

Home	Algorithms	DS	GATE	Interview Corner	Q&A	C	C++	Java	Books	Contribute	Ask a Q	About
Array	Bit Magic	C/C++	Articles	GFactS	Linked List	MCQ	Misc	Output	String	Tree	Graph	

Magic Square

A **magic square** of order n is an arrangement of n^2 numbers, usually distinct integers, in a square, such that the n numbers in all rows, all columns, and both diagonals sum to the same constant. A magic square contains the integers from 1 to n^2 .

The constant sum in every row, column and diagonal is called the **magic constant** or **magic sum**, M . The magic constant of a normal magic square depends only on n and has the following value:
 $M = n(n^2+1)/2$

For normal magic squares of order $n = 3, 4, 5, \dots$, the magic constants are: 15, 34, 65, 111, 175, 260, ...

In this post, we will discuss how programmatically we can generate a magic square of size n . Before we go further, consider the below examples:

Magic Square of size 3

```

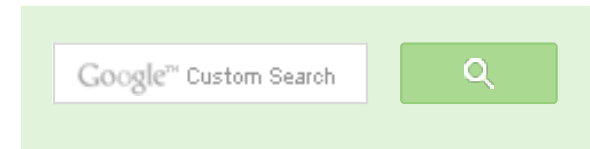
-----
2   7   6
9   5   1
4   3   8
    
```

Sum in each row & each column = $3 \cdot (3^2+1)/2 = 15$

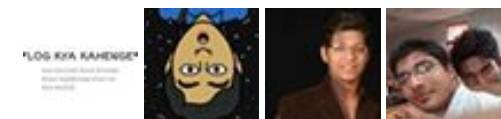
Magic Square of size 5

```

-----
9   3  22  16  15
2  21  20  14   8
    
```



53,525 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

25	19	13	7	1
18	12	6	5	24
11	10	4	23	17

Sum in each row & each column = $5 \cdot (5^2 + 1) / 2 = 65$

Magic Square of size 7

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

Sum in each row & each column = $7 \cdot (7^2 + 1) / 2 = 175$

Did you find any pattern in which the numbers are stored?

In any magic square, the first number i.e. 1 is stored at position $(n/2, n-1)$. Let this position be (i, j) . The next number is stored at position $(i-1, j+1)$ where we can consider each row & column as circular array i.e. they wrap around.

Three conditions hold:

1. The position of next number is calculated by decrementing row number of previous number by 1, and incrementing the column number of previous number by 1. At any time, if the calculated row position becomes -1, it will wrap around to $n-1$. Similarly, if the calculated column position becomes n , it will wrap around to 0.
2. If the magic square already contains a number at the calculated position, calculated column position will be decremented by 2, and calculated row position will be incremented by 1.
3. If the calculated row position is -1 & calculated column position is n , the new position would be: $(0, n-2)$.

Example:

Magic Square of size 3



Popular Posts

[All permutations of a given string](#)

[Memory Layout of C Programs](#)

[Understanding "extern" keyword in C](#)

[Median of two sorted arrays](#)

[Tree traversal without recursion and without stack!](#)

[Structure Member Alignment, Padding and Data Packing](#)

[Intersection point of two Linked Lists](#)

[Lowest Common Ancestor in a BST.](#)

[Check if a binary tree is BST or not](#)

[Sorted Linked List to Balanced BST](#)

```
-----  
2  7  6  
9  5  1  
4  3  8
```

Steps:

1. position of number 1 = $(3/2, 3-1) = (1, 2)$
2. position of number 2 = $(1-1, 2+1) = (0, 0)$
3. position of number 3 = $(0-1, 0+1) = (3-1, 1) = (2, 1)$
4. position of number 4 = $(2-1, 1+1) = (1, 2)$
Since, at this position, 1 is there. So, apply condition 2.
new position = $(1+1, 2-2) = (2, 0)$
5. position of number 5 = $(2-1, 0+1) = (1, 1)$
6. position of number 6 = $(1-1, 1+1) = (0, 2)$
7. position of number 7 = $(0-1, 2+1) = (-1, 3)$ // this is tricky, see condition 3
new position = $(0, 3-2) = (0, 1)$
8. position of number 8 = $(0-1, 1+1) = (-1, 2) = (2, 2)$ //wrap around
9. position of number 9 = $(2-1, 2+1) = (1, 3) = (1, 0)$ //wrap around

Based on the above approach, following is the working code:

```
#include<stdio.h>  
#include<string.h>  
  
// A function to generate odd sized magic squares  
void generateSquare(int n)  
{  
    int magicSquare[n][n];  
  
    // set all slots as 0  
    memset(magicSquare, 0, sizeof(magicSquare));  
  
    // Initialize position for 1  
    int i = n/2;  
    int j = n-1;  
  
    // One by one put all values in magic square  
    for (int num=1; num <= n*n; )  
    {  
        if (i== -1 && j==n) //3rd condition  
        {
```



Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 15 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 55 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 58 minutes ago

Sanjay Agarwal bool

tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

AdChoices 

[▶ Math Puzzles](#)

[▶ Magic Square](#)

[▶ C++ Code](#)

```

        j = n-2;
        i = 0;
    }
    else
    {
        //1st condition helper if next number goes to out of square
        if (j == n)
            j = 0;
        //1st condition helper if next number is goes to out of square
        if (i < 0)
            i=n-1;
    }
    if (magicSquare[i][j]) //2nd condition
    {
        j -= 2;
        i++;
        continue;
    }
    else
        magicSquare[i][j] = num++; //set number

    j++; i--; //1st condition
}

// print magic square
printf("The Magic Square for n=%d:\nSum of each row or column %d:\n",
        n, n*(n*n+1)/2);
for(i=0; i<n; i++)
{
    for(j=0; j<n; j++)
        printf("%3d ", magicSquare[i][j]);
    printf("\n");
}
}

// Driver program to test above function
int main()
{
    int n = 7; // Works only when n is odd
    generateSquare (n);
    return 0;
}

```

Output:

The Magic Square for n=7:
Sum of each row or column 175:

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

NOTE: This approach works only for odd values of n.

References:

http://en.wikipedia.org/wiki/Magic_square

This article is compiled by **Aashish Barnwal** and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

JDBC to Informix

 progress.com/Informix

Supports Latest Data Connections
J2EE Certified. Download Eval Now!




AdChoices 

[▶ C++ Code](#)

[▶ Math Geeks](#)

[▶ The Square](#)

AdChoices 

[▶ D Square](#)

[▶ Matrix Math](#)

[▶ Programming C++](#)

Related Topics:

- Backtracking | Set 8 (Solving Cryptarithmic Puzzles)
- Tail Recursion
- Find if two rectangles overlap
- Analysis of Algorithm | Set 4 (Solving Recurrences)
- Print all possible paths from top left to bottom right of a mXn matrix
- Generate all unique partitions of an integer
- Russian Peasant Multiplication
- Closest Pair of Points | O(nlogn) Implementation



3



0



0

Writing code in comment? Please use ideone.com and share the link here.

14 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion...



Vinay Singh • 2 months ago

Magic square can be generated by using one loop and testing three conditions

```
int row = 0, col = n / 2, i, j, square = n * n;
for( i=1; i <= square ; i++ ){
    magic[ row ][ col ] = i;
    if( i % n == 0 ) row++;
    else{
        if( row == 0 ) row = n - 1;
        else row--;
        if( col == ( n - 1 ) ) col = 0;
        else col++;
    }
}
```

Those who interested in a line by line description can visit <http://www.vinaysingh.in>

1 ^ | v .



leet · 2 years ago

Can someone tell me why it is working or logic behind this(some proof)?

```
/* Paste your code here (You may delete these lines if not writing c
```

^ | v .



Prachi · 2 years ago

@Ankit Gupta

Your concept is very good. But there is a minor change required. In step 2, when occupied then we should move to $(i, (j-1)\%3)$

^ | v .



Ankit Gupta → Prachi · 2 years ago

Hi Prachi,

Thanks for pointing that out. You are right. :)

^ | v .



Prachi · 2 years ago

@Ankit Gupta

Your concept is very good. But there is a minor change required. In step 2, when occupied then we should move to $(i, (j-1)\%3)$

^ | v .



Yugandhar · 2 years ago

Another approach

```
void generateMagicSquare(int n)
{
    int a[n][n];
    int i=0,j=n/2,num;
    memset(a,0,sizeof(a));
    for(num=1;num<=n*n;num++)
    {
        a[i--][j++]=num;

        if(i== -1 && j==n)    {i=i+2;    j=j-1;}

        else if(i== -1)      i=n-1;

        else if(j==n)        j=0;

        else if(a[i][j]!=0)  {i=i+2;    j=j-1;}
```

see more

1 ^ | v ·



Sandeep · 2 years ago

In the steps,

For Number 2 position it should be (0,3). In the article, it is (0,0).

2. position of number 2 = (1-1, 2+1) = (0, 0)

^ | v ·



Aashish → Sandeep · 2 years ago



As mentioned in the article,

Let this position be (i,j) . The next number is stored at position $(i-1, j+1)$ column as circular array i.e. they wrap around.

The next position of 2 is calculated as $(0, 3)$

Since, 3 goes out of square's right side, it wraps around to $(0,0)$.

^ | v .



Sandeep → Aashish · 2 years ago

Okay. They have jumped directly to $(0,0)$. I thought, they were h other steps.

^ | v .



Aashish → Sandeep · 2 years ago

Sorry, i forgot to mention the intermediate step here.

^ | v .



ashok · 2 years ago

this was the same question i was asked in an interview(qualcomm)

^ | v .



atul · 2 years ago

simple ways of saying this is after setting "1" , from there on

1) move 1 step right and the 1 step up.

2) if space is already occupied and move back to previous state then move 1 :

else

set value.

```
/* Paste your code here (You may delete these lines if not writing c
```

^ | v .



Ankit Gupta → atul · 2 years ago



Ankit Gupta → atul · 2 years ago

Thanks. That looks simpler to read and understand.

With fewer cases one can simply write :

if start position = $(i,j) = (n/2, n-1)$

1. Move to $((i-1)+3)\%3, (j+1)\%3$ say (m,n)
2. If occupied (m,n) go to (i,j) and then $((i-1) + 3)\%3, j)$
3. Repeat

^ | v ·



atul → atul · 2 years ago

simple ways of saying this is after setting "1" , from there on

1) move 1 step right and the 1 step up.

2) if space is already occupied , move back to previous state then mov
else

set value.

^ | v ·



Subscribe



Add Disqus to your site