Série 2: buts des séries sur moodle

Lien avec le MOOC Initiation à la Programmation (en C++)
Lien avec ICC-Théorie en complément du MOOC

Le MOOC C++ vous offre normalement tous les éléments pour faire les séries hebdomadaires. Cependant le cours ne se réduit au seul MOOC. C'est pourquoi chaque semaine vous trouvez sur moodle une série qui fournit :
- (optionnel) des éléments de méthode de travail, comme dans cette série (pages 1-2).
- les liens vers les exercices du MOOC qui sont recommandés pour la semaine courante
- (optionnel) des exercices complémentaires orientés ICC-théorie
- (optionnel) des exercices complémentaires orientés MT & EL (plus proche de la machine)

Informations sur le niveau des exercices : 0-1-2-3

Niveau 0 : les **tutoriels** du MOOC, ou les exercices de niveau 0 (quand le MOOC sera fini), reprennent pas à pas un exemple élémentaire et l'expliquent en détail. Ces exercices sont prévus pour vous faire assimiler le cours à votre rythme, en particulier aux débutants. Ils sont en général dans un document séparé.

Niveaux 1 à 3 : ensuite, les exercices du MOOC et de moodle offrent un thème et un niveau de difficulté (de 1 = facile à 3 = avancé). Pratiquez donc en priorité les thèmes avec lesquels vous vous sentez le moins à l'aise. De même, si vous êtes doué pour la matière, attaquez directement les exercices niveau 2 et 3. Si par contre vous êtes en difficulté, focalisez-vous sur les niveau 1 et 2 puis revenez plus tard sur le niveau 3.

Notez que les niveaux sont déterminés en fonction du moment où la série est prévue... il est clair qu'un même exercice donné plus tard au cours de l'année (par exemple au moment de l'examen) serait considéré comme beaucoup plus facile !

La totalité de la série n'est pas prévue pour être finie en 2 heures ! La série est à considérer comme un regroupement de plusieurs exercices, un peu comme un chapitre de livre d'exercices, où chacun peut choisir (à l'aide des indications ci-dessus) ce qui lui semble adapté à sa progression.

Pour finir, je vous conseille également, en plus des 2 heures d'exercices hebdomadaires à l'emploi du temps, de travailler par vous-même chez vous de façon régulière en guise d'exercices personnels. Ce travail supplémentaire devrait en moyenne correspondre à 2 à 3 heures hebdomadaires.

Préliminaire : organisation de votre travail dans le répertoire `myfiles` (compte myNAS)

La **série1** vous a décrit comment créer dans votre répertoire **`myfiles`** un nouveau répertoire **`Programmation`** avec d'une commande **`create_prog`**.

Cette étape est nécessaire pour pouvoir faire les séries suivantes car ce répertoire **`Programmation`** permet de créer le code exécutable de vos programmes.

Chaque semaine, à l'intérieur du répertoire **`Programmation`**, commencez par créer un sous-répertoire destiné à contenir les fichiers de la série. Cette semaine, donc, créez le sous-répertoire **serie02** puis déplacer vous dans ce répertoire pour travailler.

A propos…

de votre approche de résolution des problèmes ...

Le cours ICC-théorie met l'accent sur l'importance de la première phase de résolution *papier-crayon* d'un problème pour mettre au clair une ébauche de la *structure* de votre solution.

Cette ébauche, écrite sous forme de *pseudocode*, permettra d'évaluer *votre* stratégie de résolution sous l'angle des performances (calcul ou mémoire) comme nous le verrons dès la semaine prochaine.

Cette semaine vous n'avez pas encore les outils conceptuels pour le faire et les exercices sont de toute façon trop simples pour que cela présente un intérêt. Ça reviendra dès la série03.

---

de votre style d'écriture de code ...

Ce qui s'écrit lisiblement se comprend facilement

Les exemples du MOOC et du cours suivent certaines règles de présentations que nous vous demandons de suivre.

La première règle de présentation à suivre impérativement dès cette série02 est ***l'indentation*** qui impose de ***décaler le code vers la droite de 2 ou 4 caractères (selon votre préférence) à l'intérieur de la fonction main()***. Nous verrons bientôt en cours les autres contextes où cela s'applique aussi. Nous fournirons prochainement un résumé des conventions à respecter pour notre cours.

Remarque : ces conventions ne sont pas obligatoires du point de vue de la syntaxe du langage C++. Le code peut quand même compiler et produire un exécutable. Par contre c'est difficile de comprendre efficacement comment il fonctionne pour vous et toute personne qui veut vous aider ou qui doit noter votre travail. Voyez ces convention comme un outil pour trouver certains types de bug ; au bout du compte elles vous ferons gagner un temps précieux. S'il vous faut un argument supplémentaire pour vous convaincre, la lisibilité du code de votre projet sera notée.

---

des bugs ...

Tout le monde fait des erreurs, *l'erreur est humaine*, lors de la mise au point d'un programme, cela va de la faute de frappe qui empêche le compilateur de produire un exécutable jusqu'à l'erreur de conception qui produit un résultat faux.

Il est tout à fait normal qu'une part importante de votre temps soit consacrée à la traque de ces erreurs, familièrement appelées des "***bugs***". Un des objectifs du cours est de vous donner une méthode de travail pour être plus efficace dans la correction des bugs. Rappelons seulement cette distinction importante :

- **Erreur syntaxique** : échec à produire un exécutable car non-respect de l'orthographe et de la grammaire du C++. On dispose des messages d'erreurs du compilateur pour nous guider.
- **Erreur sémantique** : l'exécution ne donne pas le résultat attendu. Relire la donnée et son code. Faire des tests, des tests et encore des tests.

La ***première vidéo du MOOC*** vous montre les bons réflexes à avoir avec **geany** en matière de correction de bug. Une premier résumé des bugs fréquents (semestre1) est maintenant disponible sur moodle.

Exercices du MOOC

- **Rappel MOOC : comment configurer geany pour travailler avec C++11**
  - Dans le menu de la commande « **build** »/ « construire » de geany, choisir la dernière rubrique « **Set build commands** » / « Définir les commandes de construction »
  - Modifier la deuxième commande = « **Build** » / « Construire » qui lance le compilateur **g++**. Il faut ajouter, après l'option `-Wall` et avant l'option `-o`, l'option supplémentaire tout-attachée : `-std=c++11`
  - Ensuite valider
  - Si on ne le fait pas on obtient un message d'erreur pour les nouveaux éléments du C++ introduits avec la version C++11, comme par exemple `consteprx`

- **Document Tutoriel « Calcul de l'IMC » niveau 0 :**
  - Les entrées/sorties de base : premier dialogue avec un programme

- **Document Exercices semaine 1 du MOOC**

  - **Exercice 1 : Ages du capitaine (niveau 1)**
    - Même type d'exercice que le tutoriel

  - **Exercice 2 : fondue (niveau 1)**
    - Plus complet que le précédent ; définition d'une constante

  - **Exercice 3 : variables (niveau 2 pour le début et 3 pour l'explication)**
    - Comparaison des résultats d'opérations arithmétiques avec des variable de types variés

- **Document complémentaire du MOOC « Limitations des types entiers et réels »**
  - Ce document offre un lien parfait avec le cours ICC-théorie de cette semaine. Essayez les programmes fournis en exemple, surtout celui sur les réels. Au lieu de la valeur 37 mettez la valeur 0.01 pour la variable **a** car cela correspond à l'exemple vu en cours sur les arrondis.