# Jeu du Baguenaudier [je03] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Université Haute-Alsace

Version 22 mai 2018

## Table des matières



## C++ - Jeu du Baguenaudier (TP)

**Mots-Clés** Jeu de démontage, Récursivité croisée ■
**Requis** Axiomatique impérative, Classes, Classes (suite), Récursivité des actions, Complexité des algorithmes ■
**Difficulté** ••• (1 h 30) ■

**Objectif**
Cet exercice réalise le jeu du Baguenaudier (jeu de démontage, récursivité croisée) et étudie sa complexité. Dans le même ordre d'idées, l'exercice @[Jeu du Tracassin] réalise le jeu du Tracassin.

# 1 Présentation du jeu

### Jeu du Baguenaudier

Il se joue sur une tablette divisée en $n$ cases. « Jouer » c'est ôter ou placer un pion sur une case selon que celle-ci est ou non remplie. L'unique règle : à chaque coup les seules cases **jouables** sont la première case ou la case qui suit la première case occupée.

### Types de jeu

Les deux types de jeux sont :

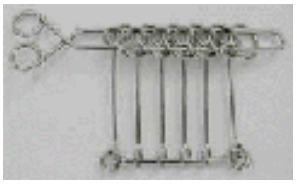
- Mettre $n$ pions sur le baguenaudier initialement vide.
- Prendre $n$ pions du baguenaudier initialement plein.

### Exemple d'exécution

```
Taille du jeu? 3
On remplit le baguenaudier...
. . .
* . .
* * .
. * .
. * *
* * *
Nombre de coups = 6
0 1 3 2 6 7
Appuyez sur une touche pour continuer...
On vide le baguenaudier...
* * *
. * *
. * .
* * .
* . .
. . .
Nombre de coups = 6
7 6 2 3 1 0
```

### Éditions du jeu

Il existe de nombreuses éditions de ce casse-tête, en métal ou en bois.



# 2 Classe Baguenaudier

On modélise le jeu du baguenaudier par une classe, et on représente un baguenaudier par un tableau $b$ de $n$ booléens telle que $b[k]$ est vrai s'il y a un pion sur la $k$-ème case et faux sinon.

Écrivez une classe `Baguenaudier` ayant pour attributs :

- Le nombre de pions `npions` (entier).
- La tablette `b` (vecteur de booléens).
- La liste des valeurs `vals` (liste d'entiers) générées lors d'un remplissage (montage) ou vidage (démontage) du baguenaudier.

Écrivez un constructeur à un paramètre d'entier `n` spécifiant le nombre de cases, qui initialise le nombre de pions ainsi que la taille du baguenaudier à `n` et la liste des valeurs à vide.

Écrivez un accesseur interne `getN` du nombre de pions.

Écrivez une méthode interne `initialiser` qui réinitialise la tablette à `Faux` avant de lancer le jeu.

Écrivez une méthode interne `afficher` qui visualise la tablette en affichant une astérisque (`*`) s'il y a un pion dans la case `k` et un point (`.`) sinon.

Écrivez une méthode interne `eval` qui calcule et renvoie l'évaluation du baguenaudier selon la méthode de Horner en base binaire.

Écrivez une méthode interne `afficherValeurs` qui affiche la liste des valeurs générées.

# 3 Stratégie du jeu

## 3.1 Analyse

« Jouer » c'est ôter ou placer un pion sur une case selon que celle-ci est ou non remplie. L'unique règle : à chaque coup les seules cases **jouables** sont la première case ou la case qui suit la première case occupée. Ce problème définit les deux types de jeux :

- **Mettre** $n$ pions sur le baguenaudier initialement vide.
- **Prendre** $n$ pions du baguenaudier initialement plein.

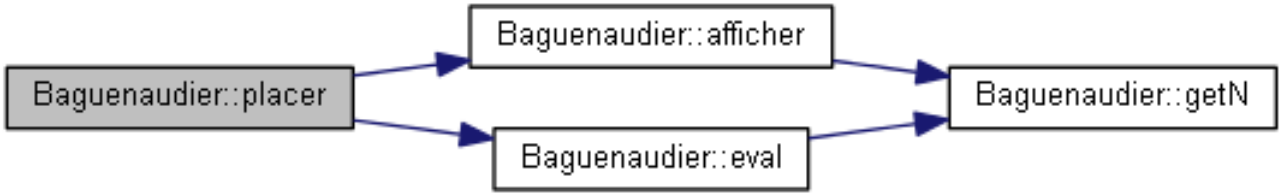
En supposant savoir placer $p$ pions ($p \leq n-1$) sur un baguenaudier $b[1..p]$, comment placer le pion $n$ ?

En supposant savoir prendre $p$ pions ($p \leq n-1$) d'un baguenaudier $b[1..p]$, comment prendre le pion $n$ ?

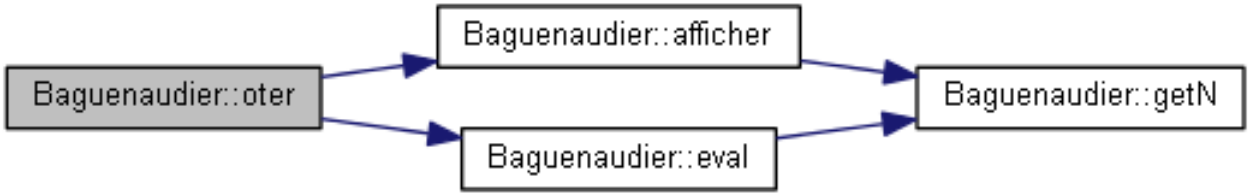
## 3.2 Algorithmique, Programmation

Ce problème réalise les stratégies de jeux.

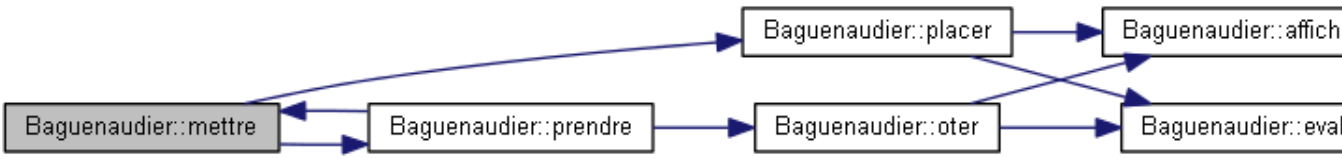
Écrivez une méthode interne `placer(k)` qui place le pion d'indice `k` (entier) sur le baguenaudier puis insère son évaluation (eval) dans la liste des valeurs.




De même, écrivez une méthode interne duale `oter(k)` qui ôte le pion d'indice `k` (entier) du baguenaudier puis insère son évaluation (eval) dans la liste des valeurs.

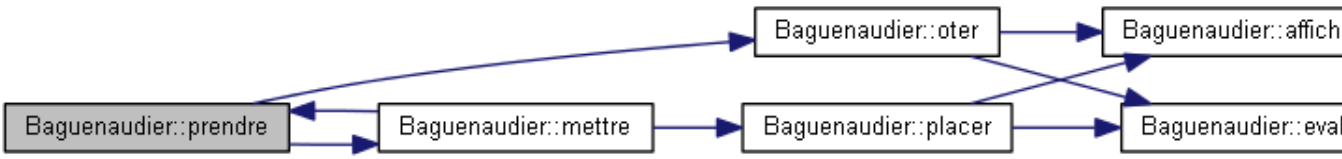



Écrivez une méthode interne **récursive croisée** `mettre(n)` qui met `n` (entier) pions sur le baguenaudier initialement vide.




**Aide simple**
Traduisez l'[Analyse].

De même, écrivez une méthode interne **récursive croisée** `prendre(n)` qui prend `n` (entier) pions du baguenaudier initialement plein.



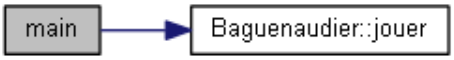

**Aide simple**
Traduisez l'[Analyse].

Déduisez une méthode interne `remplir` qui remplit le baguenaudier initialement vide.

De même, déduisez une méthode interne `vider` qui vide le baguenaudier initialement plein.

Écrivez une méthode `jouer` qui initialise le baguenaudier, le remplit puis le vide.

Écrivez un programme qui saisit la taille du jeu dans un entier `n`, instancie un `Baguenaudier` puis lance le jeu pour `n|` pions.




Testez. Exemple d'exécution :

```
Taille du jeu? 4
On remplit le baguenaudier...
. . . .
* . . .
* * . .
. * . .
. * * .
* * * .
* . * .
. . * .
. . * *
* . * *
* * * *
Nombre de coups = 11
0 1 3 2 6 7 5 4 12 13 15
Appuyez sur une touche pour continuer...
On vide le baguenaudier...
* * * *
```

```
* . * *
. . * *
. . * .
* . * .
* * * .
. * * .
. * . .
* * . .
* . . .
. . . .
Nombre de coups = 11
15 13 12 4 5 7 6 2 3 1 0
```

# 4 Etude, Complexité, Décurryfication

## 4.1 Nombres générés

Chaque case d'un baguenaudier $b$ ne peut prendre que deux états : libre ou occupée. En considérant que chaque $b[k]$ est équivalent à un chiffre binaire (0 pour libre et 1 pour occupé), chaque configuration est la représentation d'un entier en base 2.

**Exemple**
La configuration $b = [F, V, F, V, V, F, V, V]$ (où $F$ est `Faux`, $V$ est `Vrai`) représente l'entier 0b11011010 (valeur dans l'ordre inverse).

**Remarque**
Ce jeu engendre une suite de nombres et le passage d'un nombre au suivant s'effectue en changeant un unique chiffre binaire : c'est un **code de Gray**.

Quelle est la suite des nombres engendrés par le jeu `mettre` pour $n = 4$ ?

Vérifiez qu'il y a 218 nombres générés pour $n = 8$.

Déterminez la suite engendrée par chacun des jeux.

Généralisez par récurrence.

## 4.2 Complexité du jeu

Ce problème étudie la complexité du jeu en nombre d'opérations, en supposant que les opérations `oter` et `placer` un pion nécessitent le même temps.

Soient $P(n)$ et $M(n)$ le nombre d'opérations effectuées par `prendre(n)` et `mettre(n)`. Écrivez les relations vérifiées par $P(n)$ et $M(n)$.

Écrivez la relation de récurrence de la suite $A$ définie par :

$$\forall n \geq 0,\ P(n) = M(n) = A(n) + a \qquad (*)$$

Résolvez la relation pour que $A$ soit linéaire.

Concluez en terme de complexité des procédures récursives.

## 4.3 Décurryfication des procédures récursives

Ce problème (optionnel) réalise la décurryfication des procédures récursives.

Montrez que le jeu `mettre` est entièrement déterminé à chaque coup.

**Aide simple**

Vérifiez que le premier coup est imposé, puis utilisez le fait qu'un coup ne doit pas défaire ce qui a été fait au coup précédent pour constater que :

1. Le premier pion est joué un coup sur deux.
2. Les autres coups sont complètement déterminés.

Ainsi il n'y a aucun choix dans le jeu `mettre` : il est entièrement déterminé à chaque coup.

De même, montrez que le jeu `prendre` est entièrement déterminé à chaque coup.

Décurryfiez les procédures récursives.