Précisions pour l'examen

Programmation séquentielle en C, 2024-2025

Orestis Malaspinas (A401)

2024-11-05

Informatique et Systèmes de Communication, HEPIA

Administration

- L'examen se déroule sur `nexus`.
- L'examen dure au plus 4h (il est prévu pour 3 exceptions jusqu'à 4).
- Votre code devra être dans le répertoire prévu à cet effet.
- Chaque exercice dans un répertoire (`ex1`, `ex2`, `ex3`, ...).
- Compilation
  - Il est impératif de compiler avec les warnings.
  - Vous devez être capables d'écrire la ligne de compilation de votre code:

```
gcc -o ex1 ex1.c -Wall -Wextra -pedantic
```

Documents

- On fournit:
  - Les slides d'algorithmique.
  - Les slides de programmation séquentielle.
- Vous pouvez avoir:
  - Une feuille A4 recto-verso ***manuscrite***.

Les exercices

- Deux ou trois parties à chaque énoncé:
  - Une partie “théorique” qui décrit les structures de données et fonctionnalités.
  - Une partie “technique” qui propose un exemple d’exécution de votre programme avec entrées et sorties.
  - Une partie “exemple” (pas obligatoire) où d’autres exemples sont donnés afin de tester l’exécution de votre programme.
- Votre code doit avoir **exactement** le comportement des exemples donnés **sous peine de sanctions**.
- Chaque code doit être **dans un unique fichier .c** (`ex1.c`, `ex2.c`, …).
- Donc **inutile** et **interdit** d’écrire d’autres fichiers

Évaluation (technique)

- L'évaluation se base surtout sur des critères de fonctionnement:
  - le code compile-t-il? (on regarde même pas si la réponse est non)
  - s'exécute-t-il? (on regarde un peu)
  - le code demandé est-il réalisé?
  - donne-t-il des résultats corrects?
  - votre code possède-t-il des comportements indéfinis?
- Si vous laissez des warnings vous serez pénalisé · e · s.

Évaluation (style)

- Le code est-il joli?
  - Présentation (indentation cohérente, variables nommées de façon raisonnable).
  - Modularité (utilisation de fonctions).
  - Pas de variables globales inutiles.
  - Utilisation de boucles, structures de contrôle, struct, ...
  - Fonctions récursives, ...

Les exemples

- Chaque exemple contient:
  - un input à rentrer à la ligne de commande (on peut copier-coller l'input de l'énoncé).
  - un output à écrire dans le terminal (on peut comparer la sortie avec celle de l'énoncé).
- La structure de l'input doit être **exactement** la même que celle décrite dans l'énoncé.
- Pour l'input il faut impérativement utiliser `argc` et `argv[]`.
- L'output de vos exercices doit être celui de l'énoncé.

Et maintenant place à des exemples (simplifiés)!

Exercice 1

Ce programme prend en argument deux entiers et affiche leur somme sur une nouvelle ligne.

Exemple

```
./prog 12 19

31
```

Exercice 2

Ce programme prend en argument un nombre entier `N` puis des nombres à virgule flottante. Il y aura exactement `N` nombres à virgule flottante. Ensuite, votre programme multiplie chaque nombre à virgule flottante par deux et affiche la somme sur une nouvelle ligne suivi de CHF.

Exemple

```
./prog 12 12.2 45.5 1.5 65.1 89.4 567.6 112.8 67.0 35.1 112.2 3.3 9.8

2243.000000 CHF
```

Exercice 3

Ce programme prend en argument 2 chaînes de caractères (longueur max de 80), les sépare au milieu et retourne les 4 chaînes chacune sur une nouvelle ligne (si la longueur N est paire on sépare en 2 chaînes de longueur N/2, sinon la première aura une longueur de N/2 et la seconde N/2+1).

Exemple

```
./prog abcdefgh asdfghjkl

abcd
efgh
asdf
ghjkl
```