Tours de Hanoi OO [cm08] - Exercice

Karine Zampieri, Stéphane Rivière, Béatrice Amerein-Soltner

Unisciel algoprog Université Haute-Alsace

Version 20 mai 2018

Table des matières

C++ - Tours de Hanoi OO (Solution)

Mots-Clés Classes ■
Requis Structures de base, Structures conditionnelles, Algorithmes paramétrés, Structures répétitives, Tableaux, Récursivité des actions, Classes, Classes (suite) ■
Difficulté • ○ ○ (40 min) ■

Objectif
Cet exercice réalise une version OO des Tours de Hanoï permettant de visualiser les différents piliers et déplacements.

1 Tours de Hanoi OO / pgtourshanoi

Voici un extrait d'exécution du résultat attendu :

```
# de disques? 3
  -     |     |
 ---    |     |
-----   |     |
*****************

  |     |     |
 ---    |     |
-----   |     -
*****************


  |     |     -
  |     |    ---
  |     |   -----
*****************

Nombre de déplacements = 7
```

1.1 Classe HDisque

Écrivez une classe `HDisque` qui représente un disque de Hanoï. Il nécessite de stocker la taille du jeu `szjeu` (entier) et la `taille` (entier) du disque.

Écrivez un constructeur par défaut initialisant la taille du jeu à 1 et la taille du disque à zéro.

Écrivez un constructeur à deux paramètres entiers `s` et `t` initialisant les attributs.

Écrivez un accesseur `getTaille` de la taille du disque.

Écrivez une méthode `toString` qui calcule et renvoie l'équivalent « chaîne » du disque. Exemple : Pour une taille de jeu `szjeu` de 3, il faut retourner la chaîne (« . » représente l'espace) :

- `"..|..."` si la `taille` du disque vaut 0
- `"..-..."` pour 1
- `".---.."` pour 2
- `"-----."` pour 3

Aide simple

Le nombre d'espaces `nespaces` d'un côté du pilier vaut `szjeu-1` si `taille` vaut 0, `szjeu-taille` sinon. Le nombre de tirets est `2*taille-1`.

Validez votre classe et vos méthodes avec la solution.

Solution C++ @[HDisque.hpp] @[HDisque.cpp]

```
#ifndef TOURSHANOI_DISQUE_CLASS
#define TOURSHANOI_DISQUE_CLASS
#include <string>
using namespace std;

/**
  Un disque est represente par sa taille
*/
class HDisque
{
  public:
    HDisque();
    HDisque(int s, int t);
    int getTaille() const;
    string toString() const;
  private:
    int m_taille; // taille du disque
    int m_szjeu;  // taille du jeu
};
#include "HDisque.cpp"
#endif
```

```
/**
  Constructeur par defaut (tableau)
*/
HDisque::HDisque()
: m_taille(0), m_szjeu(1)
{}

/**
  Constructeur normal
  @param[in] s - taille du jeu
  @param[in] t - taille du disque
 */
HDisque::HDisque(int s, int t)
: m_taille(t), m_szjeu(s)
{}

/**
  Accesseur de la taille du disque
  @return la taille du disque
*/
int HDisque::getTaille() const
{
  return m_taille;
}

/**
```

```
  Equivalent "string" du disque
  @return Equivalent "string" du disque
*/
string HDisque::toString() const
{
  int nespaces = (getTaille() == 0 ? m_szjeu-1U : m_szjeu-getTaille());
  return string(nespaces, ' ')
       + (getTaille() == 0 ? "|" : string(2*getTaille()-1, '-'))
       + string(nespaces, ' ')
       + " ";
}
```

1.2 Classe HTour

Définissez la constante `NMAXDISQUES=10` (nombre maximum de disques).

Écrivez une classe `HTour` qui représente une tour de Hanoï. Par conséquent, elle contient un tableau `pilier` de `NMAXDISQUES` disques ainsi que l'indice `top` (entier) du sommet.

Écrivez un constructeur par défaut.

Écrivez un constructeur de paramètres `n` (entier) et `vide` (booléen) initialisant les attributs.

Écrivez des accesseurs `getDisque(k)` du disque `k` du pilier et `sommet` de l'indice du sommet.

Écrivez une méthode `push(d)` qui met un `Disque d` au sommet du pilier.

Écrivez une méthode `pop` qui dépile le disque au sommet

Écrivez une méthode `assign(taille,vide)` qui redimensionne la tour et empile les disques.

Validez votre classe et vos méthodes avec la solution.

Solution C++ @[HTour.hpp] @[HTour.cpp]

```
#ifndef TOURSHANOI_TOUR_CLASS
#define TOURSHANOI_TOUR_CLASS
#include <string>
using namespace std;
```

```
/**
  Une Tour avec ses disques
*/
#include "HDisque.hpp"
const int NMAXDISQUES = 10;
class HTour
{
  public:
    HTour();
    HTour(int n, bool vide);
    HDisque getDisque(int k) const;
    int sommet() const;
    void push(const HDisque& d);
    void pop();
    void assign(int taille, bool vide);
  private:
    HDisque m_pilier[NMAXDISQUES]; // le pilier
    int m_top; // indice du sommet
    int m_dim; // dimension du socle
};
#include "HTour.cpp"
#endif
```

```
/**
  Constructeur par defaut (tableau)
*/
HTour::HTour()
: m_top(-1)
{}

/**
  Constructeur normal: une tour peut contenir au plus n disques
  @param[in] n - nombre de tours
  @param[in] vide - si vrai alors 0
*/
HTour::HTour(int n, bool vide)
: m_top(-1)
{
  assign(n, vide);
}

/**
  Accesseur au disque k
  @param[in] k - numero du disque
  @return disque k
*/
HDisque HTour::getDisque(int k) const
{
  return m_pilier[k];
}

/**
  Accesseur a l'indice du sommet
  @return indice du sommet
*/
int HTour::sommet() const
{
  return m_top;
```

```
}

/**
  Met un disque au sommet du pilier
  @param[in] d - un Disque
*/
void HTour::push(const HDisque& d)
{
  m_pilier[++m_top] = d;
}

/**
  Depile le disque au sommet
*/
void HTour::pop()
{
  m_pilier[m_top--] = HDisque(m_dim,0);
}

/**
  Retaille la tour et empile les disques
  @param[in] taille - taille des disques
  @param[in] vide - si vrai alors 0
*/
void HTour::assign(int taille, bool vide)
{
  m_dim = taille;
  // Empile les disques par ordre de diamètre decroissant
  for (int ix = 0, t = taille; ix < taille; ++ix, --t)
  {
    m_pilier[ix] = HDisque(taille, vide ? 0 : t);
  }
  // Actualise le sommet
  m_top = (!vide) ?  taille-1 : -1;
}
```

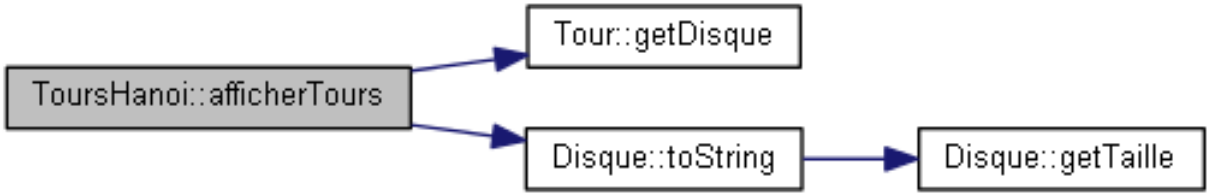
1.3 Classe ToursHanoi

Définissez la constante `NMAXTOURS=3` (nombre de tours).

Écrivez une classe `ToursHanoi` qui contient un tableau `tours` de `NMAXTOURS` tours ainsi que la `hauteur` (entier) de la tour (nombre de disques). Comme l'on souhaite également compter le nombre de déplacements, ajoutez un attribut `deplts` (entier) pour le comptage.

Écrivez un constructeur de paramètre entier `n` (nombre de disques) initialisant les attributs.

Écrivez une méthode `afficherTours` qui affiche l'état du jeu.

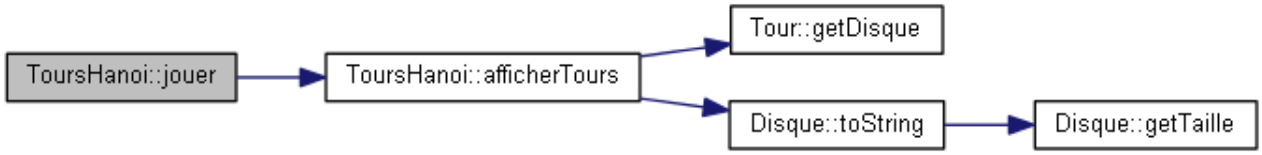


Écrivez une méthode **interne** `deplacer(o,d)` qui déplace le disque au sommet de la tour numéro `o` vers la tour numéro `d`.

Écrivez une méthode **interne** `hanoi(n,o,d,x)` qui est la procédure récursive principale des tours de Hanoï pour `n` disques de la tour d'origine numéro `o` vers la tour destination numéro `d` en passant par la tour intermédiaire numéro `x`.

Écrivez une méthode **interne** `initialiser` qui initialise le jeu.

Jouer consiste à initialiser le jeu, afficher les tours puis à lancer la procédure `hanoi(n,o,d,x)` classique. Déduisez une méthode `jouer` qui lance le jeu.



Validez votre classe et vos méthodes avec la solution.

Solution C++ @[ToursHanoi.hpp] @[ToursHanoi.cpp]

```
#ifndef TOURSHANOI_CLASS
#define TOURSHANOI_CLASS
#include <iostream>
#include <string>
using namespace std;

/**
  Les Tours de Hanoi
*/
#include "HTour.hpp"
const int NMAXTOURS = 3;
class ToursHanoi
{
  public:
    explicit ToursHanoi(int n);
    void afficherTours() const;
    void jouer();
  private:
    void deplacer(int o, int d);
    void hanoi(int n, int orig, int dest, int inter);
```

```
    void initialiser();

    HTour m_tours[NMAXTOURS]; // les tours
    int m_hauteur; // nombre de disques

    int m_ndeplts; // nombre de deplacements
};
#include "ToursHanoi.cpp"
#endif
```

```
/**
  Deplace le disque place au sommet de la tour o au sommet de la tour d
  @param[in] o - tour o(rigine)
  @param[in] d - tour d(estination)
*/
void ToursHanoi::deplacer(int o, int d)
{
  HTour &orig = m_tours[o];
  HTour &dest = m_tours[d];
  int top1 = orig.sommet();
  HDisque dk = orig.getDisque(top1);
  orig.pop();
  dest.push(dk);

  ++m_ndeplts;
}

/**
  Methode recursive principale
  @param[in] n - nombre de disques
  @param[in] orig - tour origine
  @param[in] dest - tour destination
  @param[in] inter - tour intermediaire
*/
void ToursHanoi::hanoi(int n, int orig, int dest, int inter)
{
  if (n > 0)
  {
    hanoi(n-1, orig, inter, dest);
    deplacer(orig, dest);
    afficherTours();
    hanoi(n-1, inter, dest, orig);
  }
}

/**
  Initialise le jeu
*/
void ToursHanoi::initialiser()
{
  m_tours[0].assign(m_hauteur, false);
  m_tours[1].assign(m_hauteur, true);
  m_tours[2].assign(m_hauteur, true);

  m_ndeplts = 0;
}
```

```
/**
  Constructeur normal
  @param[in] n - nombre de disques
 */
ToursHanoi::ToursHanoi(int n)
: m_hauteur(n < NMAXDISQUES ? n : NMAXDISQUES)
{
  if (n >= NMAXDISQUES)
  {
    m_hauteur = NMAXDISQUES;
  }
}

/**
  Affiche l'etat du jeu
*/
void ToursHanoi::afficherTours() const
{
  // Construit l'equivalent "string" des tours
  string s;
  for (int ix = m_hauteur-1; ix >= 0; --ix)
  {
    s += m_tours[0].getDisque(ix).toString();
    s += m_tours[1].getDisque(ix).toString();
    s += m_tours[2].getDisque(ix).toString();
    s += "\n";
  }
  // Construit la "string" du socle
  s += string(6*m_hauteur-1, '*');
  s += "\n";
  // Affiche la chaine du jeu
  cout << s << endl;
}

/**
  Lance le jeu
*/
void ToursHanoi::jouer()
{
  initialiser();
  afficherTours();
  hanoi(m_hauteur, 0, 2, 1);
  cout<<"Nombre de deplacements = "<<m_ndeplts<<endl;
}
```

1.4 Programme de test

Écrivez un programme qui :

- Demande et saisit le nombre de disques dans un entier `n`.
- Instancie une `ToursHanoi` avec `n` disques.
- Puis lance le jeu.

Testez. Exemple de l'extrait d'exécution ci-dessus :
@[rstourshanoi.txt]

Validez votre programme avec la solution.

Solution C++ @[pgToursHanoi.cpp]

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

#include "ToursHanoi.hpp"
int main()
{
  int n;
  cout<<"# de disques? ";
  cin>>n;
  ToursHanoi h(n);
  h.jouer();
}
```

2 Références générales

Comprend [Chappelier-CPP1 :c6 :ex35] ■