

Triangle de Pascal [cb04] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel  algoprogram  Version 21 mai 2018

Table des matières

1	Triangle arithmétique de Pascal	2
2	Algorithmique, Programmation / pgtpascal	3
2.1	Calcul et affichage	3
2.2	Affichage de la parité	7
2.3	Calcul réduit	8
2.4	Vrai calcul réduit	10
3	Optionnel	13
3.1	Mode graphique	13
3.2	Vrai affichage graphique	14
4	Compléments	15
5	Références générales	15

Java - Triangle de Pascal (Solution)



Mots-Clés Combinatoire ■
Requis Axiomatique impérative (sauf Fichiers) ■
Optionnel Structuration de l'information, Graphique ■
Difficulté ●●○ (2 h 30) ■



Objectif

Cet exercice calcule les coefficients binomiaux grâce à la méthode du triangle de PASCAL. Ces coefficients sont ensuite affichés en fonction de leur parité.

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
```

1 Triangle arithmétique de Pascal



Définition

Les **coefficients binomiaux**, ou encore *nombres de PASCAL*, sont les coefficients du polynôme $(1 + x)^n$:

$$(1 + x)^n = 1 + \dots + \binom{n}{k} x^k + \dots + x^n = \sum_{k=0}^n \binom{n}{k} x^k$$

avec $\binom{n}{0} = 1$ et $\binom{n}{n} = 1$.



Propriété

Les coefficients binomiaux se calculent directement par (la définition) :

$$\binom{n}{k} = C_n^k = \frac{n!}{k!(n-k)!}$$

Ils peuvent aussi se calculer de façon récurrente à partir des coefficients du binôme précédent $(1 + x)^{n-1}$ par la relation :

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Ce calcul se fait alors en construisant le **triangle arithmétique de Pascal** ou *tableau binomial*.



Construction du triangle de Pascal

Ce tableau, inventé par BLAISE PASCAL (1623-1662), était déjà connu en Chine.

- Commencez avec 1.
- Ajoutez 1 au début et à la fin de chaque ligne.
- Tous les autres éléments du triangle de PASCAL s'obtiennent en faisant la somme de deux nombres adjacents dans la ligne juste au-dessus.

n = 0	1					
n = 1	1	1				
n = 2	1	2	1			
n = 3	1	3	3	1		
n = 4	1	4	6	4	1	
n = 5			←	= 10	= 5	1



Calculez la ligne $n = 5$.

Solution simple

On applique la méthode :

```
1 5(=1+4) 10(=4+6) 10(=6+4) 5(=4+1) 1
```



Les coefficients d'une ligne étant stockés dans un tableau, calculez la ligne $n = 6$ avec les données de la ligne 5, en mémorisant les résultats dans le même tableau. (Aide : Comment avez-vous procédé pour ne pas écraser les anciens coefficients avant de vous en servir ?)

Solution simple

Si on calcule les éléments de la gauche vers la droite on obtient :

```
1 6(=1+5) 16(=6+10)...
```

Comme il faut la valeur de l'élément à gauche, il faut calculer les éléments de la droite vers la gauche, d'où :

```
1 6(=1+5) 15(=5+10) 20(=10+10) 15(=10+5) 6(=5+1) 1
```

2 Algorithmique, Programmation / pgtpascal

2.1 Calcul et affichage

Ce problème calcule et affiche les lignes du triangle de PASCAL au fur et à mesure en stockant les coefficients binomiaux d'une ligne dans un tableau.



Définissez la constante entière `TMAX=100` (nombre maximal de coefficients d'une ligne) puis le type `Ligne`, structure contenant :

- Un entier `n` du numéro de la ligne.
- Un tableau `coefs`, d'au plus `TMAX` entiers indicé à partir de zéro, mémorisant les coefficients.



Écrivez une procédure `initialiserLigne(t)` qui initialise une `Ligne t`.

Aide simple

Initialisez l'entier `n` à `-1` et créez le tableau des `coef`.



Validez vos définitions avec la solution.

Solution Java @[pgtpascal.java]

```

/**
 * Taille maximale
 */
final static int TMAX = 100;

/**
 * Représente une Ligne
 */
public static class Ligne
{
    /// dimension de coefs
    public int n;
    /// tableau des coefficients
    public int[] coefs;
}

```

```

/**
 * Initialise une Ligne
 * @param[out] t - une Ligne
 */
public static void initialiserLigne(Ligne t)
{
    t.n = -1;
    t.coefs = new int[TMAX];
}

```



Écrivez une procédure `premiereLigne(t)` qui initialise une `Ligne t` à la première ligne ($n = 0$) du triangle de PASCAL.

Aide simple

Initialisez `n` à 0 et `coefs[n]` (donc `coefs[0]`) à 1.



Écrivez une procédure `ligneSuivante(t)` qui modifie une `Ligne t` en la ligne suivante du triangle de PASCAL.

Aide simple

Voir dernière question du problème [Le triangle arithmétique de Pascal].

Aide détaillée

Incrémentez `n`, initialisez `coefs[n]` à 1, puis réalisez le calcul des coefficients de la **droite vers la gauche** (c-à-d. par indice décroissant `j` de $n-1$ vers 1) et non pas le parcours classique de la gauche vers la droite (indice croissant). Notez qu'il est inutile d'initialiser la première case de `coefs` à 1, puisqu'elle contient déjà cette valeur.



Écrivez une procédure `afficherCoefs(t)` qui affiche une `Ligne t` du triangle de PASCAL.

Aide simple

Attention, ici :

- Sur la ligne 0 : il y a un coefficient
- Sur la ligne 1 : deux coefficients
- etc.



Validez vos procédures avec la solution.

Solution Java @[pgtpascal.java]

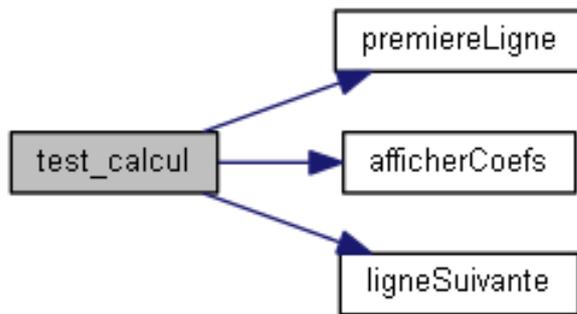
```
/**
 * Initialise la première Ligne du triangle de Pascal
 * @param[in,out] t - une Ligne
 */
public static void premiereLigne(Ligne t)
{
    t.n = 0;
    t.coefs[t.n] = 1;
}
```

```
/**
 * Passe à la Ligne suivante
 * @param[in,out] t - une Ligne
 */
public static void ligneSuivante(Ligne t)
{
    ++t.n;
    t.coefs[t.n] = 1;
    for (int j = t.n - 1; j >= 1; --j)
    {
        t.coefs[j] += t.coefs[j - 1];
    }
}
```

```
/**
 * Affiche les coefficients du triangle de Pascal
 * @param[in] t - une Ligne
 */
public static void afficherCoefs(final Ligne t)
{
    for (int j = 0; j <= t.n; ++j)
    {
        System.out.print(t.coefs[j] + " ");
    }
    System.out.println();
}
```



Écrivez une procédure `test_calcul` qui demande jusqu'à quelle ligne il faut afficher le triangle de PASCAL puis calcule et affiche chaque ligne du triangle de PASCAL jusqu'à cette ligne.



Testez. Exemple d'exécution :

```

Triangle de Pascal jusqu'à la ligne? 6
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
  
```



Validez votre procédure avec la solution.

Solution Java @[pgtpascal.java]

```

/**
 * @test
 */
public static void test_calcul()
{
    Scanner input = new Scanner(System.in);
    int nb;
    System.out.print("Triangle de Pascal jusqu'à la ligne? ");
    nb = input.nextInt();
    Ligne t = new Ligne();
    initialiserLigne(t);
    premiereLigne(t);
    afficherCoefs(t);
    for (int j = 1; j <= nb; ++j)
    {
        ligneSuiVante(t);
        afficherCoefs(t);
    }
}
  
```



Exécutez votre programme et tapez des valeurs de plus en plus élevées de `nb`. Que constatez-vous? Pourquoi?

2.2 Affichage de la parité

Ce problème affiche le triangle de PASCAL sous forme graphique (ici avec des caractères) en n'affichant non plus la valeur des coefficients mais leur parité.



Écrivez une procédure `afficherParite(t)` qui affiche une `Ligne t` du triangle de PASCAL de la façon suivante : pour chaque coefficient `coefs[j]` de `t`, affichez le caractère 'X' si le coefficient est impair, le caractère point '.' ou espace ' ' (au choix) sinon.



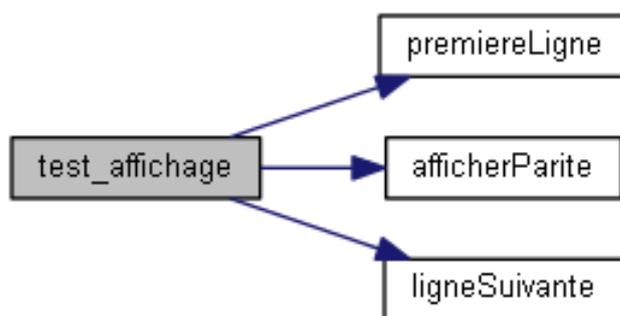
Validez votre procédure avec la solution.

Solution Java `@[pgtpascal.java]`

```
/**
 * Affiche les coefficients du triangle de Pascal
 * @param[in] t - une Ligne
 */
public static void afficherParite(final Ligne t)
{
    for (int j = 0; j <= t.n; ++j)
    {
        System.out.print((t.coefs[j] % 2 == 1 ? "X" : "."));
    }
    System.out.println();
}
```



Copiez/collez la procédure `test_calcul` en la procédure `test_affichage` puis modifiez-la pour afficher les coefficients avec la procédure `afficherParite`.



Testez. Exemple d'exécution :

```
Triangle de Pascal jusqu'à la ligne? 6
X
XX
X.X
XXXX
X...X
```

```
XX..XX
X.X.X.X
```



Validez votre procédure avec la solution.

Solution Java @[pgtpascal.java]

```
/**
 * @test
 */
public static void test_affichage()
{
    Scanner input = new Scanner(System.in);
    int nb;
    System.out.print("Triangle de Pascal jusqu'à la ligne? ");
    nb = input.nextInt();
    Ligne t = new Ligne();
    initialiserLigne(t);
    premiereLigne(t);
    afficherParite(t);
    for (int j = 1; j <= nb; ++j)
    {
        ligneSuivante(t);
        afficherParite(t);
    }
}
```

2.3 Calcul réduit

Comme vous avez pu le constater, la taille des coefficients dépasse rapidement la taille des entiers machine. Or on souhaite réaliser l'affichage graphique sur des triangles plus grands. Ce qui nous intéresse pour cet affichage, ce n'est pas la valeur des coefficients mais seulement leur parité. On remplace alors chaque coefficient par 1 s'il est impair, par 0 s'il est pair.



Écrivez une procédure `lignePariteSuivante(t)` qui modifie une `Ligne t` en la ligne suivante du triangle de PASCAL, les coefficients de `t` étant sous la forme 0/1 (pair/impair).

Aide simple

Étant donnés deux coefficients, on peut savoir en fonction de leur parité si leur somme est paire ou impaire.

Aide détaillée

Si la somme `coefs[j]+coefs[j-1]` vaut 1, affectez 1 à `coefs[j]`, 0 sinon (cas où la somme vaut 0 ou 2).



A-t-on besoin d'écrire une procédure `premiereLigneParite(t,n)` ou bien peut-on utiliser la procédure `premiereLigne` dans la nouvelle représentation de `t` ?



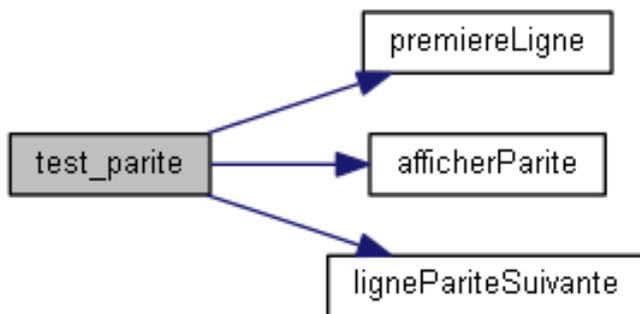
Validez votre procédure avec la solution.

Solution Java @[pgtpascal.java]

```
/**
 * Passe à la Ligne suivante
 * @param[in,out] t - une Ligne
 */
public static void lignePariteSuivante(Ligne t)
{
    ++t.n;
    t.coefs[t.n] = 1;
    for (int j = t.n - 1; j >= 1; --j)
    {
        t.coefs[j] = (t.coefs[j] + t.coefs[j - 1] == 1 ? 1 : 0);
    }
}
```



Écrivez une procédure `test_parite` qui fait comme la procédure `test_affichage` le calcul et l'affichage du triangle de PASCAL sous forme graphique mais en ne calculant que la parité des coefficients.



Testez. Exemple d'exécution :

```
Triangle de Pascal jusqu'à la ligne? 6
X
XX
X.X
XXXX
X...X
XX..XX
X.X.X.X
```



Validez votre procédure avec la solution.

Solution Java @[pgtpascal.java]

```

/**
 * @test
 */
public static void test_parite()
{
    Scanner input = new Scanner(System.in);
    int nb;
    System.out.print("Triangle de Pascal jusqu'a la ligne? ");
    nb = input.nextInt();
    Ligne t = new Ligne();
    initialiserLigne(t);
    premiereLigne(t);
    afficherParite(t);
    for (int j = 1; j <= nb; ++j)
    {
        lignePariteSuiivante(t);
        afficherParite(t);
    }
}

```

2.4 Vrai calcul réduit

Ne souhaitant calculer que la parité des coefficients, on représente la parité par un booléen : **Vrai** si le coefficient est pair, **Faux** sinon. **Attention**, dans le problème précédent, 0 est associé à pair et 1 à impair.

	Vrai 0 (pair)	Faux 1 (impair)	= coef t[j]
0 (pair) Vrai	0 (pair) Vrai	1 (impair) Faux	
1 (impair) Faux	1 (impair) Faux	0 (pair) Vrai	

= coef t[j-1]



Définissez le type `LigneParite`, structure contenant le numéro `n` (entier) de la ligne ainsi qu'un tableau `coefs` d'au plus `TMAX` booléens, indicé à partir de zéro, qui stockera la parité des coefficients d'une ligne de parité du triangle de PASCAL.



Écrivez une procédure `initialiserLigneParite(t)` qui initialise une `LigneParite t`.



Validez votre définition et procédure avec la solution.

Solution Java @[pgtpascal.java]

```

/**
 * Représente une LigneParite
 */
public static class LigneParite
{
    /// taille de coefs
    public int n;
    /// Les coefficients
    public boolean[] coefs;
}

/**
 * Initialise une LigneParite
 * @param[out] t - une LigneParite
 */
public static void initialiserLigneParite(LigneParite t)
{
    t.n = -1;
    t.coefs = new boolean[TMAX];
}

```



Écrivez une procédure `premiereLigneParite(t)` qui initialise une `LigneParite t` à la première ligne de parité du triangle de PASCAL. Pour cela, dupliquez la procédure `premiereLigne` puis modifiez-la afin de tenir compte de la nouvelle représentation.



Écrivez une procédure `lignePariteSuivante2(t)` qui modifie une `LigneParite t` en la ligne de parité suivante du triangle de PASCAL.

Aide simple

Dupliquez la procédure `lignePariteSuivante` puis modifiez-la afin de tenir compte de la nouvelle représentation.



Écrivez une procédure `afficherParite2(t)` qui affiche une `LigneParite t` du triangle de PASCAL de la façon suivante : pour chaque coefficient `coefs[j]` de `t`, affichez le caractère 'x' si le coefficient est **pair**, le caractère point '.' ou espace ' ' sinon.

Aide simple

Dupliquez la procédure `afficherParite` puis modifiez-la afin de tenir compte de la nouvelle représentation.



Validez vos procédures avec la solution.

Solution Java

@[pgtpascal.java]

```

/**
 * Initialise la première LigneParite du triangle de Pascal
 * @param[in,out] t - une LigneParite

```

```

*/
public static void premiereLigneParite(LigneParite t)
{
    t.n = 0;
    t.coefs[t.n] = false;
}

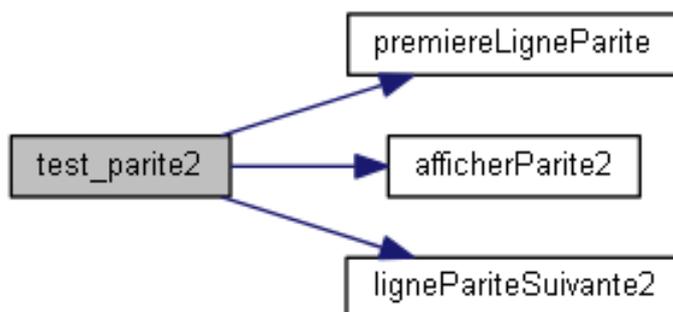
/**
    Passe à la LigneParite suivante
    @param[in,out] t - une LigneParite
*/
public static void lignePariteSuivante2(LigneParite t)
{
    ++t.n;
    t.coefs[t.n] = false;
    for (int j = t.n - 1; j >= 1; --j)
    {
        t.coefs[j] = (t.coefs[j] == t.coefs[j - 1]);
    }
}

/**
    Affiche les coefficients du triangle de Pascal
    @param[in] t - une LigneParite
*/
public static void afficherParite2(final LigneParite t)
{
    for (int j = 0; j <= t.n; ++j)
    {
        System.out.print((!t.coefs[j] ? "X" : "."));
    }
    System.out.println();
}

```



Écrivez une procédure `test_parite2` similaire à la procédure `test_parite` qui fait le calcul et l'affichage du triangle de PASCAL sous forme graphique en tenant compte de la nouvelle représentation.



Testez. Exemple d'exécution :

```
Triangle de Pascal jusqu'à la ligne? 6
X
XX
X.X
XXXX
X...X
XX..XX
X.X.X.X
```



Testez avec de grandes valeurs de `nb` (par exemple 60, @[rstpascal4b.txt]).

Commentaires

Remarquez la régularité. Le dessin obtenu ressemble beaucoup à un dessin bien connu que les mathématiciens appellent le **triangle de Sierpinski** (du nom de son inventeur). (<http://www.mathcurve.com/fractals/sierpinski/sierpinskitriangle.shtml>)



Validez votre procédure avec la solution.

Solution Java @[pgtpascal.java]

```
/**
 * @test
 */
public static void test_parite2()
{
    Scanner input = new Scanner(System.in);
    int nb;
    System.out.print("Triangle de Pascal jusqu'a la ligne? ");
    nb = input.nextInt();
    LigneParite t = new LigneParite();
    initialiserLigneParite(t);
    premiereLigneParite(t);
    afficherParite2(t);
    for (int j = 1; j <= nb; ++j)
    {
        lignePariteSuivante2(t);
        afficherParite2(t);
    }
}
```

3 Optionnel

3.1 Mode graphique



Mode standard

Il y a renversement des axes : le point (0,0) se trouve dans le coin supérieur gauche.

3.2 Vrai affichage graphique

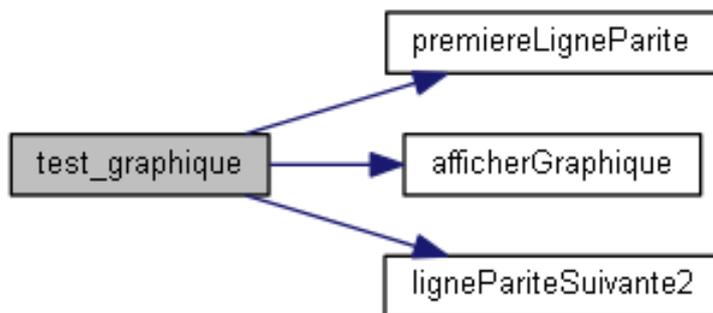
Ce problème utilise la bibliothèque graphique pour afficher le triangle de PASCAL sous forme graphique (un pixel par coefficient).



Validez votre procédure avec la solution.



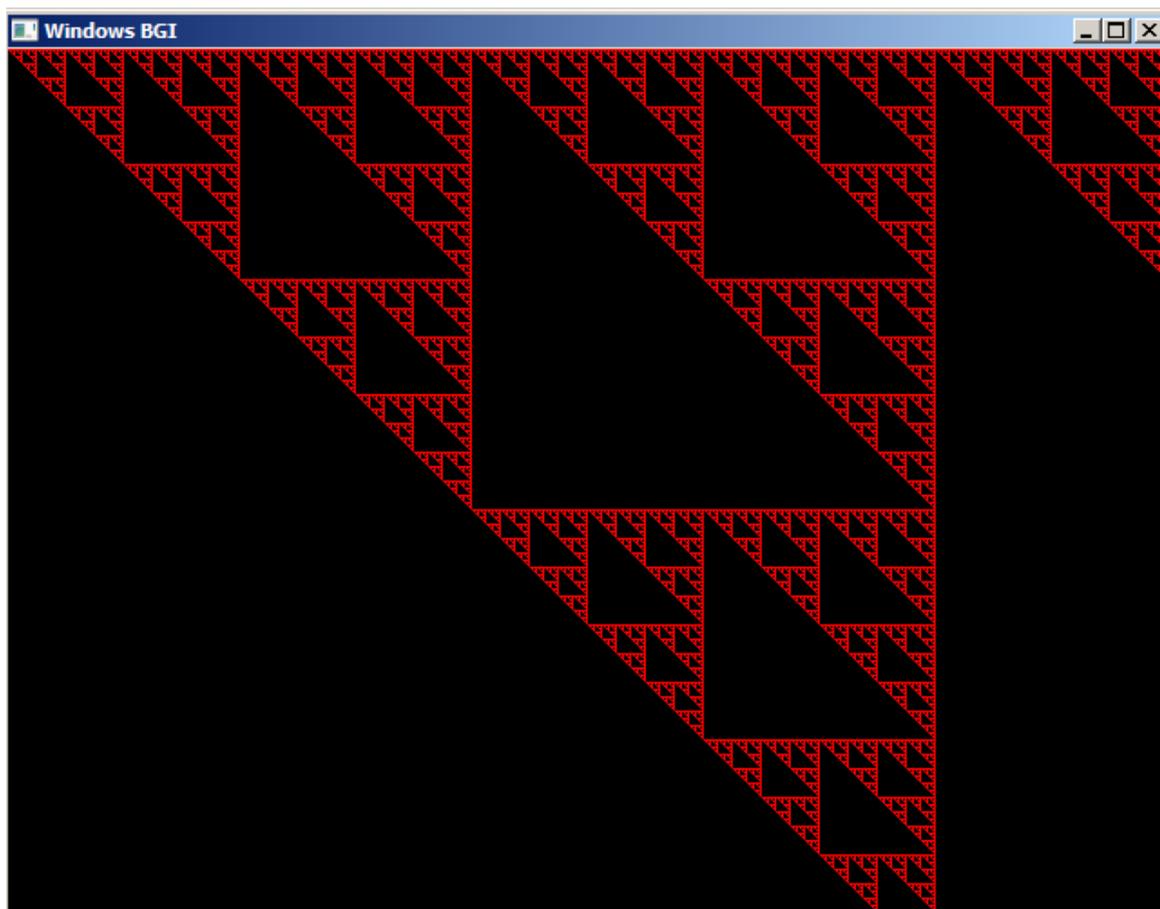
Écrivez une procédure `test_graphique` qui fait comme la procédure `test_parite2` le calcul et l’affichage du triangle de PASCAL mais sur l’écran graphique.



Modifiez `TMAX` en 800 ou 1000.



Testez (ici `nb=600`). Exemple d’exécution :



Validez votre procédure avec la solution.

4 Compléments

Ce problème utilise les propriétés des coefficients binomiaux afin de réaliser quelques optimisations de calcul.



Le tableau est symétrique : $\binom{n}{k} = \binom{n}{n-k}$

5 Références générales

Comprend ■