

Bài tập thực hành Mạng hai chiều

Site: [E-Learning Nha Trang University](#)
Course: Nhập môn lập trình - SOT315_66.KHMT - GV.
Lê Thị Bích Hằng
Book: Bài tập thực hành Mạng hai chiều

Printed by: Tiến Phạm Minh
Date: Tuesday, 27 May 2025, 10:28 PM

Table of contents

1. Một số bài toán cơ bản

- 1.1. Duyệt mảng hai chiều
- 1.2. Nhập dữ liệu vào mảng hai chiều
- 1.3. Xuất dữ liệu mảng hai chiều
- 1.4. Tìm kiếm một phần tử trong mảng hai chiều
- 1.5. Kiểm tra tính chất của mảng hai chiều
- 1.6. Tính tổng các phần tử trên hàng/cột của ma trận
- 1.7. Tìm phần tử lớn nhất/nhỏ nhất của ma trận
- 1.8. Sắp xếp trên mảng hai chiều

2. Chương trình mẫu

3. Bài tập thực hành

1. Một số bài toán cơ bản

Viết hàm thực hiện từng yêu cầu sau

- Nhập mảng
- Xuất mảng
- Tìm kiếm một phần tử trong mảng
- Kiểm tra tính chất của mảng
- Tìm giá trị nhỏ nhất/lớn nhất của mảng
- Sắp xếp mảng giảm dần/tăng dần
- Sửa giá trị của phần tử mảng
- Tính tổng các phần tử trên dòng/cột/toàn ma trận/đường chéo chính/nửa trên/nửa dưới

1.1. Duyệt mảng hai chiều

- Việc duyệt mảng hai chiều sẽ làm cơ sở cho hầu hết các bài toán như:
 - Tính tổng các phần tử trong ma trận,
 - Đếm số phần tử trong ma trận,
 - Tìm kiếm một phần tử trong ma trận,
 - Sắp xếp các phần tử,...

Ví dụ, bài toán sắp xếp một số phần tử trên mảng hai chiều (theo một thứ tự nào đó) chính là việc vận dụng phối hợp một cách duyệt mảng với một thuật toán sắp xếp.

1. Duyệt hết các phần tử của ma trận

- Cách 1: Duyệt theo từng hàng từ trên xuống dưới, với mỗi hàng duyệt từ trái sang phải (tương tự như phần nhập mảng theo từng dòng)

```
for (i = 0; i < so_hang; i++)
    for (j = 0; j < so_cot; j++)
    {
        /* Xử lý phần tử a[i][j] */
    }
```

- Cách 2: Duyệt theo từng cột từ trái sang phải, với mỗi cột duyệt từ trên xuống dưới (tương tự như phần nhập mảng theo từng cột).

```
for (j = 0; j < so_cot; j++)
    for (i = 0; i < so_hang; i++)
    {
        /* Xử lý phần tử a[i][j] */
    }
```

2. Duyệt các phần tử nằm trên cùng một hàng hoặc nằm trên cùng một cột

- Duyệt các phần tử trên **hàng thứ k** ($0 \leq k < \text{số hàng}$)

```
for (i = 0; i < so_cot; i++)
{
    /* Xử lý phần tử a[k][i] */
}
```

- Duyệt các phần tử trên **cột thứ k** ($0 \leq k < \text{số cột}$)

```
for (i = 0; i < so_hang; i++)
{
    /* Xử lý phần tử a[i][k] */
}
```

3. Duyệt các phần tử nằm trên đường chéo chính của ma trận vuông

- Ma trận vuông là ma trận có n hàng, n cột (chỉ số hàng và cột từ 0 đến $n-1$). Giả sử ta có một ma trận vuông có 4×4 phần tử, khi đó các phần tử trong ma trận được sắp xếp như sau:

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]
a[3][0]	a[3][1]	a[3][2]	a[3][3]

- Trong một ma trận vuông, các phần tử nằm trên đường chéo chính là các phần tử $a[i,i]$ với $0 \leq i \leq n-1$. Với bảng minh họa trên, các phần tử nằm trên đường chéo chính là: $a[0][0]$, $a[1][1]$, $a[2][2]$, $a[3][3]$.
- Hai cách sau duyệt các phần tử nằm trên đường chéo chính:
 - Cách 1: Sử dụng một vòng lặp

```
for (i = 0; i < n; i++)
{
    /* Xử lý phần tử a[i][i] */
}
```

- Cách 2: Sử dụng hai vòng lặp. Cách này kém hiệu quả hơn do phải duyệt toàn bộ ma trận

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
    {
        if (i == j)
            /* Xử lý phần tử a[i][j] */
    }
```

4. Duyệt các phần tử nằm trên đường chéo phụ của ma trận vuông

- Trong một ma trận vuông có n hàng, n cột (chỉ số dòng và cột từ 0 đến $n-1$) thì các phần tử nằm trên đường chéo phụ là các phần tử $a[n-1-i, i]$ với $0 \leq i \leq n-1$. Giả sử ta có một ma trận vuông có 4×4 phần tử, khi đó các phần tử trong ma trận được sắp xếp như sau:

$a[0][0]$	$a[0][1]$	$a[0][2]$	$a[0][3]$
$a[1][0]$	$a[1][1]$	$a[1][2]$	$a[1][3]$
$a[2][0]$	$a[2][1]$	$a[2][2]$	$a[2][3]$
$a[3][0]$	$a[3][1]$	$a[3][2]$	$a[3][3]$

- Với bảng minh họa trên, các phần tử nằm trên đường chéo phụ là: $a[0][3]$, $a[1][2]$, $a[2][1]$, $a[3][0]$.
- Việc duyệt các phần tử nằm trên đường chéo phụ được thực hiện như sau:
 - Cách 1: Sử dụng một vòng lặp

```
for (i = 0; i < n; i++)
{
    /* Xử lý phần tử a[n-1-i][i] */
}
```

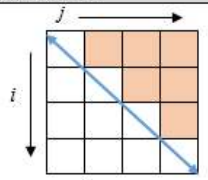
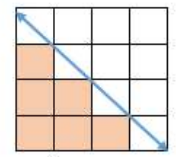
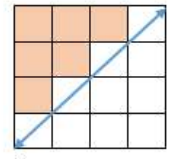
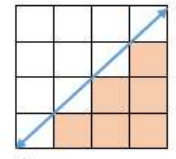
- Cách 2: Nhận xét rằng các phần tử nằm trên đường chéo phụ thỏa điều kiện $i + j = n - 1$

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
    {
        if (i + j == n-1)
            /* Xử lý phần tử a[i][j] */
    }
```

5. Duyệt phân nửa ma trận vuông

- Có 4 trường hợp khác nhau của nửa ma trận vuông:
 - Nửa tam giác phía trên đường chéo chính
 - Nửa tam giác phía dưới đường chéo chính
 - Nửa tam giác phía trên đường chéo phụ
 - Nửa tam giác phía dưới đường chéo phụ

Các hình vẽ dưới đây minh họa bốn trường hợp này và các điều kiện đặc trưng cho mỗi trường hợp được sử dụng khi duyệt một nửa ma trận vuông:

Các phần tử ở nửa tam giác phía trên đường chéo chính	Các phần tử ở nửa tam giác phía dưới đường chéo chính
 <p>Điều kiện: $i < j$</p>	 <p>Điều kiện: $i > j$</p>
Các phần tử ở nửa tam giác phía trên đường chéo phụ	Các phần tử ở nửa tam giác phía dưới đường chéo phụ
 <p>Điều kiện: $i + j < n - 1$</p>	 <p>Điều kiện: $i + j > n - 1$</p>

1.2. Nhập dữ liệu vào mảng hai chiều

- Yêu cầu:
 - Viết hàm nhập giá trị cho một ma trận a có m hàng, n cột.
- Ý tưởng:
 - + Sử dụng `#define` định nghĩa số hàng tối đa là MAXD, số cột tối đa là MAXC.
 - + Nhập số lượng phần tử thực sự m, n của mỗi chiều.
 - + Lặp lượt nhập từng phần tử từ $[0][0]$ đến $[m-1][n-1]$.

```
void NhapMaTran(int a[][MAXC], int &m, int &n)
```

```
{    printf("Nhap so hang, so cot cua ma tran: ");
```

```
    scanf("%d%d", &m, &n);
```

```
    int i, j;
```

```
    for (i = 0; i < m; i++)
```

```
    {    for (j = 0; j < n; j++)
```

```
    {        printf("Nhap a[%d][%d]: ", i, j);
```

```
        scanf("%d", &a[i][j]);
```

```
    }
```

```
}
```

1.3. Xuất dữ liệu mảng hai chiều

- Yêu cầu:
 - Viết hàm xuất một ma trận a có m hàng, n cột ra màn hình.
- Ý tưởng: Xuất giá trị từng phần tử của ma trận từ hàng 0 đến hàng $m-1$, mỗi hàng xuất giá trị của cột 0 đến cột $n-1$ trên hàng đó.

```
void XuatMaTran(int a[][MAXC], int m, int n)
{
    int i, j;
    for (i = 0; i < m; i++)    // i lap tu 0 den m-1 hang
    {
        for (j = 0; j < n; j++)    // j lap tu 0 đến n-1 cot
            printf("%5d", a[i][j]);
        printf("\n");    // Xuong dong de in hang ke tiep
    }
}
```


1.4. Tìm kiếm một phần tử trong mảng hai chiều

- Yêu cầu:
 - Viết hàm tìm xem phần tử x có nằm trong ma trận a kích thước $m \times n$ hay không?
- Ý tưởng:
 - Duyệt từng phần tử của ma trận a . Nếu phần tử đang xét bằng x thì quá trình duyệt mảng kết thúc và trả về 1, ngược lại trả về 0.

```
int TimKiem(int a[][MAXC], int m, int n, int x)
```

```
{    int i, j;
```

```
    for (i = 0; i < m; i++)
```

```
        for (j = 0; j < n; j++)
```

```
            if (a[i][j] == x)
```

```
                return 1;
```

```
    return 0;
```

```
}
```

1.5. Kiểm tra tính chất của mảng hai chiều

- Yêu cầu:
 - Cho trước ma trận a kích thước $m \times n$. Viết hàm kiểm tra ma trận a có phải là ma trận toàn các số chẵn không?
- Ý tưởng: Tương tự với mảng một chiều, có nhiều cách để kiểm tra tính chất của mảng hai chiều.
- + **Cách 1:** Đếm số lượng số chẵn có trong ma trận. Nếu số lượng này bằng đúng $m \times n$ thì mảng toàn số chẵn.

```
int KiemTra_C1(int a[][MAXC], int m, int n)
```

```
{    int i, j, dem = 0;
```

```
    for (i = 0; i < m; i++)
```

```
        for (j = 0; j < n; j++)
```

```
            if (a[i][j] % 2 == 0)
```

```
                dem++;
```

```
        if (dem == m*n)
```

```
            return 1;
```

```
        return 0;
```

```
}
```

- + **Cách 2:** Đếm số lượng phần tử không phải là số chẵn trong ma trận. Nếu số lượng này bằng 0 thì ma trận sẽ chứa toàn số chẵn.

```
int KiemTra_C2(int a[][MAXC], int m, int n)
```

```
{    int i, j, dem = 0;
```

```
    for (i = 0; i < m; i++)
```

```
        for (j = 0; j < n; j++)
```

```
            if (a[i][j] % 2 != 0)
```

```
                dem++;
```

```
        if (dem == 0)
```

```
            return 1;
```

```
        return 0;
```

```
}
```

- + **Cách 3:** Tìm xem có phần tử nào không phải là số chẵn hay không. Nếu có thì ma trận không chứa toàn số chẵn => trả về 0.

```
int KiemTra_C3(int a[][MAXC], int m, int n)
{
    int i, j;
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            if (a[i][j] % 2 == 1)
                return 0;
    return 1;
}
```

1.6. Tính tổng các phần tử trên hàng/cột của ma trận

- Yêu cầu:
 - Cho trước ma trận a kích thước $m \times n$. Viết hàm tính tổng các phần tử trên hàng d (hoặc cột c).
- Ý tưởng:
 - Duyệt ma trận và cộng dồn các phần tử có tọa độ hàng (hoặc cột) thỏa yêu cầu.

+ Tính tổng các phần tử trên **hàng d**:

```
int TongHang(int a[][MAXC], int n, int d)
{
    int j, tong;
    tong = 0;
    for(j = 0; j < n; j++)
        tong = tong + a[d][j];
    return tong;
}
```

+ Tính tổng các phần tử trên **cột c**:

```
int TongCot(int a[][MAXC], int m, int c)
```

```
{ int i, tong;
```

```
    tong = 0;
```

```
    for(i = 0; i < m; i++)
```

```
        tong = tong + a[i][c];
```

```
    return tong;
```

```
}
```

1.7. Tìm phần tử lớn nhất/nhỏ nhất của ma trận

- Yêu cầu:
 - Cho trước ma trận a kích thước $m \times n$. Viết hàm tìm giá trị lớn nhất trong ma trận a (gọi là max).
- Ý tưởng:
 - Giả sử giá trị lớn nhất hiện tại (gọi là max) là giá trị phần tử đầu tiên $a[0][0]$. Ta lần lượt so sánh max với các phần tử còn lại: nếu có phần tử nào có giá trị lớn hơn max thì ta cập nhật lại giá trị của max chính là bằng giá trị của phần tử đó.

```
int TimMax(int a[][MAXC], int m, int n)
{
    int i, j, max;
    max = a[0][0];
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            if (a[i][j] > max)
                max = a[i][j];
    return max;
}
```

1.8. Sắp xếp trên mảng hai chiều

- Yêu cầu:
 - Cho trước ma trận a kích thước $m \times n$. Viết hàm sắp xếp ma trận a sao cho các phần tử tăng dần từ trái qua phải, từ trên xuống dưới.
- Ý tưởng:
 - Bước 1: Chuyển đổi mảng hai chiều a thành mảng một chiều b ,
 - Bước 2: Sắp xếp mảng b tăng dần
 - Bước 3: Chuyển đổi mảng b trở lại thành mảng a .

```

void SapXepMang2ChieuTangDan(int a[][MAXC], int m, int n)
{
    int i, j, k = 0;

    int b[MAXD*MAXC];

    for(i = 0; i < m; i++)        /* Tạo mảng một chiều b */
    {
        for(j = 0; j < n; j++)
        {
            b[k] = a[i][j];
            k++;
        }
    }

    SapXepMang1ChieuTangDan(b, k); /* Tham khảo Chương 5 - Mảng một chiều */

    k = 0;

    for(i = 0; i < m; i++)        /* Đưa phần tử trong b vào a */
    {
        for(j = 0; j < n; j++)
        {
            a[i][j] = b[k];
            k++;
        }
    }
}

```

2. Chương trình mẫu

- Ví dụ: Viết chương trình nhập vào 1 ma trận số nguyên $m \times n$ ($2 \leq m, n \leq 10$). Sau đó hãy in ra ma trận vừa nhập.
- Chương trình gồm các hàm *NhapMatran* (Nhập các số), *XuatMatran* (Hiển thị ma trận). Các tham số hình thức của các hàm này là một mảng hai chiều có số phần tử tối đa của chiều thứ nhất (hàng) không cần khai báo, nhưng số phần tử của chiều thứ hai (cột) phải được khai báo, và giá trị thực sự của hàng và cột ma trận được đưa vào hai biến m và n tương ứng.
- Chương trình minh họa như sau:

```

1  /* Chương trình tạo ma tran so nguyen kích thước m x n */
2  #include<stdio.h>
3  #define MAXD 10
4  #define MAXC 10
5  /* Khai báo các nguyên mẫu hàm */
6  void NhapMaTran(int a[][MAXC], int &m, int &n);
7  void XuatMaTran(int a[][MAXC], int m, int n);
8  /* Hàm main */
9  int main()
10 {   int b[MAXD][MAXC], m, n;
11     NhapMaTran(b, m, n);
12     printf("\nMa tran vua nhap: \n");
13     XuatMaTran(b, m,n);
14     return 0;
15 }
16 /* Định nghĩa các hàm */
17 void NhapMaTran(int a[][MAXC], int &m, int &n)
18 {   do{
19
20     printf("Nhap so dong, so cot cua ma tran: ");
21
22     scanf("%d%d", &m, &n);
23
24     }while(!(m>=2 && m<=10) && !(n>=2 && n <=10));
25     for(int i = 0; i < m; i++)
26         for(int j = 0; j < n; j++)
27             {   printf("Nhap a[%d][%d]: ", i, j);
28                 scanf("%d", &a[i][j]);
29             }
30 }
31 void XuatMaTran(int a[][MAXC], int m, int n)
32 {   for(int i = 0; i < m; i++)
33     {   for (int j = 0; j < n; j++)
34         printf("%5d", a[i][j]);
35         printf("\n");
36     }
37 }
```

3. Bài tập thực hành

Bài 1. Viết chương trình nhập vào một ma trận vuông có $n \times n$ phần tử là số nguyên ($2 \leq n \leq 5$).

- Xuất ra màn hình ma trận đó
- Tính tổng các phần tử nằm trên đường chéo chính.
- Đếm số phần tử chẵn trong ma trận.
- In ra các phần tử thuộc dòng chẵn và cột lẻ.

Bài 2. Viết chương trình nhập vào một ma trận các số nguyên có $m \times n$ phần tử ($2 \leq m \leq 5, 2 \leq n \leq 10$).

- Tính tổng các số dương trong ma trận.
- Tính tổng các giá trị trên một dòng trong ma trận
- Kiểm tra ma trận có tồn tại số dương hay không
- Kiểm tra ma trận có toàn dương hay không

Bài 3. Viết chương trình nhập vào một ma trận các số nguyên có $m \times n$ phần tử ($2 \leq m \leq 5, 2 \leq n \leq 10$).

- Tìm giá trị lớn nhất trên một dòng trong ma trận
- Tìm số chẵn xuất hiện đầu tiên trong ma trận
- Đếm số phần tử âm trên một cột trong ma trận
- Tính tích các giá trị dương trên một dòng trong ma trận

Bài 4. Viết chương trình nhập vào một ma trận các số nguyên có $m \times n$ phần tử ($2 \leq m \leq 5, 2 \leq n \leq 10$).

- Tính tích các giá trị lẻ trong ma trận.
- Tính tích các số chẵn trên một cột trong ma trận
- Tính tổng các số hoàn thiện trong ma trận
- Hãy biến đổi ma trận bằng cách thay các giá trị âm bằng trị tuyệt đối của nó.

Bài 5. Viết chương trình nhập vào một ma trận các số nguyên có $m \times n$ phần tử ($2 \leq m \leq 10, 1 \leq n \leq 10$).

- Kiểm tra ma trận có tồn tại số hoàn thiện hay không
- Kiểm tra các phần tử trong một hàng của ma trận có tăng dần hay không.
- Kiểm tra một cột trong ma trận có giảm dần hay không
- Liệt kê các cột trong ma trận có chứa số chính phương

Bài 6. Viết chương trình nhập vào một ma trận vuông có $n \times n$ phần tử là số nguyên ($2 \leq n \leq 5$).

- Đếm số phần tử âm trong ma trận
- In ra các phần tử dương chẵn nằm trên đường chéo chính
- Tính tổng các phần tử thuộc ma trận tam giác trên (không tính đường chéo) trong ma trận vuông.
- Tính tổng các phần tử trên đường chéo phụ

Bài 7. Viết chương trình nhập vào một ma trận vuông có $n \times n$ phần tử là số nguyên ($2 \leq n \leq 5$).

- a. Tìm giá trị lớn nhất trên đường chéo chính
- b. Kiểm tra ma trận nhập vào có phải là ma trận chéo không? (Ma trận chéo là ma trận vuông thỏa điều kiện $a_{ij} = 0, \forall i \neq j$; có nghĩa là: Các phần tử nằm ngoài đường chéo chính có giá trị là 0, các phần tử nằm trên đường chéo chính có giá trị khác 0)
- c. Kiểm tra ma trận nhập vào có phải là ma trận đơn vị không? (Ma trận đơn vị là ma trận chéo thỏa điều kiện $a_{ij} = 1, \forall i = 1, 2, \dots, n$; có nghĩa là: Các phần tử nằm ngoài đường chéo chính có giá trị là 0, các phần tử nằm trên đường chéo chính có giá trị là 1)
- d. Tìm giá trị lớn nhất trong ma trận tam giác trên (Ma trận tam giác trên là ma trận vuông thỏa điều kiện $a_{ij} = 0, \forall i > j$; có nghĩa là: Các phần tử nằm phía dưới đường chéo chính có giá trị là 0)