Cours de C++, 1ère année, HE-Arc

Serie 1: De C à C++

Exercice 1: Affichage

- Compléter le code suivant :

```
bool isNumber = true;
std::cout <<  isNumber << " " << !isNumber << std::endl;
std::cout <<  isNumber << " "  << !isNumber << std::endl;

int x = 15;
std::cout  << x << " " << x << " " << x << std::endl;
std::cout  << x << " " << x << " " << x << std::endl;

double dbl = -5345.123456789;
std::cout << dbl << std::endl;
std::cout << dbl << std::endl;
std::cout << dbl << std::endl;
std::cout << dbl << std::endl;
```

de manière à obtenir le résultat suivant:

```
true false
1 0

hexadecimal: f decimal: 15 octal: 17
15 15 15

-5345.12
-5345.12345679
-5.3451e+003
-5345.12
```

Exercice 2: Surcharge de fonctions

1. Écrire deux fonctions `minimum()` qui retournent le plus petit des 2 nombres passés en arguments. La première utilise des nombres entiers (`int`) et la seconde des nombres décimaux (`double`).

2. Laquelle de ces fonctions sera appelée si on lui passe les paramètres suivants :

   `minimum(7, 3)`, `minimum(7.2, 3)`, `minimum(7, 3.1)` et `minimum(7.2, 3.1)` ?

- Que se passe-t-il si on supprime la méthode qui reçoit deux entiers en arguments ?

Exercice 3: Passage de paramètres

- Écrire deux fonctions `divide(dividend, divisor, remainder)` qui à partir de deux entiers, passés en paramètre, doivent retourner le **quotient** et le **reste** de la division entière.

  Par exemple: 13 divisé par 2 donne un quotient de 6 et un reste de 1.

- Tester les différentes possibilités de passage de paramètres en C++. Lesquelles vous semblent les plus appropriées à ce problème ?

Exercice 4: string

- Ecrire un programme qui: utilise une fonction `askForAString()` demandant à l'utilisateur de saisir une phrase et renvoyant cette phrase au programme principal sous la forme d'un pointeur char*. Le programme déclarera ensuite une variable de type `string` pour y copier cette phrase et en calculer sa longueur avec la fonction `size()`. Une fois la copie faite, libérer la mémoire allouée dynamiquement et afficher le message contenu dans le *string*.

Bonus : Concaténer au string des points ('.') jusqu'à concurrence de sa capacité.

Consigne : gérer l'allocation dynamique et la récupération de la mémoire avec les opérateurs C++ `new`et `delete`.

Exercice 5: Range-based for loop

- Compléter le programme suivant :

  1. Remplacer la boucle **`for`** suivante par une boucle **`range-based`**
  2. Copier les valeurs du tableau **`primeNumbers`** dans le tableau **`copie`**

  Pourquoi ne peut-on pas faire cette copie SIMPLEMENT avec le nouveau **`for`** ?

```
#include <iostream>
using namespace std;
int main()
{
    int primeNumbers[] = { 1, 2, 3, 5, 7, 11, 13};
    int copy[7]  = { 0, 0, 0, 0, 0, 0, 0 };
    int sizeArray = sizeof(primeNumbers) / sizeof(int);

    // AFFICHAGE DU TABLEAU
    //----------------------
    // 1. remplacer la boucle for suivante par une boucle range-based
    for (int i=0; i < sizeArray; i++)
    {
        std::cout << primeNumbers [i] << std::endl;
    }
```

```
        // COPIE DU TABLEAU
        // --------------------
        // 2. copier les valeurs du tableau primeNumbers dans copie

        return 0;
    }
```

Exercice 6: Structure

- Définir une structure `Room` capable de contenir les dimensions d'une chambre (largeur, longueur, hauteur), de lui donner un nom à l'aide d'un champ de type `string`, et possédant les fonctions suivantes :
    - `surfaceFloor()` // calcule et renvoie la surface au sol de la pièce
    - `surfaceWalls()` // calcule et renvoie la surface des murs de la pièce
    - `volume()` // calcule et renvoie le volume de la pièce
- Écrire un programme utilisant ce nouveau type, qui lui affecte des valeurs et utilise ses fonctions (méthodes)