MOOC Init. Prog. C++
Exercices supplémentaires facultatifs semaine 5

Écrivez une fonction appelée `elements_en_indice` prenant un tableau `T` d'entiers en paramètres. La fonction devra retourner un nouveau tableau, qui sera construit de la façon suivante : Les éléments de `T` d'indice pair seront placés dans ce nouveau tableau à l'indice donné par l'élément suivant de `T`:

- `T[0]` sera placé dans le nouveau tableau à l'indice `T[1]`,
- `T[2]` sera placé dans le nouveau tableau à l'indice `T[3]`,
- etc...

Testez votre fonction avec le code:

```
vector<int> A(6);
A[0] = 4;
A[1] = 2;
A[2] = 8;
A[3] = 0;
A[4] = 7;
A[5] = 1;
vector<int> T = elements_en_indice(A);
for(unsigned int i(0); i < T.size(); ++i) {
  cout << T[i] << " ";
}
cout << endl;
```

qui devrait afficher `8 7 4`.

Crible d'Ératosthène (niveau 2)

Un nombre est dit premier s'il admet exactement 2 diviseurs *distincts* (1 et lui-même). 1 n'est donc pas premier.

Le crible d'Ératosthène est une méthode de recherche des nombres premiers plus petits qu'un entier naturel *n* donné. Cette méthode est simple:

- On commence par supprimer tous les multiples de 2 inférieurs à *n*.
- L'entier 3 n'a pas été supprimé et il ne peut être multiple des entiers qui le précèdent, sinon on l'aurait supprimé; il est donc premier. Supprimons alors tous les multiples de 3 inférieurs à *n*.
- L'entier 5 n'a pas été supprimé, il est donc premier. Supprimons tous les multiples de 5 inférieurs à *n*.
- Et ainsi de suite jusque *n*. Les valeurs n'ayant pas été supprimées sont les nombres entiers plus petits que *n*.

Écrivez le code qui applique cette méthode pour trouver les nombres premiers inférieurs à 100. Vous devez trouver: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.

On utilisera un tableau de booléens :

```
vector<bool> supprimes(100, false);
```

pour mémoriser les entiers qui ont été supprimés. Notez que nous avons initialisé chacun de ses éléments à `false`.

Placement sans recouvrement (niveau 2)

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

Le but de cet exercice est de placer sans recouvrement des objets rectilignes sur une grille carrée. Cela pourrait être par exemple une partie d'un programme de bataille navale.

Dans le fichier `recouvrement.cc` :

1. Définissez une constante globale entière non-signée, nommée `DIM` et de valeur 10. Elle représentera la taille de la grille (carrée).

2. Prototypez et écrivez une fonction :

   ```
   bool remplitGrille(array<array<bool, DIM>, DIM>& grille,
        unsigned int ligne, unsigned int colonne,
                     char direction, unsigned int longueur);
   ```

   dont le rôle est de vérifier si le placement dans une `grille` (voir ci-dessous) d'un objet de dimension `longueur` est possible, en partant de la coordonnée (`ligne`,`colonne`) et dans la direction définie par `direction` (Nord, Sud, Est ou Ouest).

   Si le placement est possible, la fonction devra de plus effectuer ce placement (voir ci-dessous la description de la grille).

   La fonction devra indiquer (par la valeur de retour) si le placement a pu être réalisé ou non.

3. Dans le `main()`
   - Définissez une **variable** nommée `grille`, de type `array<array<bool, DIM>, DIM>`.
     La valeur `true` dans une case `[i][j]` de cette grille représente le fait qu'un (bout d')objet occupe la case de coordonnées (i, j).
   - Utilisez la fonction précédente pour remplir interactivement la grille, en demandant à l'utilisateur de spécifier la position, la taille et la direction des objets à placer.
     Indiquez à chaque fois à l'utilisateur si l'élément a pu ou non être placé dans la grille.
     Le remplissage se termine lorsque l'utilisateur entre une position/coordonnée strictement inférieure à 0.
   - Terminer alors en "dessinant" la grille : afficher un `'.'` si la case est vide et un `'#'` si la case est occupée.

Remarques :

- Dans l'interface utilisateur, pour indiquer les positions, utilisez au choix soit les coordonnées du C++ : 0 à DIM-1 (plus facile), soit les coordonnées usuelles (1 à DIM, un peu plus difficile) , **MAIS dans tous les cas utilisez les indices de 0 à DIM-1** pour votre tableau (aspect programmeur).
- Pensez à effectuer des tests de validité sur les valeurs entrées (débordement du tableau).
- pour représenter la direction, vous pouvez soit utiliser un caractère (`'N'` pour nord. `'S'` pour sud, etc..., plus facile), soit un type enuméré (plus difficile : pensez à l'interface avec l'utilisateur).
- N'oubliez pas d'initialiser la grille en tout début de programme !

Exemple de fonctionnement (version facile : coordonées de 0 à 9 et lettres pour les directions) :

```
Entrez coord. x: 2
Entrez coord. y: 8
Entrez direction (N,S,E,O): E
Entrez taille: 2
Placement en (2,8) direction E longueur 2 -> succès
Entrez coord. x: 0
Entrez coord. y: 8
Entrez direction (N,S,E,O): S
Entrez taille: 5
Placement en (0,8) direction S longueur 5 -> ECHEC
```

```
Entrez coord. x: 0
Entrez coord. y: 9
Entrez direction (N,S,E,O): O
Entrez taille: 5
Placement en (0,9) direction O longueur 5 -> succès
Entrez coord. x: -1
```

Résultat des placements :

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5										
6										
7										
8										
9										