Travaux Pratiques 1: Algorithme et programmation structurée en C appliquées aux systèmes numériques (Prise en main)

Responsable: John SAMUEL

Année: 2018-2019

Contact: john.samuel@cpe.fr

Lisez bien tout le document avant de commencer à programmer.

Instructions

1. Tous les travaux pratiques sont basés sur les aspects que vous avez déjà appris pendant vos cours. Utilisez bien vos supports cours. Pour votre premier travail pratique, vous avez des slides sur e-campus.
2. Travailler en binôme. N'oubliez pas de numéroter vos groupes (groupeA01, groupeB01,.. etc.)
3. Il est obligatoire de citer toutes les sources (e.g., internet, groupes)
4. Les séances sont encadrées par 2 enseignants.
5. Ne pas utiliser des bibliothèques externes.

Evaluation

1. Les travaux pratiques correspondent à 40% de votre note finale
2. Chaque exercice est noté. Total points pour tous les exercices: 20
3. Rendu en ligne sur e-campus.
4. Chaque question a un niveau de difficulté
    a. ★ : Facile
    b. ★★ : Difficulté moyenne
    c. ★★★ : Difficile

Rendus

1. Il y a deux parties de rendu: rapport d'auto-évaluation et les sources
2. Votre dossier de rendu doit contenir les fichiers suivants:
    a. `README`: rapport d'auto-évaluation
    b. `CONTRIBUTORS`: Noms et prénoms de contributeurs
    c. `src/` : les sources
3. Votre rendu doit renommer comme groupe[ABCD]NN, où NN est le numéro de votre groupe (e.g., groupeA01, groupeB01 etc.).
4. Ecrivez `README` et `CONTRIBUTORS` en format markdown.
5. Le contenu de README rapport d'auto-évaluation :
    a. Objectif: Quel est l'objectif de votre projet? (Remplissez ça après votre premier exercice)

b. Exercice N (N: [1..7])
   i. Fichiers: Noms de fichiers
   ii. Bibliothèques: les bibliothèques standards (e.g., stdio.h)
   iii. Références: les URLs, les numéros de groupes
   iv. Difficulté: niveau de difficulté (facile, moyenne, difficile)
   v. Commentaires (optionnels): remarques etc.

Exercice 1 (1 point)

Objectifs

Ecrire, compiler et exécuter les programmes C.

Instructions

1. ★ Créer un fichier `bonjour.c` et écrire un programme qui affiche “bonjour le monde!” à l’écran. Compiler ce fichier en utilisant gcc et exécuter le code.
2. ★ Ecrire un programme `circle.c` qui calcule l’aire et le périmètre d’un cercle
   a. l’aire: Utiliser une variable rayon : float ou double
   b. le périmètre: Utiliser une variable rayon : float ou double
   c. Compiler `circle.c` et créer un fichier éxecutable nommé circle
   d. Exécuter ‘`circle`’
3. ★ Ecrire un programme `sizeof_types.c` qui affiche la taille de différents types de base (en octets) : char, short, int, long int, long long int, float, double, long double. Ne pas oublier d’utiliser les versions signées et non-signées. Tester le programme (compiler et exécuter).
4. ★ Ecrire un programme `variables.c` qui affecte et affiche les valeurs des variables de différents types de base : char, short, int, long int, long long int, float, double, long double. N’oublier pas d’utiliser les versions signées et non-signées. Tester le programme.
5. ★★ Ecrire un programme `operators.c` qui utilise deux variables a = 16 et b = 3 et tester les différents opérateurs arithmétiques et logiques
6. ★★ Ecrire un programme `boucles.c` qui utilise `for`, # et * et qui affiche un triangle rectangle. La taille du triangle est dependent de la valeur de count (count < 4 inacceptable). Exemple, si count = 5, le programme affiche

   *

   * *

   * # *

   * # # *

   * * * * *

   ….

   Tester le code avec différentes valeurs de count. Ecrire une nouvelle versions du code en utilisant `while` ou `do..while`.

7. ★★ Ecrivez un programme `operators2.c` qui utilise trois variables num1 (entier), num2 (entier) et op (un caractère). La variable c contient un de ces différents opérateurs. (+, -, *, /, %, &, |, ~). Utilisez `switch` et réutiliser le code de votre premier exercice. Si c'est égal à '+', le programme fait l'addition de deux variables num1 et num2, si c est égal à '&', le programme fait l'opération ET etc. Rappelez-vous bien que l'on ne peut pas utiliser tableau de caractères comme condition en `switch`. Testez votre programme avec différents valeurs de num1, num2, op
8. ★★ Ecrivez un programme `conditions.c` qui utilise les boucles (`for`, `while` ou `do..while`) et les branchements inconditionnels (break ou continue) pour l'affichage de numéros <= 100 qui sont divisés par :
    a. 2 et 15
    b. 103 ou 107
    c. 7 ou 5, mais pas par 3
9. ★★★ Ecrivez un programme `binary.c` qui utilise `for` pour l'affichage d'une variable int en format binaire. Rappelez-vous bien que `printf` n'a pas de code de conversion comme x (l'affichage d'un numéro en notation hexadécimale) ou o (l'affichage d'un numéro en notation octale) pour l'affichage en notation binaire. Testez votre code avec les 5 numéros suivants : 0, 4096, 65536, 65535, 1024
10. ★★★ Imaginez que vous gérez les notes de cinq étudiants. En utilisant seulement des tableaux, écrivez un programme `etudiant.c` qui déclare, initialise et affiche les détails de ces cinq étudiants. Pour chaque étudiant, on est intéressé par son nom, son prénom, son adresse, et ses notes dans 2 modules (Programmation en C, Système d'exploitation).

Exemple

```
1. CONTRIBUTORS
   1. NOM Prénom
   2. NOM Prénom
2. README
   L'objectif …
   * Exercice 1
    * Fichiers: bonjour.c,…
    * Bibliothèques:
     * stdio.h
     * ..
    * Références:
     * groupe..
     * http://www.example.com
     * ..
    * Difficulté: ..
    * Commentaires
     * commentaire 1
     * commentaire 2
     * ..
```