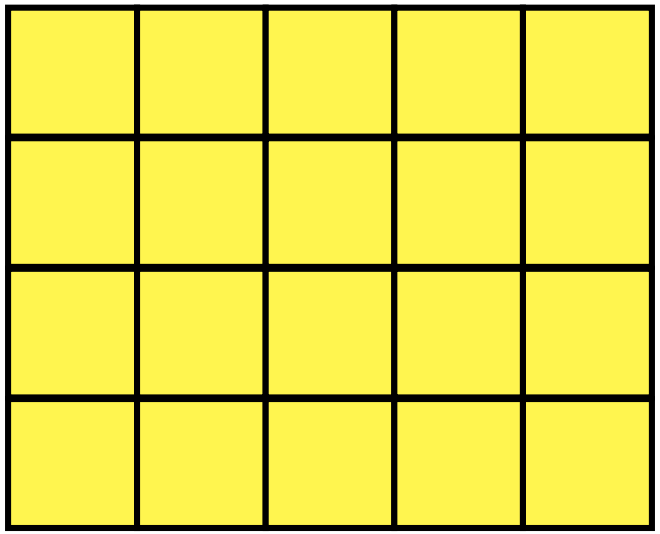
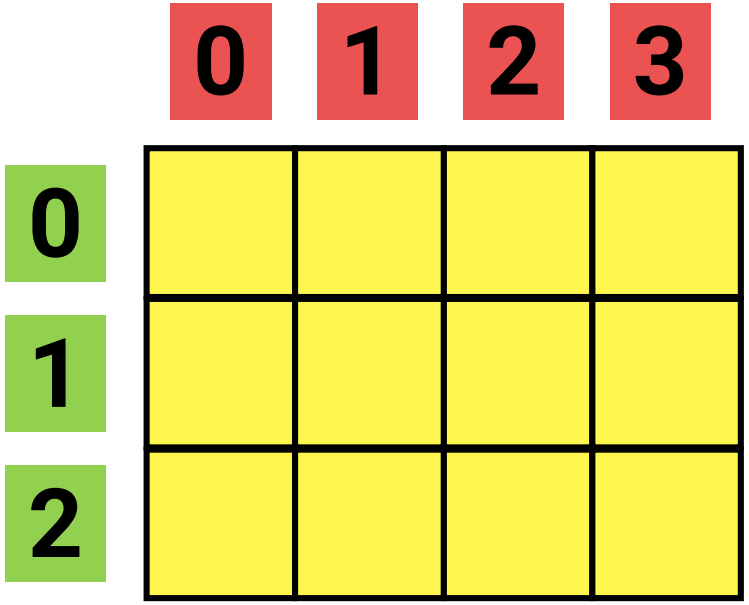
MẢNG HAI CHIỀU (2D ARRAY)



NỘI DUNG

Khái quát về mảng hai chiều:

Mảng 2 chiều được sử dụng trong các bài toán liên quan tới ma trận, bảng số. Bạn có thể coi mảng 2 chiều chính là các mảng một chiều được xếp chồng lên nhau



- Chỉ số hàng
- Chỉ số cột

1. Khai báo mảng hai chiều

Khi khai báo mảng hai chiều, các bạn cần chỉ ra số hàng, số cột của ma trận.

```c
#include <stdio.h>
int main(){
    //Mảng a gồm 3 hàng, mỗi hàng 3 cột
    int a[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9};
    }
    //Mảng b có 10 hàng, mỗi hàng 10 cột
    int b[10][10];
}
```

a[3][3]

1	2	3
4	5	6
7	8	9

2. Truy cập phần tử trong mảng:

Để truy cập vào phần tử trong mảng, các bạn dùng chỉ số hàng và chỉ số cột.

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

```
a[0][2] = 3
a[2][1] = 8
```

Chỉ số hàng và cột của mảng 2 chiều được đánh số từ 0 như mảng 1 chiều.

3. Nhập và duyệt mảng hai chiều:

```
#include <stdio.h>

int main(){
    int n, m; scanf("%d%d", &n, &m);
    int a[n][m];
    for(int i = 0; i < n; i++){
       for(int j = 0; j < m; j++){
           scanf("%d", &a[i][j]);
       }
    }
    for(int i = 0; i < n; i++){
       for(int j = 0; j < m; j++){
           printf("%d ", a[i][j]);
       }
       printf("\n");
    }
}
```

4. Các bài toán cơ bản trên mảng hai chiều:

Tìm phần tử lớn nhất, nhỏ nhất

```
#include <stdio.h>

int main(){
    int n, m; scanf("%d%d", &n, &m);
    int a[n][m];
    int max_val = -1e9, min_val = 1e9;
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            scanf("%d", &a[i][j]);
            max_val = fmax(max_val, a[i]);
            min_val = fmin(min_val, a[i]);
        }
     }
     printf("%d %d", min_val, max_val);
}
```

4. Các bài toán cơ bản trên mảng hai chiều:

Tính tổng từng hàng của mảng 2 chiều

```
#include <stdio.h>

int main(){
    int n, m; scanf("%d%d", &n, &m);
    int a[n][m];
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            scanf("%d", &a[i][j]);
        }
    }
    for(int i = 0; i < n; i++){
        int sum = 0;
        for(int j = 0; j < m; j++){
            sum += a[i][j];
        }
        printf("%d\n", sum);
    }
}
```

4. Các bài toán cơ bản trên mảng hai chiều:

Cộng trừ hai ma trận:

Trong đại số tuyến tính, ma trận tương tự như một mảng 2 chiều gồm n hàng và m cột. Để 2 ma trận có thể cộng hoặc trừ cho nhau thì chúng phải có cùng số hàng và số cột.

1	2	0
0	4	1

+

1	4	8
9	2	3

=

2	6	8
9	6	4

1	2	0
0	4	1

−

1	4	8
9	2	3

=

0	-2	-8
-9	2	-2

4. Các bài toán cơ bản trên mảng hai chiều:

Cộng 2 ma trận cỡ n hàng m cột

```
#include <stdio.h>
int main(){
    int n, m; scanf("%d%d", &n, &m);
    int a[n][m], b[n][m];
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            scanf("%d", &a[i][j]);
        }
    }
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            scanf("%d", &b[i][j]);
        }
    }
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            printf("%d ", a[i][j] + b[i][j]);
        }
        printf("\n");
    }
}
```

Trừ 2 ma trận cỡ n hàng m cột

```
#include <stdio.h>
int main(){
    int n, m; scanf("%d%d", &n, &m);
    int a[n][m], b[n][m];
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            scanf("%d", &a[i][j]);
        }
    }
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            scanf("%d", &b[i][j]);
        }
    }
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            printf("%d ", a[i][j] - b[i][j]);
        }
        printf("\n");
    }
}
```

4. Các bài toán cơ bản trên mảng hai chiều:

Nhân hai ma trận:

Giả sử có 2 ma trận a cỡ nxm, ma trận b cỡ pxq, để ma trận a có thể nhân với ma trận b thì số cột của ma trận a, tức là m phải bằng số hàng của ma trận b, tức là p.

```
a[n][m] x b[p][q] = c[n][q]
```

Khi đó m = p thì ma trận tích của a với b sẽ là ma trận c có cỡ nxq. Phần tử ở chỉ số (i, j) của ma trận tích c được tính bằng cách nhân từng cặp phần tử ở hàng i của ma trận a với các phần tử ở cột j của ma trận b.

4. Các bài toán cơ bản trên mảng hai chiều:

Nhân hai ma trận:

Nhập 2 ma trận:

```
int n, m, p;
scanf("%d%d%d", &n, &m, &p);
int a[n][m], b[m][p], c[n][p];
for(int i = 0; i < n; i++){
    for(int j = 0; j < m; j++){
        scanf("%d", &a[i][j]);
    }
}
for(int i = 0; i < m; i++){
    for(int j = 0; j < p; j++){
        scanf("%d", &b[i][j]);
    }
}
```

Tính ma trận tích và in kết quả

```
for(int i = 0; i < n; i++){
   for(int j = 0; j < p; j++){
       c[i][j] = 0;
       for(int k = 0; k < m; k++){
          c[i][j] += a[i][k] * b[k][j];
       }
   }
}
for(int i = 0; i < n; i++){
   for(int j = 0; j < p; j++){
       printf("%d ", c[i][j]);
   }
   printf("\n");
 }
```

5. Kĩ thuật duyệt các ô liền kề:

i-1, j-1	i-1, j	i-1, j+1
i, j-1	i, j	i, j+1
i+1, j-1	i+1, j	i+1, j+1

5. Kĩ thuật duyệt các ô liền kề:

Duyệt 4 ô chung cạnh với ô [i][j]

```
#include <stdio.h>

int dx[4] = {-1, 0, 0, 1};
int dy[4] = {0, -1, 1, 0};

int main(){
    int a[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };
    int i = 1, j = 1;
    for(int k = 0; k < 4; k++){
        int i1 = i + dx[k], j1 = j + dy[k];
        printf("%d ", a[i1][j1]);
    }
}
```

OUTPUT: 2 4 6 8

5. Kĩ thuật duyệt các ô liền kề:

Duyệt 8 ô chung đỉnh với ô [i][j]

```
#include <stdio.h>

int dx[8] = {-1, -1, -1, 0, 0, 1, 1, 1};
int dy[8] = {-1, 0, 1, -1, 1, -1, 0, 1};

int main(){
    int a[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };
    int i = 1, j = 1;
    for(int k = 0; k < 8; k++){
        int i1 = i + dx[k], j1 = j + dy[k];
        printf("%d ", a[i1][j1]);
    }
}
```

OUTPUT: 1 2 3 4 6 7 8 9

5. Kĩ thuật duyệt các ô liền kề:

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

Duyệt 8 ô xung quanh nước đi của quân mã

```
#include <stdio.h>

int dx[8] = {-2, -2, -1, -1, +1, +1, +2, +2};
int dy[8] = {-1, +1, -2, +2, -2, +2, -1, +1};

int main(){
    int a[6][6] = {
        {1, 2, 3, 4, 5, 6},
        {7, 8, 9, 1, 2, 5},
        {1, 2, 1, 0, 3, 5},
        {1, 2, 1, 3, 4, 9},
        {1, 2, 1, 3, 0, 4},
        {1, 8, 7, 6, 2, 9}
    };
    int i = 2, j = 3;
    for(int k = 0; k < 8; k++){
        int i1 = i + dx[k], j1 = j + dy[k];
        printf("%d ", a[i1][j1]);
    }
}
```

OUTPUT: 3 5 8 5 2 9 1 0