

Partie I Construction d'un carré magique

Il existe plusieurs méthode pour construire un carré magique d'ordre impaire (Damier crénelé, siamoise, losange, serpent ...).

Dans cette épreuve, on s'intéresse seulement à la méthode du damier crénelé.

Question 1)

- a) On sait que le tableau est composé de n^2 cases avec n lignes et n colonnes dont chaque case est affectée par un nombre compris entre 1 et n^2 (sans répétition).

Donc la somme de tous les nombres dans les n^2 cases est :

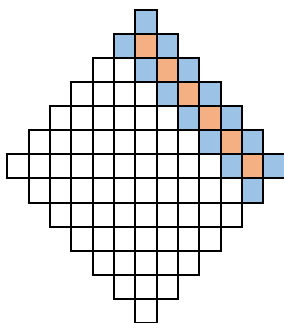
$$1 + 2 + 3 \dots + n^2 = n^2 \times \frac{n^2 + 1}{2}$$

Et puisqu'il y a n cases et que chaque ligne (resp. colonnes) possède la même somme, soit m et la somme pour une ligne (resp. colonne).

On a $1 + 2 + 3 \dots + n^2 = m + m + \dots + m$ (n fois) donc

$$m = \frac{1}{n} \frac{n^2(n^2+1)}{2} = \frac{n}{2} \times (n^2 + 1)$$

- b) On a



On a entre chaque deux ligne consécutives (bleu) de longueur n (n cases), existe une ligne (orange) de $n-1$ cases vides.

- c) Pour un damier qui est assimilable à un losange de longueur n , existe n lignes, entre chaque 2 lignes existe une ligne de longueur $n-1$, donc $n-1$ ligne de longueur $n-1$ et n lignes de longueur n : le nombre totale de cas :

$$n \times n + (n - 1) \times (n - 1) = n^2 + (n - 1)^2$$

Question 2)

Initialisation d'une matrice carrée d'ordre $(2n-1)$

```
int main(void) {
    int **P,n,i,j,c=1;
    //Entrer la taille du tableau
    scanf("%d",&n);
    //Allocation dynamique d'une matrice d'ordre  $(2*n-1)*(2*n-1)$ 
    P=(int **)malloc(sizeof(int**)*(2*n-1));
    for(i=0;i<2*n-1;i++)
        P[i]=(int *)malloc(sizeof(int *)*(2*n-1));

    //Initialisation des elements du tableau par 0
    //Normalement le tableau s'initialise automatiquement par 0
    /*for(i=0;i<2*n-1;i++)
        for(j=0;j<2*n-1;j++)
            P[i][j]=0;*/
```

Question 3)

Méthode 1 :

```
//Remplissage du tableau
for(j=0;j<n;j++)
    for(i=n-1;i<2*n-1;i++)
    {
        P[j+(i+1-n)][i-j]=c++;
    }
```

Méthode 2 :

```
for(i=0;i<n;i++)
{
    for(j=0;j<n;j++)
        P[i+j][n-1-i+j]=c++;
}
```

Explication :

Méthode 1 :

L'indice $j+(i+1-n)$ prend les valeurs $j, j+1, \dots, j+n-1$ (Décalage au bas).

L'indice $i-j$ prends les valeur $n-1-j, n-j, \dots, 2n-2-j$ (Décalage à droite).

Le point de départ et $(j, n-1-j)$ qui prend les valeurs $(0, n-1) ; (1, n-2) \dots (n-1 ; 0)$.

Pour notre exemple : (0,6),(1,5)...(6,0)

Méthode 2 :

L'indice $i+j$ prend les valeurs $0,1,2,\dots,n-1$; $1,2,3,\dots,n$; .. ; $n-2,n,\dots,2n-1$.

L'indice $n-1-i+j$ prend les valeurs $n-1,n,\dots,2n-1$; $n-2,n-1,\dots,2n-2$; $0,1,\dots,n-1$

j \ i	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	8	0	2
2	0	0	0	0	15	0	9	0
3	0	0	0	22	0	16	0	10
4	0	0	29	0	23	0	17	0
5	0	36	0	30	0	24	0	18
6	43	0	37	0	31	0	25	0

Question 4)

Méthode 1 :

```
//Decalage
for(i=0;i<n/2;i++)
{
    for(j=n/2+1;j<2*n-1-(n/2+1);j++)
    {
        if(P[i][j]!=0)
        {
            //On peut affecter les 3 quadrants
            //en utilisant un seul parcours
            P[i+n][j]=P[i][j];
            P[j][i+n]=P[j][i];
            P[n-2-i][j]=P[2*(n-1)-i][j];
            P[j][n-2-i]=P[j][2*(n-1)-i];
        }
    }
}
```

Explication :

Pour cet exemple, i prend les valeurs $0, 1, 2$; j prend les valeurs $4, 5, 6, 7, 8$;

Il balaye la partie verte de la matrice.

Si on prend $(i,j)=(2,4)$ ($P[i][j]=15$)

On aura :

- 1- $P[i+n][j]=P[9][4]$ affectée par $P[2][4]$ (a pour valeur 15) (1^{ère} quadrant)
- 2- $P[j][n+i]=P[4][9]$ affectée par $P[4][2]$ (a pour valeur 29) (2^{ème} quadrant)
- 3- $P[n-2-i][j]=P[3][4]$ affectée par $P[10][4]$ (a pour valeur 47) (3^{ème} quadrant)
- 4- $P[j][n-2-i]=P[4][3]$ affectée par $P[4][10]$ (a pour valeur 5) (4^{ème} quadrant)

Method 2 :

Le problème avec l'autre méthode, c'est qu'on doit comparer $P[i][j]$ avec 0 donc on doit initialiser toutes les cases par 0 dès le début.

Normalement pour certain compilateurs, ils initialisent automatiquement les cases par 0, pour enlever l'ambiguïté, on peut tester si $P[i][j]$ se trouve dans les diagonales $((1,8,15,...) ; (2,9,16,...) ; (3,10,17,...)..)$.

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	8	0	2	0	0	0	0	0
2	0	0	0	0	15	0	9	0	3	0	0	0	0
3	0	0	0	22	0	16	0	10	0	4	0	0	0

Si vous remarque bien les positions des éléments (1,8,15,9,2,3)

Vous allez constater qu'ils se trouvent aux points dont les indices respectent l'inégalité suivante

$$\begin{cases} j+i \geq n-1 \\ j-i \leq n-1 \\ j+i \text{ paire} \\ j-i \text{ paire} \end{cases} \rightarrow \begin{cases} j+i \geq n-1 \\ j-i \leq n-1 \\ (j\%2) == (i\%2) \end{cases}$$

Pour cela il faut seulement comparer $i+j$ et $j-i$ avec $n-1$ et tester leur parité.

Pour notre exemple on les seuls points respectant l'inégalité sont

$(4,2); (5,1); (6,0); (6,2); (7,1); (8,0)$

Le code sera :

```
//2eme Damier
for(i=0;i<n/2;i++)
{
    for(j=n/2+1;j<2*n-1-(n/2+1);j++)
    {
        if((j+i>=n-1)&&(j-i<=n-1)&&(j%2==i%2))
        {
            //On peut affecter les 3 quadrants
            //En utilisant un seul parcours
            P[i+n][j]=P[i][j];
            P[j][i+n]=P[j][i];
            P[n-2-i][j]=P[2*(n-1)-i][j];
            P[j][n-2-i]=P[j][2*(n-1)-i];
        }
    }
}
```

Method 3 :

C'est une méthode banale consiste à parcourir chaque quadrant, elle est couteuse en terme de mémoire et de temps.

j \ i	0	1	2	3	4	5	6	7	8	9	10	11	12
0							1						
1						8		2					
2					15		9		3				
3				22		16		10		4			
4			29		23		17		11		5		
5		36		30		24		18		12		6	
6	43		37		31		25		19		13		7
7		44		38		32		26		20		14	
8			45		39		33		27		21		
9				46		40		34		28			
10					47		41		35				
11						48		42					
12							49						

Tableau pour n=7 illustrant les quadrants, diagonale ...

Le **vert** désigne le quadrant principal

Le **jaune** désigne les 3 quadrants restants qu'on déduit partir du quadrant principale

Question 5)

```
//Vider la memoire
for(i=0;i<2*n-1;i++)
    free(P[i]);
free (P);
```

Exemples :

Pour n=3 on a :

4	9	2
3	5	7
8	1	6

Carré magique.

0	0	1	0	0
0	4	0	2	0
7	0	5	0	3
0	8	0	6	0
0	0	9	0	0

Matrice.

Pour n=5 on a ;

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

Carré magique.

0	0	0	0	1	0	0	0	0
0	0	0	6	0	2	0	0	0
0	0	11	0	7	0	3	0	0
0	16	0	12	0	8	0	4	0
21	0	17	0	13	0	9	0	5
0	22	0	18	0	14	0	10	0
0	0	23	0	19	0	15	0	0
0	0	0	24	0	20	0	0	0
0	0	0	0	25	0	0	0	0

Matrice.

Correction Examen Langage C et Algorithmique (Corrigé par Ayoub Dergaoui)

Pour $n=7$ (déjà cité en exemple)

22	47	16	41	10	35	4
5	23	48	17	42	11	29
30	6	24	49	18	36	12
13	31	7	25	43	19	37
38	14	32	1	26	44	20
21	39	8	33	2	27	45
46	15	40	9	34	3	28

Carré magique

j \ i	0	1	2	3	4	5	6	7	8	9	10	11	12
0							1						
1						8		2					
2					15		9		3				
3				22	47	16	41	10	35	4			
4			29	5	23	48	17	42	11	29	5		
5		36		30	6	24	49	18	36	12		6	
6	43		37	13	31	7	25	43	19	37	13		7
7		44		38	14	32	1	26	44	20		14	
8			45	21	39	8	33	2	27	45	21		
9				46	15	40	9	34	3	28			
10					47		41		35				
11						48		42					
12							49						

1^{er} Damier et 2^{ème} Damier.

Correction Examen Langage C et Algorithmique (Corrigé par Ayoub Dergaoui)

Pour n=9 : (Avec couleur indicative)

37	78	29	70	21	62	13	54	5
6	38	79	30	71	22	63	14	46
47	7	39	80	31	72	23	55	15
16	48	8	40	81	32	64	24	56
57	17	49	9	41	73	33	65	25
26	58	18	50	1	42	74	34	66
67	27	59	10	51	2	43	75	35
36	68	19	60	11	52	3	44	76
77	28	69	20	61	12	53	4	45

Carré magique.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	10	0	2	0	0	0	0	0	0	0
2	0	0	0	0	0	0	19	0	11	0	3	0	0	0	0	0	0
3	0	0	0	0	0	28	0	20	0	12	0	4	0	0	0	0	0
4	0	0	0	0	37	78	29	70	21	62	13	54	5	0	0	0	0
5	0	0	0	46	6	38	79	30	71	22	63	14	46	6	0	0	0
6	0	0	55	0	47	7	39	80	31	72	23	55	15	0	7	0	0
7	0	64	0	56	16	48	8	40	81	32	64	24	56	16	0	8	0
8	73	0	65	0	57	17	49	9	41	73	33	65	25	0	17	0	9
9	0	74	0	66	26	58	18	50	1	42	74	34	66	26	0	18	0
10	0	0	75	0	67	27	59	10	51	2	43	75	35	0	27	0	0
11	0	0	0	76	36	68	19	60	11	52	3	44	76	36	0	0	0
12	0	0	0	0	77	28	69	20	61	12	53	4	45	0	0	0	0
13	0	0	0	0	0	78	0	70	0	62	0	54	0	0	0	0	0
14	0	0	0	0	0	0	79	0	71	0	63	0	0	0	0	0	0
15	0	0	0	0	0	0	0	80	0	72	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	81	0	0	0	0	0	0	0	0

1^{er} Damier et 2^{ème} Damier.

NB : vous pouvez consulter le code complet, ainsi le fichier PDF sur le lien

<https://github.com/AyoubDergaoui/CorrectionExamen>