Exercice : tours de Hanoï

> Exercice n°35 (pages 85 et 257) de l'ouvrage *C++ par la pratique*.

Cet exercice est de niveau 3 en raison de sa longueur et de l'appel récursif double (`hanoi` s'appelle deux fois) qui peut, peut être, troubler une première fois.

Cependant l'énoncé est assez bien détaillé et guide relativement bien.
Voici donc directement le code solution en C++11. Pour une version compilant avec l'ancien standard (C++98), **voir ci-dessous**.

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

constexpr unsigned int N(4); // la taille de la tour de départ

// un disque sera représenté simplement par sa taille (rayon)
typedef unsigned int Disque;
// 0 signifiant pas de disque
constexpr Disque PAS_DE_DISQUE(0);

typedef vector<Disque> Pilier; // un pilier est une pile de disques
typedef array<Pilier, 3> Jeu;  // le jeu est constitué de 3 piliers

// les fonctions
void affiche(char c, unsigned int n = 1);
void affiche(const Disque& d);
void affiche(const Jeu& jeu);
void init(Jeu& jeu, unsigned int taille = N);
size_t sommet(const Pilier& p);
void deplace(Pilier& origine, Pilier& destination);
unsigned int autre(unsigned int p1, unsigned int p2);
void hanoi(unsigned int taille, unsigned int origine,
           unsigned int destination, Jeu& jeu);

// ----------------------------------------------------------------------
int main()
{
  Jeu monjeu;

  init(monjeu);
  affiche(monjeu);
  hanoi(N, 0, 2, monjeu);

  return 0;
}

/* ----------------------------------------------------------------------
 * Initialise le jeu
 * ---------------------------------------------------------------------- */
void init(Jeu& jeu, unsigned int taille)
{
  // crée un jeu vide :
  for (auto& pilier : jeu) {
    pilier = Pilier(taille, PAS_DE_DISQUE);
  }

  // remplit le 1er pilier :
  for (size_t i(0); (i < jeu[0].size()) and (i < taille); ++i) {
    jeu[0][i] = Disque(i+1);
  }
}
```

```
/* ------------------------------------------------------------------------
 * Affiche n fois un caractère
 * Entrée : le caractère à afficher et le nombre de fois
 * ------------------------------------------------------------------------ */
void affiche(char c, unsigned int n)
{
  for (unsigned int i(0); i < n; ++i) cout << c;
}

/* ------------------------------------------------------------------------
 * Affiche un dique
 * Entrée : le dique à afficher
 * ------------------------------------------------------------------------ */
void affiche(const Disque& d)
{
  if (d == PAS_DE_DISQUE) {
    affiche(' ', N-1);
    affiche('|');
    affiche(' ', N); // N = N-1 de tour vide + 1 espace intermédiaire
  }
  else {
    affiche(' ', N-d);
    affiche('-', 2*d-1);
    affiche(' ', N-d+1); // +1 pour l'espace intermédiaire
  }
}

/* ------------------------------------------------------------------------
 * Affiche un jeu
 * Entrée : le jeu à afficher
 * ------------------------------------------------------------------------ */
void affiche(const Jeu& jeu)
{
  for (size_t i(0); i < jeu[0].size(); i++) {
    affiche(jeu[0][i]);
    affiche(jeu[1][i]);
    affiche(jeu[2][i]);
    cout << endl;
  }
  // le socle
  affiche('#', 6*N-1); // 3*(2*N-1)+2 = 6*N-1, ou autre choix...
  cout << endl << endl;
}

/* ------------------------------------------------------------------------
 * Retourne l'indice du sommet d'une tour sur un pilier. Retourne N si pas
 * de tour sur ce pilier
 * ------------------------------------------------------------------------ */
size_t sommet(const Pilier& p)
{
  size_t top;
  for (top = 0; (top < p.size()) and (p[top] == PAS_DE_DISQUE); ++top);
  return top;
}

/* ------------------------------------------------------------------------
 * Déplace le disque du top d'une tour à un autre pilier.
 * Vérifie que le mouvement est valide.
 * Entrée : le pilier d'où enlever le disque, et le pilier destination
 * ------------------------------------------------------------------------ */
void deplace(Pilier& origine, Pilier& destination)
{
  size_t top1(sommet(origine));
  if (top1 < origine.size()) { // si la tour d'origine existe bien

    size_t top2(sommet(destination));
    if ((top2 < destination.size()) and (destination[top2] < origine[top1])) {
      /* on essaye de mettre un disque plus gros (origine[top1])
       * sur un disque plus petit (destination[top2]) : ce n'est pas
```

```
        * un déplacement autorisé !
        */
       cerr << "ERREUR : on ne peut pas déplacer un disque de taille "
    << origine[top1] << " sur un disque de taille "
    << destination[top2] << " !" << endl;
       return;
    }

    // effectue le mouvement
    destination[top2-1] = origine[top1];
    origine[top1] = PAS_DE_DISQUE;
  }
}

/* ------------------------------------------------------------------------
 * Étant donnés deux numéros de pilier p1 et p2, retourne l'indice du 3e
 * pilier
 * ------------------------------------------------------------------------ */
unsigned int autre(unsigned int p1, unsigned int p2)
{
  return 3 - p1 -p2;
}

/* ------------------------------------------------------------------------
 * Déplace une tour d'un pilier à un autre.
 * Vérifie que le mouvement est valide.
 * Entrée : la taille de la tour, le pilier de départ, le pilier de destination
 * ------------------------------------------------------------------------ */
void hanoi(unsigned int n, unsigned int origine, unsigned int destination,
    Jeu& jeu)
{
  if (n != 0) {
    const unsigned int auxiliaire(autre(origine, destination));
    hanoi(n-1, origine, auxiliaire, jeu);
    deplace(jeu[origine], jeu[destination]);
    affiche(jeu);
    hanoi(n-1, auxiliaire, destination, jeu);
  }
}
```

Version C++98, les différences avec la version C++11 sont mises en évidence :

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

const unsigned int N(4); // la taille de la tour de départ

// un disque sera représenté simplement par sa taille (rayon)
typedef unsigned int Disque;
// 0 signifiant pas de disque
const Disque PAS_DE_DISQUE(0);

typedef Disque Pilier[N]; // un pilier est une pile d'au plus N disques
typedef Pilier Jeu[3]; // le jeu est constitué de 3 piliers


// les fonctions
void affiche(char c, unsigned int n = 1);
void affiche(Disque d);
void affiche(Jeu jeu);
void init(Jeu& jeu); /* Note: le passage par référence est ici facultatif
        * vu que Jeu est un tableau "à la C".
        * Mais je préfère marquer clairement par ce passage
        * par référence que l'argument jeu est modifié par
        * cette fonction */

unsigned int sommet(Pilier p);
```

```
void deplace(Pilier& origine, Pilier& destination); // même remarque (bis)
unsigned int autre(unsigned int p1, unsigned int p2);
void hanoi(unsigned int n, unsigned int origine, unsigned int destination,
    Jeu& jeu); // même remarque (ter)

// ----------------------------------------------------------------------
int main()
{
  Jeu monjeu;

  init(monjeu);
  affiche(monjeu);
  hanoi(N, 0, 2, monjeu);

  return 0;
}

/* ----------------------------------------------------------------------
 * Initialise le jeu
 * ---------------------------------------------------------------------- */
void init(Jeu& jeu)
{
  for (unsigned int i(0); i < N; ++i) {
    jeu[0][i] = Disque(i+1);
    jeu[1][i] = PAS_DE_DISQUE;
    jeu[2][i] = PAS_DE_DISQUE;
  }
}

/* ----------------------------------------------------------------------
 * Affiche n fois un caractère
 * Entrée : le caractère à afficher et le nombre de fois
 * ---------------------------------------------------------------------- */
void affiche(char c, unsigned int n)
{
  for (unsigned int i(0); i < n; ++i) cout << c;
}

/* ----------------------------------------------------------------------
 * Affiche un dique
 * Entrée : le dique à afficher
 * ---------------------------------------------------------------------- */
void affiche(Disque d)
{
  if (d == PAS_DE_DISQUE) {
    affiche(' ', N-1);
    affiche('|');
    affiche(' ', N); // N = N-1 de tour vide + 1 espace intermédiaire
  }
  else {
    affiche(' ', N-d);
    affiche('-', 2*d-1);
    affiche(' ', N-d+1); // +1 pour l'espace intermédiaire
  }
}

/* ----------------------------------------------------------------------
 * Affiche un jeu
 * Entrée : le jeu à afficher
 * ---------------------------------------------------------------------- */
void affiche(Jeu jeu)
{
  for (unsigned int i(0); i < N; ++i) {
    affiche(jeu[0][i]);
    affiche(jeu[1][i]);
    affiche(jeu[2][i]);
    cout << endl;
  }
  // le socle
```

```
  affiche('#', 6*N-1); // 3*(2*N-1)+2 = 6*N-1, ou autre choix...
  cout << endl << endl;
}

/* ------------------------------------------------------------------------
 * Retourne l'indice du sommet d'une tour sur un pilier. Retourne N si pas
 * de tour sur ce pilier
 * ------------------------------------------------------------------------ */
unsigned int sommet(Pilier p)
{
  unsigned int top;
  for (top = 0; (top < N) and (p[top] == PAS_DE_DISQUE); ++top);
  return top;
}

/* ------------------------------------------------------------------------
 * Déplace le disque du top d'une tour à un autre pilier.
 * Vérifie que le mouvement est valide.
 * Entrée : le pilier d'où enlever le disque, et le pilier destination
 * ------------------------------------------------------------------------ */
void deplace(Pilier& origine, Pilier& destination)
{
  unsigned int top1(sommet(origine));
  if (top1 < N) { // si la tour d'origine existe bien

    unsigned int top2(sommet(destination));
    if ((top2 < N) and (destination[top2] < origine[top1])) {
      /* on essaye de mettre un disque plus gros (origine[top1])
       * sur un disque plus petit (destination[top2]) : ce n'est pas
       * un déplacement autorisé !
       */
      cerr << "ERREUR : on ne peut pas déplacer un disque de taille "
    << origine[top1] << " sur un disque de taille "
    << destination[top2] << " !" << endl;
      return;
    }

    // effectue le mouvement
    destination[top2-1] = origine[top1];
    origine[top1] = PAS_DE_DISQUE;
  }
}

/* ------------------------------------------------------------------------
 * Étant donnés deux numéros de pilier p1 et p2, retourne l'indice du 3e
 * pilier
 * ------------------------------------------------------------------------ */
unsigned int autre(unsigned int p1, unsigned int p2)
{
  return 3 - p1 -p2;
}

/* ------------------------------------------------------------------------
 * Déplace une tour d'un pilier à un autre.
 * Vérifie que le mouvement est valide.
 * Entrée : la taille de la tour, le pilier de départ, le pilier de destination
 * ------------------------------------------------------------------------ */
void hanoi(unsigned int n, unsigned int origine, unsigned int destination,
    Jeu& jeu)
{
  if (n != 0) {
    const unsigned int auxiliaire(autre(origine, destination));
    hanoi(n-1, origine, auxiliaire, jeu);
    deplace(jeu[origine], jeu[destination]);
    affiche(jeu);
    hanoi(n-1, auxiliaire, destination, jeu);
  }
}
```