'affichage ;
   — `j_sem` pour stocker le numéro dans la semaine du jour en cours d'affichage.
2. Écrire un programme qui demande à l'utilisateur le jour du premier janvier sous la forme (0 = lundi, 1 = mardi, 2 = mercredi, etc.) et qui affiche le calendrier de l'année.