

# 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

### Solution de l'exercice 1

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int n;
6     printf("Entrez un nombre entier : ");
7     scanf("%d", &n);
8
9     printf("Les diviseurs de %d sont ", n);
10    for (int i = 1; i < n; i++)
11    {
12        if (n % i == 0)
13        {
14            printf("%d, ", i);
15        }
16    }
17    printf("et %d\n", n);
18 }
```

## 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 définissez 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 chaîne `"Bonjour!"` de longueur 8.

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

### Solution de l'exercice 2

```
1 #include <stdio.h>
2
3 int main()
4 {
5     const int longueur_max = 100;
6     char str[longueur_max + 1];
7
8     printf("Entrez une chaîne de caractères : ");
9     scanf("%100s", str);
10    int longueur;
11    for (longueur = 0; longueur <= longueur_max; longueur++)
12        if (str[longueur] == '\0')
13            {
14                printf(
15                    "Vous avez entré la chaîne \"%s\" de longueur %d\n",
16                    str, longueur);
17                break;
18            }
19 }
```

## Exo3     **La croissance**

(Inspiré de Advent of Code 2021)

Votre chef veut un rapport sur la croissance du ARR (Annual 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...

### Solution de l'exercice 3

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int n;
6     scanf("%d", &n);
7
8     int num_increase = 0;
9     int current_year_arr, last_year_arr;
10    scanf("%d", &last_year_arr);
11    for (int i=1; i<n; i++) {
12        scanf("%d", &current_year_arr);
13        if (current_year_arr > last_year_arr) {
14            num_increase++;
15        }
16        last_year_arr = current_year_arr;
17    }
18    printf("%d\n", num_increase);
19 }
```

### 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 écrivez 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 ?

#### Solution de l'exercice 4

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int m, n;
6     int div1[200], div2[200];
7
8     scanf("%d", &m);
9     for (int i = 0; i < m; i++)
10    {
11        scanf("%d", &div1[i]);
12    }
13
14    scanf("%d", &n);
15    for (int i = 0; i < n; i++)
16    {
17        scanf("%d", &div2[i]);
18    }
19
```

```

20  int pgcd = 1;
21  int i = 0, j = 0;
22  while (i < m && j < n)
23  {
24      if (div1[i] < div2[j])
25      {
26          i++;
27      }
28      else if (div1[i] > div2[j])
29      {
30          j++;
31      }
32      else
33      {
34          // On a trouvé un nouveau diviseur commun
35          // car div1[i] == div2[j]
36          // et c'est le plus grand trouvé jusqu'à présent
37          // car les deux tableaux sont triés
38          pgcd = div1[i];
39          i++;
40          j++;
41      }
42  }
43  printf("%d\n", pgcd);
44  }

```