CS-119(a) – ICC-C Série 4

2024-03-12

Exo1 Diviseurs

On lit depuis l'entrée standard un entier strictement positif. On aimerait afficher tous ses diviseurs. Par exemple, si on rentre `13`, on devrait voir s'afficher à l'écran

```
Les diviseurs de 13 sont 1 et 13
```

et si on rentre `110`, on devrait voir

```
Les diviseurs de 110 sont 1, 2, 5, 10, 11, 20, 22, 55 et 110
```

Exo2 `strlen`

On lit avec `scanf` une chaîne de caractères (sans espace) de longueur au plus 100. Pour ce faire, utilisez le marqueur `"%100s"` qui indique à la fonction `scanf` de lire au plus 100 caractères, et definissez votre variable de type `char[101]`.

Déterminez la longueur de cette chaîne et imprimez-la!

Par exemple, pour `Bonjour!` on devrait afficher

```
Vous avez entré la chaine "Bonjour!" de longueur 8.
```

Indice : la chaîne se termine toujours avec le caractère de fin de chaîne `'\0'`.

Exo3 La croissance

(Inspiré de Advent of Code 2021)

Votre chef veut un rapport sur la croissance du ARR (Anual Recurring Revenue). Il fait des économies, et donc vous n'avez pas de licence pour Excel. Il vous donne un fichier texte qui a sur la première ligne un entier `N` représentant le nombre d'années que doit couvrir cette étude (plus petit que 100). Sur la deuxième ligne il y a `N` entiers séparés par des espaces représentant le ARR des `N` années successives.

Vous devez compter combien de fois il y a eu une augmentation du ARR par rapport à l'année précédente.

Par exemple, si on vous donne le fichier

```
10
4 3 6 5 7 13 2 3 1 1
```

vous devez afficher `4` car il y a quatre valeurs qui sont plus grandes que les valeurs précédentes : `6` (précédé par `3`), `7` (précédé par `5`), `13` (précédé par `7`), et `3` (précédé par `2`).

Souvenez-vous que pour lire des valeurs d'un fichier texte il suffit d'utiliser la redirection de l'entrée :

```
./exo < fichier.txt
```

Dans votre code utilisez tout simplement `scanf` sans intercaler des messages pour l'utilisateur avec `printf`. Affichez seulement le résultat à la fin du programme.

Pour tester votre code écrivez vous-même quelques fichiers texte d'entrée. Utilisez VS Code, **pas** un processeur de texte comme Word - les fichiers produits par Word ne sont pas des fichiers texte...

Exo4 PGCD ?

Étant donnés deux nombres entiers positifs, Bob veut calculer leur PGCD. Il ne se souvient plus de l'algorithme d'Euclide, mais il sait calculer les diviseurs d'un nombre grâce à un exercice de la série du cours ICC. Il aimerait du coup que vous lui écriviez un autre programme qui prend en entrée deux listes **triées** de diviseurs et retourne le plus grand diviseur qui se trouve dans les deux listes.

Les deux listes sont données dans un fichier texte que vous pouvez lire en utilisant la redirection de l'entrée standard.

Le fichier contient d'abord le nombre `N` de diviseurs du premier nombre. S'ensuivent `N` entiers dans l'ordre croissant - les diviseurs du premier nombre. Vous pouvez supposer que `1` est toujours le premier élément de cette liste. Ensuite vient `M` le nombre de diviseurs du deuxième nombre suivi des `M` diviseurs du deuxième nombre. Pareil, le premier élément est toujours `1` et les diviseurs sont donnés dans l'ordre croissant.

Par exemple, si les deux nombres sont `200` et `110`, le fichier contient

```
9
1 2 5 10 11 20 22 55 110
12
1 2 4 5 8 10 20 25 40 50 100 200
```

On suppose qu'on ne traite jamais plus de 200 diviseurs.

Vous devez trouver dans ces listes l'élément le plus grand qu'elles ont en commun. Pour l'exemple ci-dessus ce nombre est bien sûr `20`.

Après avoir lu les deux listes, est-ce que vous arrivez à calculer ce PGCD avec une seule boucle ?