MOOC Init Prog C++
Corrigés semaine 3

Les corrigés proposés correspondent à l'ordre des apprentissages : chaque corrigé correspond à la solution à laquelle vous pourriez aboutir au moyen des connaissances acquises jusqu'à la semaine correspondante.

Exercice 7 : tables de multiplication

> Cet exercice correspond à l'exercice n°3 (pages 19 et 199) de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

```
#include <iostream>
using namespace std;

int main()
{
    cout << "     Tables de multiplication" << endl;
    for (int i(2); i <= 10; ++i) {
        cout << endl << " Table de " << i << " :" << endl;
        for (int j(1); j <= 10; ++j) {
            cout << "    " << i << " * " << j << " = " << i*j << en
        }
    }
    return 0;
}
```

Exercice 8 : rebonds de balles (1)

Cet exercice correspond à l'exercice n°6 (pages 21 et 202) de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

```
#include <iostream>
#include <cmath>
using namespace std;

constexpr double g(9.81); // la constante de gravité terrestre

int main()
{
  // Saisie des valeurs, avec test de validité ------------
  double eps; // coefficient de rebond de la balle
  do {
    cout << "Coefficient de rebond (0 <= coeff < 1) : ";
    cin >> eps;
  } while ( (eps < 0.0) or (eps >= 1.0) );

  double h0; // hauteur avant rebond
  do {
    cout << "Hauteur initiale      (h0 > 0)         : ";
    cin >> h0;
  } while (h0 <= 0.0);

  int n; // nombre de rebonds à simuler
  do {
    cout << "Nombre de rebonds     (n >= 0)         : ";
    cin >> n;
  } while (n < 0);

  // Declarations -----------------------------------------
  double h1;            // hauteur après le rebond
  double v0, v1;        // vitesses avant et après le rebond

  // Boucle de calcul -------------------------------------
  for (int rebonds(1); rebonds <= n; ++rebonds) {
    // on fait une iteration par rebond
    v0 = sqrt(2.0 * g * h0);   // vitesse avant rebond
    v1 = eps * v0;             // vitesse apres le rebond
    h1 =(v1 * v1) / (2.0 * g); // hauteur après rebond
    h0 = h1;   // qui devient nouvelle haut. avant rebond suivant
  }

  // Affichage du resultat --------------------------------
```

```
  cout << "Au " << n
       << "e rebond, la hauteur atteinte est de " << h0 << endl;
  return 0;
}
```

Exercice 9 : rebonds de balles (2)

> Cet exercice correspond à l'exercice n°7 (pages 22 et 203) de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

La différence entre cet exercice et le précédent réside dans la condition d'arrêt du calcul. Au lieu d'avoir un nombre fixe d'itération entré par l'utilisateur, facilement utilisable dans une boucle `for`, celui-ci est variable. Une boucle `do..while` qui compare le résultat du dernier calcul avec la valeur `h_fin` entrée par l'utilisateur est donc plus judicieuse.

```
#include <iostream>
#include <cmath>
using namespace std;

constexpr double g(9.81); // la constante de gravité terrestre

int main()
{
  // Saisie des valeurs, avec test de validité ------------
  double eps; // coefficient de rebond de la balle
  do {
    cout << "Coefficient de rebond (0 <= coeff < 1) : ";
    cin >> eps;
  } while ( (eps < 0.0) or (eps >= 1.0) );

  double h0; // hauteur avant rebond
  do {
    cout << "Hauteur initiale      (h0 > 0)          : ";
    cin >> h0;
  } while (h0 <= 0.0);

  double h_fin; // hauteur finale
  do {
    cout << "Hauteur finale        (0 < h_fin < h0) : ";
    cin >> h_fin;
  } while ( (h_fin <= 0.0) or (h_fin >= h0) );

  // Declarations ----------------------------------------
  double h1;            // hauteur après le rebond
  double v0, v1;        // vitesses avant et après le rebond
  int rebonds(0);       // nombre de rebonds

  // Boucle de calcul ------------------------------------
  do {
    ++rebonds;                  // une iteration par rebond
    v0 = sqrt(2.0 * g * h0);    // vitesse avant rebond
```

```
    v1 = eps * v0;                // vitesse apres le rebond
    h1 = (v1 * v1) / (2.0 * g); // hauteur après rebond
    h0 = h1;    // qui devient nouvelle haut. avant rebond suivant
  } while (h0 > h_fin);

  // Affichage du resultat --------------------------------
  cout << "La balle rebondit " << rebonds
       << " fois avant que la hauteur de rebond (" << h0
       << ") soit inférieure à la limite de " << h_fin << endl;
  return 0;
}
```

Exercice 10 : nombres premiers

Cet exercice correspond à l'exercice n°9 (pages 22 et 205) de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    int n;                // le nombre à tester
    bool premier(true);   // résultat du test de primalité
    int diviseur(1);

    // Saisie du nombre à tester
    do {
        cout << "Entrez un nombre entier > 1 : ";
        cin >> n;
    } while (n <= 1);

    if (n % 2 == 0) {
        // le nombre est pair
        if (n != 2) {
            premier  = false;
            diviseur = 2;
        }
    } else {
        const double borne_max(sqrt(n));
        for (int i(3); (premier) and (i <= borne_max); i += 2) {
            if (n % i == 0) {
                premier  = false;
                diviseur = i;
            }
        }
    }

    cout << n ;
    if (premier) {
        cout << " est premier" << endl;
    } else {
        cout << " n'est pas premier, car il est divisible par "
             << diviseur << endl;
    }
    return 0;
}
```

Résultats :

```
2 est premier
16 n'est pas premier, car il est divisible par 2
17 est premier
91 n'est pas premier, car il est divisible par 7
589 n'est pas premier, car il est divisible par 19
1001 n'est pas premier, car il est divisible par 7
1009 est premier
1299827 est premier
2146654199 n'est pas premier, car il est divisible par 46327
```

[Niveau Avancé]

Remarques :

Si vous souhaitez tester des nombres plus grands que 2147483647 (c'est-à-dire 2^{31}-1, qui d'ailleurs est premier !), remplacez
`int n;`
par
`unsigned long int n;`

Il faut aussi changer la déclaration de `i` dans le `for` :

```
for (unsigned long int i(3);
```

Vous pouvez alors tester jusqu'à 4294967295 (Essayez par exemple 4292870399). La signification de tout cela sera vue plus tard dans le cours.

Pour ceux qui aimeraient tester de plus grand nombre encore, vous pouvez, sur les stations Sun et avec le compilateur *gcc* (C++), remplacer les entiers précédents (`int` puis `unsigned long int`) par (n'oubliez pas de le faire pour `n` et pour `i`) :

```
unsigned long long int n;
```

[attention, ce type étendu n'est pas standard]

Ceci vous permet d'aller jusqu'à 18446744073709551615 (Essayez par exemple 18446744073709551577 ou 18446744073709551557 (il faut attendre un moment !)).
