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, trois 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.

L'environnement de développement `Code::Blocks`

L'environnement de développement `Code::Blocks` vous permet de :

- Écrire votre code
- Compiler en langage machine
- Exécuter le programme
- Corriger les erreurs (« débugger ») et mettre au point

Pour lancer l'environement `Code::Blocks`, cliquez sur « Démarrer », ensuite sur « Programmes », puis sur « `Code::Blocks` ». Noter que `Code::Blocks` est un logiciel libre et gratuit, que vous pouvez facilement télécharger et utiliser chez vous.

Si vous ne l'avez pas fait au premier semestre, il faut configurer `Code::Blocks` comme suit :

Réglage du compilateur : Menu Settings ↦ Compiler... ↦ Rubrique : Global compiler settings ↦ Onglet : Compiler settings ↦ sous onglet : Compiler Flags ↦ Cocher les cases :

- `Have G++ follows the C++11 ISO C++ langage standard [-std=c++11]`
- `Enable all common compiler warnings (Overide many other settings) [-Wall]`

Réglage de l'éditeur : Menu Settings ↦ Editor ↦ Rubrique : Source formatter ↦ Onglet : Style ↦ Cocher la case : Java

Création de projet S'il est possible de travailler sur un fichier `C/C++` seul sous `Code::Blocks`, pour pouvoir debugger, nous allons devoir créer un projet. On conseille de créer un nouveau projet pour chaque exercice. Pour ceci, on clique sur « Create a new project ». Choisir ensuite « Console application » comme type de projet, puis « C++ » comme langage. Et enfin choisir un répertoire pour stocker le projet.

Pour démarrer le TP, vous pouvez télécharger l'archive `tp1.zip` sur `http://www.lri.fr/~hivert/cours.html`

▸ **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, déterminez votre fréquence cardiaque appelée F_0.

— Faites ensuite 30 flexions sur les jambes en 45 secondes. Faites attention à ce que vos pieds soient à plat sur le sol. Asseyez-vous ou couchez-vous rapidement puis déterminez votre fréquence cardiaque que nous appellerons F_1.

— Une minute après l'effort, déterminez à nouveau votre fréquence cardiaque que nous appellerons F_2.

— Effectuez l'opération suivante : $0{,}1 * (F_0 + F_1 + F_2 - 200)$.

Si le résultat est

— inférieur à 0, la condition est excellente ;

— situé entre 0 et 5, la condition est très bonne ;

— situé entre 5 et 10, la condition est bonne ;

— situé entre 10 et 15, la condition est moyenne ;

— et elle est faible pour toutes les autres valeurs.

1. Extraire le patron de programme `forme.cpp` de l'archive.
2. 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.

▸ **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. Extraire le patron de programme `convertHMS.cpp` de 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> hms convertS2HMS(int)
   ```

   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 (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 $0 \leq$ `hms[0]` < 24, $0 \leq$ `hms[1]` < 60 et $0 \leq$ `hms[2]` < 60. On écrira les vérifications en utilisant la commande `ASSERT` fournie.
5. 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 que l'on retrouve bien la durée initiale.

▸ Exercice 3. Nombre Médian

Écrire un programme qui prend en entrée trois nombres entiers et qui retourne 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 l'utilisateur tape `5 3 12` on doit afficher `5`. S'il tape `5 12 5`, on doit afficher `5`.

▸ 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_ann` pour stocker le numéro du jour en cours d'affichage dans l'année ;
   — `j_mois` pour stocker le numéro du jour dans le mois.
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.