# Le Carré Magique

On appelle carré magique d'ordre n un carré n x n dont les éléments sont des nombres entiers 1, 2, …, $n^2$ disposés de telle sorte que la somme des n nombres situés sur une ligne, sur une colonne ou sur une diagonale soit toujours égale à : $\frac{n(n^2+1)}{2}$.

Exemple : carré magique d'ordre 5 (n=5 et $\frac{n(n^2+1)}{2} = 65$)

| | | | | |
|---|---|---|---|---|
| 17 | 24 | 1 | 8 | 15 |
| 23 | 5 | 7 | 14 | 16 |
| 4 | 6 | 13 | 20 | 22 |
| 10 | 12 | 19 | 21 | 3 |
| 11 | 18 | 25 | 2 | 9 |

Dans tout ce qui suit, on considère n impair.

Le but de ce développement est :

- De permettre à un utilisateur de saisir une matrice d'entiers et de déterminer si celle-ci représente un carré magique.
- De générer des carrés magiques pout tout n impair < 100.
- De présenter ces deux fonctionnalités sous forme d'un menu.

Vous allez développer ces fonctionnalités de manière incrémentale :

1. La version 1 permet de renseigner et d'afficher une matrice carrée d'ordre n.
2. La version 2 permet de tester si une matrice carrée est magique ou non.
3. La version 3 permet de générer et afficher une matrice carrée d'ordre n avec n impair et n < 100.
4. La version 4 permet de regrouper ces deux fonctionnalités dans un menu.
5. Qu'en est-il des carrés magiques d'ordre pair ? Ils existent aussi et sont beaucoup plus difficiles à générer. Nous en verrons un exemple pour les matrices carrés d'ordre 6 !

## 1. Travail à faire pour la Version 1

Définir en C++ le type matrice carrée d'entiers d'ordre TAILLE (Carre) où TAILLE désigne la constante 100.

Définir et implémenter en C++ les fonctions de prototype suivant :

```
 -- initialise une matrice carrée d'entiers d'ordre n à zéro
-- lève l'exception "ORDRE INCORRECT" si n ne vérifie pas 0 ≤ n ≤ TAILLE
procédure initialiserCarré (sortie carré <Carré>, entrée n <Entier>);

-- lit une matrice carrée d'ordre n d'entiers désignée par carré à partir
-- de valeurs saisies au clavier
-- lève l'exception "ORDRE INCORRECT" si n ne vérifie pas 0 ≤ n ≤ TAILLE
procédure lireCarré (sortie carré <Carré>, entrée n <Entier>) ;

-- affiche une matrice carrée d'ordre n d'entiers désignée par carré
-- lève l'exception "ORDRE INCORRECT" si n ne vérifie pas 0 ≤ n ≤ TAILLE
procédure écrireCarré (entrée carré <Carré>, entrée n <Entier>) ;
```

Ecrire un programme de test permettant de saisir une valeur n, d'initialiser la matrice carré et de l'afficher.

Ecrire un programme de test permettant de saisir une valeur n, de renseigner une matrice carré et de l'afficher.

L'affichage permettra de visualiser une matrice et non une suite de valeurs.
On supposera les valeurs de n renseignées correctement.

***Remarques :***

- Il est conseillé d'utiliser les variables nommées i et j pour gérer les parcours sur les tableaux et les matrices. La variable i représente l'indice d'un élément dans un tableau ou l'indice d'une ligne dans une matrice. La variable j représente l'indice d'une colonne dans une matrice. Matrice[i][j] représente alors l'élément présent à la ligne i et à la colonne j.
- Vous utiliserez des boucles for imbriquées pour le traitement séquentiel des éléments dans une matrice.

## 2. Travail à faire pour la Version 2

Soit la donnée d'une matrice carrée d'ordre impair, il s'agit maintenant d'implémenter le test d'un carré magique. Il s'agit de calculer la somme des lignes, des colonnes, de la première et deuxième diagonale, de vérifier que tous ces nombres sont égales à la valeur escomptée. Plus formellement :

$$
\begin{aligned}
&estCarreMagique(M) = vraie \\
&\Leftrightarrow (\forall\, i \in [0, n-1]: sommeLigne(M, i) = valeur) \wedge \\
&\quad (\forall\, j \in [0, n-1] : sommeColonne(M, j) = valeur) \wedge \\
&\quad somme1ereDiagonale(M) = somme2emeDiagonale(M) = valeur \\
&avec\ M\ \ de\ type\ \text{int}[n][n] \wedge\ \ 0 < n < 100\ \ \wedge\ \ impair(n) \wedge valeur = \frac{n(n^2+1)}{2}
\end{aligned}
$$

Les résultats des calculs seront stockés dans un tableau de 2n + 2 valeurs et le test d'égalité des valeurs se fera par l'appel à la fonction élémentsEgaux.

Définir en C++ le type tableau de (TAILLE * 2 + 2) entiers (TabEntiers) où TAILLE désigne la constante 100.

Définir et implémenter dans l'ordre donné les fonctions de prototype suivant :

```
-- détermine si l'entier i est un nombre impair
fonction estImpair (entrée i <Entier>) retourne <Booléen> ;

-- calcule et renvoie la somme d'une ligne i d'une matrice carrée
-- d'entiers d'ordre n désignée par carré
-- lève l'exception "LIGNE INVALIDE" si i ne vérifie pas 0 ≤ i ≤ n-1
-- lève l'exception "ORDRE INCORRECT" si n ne vérifie pas 0 ≤ n ≤ TAILLE
fonction sommeLigne (entrée carré <Carré>, entrée n <Entier>, entrée i <Entier>)
retourne <Entier> ;

-- calcule et renvoie la somme d'une colonne j d'une matrice carrée
-- d'entiers d'ordre n désignée par carré
-- lève l'exception "COLONNE INVALIDE" si j ne vérifie pas 0 ≤ j ≤ n-1
-- lève l'exception "ORDRE INCORRECT" si n ne vérifie pas 0 ≤ n ≤ TAILLE
fonction sommeColonne (entrée carré <Carré>, entrée n <Entier>, entrée j <Entier>)
retourne <Entier> ;

-- calcule et renvoie la somme de la première diagonale d'une matrice carrée
-- d'entiers d'ordre n désignée par carré
-- lève l'exception "ORDRE INCORRECT" si n ne vérifie pas 0 ≤ n ≤ TAILLE
fonction somme1èreDiagonale (entrée carré <Carré>, entrée n <Entier>)
retourne <Entier> ;

-- calcule et renvoie la somme de la deuxième diagonale d'une matrice carrée
-- d'entiers d'ordre n désignée par carré
-- lève l'exception "ORDRE INCORRECT" si n ne vérifie pas 0 ≤ n ≤ TAILLE
fonction somme2èmeDiagonale (entrée carré <Carré>, entrée n <Entier>)
retourne <Entier> ;

-- détermine si tous les éléments d'un tableau d'entiers tab de n éléments
-- sont égaux à la valeur val
-- lève l'exception "LONGUEUR INCORRECTE" si n ne vérifie pas 0 ≤ n ≤ NB_SOMMES
fonction elémentsEgaux
    (entrée tab <TabEntier>, entrée n <Entier>, entrée val <Entier>)
retourne <Booléen> ;

-- détermine si une matrice carrée d'ordre n désignée par carré est magique
-- lève l'exception "ORDRE INCORRECT" si n ne vérifie pas 0 ≤ n ≤ TAILLE
-- lève l'exception "ORDRE PAIR" si n est pair
fonction estCarréMagique (entrée carré <Carré>, entrée n <Entier>)
retourne <Booléen> ;
```

### *Remarques :*

- Pour faire les tests de vos fonctions au fur et à mesure, vous n'allez pas saisir des matrices carrés mais utiliser un jeu de test que vous aurez construit en initialisant des variables de type Carre.
- Les traitements des fonctions sommeXXXX se feront à l'aide d'une boucle simple for. Il est inutile de parcourir tous les éléments de la matrice.

Exemple d'un test :

```
// déclaration d'une matrice carre 5 x 5
   int ok5[5][5] = { 17, 24,  1,  8, 15,
                     23,  5,  7, 14, 16,
                      4,  6, 13, 20, 22,
                     10, 12, 19, 21,  3,
                     11, 18, 25,  2,  9  };
   // recopie dans structure de données Carré
   Carre ok;
   for (int i =0; i < 5; i++)
   {
       for (int j =0; j < 5; j++)
       {
           ok[i][j] = ok5[i][j];
       }
   }
```

## 3. Travail à faire pour la Version 3

Il nous reste à générer un carré magique pout tout n impair. Ce procédé « simple » a été ramené de Chine par monsieur De la Loubère, mathématicien et accessoirement ambassadeur de France au Siam. Ce procédé ne fonctionne que pour les matrices carrés d'ordre impair, il en existe d'autres plus complexes traitant les carrés magiques d'ordre pair.
Implémentez la procédure suivante et testez !

```
// construit un carré magique d'ordre n
// lève l'exception "ORDRE PAIR" si n est pair
// lève l'exception "ORDRE INCORRECT" si n ne vérifie pas 0 <= n <= TAILLE
void construireCarreMagique (Carre carre, const int n) throw (Chaine) ;
```

Le procédé est le suivant :

- ***Initialisation :***

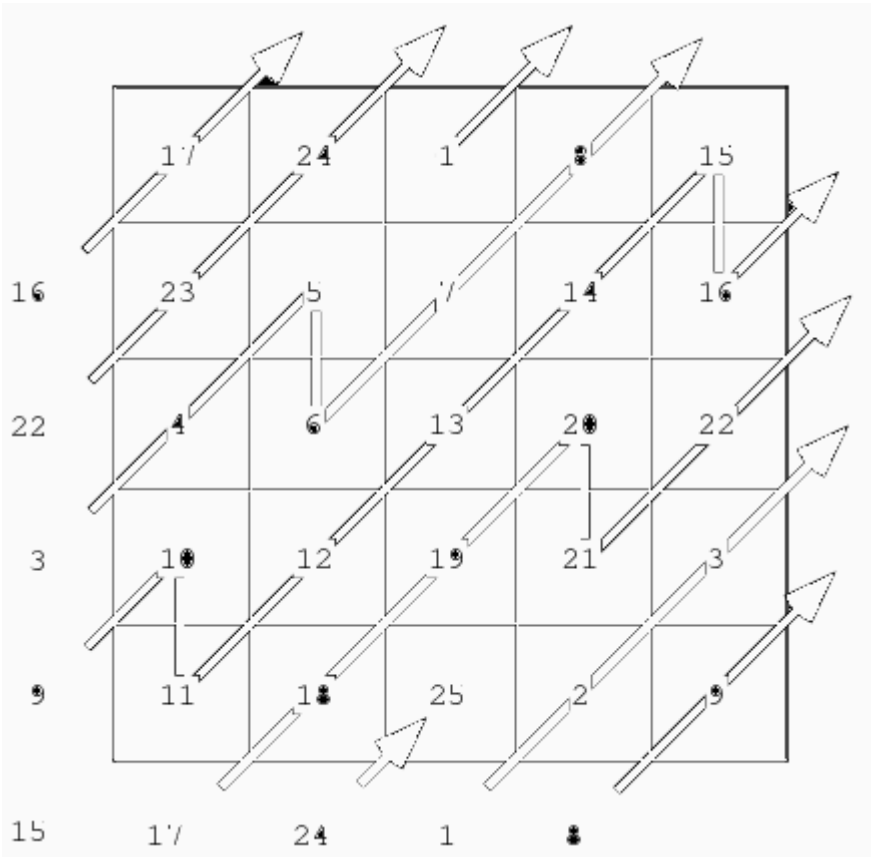
Placer le nombre 1 dans la case du milieu de la ligne supérieure.

- ***Traitement itératif :***

Puis écrire dans l'ordre les nombres de 2 à $n^2$ en s'élevant pour chacun diagonalement vers la droite et en observant les règles suivantes :

1. Lorsqu'on arrive à la ligne horizontale supérieure, placer le nombre suivant dans la ligne horizontale inférieure (comme si cette dernière était transportée à la partie supérieure du carré).
2. Lorsqu'on atteint la colonne de droite, placer le nombre suivant dans la première colonne de gauche (comme si cette dernière suivait immédiatement la dernière de droite).
3. Lorsqu'on arrive sur une case occupée, placer le nombre suivant dans la case immédiatement au dessous de celle qui contient le dernier nombre inscrit.

Ce traitement est simulé à la figure suivante pour une matrice carré d'ordre 5.



## 4. Travail à faire pour la Version 4

Ecrire un programme C++ utilisant les fonctions implémentées et qui à l'aide d'un menu permet d'aiguiller sur les options suivantes :

```
Que voulez-vous faire ?

1.   Saisir une matrice carree et determiner si elle correspond a un carre magique
2.   Calculer et afficher le carre magique d'une matrice carree d'ordre n
3.   Quitter l'application

Votre choix : _
```

## 5. Les carrés magiques d'ordre 6

Pour ce cas précis et en fonction des matrices de départ données ce n'est pas plus difficile que pour les carrés d'ordre impair

Soit la donnée d'un carré magique d'ordre 3, on supposera qu'il est renseigné correctement par une constante nommée Ordre3 de type matrice 3x3 d'entiers.

| **Ordre 3** | | |
|---|---|---|
| 8 | 1 | 6 |
| 3 | 5 | 7 |
| 4 | 9 | 2 |

Soit la donnée d'un carré particulier composé uniquement des chiffres 0, 1, 2 et 3 de telle sorte que toutes les sommes soient égales à 9. On supposera qu'il est renseigné correctement par une constante nommée Medjig de type matrice 6x6 d'entiers.

| **Medjig** | | | | | |
|---|---|---|---|---|---|
| 2 | 3 | 0 | 2 | 0 | 2 |
| 1 | 0 | 3 | 1 | 3 | 1 |
| 3 | 1 | 1 | 2 | 2 | 0 |
| 0 | 2 | 0 | 3 | 3 | 1 |
| 3 | 2 | 2 | 0 | 0 | 2 |
| 0 | 1 | 3 | 1 | 1 | 3 |

Le procédé pour calculer un carré magique d'ordre 6 est alors le suivant :

- Multiplier toutes les cases du carré Medjig par 9.
- Divisez votre carré 6x6 obtenu en 9 carrés de 2x2. Vous obtenez alors un carrré 3x3 de carré 2x2, comme indiqué sur la figure.
- A chaque case d'un carré 2x2 rajoutez l'entier correspondant dans le carré magique d'ordre 3.

Et vous devriez obtenir ce carré magique d'ordre 6 :

| **Ordre 6** | | | | | |
|---|---|---|---|---|---|
| 26 | 35 | 1 | 19 | 6 | 24 |
| 17 | 8 | 28 | 10 | 33 | 15 |
| 30 | 12 | 14 | 23 | 25 | 7 |
| 3 | 21 | 5 | 32 | 34 | 16 |
| 31 | 22 | 27 | 9 | 2 | 20 |
| 4 | 13 | 36 | 18 | 11 | 29 |

Ecrivez alors une petite procédure permettant d'obtenir ce carré magique.