

La Porte du Louvre [lp07] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel  algoprogram  UNIVERSITÉ HAUTE-ALSACE Version 17 mai 2018

Table des matières

1	Porte du Louvre / pgfserrure	2
1.1	Énoncé	2
1.2	Itératives imbriquées	2
1.3	Une seule itérative	3
1.4	Validation	4
2	Références générales	5

C++ - Porte du Louvre (Solution)



Mots-Clés Structures répétitives ■

Requis Structures de base, Structures conditionnelles, Structures répétitives ■

Difficulté ●●○ (20 min) ■



Objectif

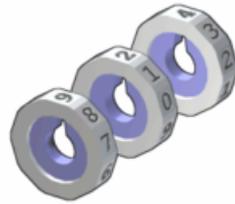
L'exercice détermine par force brute toutes les combinaisons valides d'une serrure à combinaison à trois disques rotatifs à 10 chiffres (0 à 9).

...(énoncé page suivante)...

1 Porte du Louvre / pgfbserrure

1.1 Énoncé

Un programme de type jeu d'aventure propose au joueur l'énigme suivante : il s'agit d'ouvrir une porte secrète verrouillée par une serrure à combinaison. Une telle serrure se compose d'un cadran avec trois disques rotatifs à 10 chiffres (0 à 9) comme illustré sur la figure ci-après. Au moment où le joueur découvre le mécanisme, il dispose d'un ou plusieurs indices. (image : Disques rotatifs (Wikipedia-Combination-discs))



Objectif

Rechercher **toutes** les combinaisons de la porte secrète par **force brute**.

Force brute

Technique qui consiste à passer en revue toutes les possibilités et d'effectuer les tests systématiquement.

Chiffre

C'est un entier dans [0..9].

1.2 Itératives imbriquées

Pour la recherche des combinaisons, une première méthode consiste à utiliser des **itératives imbriquées** pour passer en revue tous les nombres de 0 à 999 :

- Pour les chiffres de la centaine de 0 à 9
 - Pour les chiffres de la dizaine de 0 à 9
 - Pour les chiffres de l'unité de 0 à 9
 - Tester la combinaison (centaine, dizaine, unité)



Écrivez un programme de sorte qu'il traverse tous les nombres de 0 à 999 en utilisant **trois** répétitives imbriquées **Pour** de compteurs **c** (centaine), **b** (dizaine) et **a** (unité).



Les indices de la porte secrète sont :

- La somme des chiffres du code est égale au nombre de signes du zodiac (= 12).
- Les chiffres de la centaine et de la dizaine sont divisibles par 3.

Soient donc :

- c le chiffre de la centaine d'une combinaison,
- b celui de la dizaine,
- et a celui des unités.

Quelles sont les conditions en terme de a , b et c ?



Par conséquent, **complétez** la boucle interne afin de tester et afficher les bonnes combinaisons au fur et à mesure.



Testez (douze solutions).

Résultat d'exécution :

```
==> 039
==> 066
==> 093
==> 309
==> 336
==> 363
==> 390
==> 606
==> 633
==> 660
==> 903
==> 930
FIN
```

1.3 Une seule itérative

Une autre méthode consiste à utiliser **une seule itérative** :

- Pour les entiers de 0 à 999
 - Récupérer les chiffres de la centaine, la dizaine et l'unité.
 - Tester la combinaison (centaine, dizaine, unité)



Soit n un entier.

- Quelle est l'action d'une division par 10 sur n ?
- Et celle du modulo ?

Exemple : n vaut 395.

Solution simple

Une division par 10 a pour action de décaler les chiffres de n d'un cran vers la droite. Le modulo par 10 a pour action d'extraire le chiffre de droite de n . Par exemple, si n vaut 395 alors le modulo par 10 donne 5 et la division entière par 10 donne 39.



Complétez votre programme de sorte que, utilisant **une seule** répétitive de compteur n :

- Il calcule les entiers a (unité), b (dizaine) et c (centaine).
- Puis il teste et affiche les bonnes combinaisons.

Aide détaillée

D'après la question ci-dessus, on a :

- L'unité est le **chiffre de droite**, c.-à-d. le modulo de n par 10.
- La dizaine est le **chiffre de droite** de $(n \text{ div } 10)$.
- La centaine est le **chiffre de droite** de $(n \text{ div } 100)$.



Testez. (Les mêmes douze solutions ci-dessus.)

Notez que, selon l'affichage de « $c b a$ » ou de « n », les zéros non significatifs sont ou ne seront pas affichés. Dans le second cas, 039 s'affiche en 39.

1.4 Validation



Validez votre programme avec la solution.

Solution C++ @[pgfbserrure.cpp]

```
#include <iostream>
using namespace std;
const int SIGNES_ZODIAC = 12;

int main()
{
    for (int c = 0; c <= 9; ++c)
    {
        for (int b = 0; b <= 9; ++b)
        {
            for (int a = 0; a <= 9; ++a)
            {
                bool c1 = (a + b + c == SIGNES_ZODIAC);
                bool c2 = (c % 3 == 0) && (b % 3 == 0);
                if (c1 && c2)
                {
                    cout<<"==>"<<c<<b<<a<<endl;
                }
            }
        }
    }
    cout<<"FIN"<<endl;
    for (int n = 0; n <= 999; ++n)
```

```
{
  int a = n % 10;
  int b = (n / 10) % 10;
  int c = (n / 100) % 10;
  bool c1 = (a + b + c == SIGNES_ZODIAC);
  bool c2 = (c % 3 == 0) && (b % 3 == 0);
  if (c1 && c2)
  {
    cout<<"==>"<<n<<endl;
  }
}
cout<<"FIN"<<endl;
}
```

2 Références générales

Comprend [Rousselet-PY :c8 :ex8] ■