CHƯƠNG 6 MẢNG HAI CHIỀU

Đây là kiểu dữ liệu dùng để biểu diễn dữ liệu kiểu bảng, kiểu dữ liệu này rất thích hợp cho các bài toán liên quan đến đồ thị, biểu diễn ảnh, ...

I. TÓM TẮT LÝ THUYẾT

I.1. Khái niệm

Mảng hai chiều thực chất là mảng một chiều trong đó mỗi phần tử của mảng là một mảng một chiều, và được truy xuất bởi hai chỉ số dòng và cột.

Từ khái niệm trên ta có thể đưa ra một khái niệm về mảng nhiều chiều như sau: ***mảng có từ hai chiều trở lên gọi là mảng nhiều chiều***.

I.2. Khai báo mảng

Từ khái niệm trên ta có cú pháp khai báo mảng hai chiều như sau:

- Cách 1: Con trỏ hằng

< Kiểu dữ liệu > < Tên mảng > [< Số dòng tối đa >][< Số cột tối đa>];

Ví dụ:

```
int A[10][10];      // Khai báo mảng 2 chiều kiểu int gồm 10 dòng, 10 cột
float b[10][10]; // Khai báo mảng 2 chiều kiểu float gồm 10 dòng, 10 cột
```

- Cách 2 : Con trỏ

< Kiểu dữ liệu > **<Tên mảng>;

Ví dụ :

```
int **A ;  // Khai báo mảng động 2 chiều kiểu int
float **B ;  // Khai báo mảng động 2 chiều kiểu float
```

Tương tự như mảng một chiều, để sử dụng ta phải cấp phát vùng nhớ cho nó bằng malloc hoặc calloc và huỷ sau khi dùng bằng free

Ví dụ : Khai báo mảng các số nguyên A có kích thước 5x6

```
int **A;
A = ( int **) malloc (5) ;
for ( int  i = 0 ; i < 5 ; i ++ )
        A[i]=(int *) malloc (6) ;
```

I.3. Truy xuất phần tử của mảng

Để truy xuất các thành phần của mảng hai chiều ta phải dựa vào chỉ số dòng và chỉ số cột.

Ví dụ:

int A[3][4] = { {2,3,9,4} , {5,6,7,6} , {2,9,4,7} };

Với các khai báo như trên ta có :

A[0][0] = 2; A[0][1] = 3;

A[1][1] = 6; A[1][3] = 6;

Với ví dụ trên ta có hình dạng của một ma trận như sau

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

✍ *Lưu ý: Khi nhập liệu cho mảng hai chiều, nếu là mảng các số nguyên thì ta nhập liệu theo cách thông thường. Nhưng nếu là mảng các số thực thì ta phải thông qua biến trung gian.*

Ví dụ :

```
float a[10][10];          // Mang so thuc a
float  tmp;               // Bien trung gian tmp
scanf ("%f", &tmp);       // Nhap lieu cho bien trung gian
a[2][2] = tmp;            // Gan du lieu vao phan tu a[2][2]
```

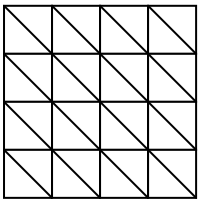
I.4. Ma trận vuông và các khái niệm liên quan

a. *Khái niệm*

Là ma trận có số dòng và số cột bằng nhau.

b. *Tính chất của ma trận vuông*

- **Đường chéo loại 1**



  - Đường chéo loại 1 bao gồm đường chéo chính và những đường chéo song song với đường chéo chính. Trong đó đường chéo chính là đường chéo có :

chỉ số dòng = chỉ số cột

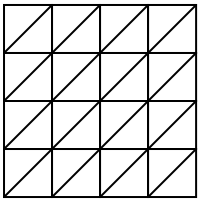
- Truy xuất các phần tử trên đường chéo loại 1 : để truy xuất các phần tử trên các đường chéo loại 1 ta có thể dựa vào chỉ số dòng và chỉ số cột như sau :

cột – dòng = hằng số

Ví dụ : Cho ma trận vuông $A_{(n \times n)}$. Gọi (i_o, j_o) là toạ độ điểm xuất phát, ta có thể duyệt đừong chéo xuất phất từ (i_o, j_o) như sau :

```
for ( i = io, j = jo ; i < n ; i ++, j ++ )
        printf ("%4d",A[i][j]);
```

- **Đường chéo loại 2:**



- Đường chéo loại 2 bao gồm đường chéo phụ và những đường song song với nó. Trong đó đường chéo phụ là đường chéo có:

chỉ số cột + chỉ số dòng = số dòng (hoặc số cột)

- Truy xuất các phần tử trên đường chéo loại 2 : để truy xuất các phần tử trên các đường chéo loại 1 ta có thể dựa vào chỉ số dòng và chỉ số cột như sau :

cột + dòng = hằng số

Ví dụ: Cho ma trận vuông A(n x n). Gọi (io, jo) là toạ độ điểm xuất phát, ta có thể duyệt đường chéo xuất phất từ (io, jo) như sau :

```
for ( i = io , j = jo ;  i < n && j > = 0 ; i ++ , j --)
        printf ("%4d",A[i][j]);
```

II. BÀI TẬP

Để đơn giản trong việc khai báo ma trận, ta định nghĩa **kiểu ma trận các phần tử với kiểu dữ liệu bất kỳ** như sau:

```
#define MAX 100
typedef <kiểu dữ liệu> MATRAN[MAX][MAX];
```

Ví dụ: Khai báo **ma trận các số nguyên a.**

```
#define MAX 100
```

```
typedef int MATRAN[MAX][MAX];
MATRAN a;
```

II.1. Một số kĩ thuật cơ bản

- ***Phương pháp nhập xuất ma trận***

```
void Nhap (MATRAN  a, int &d, int &c )
{
    printf ("\nNhap so dong: ");
    scanf (" %d", &d );
    printf ("\nNhap so cot: ");
    scanf ("%d", &c );
    for ( int i = 0; i < d; i ++ )
            for (int j = 0; j < c; j ++)
            {
                    printf (" a[%d][%d] = ", i, j );
                    scanf ("%d", &a[i][j]);
            }
}

void Xuat (MATRAN  a, int d, int c)
{
    printf ("\nNoi dung ma tran:\n");
    for (int i = 0; i < d; i++)
    {
            for (int j = 0; j < c; j++)
                    printf (" \t %d ", a[i][j] );
            printf ("\n");
        }
}
```

- ***Kĩ thuật đặt cờ hiệu***

Viết hàm kiểm tra xem trong ma trận các số nguyên có tồn tại các số nguyên lẻ lớn hơn 100 không?

```
int KiemTraLe (MATRAN  a, int d, int c)
{
    int flag = 0;                   //tra ve 1 neu co nguoc lai tra ve 0

    for (int i = 0; i < d; i ++ )
            for (int j = 0; j < c; j++)
                    if ( a[i][j] % 2 != 0 && a[i][j] > 100 )
                    {
                            flag = 1;
                            break;
                    }
    return flag;
}
```

- ***Kĩ thuật đặt lính canh***

  Viết hàm tìm phần tử nhỏ nhất trong ma trận.

```
int Min (MATRAN  a, int d, int c )
{
    int min = a[0][0];
    for ( int i = 0 ; i < d ; i ++ )
            for (int j = 0 ; j < c ; j ++)
                    if ( a[i][j] < min )
                            min = a[i][j];
    return min;
}
```

- ***Phương pháp tính tổng***

  Viết hàm tính tổng các phần tử trong ma trận.

```
long Tong (MATRAN  a, int d, int c)
{
    long tong = 0;

    for ( int i = 0; i < d; i ++ )
            for ( int j = 0; j < c; j ++)
                    tong + = a[i][j];
    return tong;
}
```

- ***Phương pháp sắp xếp***

  Viết hàm sắp xếp ma trận tăng dần từ trên xuống dưới và từ trái sang phải không dùng mảng phụ.

```
void SapTang(MATRAN  a, int d, int c)
{
    for (int i = 0; i <= d*c-2; i ++)
            for (int j = 0; j <= d*c-1; j ++)
                    if (a[i/c][i%c] < a[j/c][j%c])
                    {
                            int tmp = a[i/c][i%c] ;
                            a[i/c][i%c] = a[j/c][j%c] ;
                            a[j/c][j%c] = tmp ;
                    }
}
```

- ***Phương pháp đếm***

  Viết hàm đếm các phần tử chẵn trong ma trận.

```
int DemChan (MATRAN  a, int d, int c)
{
    int dem = 0;
```

```
    for ( int i = 0 ; i < d ; i ++)
            for ( int  j = 0 ; j < c ; j ++)
                    if ( a[i][j] % 2 = = 0 )
                            dem ++;
    return dem;
}
```

II.2. Bài tập cơ bản

a. Bài tập nhập xuất

1. Viết hàm nhập ma trận các số nguyên dương *(nhập sai báo lỗi và không cho nhập).*
2. Viết hàm nhập/ xuất ma trận các số thực.
3. Viết hàm in ra những phần tử có ký số tận cùng là 5.
4. Viết chương trình in ra các phần tử nằm trên 2 đường chéo.
5. Viết hàm in ra các phần tử nằm phía trên đường chéo phụ của ma trận vuông các số nguyên.
6. Viết hàm in ra các phần tử nằm phía dưới đường chéo phụ của ma trận vuông các số nguyên.
7. Viết hàm in ra các phần tử nằm phía trên đường chéo chính của ma trận vuông các số nguyên.
8. Viết hàm in ra các phần tử nằm phía dưới đường chéo chính của ma trận vuông các số nguyên.
9. Viết chương trình khởi tạo giá trị các phần tử là ngẫu nhiên cho ma trận các số nguyên kích thước $m \times n$.
10. Viết hàm tạo ma trận a các số nguyên gồm 9 dòng 14 cột. Trong đó phần tử a[i][j] = i * j
11. Viết hàm in tam giác Pascal với chiều cao h.

    Ví dụ : h = 5

    1
    1 1
    1 2 1
    1 3 3 1
    1 4 6 4 1

b. Bài tập tính tổng

12. Viết hàm tính tổng các phần tử trên cùng một dòng.

13. Viết hàm tính tổng các phần tử trên cùng một cột.
14. Viết hàm tính tổng các phần tử chẵn có trong ma trận.
15. Viết hàm tính tổng các phần tử nằm trên đường chéo chính của ma trận vuông.
16. Viết hàm tính tổng các phần tử là số nguyên tố có trong ma trận.
17. Viết hàm tính tổng các số hoàn thiện trong ma trận các số nguyên.
18. Viết hàm tính tổng các giá trị lớn nhất trên mỗi dòng.
19. Viết hàm tính giá trị trung bình của các phần tử nhỏ nhất trên mỗi cột.
20. Viết hàm tính tổng các giá trị nhỏ nhất nằm trên từng đường chéo loại 2.
21. Viết hàm tìm đường chéo có tổng lớn nhất trong các đường chéo loại 1.

c. Bài tập tìm kiếm

22. Viết hàm tìm vị trí phần tử lớn nhất trong ma trận các số nguyên.
23. Viết hàm tìm vị trí phần tử nhỏ nhất trong ma trận các số nguyên.
24. Viết hàm tìm vị trí phần tử chẵn cuối cùng trong ma trận các số nguyên.
25. Viết hàm tìm phần tử âm lẻ lớn nhất trong ma trận.
26. Viết hàm tìm phần tử chẵn dương và nhỏ nhất trong ma trận.
27. Viết hàm tìm số hoàn thiện đầu tiên trong ma trận các số nguyên.
28. Viết hàm tìm số hoàn thiện lớn nhất trong ma trận các số nguyên.
29. Viết hàm tìm vị trí phần tử nguyên tố cuối cùng trong ma trận các số nguyên.
30. Viết hàm tìm phần tử lớn nhất nằm trên đường chéo chính của ma trận vuông.
31. Viết hàm in các số nguyên tố nằm trên đường chéo phụ của ma trận vuông.
32. Viết hàm tìm trong 2 ma trận các số nguyên, những phần tử giống nhau.
33. Viết hàm tìm phần tử nhỏ nhất trên mỗi đường chéo loại 2 của ma trận.
34. Viết hàm tìm và liệt kê những phần tử cực đại trong ma trận (một phần tử được coi là cực đại khi nó lớn hơn các phần tử xung quanh nó).
35. Viết hàm tìm dòng có tổng lớn nhất trong ma trận các số thực.
36. Viết hàm tìm cột có tổng nhỏ nhất trong ma trận các số nguyên.

d. Bài tập đếm

37. Viết hàm đếm các giá trị âm, dương trong ma trận các số thực.
38. Viết hàm đếm các giá trị chẵn, lẻ trong ma trận các số nguyên.

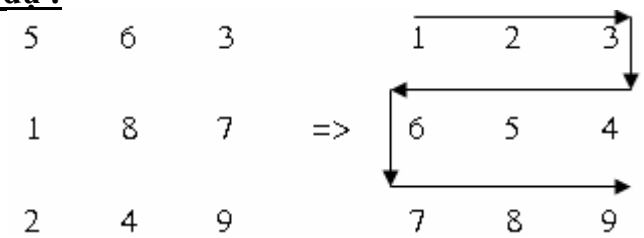
39. Viết hàm đếm số lần xuất hiện của phần tử x trong ma trận các số thực.
40. Viết hàm đếm các giá trị nhỏ hơn x trong ma trận các số thực.
41. Viết hàm đếm các phần tử nguyên tố trong ma trận các số nguyên.
42. Viết hàm đến các phần tử nguyên tố trên đường chéo chính của ma trận vuông các số nguyên.
43. Viết hàm đếm các giá trị chẵn trên đường chéo chính của ma trận vuông các số nguyên.
44. Viết hàm đếm các giá trị là bội của 3 và 5 trên đường chéo chính của ma trận các số nguyên.
45. Viết hàm đếm các giá trị nguyên tố trên 2 đường chéo (chính, phụ) của ma trận vuông các số nguyên.
46. Viết hàm đếm các giá trị cực đại trong ma trận các số nguyên.
47. Viết hàm đếm các giá trị cực tiểu trong ma trận các số nguyên.
48. Viết hàm đếm các cực trị trong ma trận các số nguyên (một phần tử được coi là cực trị khi nó là giá trị cực đại hay cực tiểu).
49. Viết hàm đếm các giá trị là số hoàn thiện trong ma trận các số nguyên.

e. Bài tập sắp xếp

50. Viết hàm sắp xếp ma trận theo thứ tự tăng dần từ trên xuống dưới và từ trái qua phải theo phương pháp dùng mảng phụ.

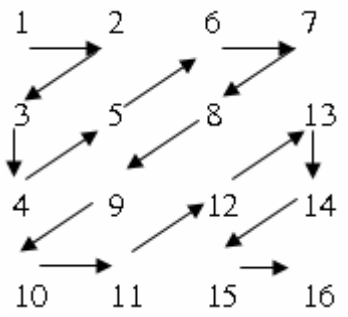
    Hướng dẫn: Đổ ma trận sang mảng một chiều, sắp xếp trên mảng một chiều theo thứ tự tăng dần, sau đó chuyển ngược mảng một chiều thành ma trận kết quả.
51. Viết hàm sắp xếp ma trận theo thứ tự giảm dần từ trên xuống dưới và từ trái sang phải.
52. Viết hàm sắp xếp các dòng trên ma trận theo thứ tự tăng dần.
53. Viết hàm sắp xếp các cột trên ma trận theo thứ tự giảm dần.
54. Viết hàm sắp xếp ma trận theo đường ziczắc ngang.

    Ví dụ :



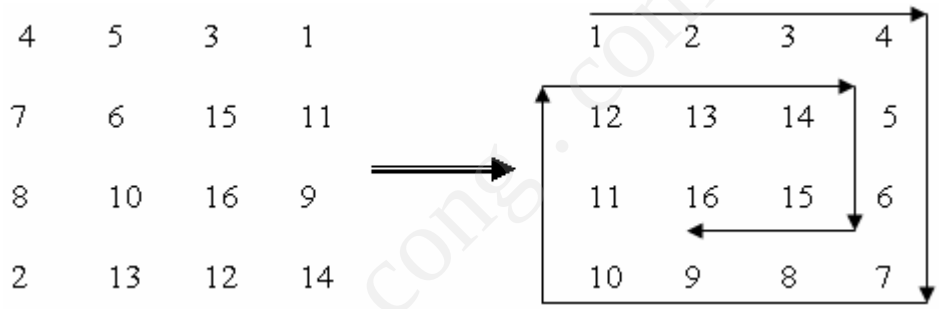
55. Viết hàm sắp xếp ma trận theo đường ziczắc chéo

Ví dụ :



56. Viết hàm sắp xếp ma trận theo đường xoắn ốc từ ngoài vào trong theo chiều kim đồng hồ.

Ví dụ :



57. Cho ma trận vuông, viết hàm sắp xếp tăng dần các phần tử nằm trên các đường chéo song song với đường chéo chính.
58. Viết chương trình nhập một ma trận vuông các số nguyên, và thực hiện những công việc sau :
    - Sắp xếp các phần tử nằm trên các đường chéo loại 1 tăng dần
    - Sắp xếp các phần tử nằm trên các đường chéo loại 2 giảm dần.
    - Sắp xếp với điều kiện: các phần tử trên đường chéo chính tăng, các phần tử trên các đường chéo song song với đường chéo chính giảm.

f. Bài tập Thêm – Xoá – Thay thế

59. Viết hàm xoá một dòng i trên ma trận.
60. Viết hàm xoá một cột j trên ma trận.
61. Viết hàm xoá dòng có tổng lớn nhất trên ma trận.
62. Viết hàm hoán vị dòng có tổng lớn nhất với dòng có tổng nhỏ nhất.
63. Viết hàm tìm và thay thế các phần tử chẵn trong ma trận bằng ước số nhỏ nhất của nó.
64. Viết hàm thay thế những phần tử có giá trị x thành phần tử có giá trị y trong ma trận (x , y nhập từ bàn phím).

II.3. Bài tập luyện tập và nâng cao

65. Viết chương trình tính tổng, tích của hai ma trận các số nguyên.
66. Viết hàm kiểm tra xem ma trận vuông các số nguyên có đối xứng qua đường chéo chính hay không.
67. Viết hàm kiểm tra xem trong ma trận vuông cấp n có hàng nào trùng nhau hay không, nếu có thì chỉ rõ những hàng nào. (Trùng giá trị và vị trí).
68. Viết chương trình nhập vào ma trận vuông kích thước n x n ($2 \le n \le 100$). Hãy viết hàm thực hiện những công việc sau :
    - In ra các phần tử trên 4 đường biên của ma trận.
    - Tính tổng các phần tử trên biên.
69. **(*)** Viết chương trình xoay ma trận các số thực 90^0 ngược chiều kim đồng hồ.

    Ví dụ:

1	2	3	4		4	8	12	16
5	6	7	8	=>	3	7	11	15
9	10	11	12		2	6	10	14
13	14	15	16		1	5	9	13

70. Viết chương trình dịch phải xoay vòng một cột trong ma trận các số thực.
71. Viết chương trình dịch xuống xoay vòng một dòng trong ma trận các số thực.
72. **(*)** Cho ma trận A ($m \times n$) các số nguyên hãy phát sinh ma trận B sao cho B là ma trận lật ngược của ma trận A.

    Ví dụ :

1	2	3	4		4	3	2	1
5	6	7	8	=>	8	7	6	5
9	10	11	12		12	11	10	9
13	14	15	16		16	15	14	13

73. **(**)** Cho ma trận A ($m \times n$) hãy phát sinh ma trận B ($m \times n$) sao cho phần tử B (i, j) là trung bình cộng của các phần tử trong hình vuông 3x3 tâm tại (i,j) của A.

    Ví dụ :

1	5	2	6		3	2	4	4
4	2	3	6	=>	4	4	4	4
8	7	9	1		5	6	6	7
10	2	12	13		6	8	7	8

74. **(**)** Cho ma trận các số nguyên dương A ($m \times n$) . Hãy xây dựng ma trận B ($m \times n$). Sao cho phần tử B (i, j) là số lớn nhất trong ô vuông 3 x 3 tâm tại (i, j) của A.

Ví dụ :

1	5	2	6		5	5	6	6
4	2	3	6	=>	8	9	9	9
8	7	9	1		10	12	13	13
10	2	12	13		10	12	13	13

75. **(**)** Cho ma trận A ($m \times n$). Hãy xây dựng ma trận B ($m \times n$) với phần tử B(i,j) được xác định theo qui tắc sau: tại vị trí (i, j) trên mảng A kẻ hai tia vuông góc với nhau, tạo thành với trục hoành một góc 45^0 từ trên xuống dưới; B(i, j) là tổng của tất cả các số của vùng mặt phẳng tạo bởi hai tia này và các cạnh của bảng.

Ví dụ :

1	3	2		25	30	23
6	4	3	=>	13	18	15
2	5	7		2	5	7

76. **(**)** Cho ma trận vuông A ($n \times n$). Hãy xây dựng mảng B ($n \times n$) bằng cách: phần tử B (i, j) là số lớn nhất trong tam giác vuông vẽ từ A (i, j) tới đường chéo chính.

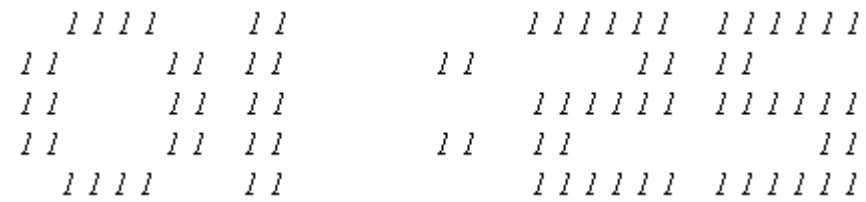
Ví dụ :

1	3	2		1	4	7
6	4	3	=>	6	4	7
2	5	7		7	7	7

77. **(*)** Viết chương trình hiển thị đồng hồ điện tử (gồm giờ phút), với giờ lấy từ hệ thống và đồng hồ được cập nhật theo phút.

Hướng dẫn: Tạo 1 ma trận giá trị gồm 0 hoặc 1, vị trí nào cần hiển thị thì gán giá trị là 1, ngược lại có giá trị là 0. Sau mỗi phút cập nhật lại ma trận và hiển thị lên màn hình.

Ví dụ: 01 giờ 25 phút



78. Nhập vào mảng hai chiều gồm n dòng và m cột các số nguyên. Hãy tìm phần tử lớn nhất trên mỗi dòng và đồng thời nhỏ nhất trên mỗi cột, hoặc lớn nhất trên mỗi cột và đồng thời nhỏ nhất trên mỗi dòng. Có bao nhiêu phần tử như thế?

Ví dụ:

3	**6**	2	1
4	7	6	9
5	15	8	7

79. Viết chương trình tạo ngẫu nhiên một ma trận các số nguyên (0 -> 50), tìm những phần tử cực đại (là phần tử lớn hơn các phần tử xung quanh).

Ví dụ :

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

80. **(**)** Cho ma trận các số nguyên A $m \times n$ $(n \geq 3, m \geq 3)$. Hãy tìm ma trận con (3x3) có tổng lớn nhất.

Ví dụ :

1	2	3	4		6	7	8
5	6	7	8	=>	10	11	12
9	10	11	12		14	15	16
13	14	15	16				

81. Nhập ma trận vuông cấp $n \times n$ (n < 10). In ra các phần tử của ma trận này theo hướng của đường chéo chính.

Ví dụ **:** n = 4

					1			
					9	3		
1	3	7	4		3	5	7	
9	5	6	2	=>	2	4	6	4
3	4	7	5		3	7	2	
2	3	1	6		1	5		
					6			

82. **(**)** Hãy điền các số từ 1 đến n^2 vào ma trận cấp n (n > 2), chỉ xét trường hợp n là số lẻ với tính chất P là tổng các số bằng nhau.

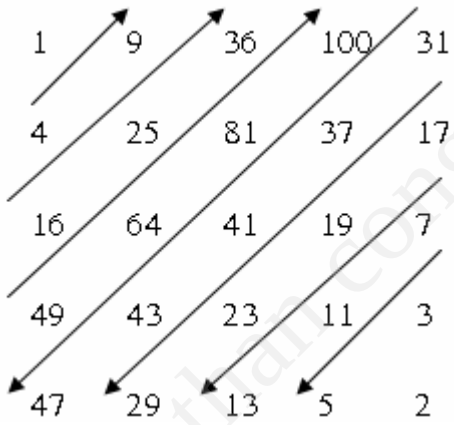
Hướng dẫn : *Ma phương của một bảng vuông cấp n, trong mỗi ô nhận một giá trị sao cho, mỗi hàng, mỗi cột và mỗi đường chéo đều thoả mãn một tính chất P nào đó cho trước.*

Ví dụ : Với n = 5

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

83. **(*)** Viết hàm in ma trận các số nguyên dương theo qui luật được mô tả như sau : các phần tử phía trên đường chéo phụ là giá trị bình phương của các giá trị $1 \to n \times 2$, các giá trị từ đường chéo phụ trở xuống là các số nguyên tố. Ma trận được sắp xếp như ví dụ bên dưới.

Ví dụ : n = 5



84. Cho ma trận vuông a cấp n (n lẻ, $3 \le n \le 15$), mỗi phần tử đều có giá trị nguyên dương. Hãy xây dựng hàm kiểm tra xem ma trận a có phải là ma phương hay không?

85. **(**)** Viết chương trình giải bài toán 8 hậu. Hãy đặt 8 con hậu trên bàn cờ 8x8 sao cho chúng không ăn nhau (2 hậu ăn nhau khi cùng hàng, cùng cột và cùng nằm trên đường chéo).

Hướng dẫn:

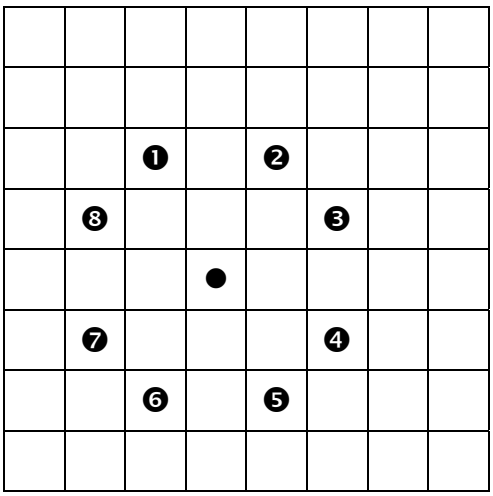
Dùng ma trận 8x8 để lưu bàn cờ. Mỗi ô có 3 trạng thái :

- *Có hậu* *1*
- *Ô trống* *0*
- *Ô không được đi* *-1*

86. **(**)** Viết chương trình giải bài toán mã đi tuần. Hãy đi con mã 64 lượt đi trên bàn cờ 8x8 sao cho mỗi ô chỉ đi qua một lần (xuất phát từ một ô bất kỳ)

Hướng dẫn :

Đứng tại một ô trên bàn cờ con mã có thể đi được 1 trong 8 hướng sau .



Khai báo 8 hướng đi của mã như sau:

```
typedef struct DIEM
{
        int x, y;
};
DIEM huongdi[8]={{-2,-1},{-2,1},{-1,2},{1,2},{2,1},{2,-1},{1,-2},{-1,-2}};
```

Trong đó mỗi thành phần của huongdi là độ lệch của dòng và cột so với vị trí của con mã.

Ví dụ: huongdi[0] (tức đi đến vị trí ❶ như hình vẽ) có độ lệch 2 dòng và 1 cột. (Giá trị âm biểu thị độ lệch về bên trái cột hay hướng lên của dòng).

Chọn vị trí đi kế tiếp sao cho vị trí đó phải gần với biên hay góc nhất (tức số đường đi có thể đi là ít nhất).

87. Viết chương trình giải bài toán Taci. Cho ma trận vuông 3x3 gồm các số nguyên từ 0 -> 8 trong đó 0 là ô trống. Bài toán đặt ra là hãy đưa ma trận ở một trạng thái đầu về trạng thái đích, mỗi lần chỉ dịch chuyển được 1 ô.

Ví dụ : Trạng thái đầu Trạng thái đích

1	3	0		1	2	3
8	2	5	=>	8	0	4
7	4	6		7	6	5

III. KẾT LUẬN

- Kiểu dữ liệu mảng hai chiều được ứng dụng rộng rãi trong các bài toán về tìm đường đi trong đồ thị, xử lý ảnh, xử lý những dữ liệu dạng bảng, …
- Lưu ý khi nhập mảng hai chiều **các số thực** phải thông qua 1 biến trung gian.