MOOC Init. Prog. C++
Correction des exercices supplémentaires semaine 5

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 13 : Éléments en indice

```
vector<int> elements_en_indice(vector<int> & T)
{
  // la taille du tableau resultat est la moitie du tableau T :
  vector<int> R = vector<int>(T.size() / 2);

  for(size_t i(0); i < R.size(); ++i) {
    // on copie l'element T[2 * i] dans R a` l'indice T[2 * i + 1] :
    R[ T[2 * i + 1] ] = T[2 * i];
  }

  return R;
}
```

Exercice 14 : Crible d'Ératosthène

La meilleure technique pour résoudre cet exercice consiste à se servir d'un tableau de `bool`, initialisé à `false`. On parcourt ensuite le tableau en enlevant (c'est-à-dire en mettant à `true`) les multiples des nombres premiers.

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

int main()
{
  array<bool, 100> supprimes;

  for(auto & element : supprimes) {
    element = false;
  }
  supprimes[0] = true;    // 0 n'est pas premier
  supprimes[1] = true;    // 1 n'est pas premier

  for(size_t i(2); i < supprimes.size(); ++i) {
    if (not supprimes[i]) {
      size_t multiple(2 * i);
      while (multiple < supprimes.size()) {
 supprimes[multiple] = true;
 multiple = multiple + i;
      }
    }
  }

  for(size_t i(0); i <  supprimes.size(); ++i) {
    if (not supprimes[i]) {
      cout << i << " ";
    }
  }
  cout << endl;

  return 0;
}
```

Exercice 15 : placements sans recouvrements

Cet exercice correspond à l'exercice n°21 (pages 58 et 224) de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

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

constexpr size_t DIM(10);

// ----------------------------------------------------------------------
bool remplitGrille(array<array<bool, DIM>, DIM>& grille,
                   unsigned int ligne, unsigned int colonne,
                   char direction, unsigned int longueur)
{
  int dx, dy;

  switch (direction) {
  case 'N': dx = -1; dy =  0; break;
  case 'S': dx =  1; dy =  0; break;
  case 'E': dx =  0; dy =  1; break;
  case 'O': dx =  0; dy = -1; break;
  }

  unsigned int i(ligne);     // les coordonnées de la case...
  unsigned int j(colonne);   // ...à modifier
  unsigned int l;            // la longueur modifiée

  bool possible(true); // est-ce possible de mettre tout l'élément ?

  // avant de modifier la grille, il faut vérifier si c'est possible de
  // mettre tout l'objet.
  for (// Initialisation
       l = 0;

       /* Condition de continuation. Vu que i et j sont des "unsigned" *
        * pas besoin de tester ici les condition (i >= 0) et (j >= 0), *
        * lesquelles sont forcément vraies                             */
       (possible) and (i < grille.size()) and (j < grille[0].size())
         and (l < longueur);

       // Incréments
       ++l, i += dx, j += dy) {
    if (grille[i][j]) // c'est-à-dire la grille est déjà occupée
    {
      possible = false;  // on ne peut donc pas mettre tout l'objet voulu
    }
  }

  /* Si l == longueur c'est qu'on a pu tout placer.
   * Il se pourrait en effet qu'on soit sorti de la boucle ci-dessus
   * parce que i >= DIM ou j >= DIM... ...ce qui n'a pas modifié
   * "possible" jusqu'ici.
   */
```

```
  possible = possible and (l == longueur);

  if (possible) {
    // on met effectivement l'objet, plus besoin de test ici
    for (l = 0, i = ligne, j = colonne; l < longueur;
         ++l, i += dx, j += dy) {
      grille[i][j] = true;
    }
  }

  return possible;
}

// ----------------------------------------------------------------------
void initGrille(array<array<bool, DIM>, DIM>& grille)
{
  for (auto& ligne : grille)
    for (auto& cell : ligne)
      cell = false;
}

// ----------------------------------------------------------------------
void ajouterElements(array<array<bool, DIM>, DIM>& grille)
{
  int x, y;
  do {
    cout << "Entrez coord. x : ";
    cin >> x;

    if (x >= 0) {
      cout << "Entrez coord. y : ";
      cin >> y;

      if (y >= 0) {

        char dir;
        do {
          cout << "Entrez direction (N,S,E,O) : ";
          cin >> dir;
        } while ((dir != 'N') and (dir != 'S') and
                 (dir != 'E') and (dir != 'O'));

        cout << "Entrez taille : ";
        unsigned int l;
        cin >> l;

        cout << "Placement en (" << x << "," << y << ") direction "
             << dir << " longueur " << l << " -> ";

        if (remplitGrille(grille, x, y, dir, l))
          cout << "succès";
        else
          cout << "ECHEC";
        cout << endl;

      }
    }
  } while ((x >= 0) and (y >= 0));
```

```
}

// ------------------------------------------------------------------------
void afficheGrille(const array<array<bool, DIM>, DIM>& grille)
{
  for (auto ligne : grille) {
    for (auto cell : ligne) {
      if (cell) cout << '#';
      else      cout << '.';
    }
    cout << endl;
  }
}

// ------------------------------------------------------------------------
int main()
{
  array<array<bool, DIM>, DIM> grille;
  initGrille(grille);
  ajouterElements(grille);
  afficheGrille(grille);
  return 0;
```
