# LAB 6. CÁC KỸ THUẬT XỬ LÝ MẢNG HAI CHIỀU (MA TRẬN)

THỜI LƯỢNG: 6 TIẾT

## A. Mục tiêu

- Giúp sinh viên hiểu rõ và thực hiện thuần thục các kỹ thuật xử lý trên mảng hai chiều.
- Tiếp tục rèn luyện kỹ năng phát triển chương trình theo chức năng, chương trình tổ chức theo thư viện hàm và hệ thống menu.
- Sau khi hoàn thành bài thực hành này, sinh viên :
  - Nắm vững các khái niệm và thao tác nhập, xuất trên mảng hai chiều.
  - Nắm vững các kỹ thuật xử lý cơ bản trên mảng hai chiều, đặc biệt là truy cập được các phần tử của nó.
  - Biết cách định nghĩa và sử dụng kiểu dữ liệu ma trận bằng từ khóa **typedef**.
  - Truyền tham số thực, truyền tham biến.

## B. Yêu cầu

- Kết quả thực tập phần D (hướng dẫn thực hành) được thực hiện tại phòng Lab theo yêu cầu :
  - Thời gian thực hiện : 4 tiết
  - Tạo thư mục, đặt tên là **MSSV_Lab06_D_HD,** để lưu bài làm. Trong đó, MSSV là mã số của sinh viên.
  - Các bài 1,2 3, 4 : tạo các project theo hướng dẫn, lưu trử trong thu mục trên
  - Xóa thư mục Debug trong các project
  - nén thư mục **MSSV_Lab06_D_HD**
  - ***Giáo viên thu bài qua mạng tại phòng lab vào cuối buổi thực tập thứ 10***
- Kết quả thực tập phần E (bài tập bắt buộc) được thực hiện tại phòng Lab theo yêu cầu :
  - Thời gian thực hiện : 2 tiết
  - Tạo thư mục, đặt tên là **MSSV_Lab06_E_BB,** để lưu bài làm.
  - Project này thực hiện các thao tác trên ma trận vuông, gồm có 5 chức năng : Bài 1 là một chức năng; trong 2 bài : bài 2 và bài 3, mỗi bài chọn tùy ý 2 chức năng.
  - Xóa thư mục Debug trong project
  - Nén thư mục **MSSV_Lab06_E_BB**
  - ***Giáo viên thu bài qua mạng tại phòng lab vào cuối tiết 2 buổi thực tập thứ 11***

## C. Ôn tập lý thuyết

### 1. Cú pháp khai báo biến mảng hai chiều (ma trận)

Cú pháp: **KDL Tên_biến_mảng [ Số_dòng_tối_đa ] [ Số_cột_tối_đa];**

Trong đó:

- **KDL**: là kiểu dữ liệu của các phần tử chứa trong ma trận.
- **Tên_biến_mảng**: là tên của ma trận, do người lập trình đặt và phải tuân theo quy tắc đặt tên.
- **Số_dòng_tối_đa** và **Số_cột_tối_đa** là hai nguyên dương, cho biết số dòng và số cột tối đa của ma trận.

### 2. Cú pháp định nghĩa kiểu dữ liệu mảng 2 chiều

Cú pháp: **typedef KDL Tên_kiểu_mảng [ Số_dòng_tối_đa ] [ Số_cột_tối_đa];**

Trong đó:

- **KDL**: là kiểu dữ liệu của các phần tử chứa trong ma trận.
- **Tên_kiểu_mảng**: là tên của kiểu dữ liệu (mới) ma trận.
- **Số_dòng_tối_đa** và **Số_cột_tối_đa** cho biết số dòng và số cột tối đa của ma trận.

**3. Các thao tác nhập - xuất mảng hai chiều**

***a. Trường hợp không sử dụng kiểu mảng hai chiều***

```
25  // Định nghĩa hàm nhập giá trị các phần tử của ma trận
26  // bằng cách nhập lần lượt từ bàn phím.
27  // Input : a = ma trận chứa tối đa SIZE dòng, SIZE cột.
28  //         m = số dòng thực sự của ma trận.
29  //         n = số cột thực sự của ma trận.
30  // Output: Không có.
31  void NhapMang(int a[SIZE][SIZE], int m, int n)
32  {
33      // Duyệt qua từng dòng từ dòng 0 tới m-1
34      for (int i=0; i<m; i++)
35      {
36          cout << endl << "Dong thu : " << i << endl;
37
38          // Duyệt qua từng cột từ cột 0 tới n-1
39          for (int j=0; j<n; j++)
40          {
41              // Xuất thông báo yêu cầu người dùng nhập
42              cout << "a[" << i << " , " << j << "] = ";
43
44              // Chờ người dùng nhập giá trị cho ô [i,j]
45              cin >> a[i][j];
46          }
47      }
48  }
```

```
50 // Định nghĩa hàm nhập giá trị cho các phần tử của
51 // ma trận bằng cách sinh các số ngẫu nhiên
52 // Input : a = ma trận chứa tối đa SIZE dòng, SIZE cột.
53 //         m = số dòng thực sự của ma trận.
54 //         n = số cột thực sự của ma trận.
55 // Output: Không có.
56 void NhapTuDong(int a[SIZE][SIZE], int m, int n)
57 {
58     // Gieo số ngẫu nhiên đầu tiên
59     srand(time(NULL));
60
61     // Duyệt qua từng dòng từ dòng 0 tới m-1
62     for (int i=0; i<m; i++)
63     {
64         // Duyệt qua từng cột từ cột 0 tới n-1
65         for (int j=0; j<n; j++)
66         {
67             // Sinh một số ngẫu nhiên trong phạm vi
68             // [-MAX/2..MAX/2) rồi gán vào ô [i,j]
69             a[i][j] = rand() % MAX - MAX / 2;
70         }
71     }
72 }
74 // Định nghĩa hàm xuất các phần tử của mảng ra màn hình
75 // Input : a = ma trận chứa tối đa SIZE dòng, SIZE cột.
76 //         m = số dòng thực sự của ma trận.
77 //         n = số cột thực sự của ma trận.
78 // Output: Không có. Chỉ xuất ra màn hình.
79 void XuatMang(int a[SIZE][SIZE], int m, int n)
80 {
81     cout << endl << "Cac phan tu cua ma tran : " << endl;
82
83     // Duyệt qua từng dòng từ dòng 0 tới m-1
84     for (int i=0; i<m; i++)
85     {
86         // Duyệt qua từng cột từ cột 0 tới n-1
87         for (int j=0; j<n; j++)
88         {
89             // Xuất phần tử ở ô [i,j]
90             cout << a[i][j] << TAB;
91         }
92
93         // Kết thúc 1 dòng -> xuống dòng
94         cout << endl;
95     }
96     cout << endl << endl;
97 }
```

***Lưu ý quan trọng:***

- Đối với ma trận, có các giá trị sau thường đi theo:
  - ***SIZE*** là kích thước khai báo, cho biết số dòng và số cột tối đa của ma trận. Giá trị này thường được định nghĩa trước như một hằng số.
  - ***m*** (số dòng), ***n*** (số cột): là kích thước thực sự của ma trận trong mỗi lần chạy chương trình, 0 < ***m, n*** < ***SIZE***.
- Nếu là ma trận vuông, ta có ***m*** = ***n***. Khi đó, ta chỉ cần một tham số ***n***. Nguyên mẫu của các hàm nhập xuất trở thành:
  - NhapMang(int a[SIZE][SIZE], int n)
  - NhapTuDong(int a[SIZE][SIZE], int n)
  - XuatMang(int a[SIZE][SIZE], int n)
- Truyền tham số

| Tham số | Đối số |
|---|---|
| Tên biến ma trận | Tên biến ma trận (có cùng kiểu, cùng kích thước) |

***b. Trường hợp định nghĩa và sử dụng kiểu dữ liệu ma trận (MaTran)***

Trước hết, cần định nghĩa kiểu dữ liệu ma trận. Đặt tên kiểu dữ liệu mới là MaTran.

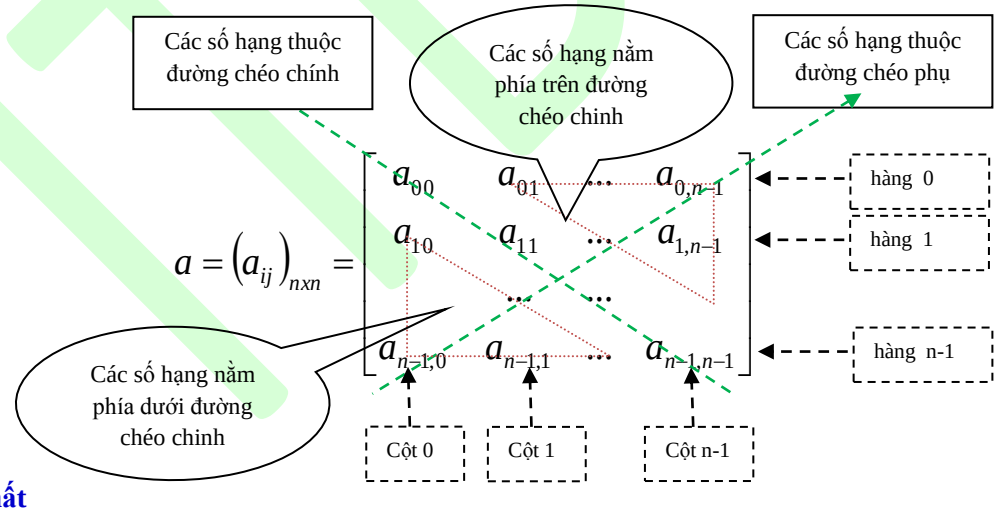
```
10  // Định nghĩa kiểu dữ liệu mảng 1 chiều
11  typedef     int     MaTran[SIZE][SIZE];
```

Sau đó, thay thế các tham số **int a[SIZE][SIZE]** trong ba hàm nhập, xuất ở trên bởi **MaTran a**. Phần nội dung bên trong hàm không thay đổi.

```
17  void NhapMang(MaTran a, int m, int n);
18  void NhapTuDong(MaTran a, int m, int n);
19  void XuatMang(MaTran a, int m, int n);
```

Với cách này, nếu cần thay đổi kiểu dữ liệu của các phần tử trong mảng hoặc kích thước mảng, ta chỉ cần sửa đổi mã lệnh ở dòng định nghĩa kiểu dữ liệu mảng (lệnh **typedef**).

**4. Kỹ thuật truy cập các phần tử của ma trận vuông**




**Các tính chất**

| STT | Đặc trưng | Truy cập chỉ số | Giá trị biến điều khiển |
|---|---|---|---|
| 1 | Các phần tử thuộc đường chéo chính | $a_{ij}$; i = j | $a_{ii}$ |
| 2 | Các phần tử thuộc đường chéo phụ | $a_{ij}$; i + j = n-1 | j = n – 1 – i |
| 3 | Các phần tử nằm dưới đường chéo chính | $a_{ij}$; i > j | i = 1→n-1, j = 0→i-1 |
| 4 | Các phần tử nằm trên đường chéo chính | $a_{ij}$; i < j | i = 0→n-2, j = i+1→n-1 |
| 5 | Các phần tử nằm trên hàng i | $a_{ij}$; j = 0→n-1 | $a_{i0}$, $a_{i1}$, $a_{i2}$, …, $a_{i,n-1}$ |
| 6 | Các phần tử nằm trên cột j | $a_{ij}$; i = 0→n-1 | $a_{0j}$, $a_{1j}$, $a_{2j}$, …, $a_{n-1,j}$ |

## D. Hướng dẫn thực hành

***Phần này xây dựng chương trình thực hiện các thao tác trên cấu trúc dữ liệu mảng 2 chiều với các chức năng cơ bản.***
***Tiếp tục tổ chức chương trình theo thư viện và có/không có hệ thống menu như lab 4, lab 5.***

***Bài 1:*** Truy cập vào các phần tử của ma trận vuông

Viết chương trình thực hiện các thao tác trên các ma trận vuông cấp ***n***. Yêu cầu của chương trình:

- Tính tổng các phần tử của ma trận
- Xuất các phần tử thuộc đường chéo chính
- Xuất các phần thuộc đường chéo phụ.
- Tính tổng các phần tử nằm phia trên đường chéo chính
- Tính tích các phần tử nằm phía dưới đường chéo phụ
- Xuất các phần tử thuộc các đường chéo song song với đường théo chính. Mỗi đường chéo xuất trên 1 dòng.

Bước 1. Tạo dự án Win32 Console Application mới. Đặt tên là **Lab06_D_Bai1**

Bước 2. Tạo cấu trúc cho chương trình như đã hướng dẫn trong **mục 2 lab 4** (từ 1- 8 để có cấu trúc nội dung tối thiểu chạy được chương trình).

Tức là ta có kết quả chương trình ở bước này như sau :

- Tập tin ***thuvien.h*** : Rỗng
- Tập tin ***menu.h*** : Rỗng
- Tập tin ***program.cpp*** có nội dung như sau :

```
#include <iostream>
#include <conio.h>
using namespace std;

#include "thuvien.h"
#include "menu.h"

void ChayChuongTrinh();

int main()
{
	ChayChuongTrinh();
	return 1;
}
void ChayChuongTrinh()
{
	_getch();
}
```

Nhấn Ctrl + F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra hoạt động của chương trình.

Bước 3:

Bước này ta định nghia kiểu dữ liệu mới, tổ chức và vận hành hệ thống menu.

- **Trong tập tin *thuvien.h* :**

Ta bổ sung định nghia hằng, kiểu dữ liệu mới :

Vì chương trình thực hiện các thao tác trên ma trận vuông, nên ta cần định nghĩa một hằng là giá trị kích thước khai báo của mảng. Ngoài ra, ta có thể định nghĩa một kiểu dữ liệu ma trận vuông (lưu ý rằng có thể không cần thực hiện định nghĩa này vì ta có thể làm trực tiếp trên biến mảng 2 chiều)

```
//Dinh nghia hang
#define SIZE 5//kich thuoc khai bao mang 2chieu
#define   TAB  '\t'

//Dinh nghia kieu du lieu moi:
typedef  int  MaTranVuong[SIZE] [SIZE];

//Khai bao nguyen mau cac ham xu ly, nhap xuat
        //bổ sung sau

//Dinh nghia cac ham xu ly, nhap xuat
        //bổ sung sau
```

- **Trong tập tin *menu.h* :**

```
// Khai báo nguyên mẫu các hàm xử lý menu
        //bổ sung sau
 // Định nghĩa các hàm xử lý menu

3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình
// Định nghĩa hàm xuất danh sách chức năng ra màn hình
//Ngoai cac chuc nang cua bai toan, ta them chuc nang xem du lieu day so
// Input : Không có
// Output: Không có
void XuatMenu()
{
        cout << endl << "============== CHON CHUC NANG =============";
        cout << endl << "0. Thoat khoi chuong trinh";
        cout << endl << "1. Nhap ma tran vuong";
        cout << endl << "2. Xem ma tran vuong";
        cout << endl << "3. Tinh tong cac phan tu cua ma tran";
        cout << endl << "4. Xuat cac phan tu thuoc duong cheo chinh";
        cout << endl << "5. Xuat cac phan tu thuoc duong cheo phu";
        cout << endl << "6. Tinh tong cac phan tu nam phia tren duong cheo chinh";
        cout << endl << "7. Tinh tich cac phan tu nam phia duoi duong cheo phu ";
        cout << endl << "8.Xuat cac duong cheo // duong cheo chinh ";
        cout << endl << "=========================================";
}
Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chọn một menu trong danh sách
// Input : soMenu = Số lượng menu có thể chọn.
// Output: Số thứ tự menu do người dùng nhập vào.
int ChonMenu(int soMenu)
{
        int stt;
        for (;;)
```

```
	{
		system("CLS");
		XuatMenu();
		cout<<"\nNhap 1 so khong khoang [0,..," << soMenu << "] de chon chuc nang, stt = ";
		cin >> stt;
		if (0 <= stt && stt <= soMenu)
			break;
	}
	return stt;

}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

### 3.3 Định nghĩa hàm xử lý menu :

Các thao tác đều thực hiện trên cùng một đầu vào là ma trận vuông, nên ta bổ sung thêm biến a kiểu MaTranVuong với kích thước mảng thực dùng trong mỗi lần thực hiện chương trình là số nguyên dương n làm đối của hàm **XuLyMenu**, ngoài tham số đã có là tham số menu. Vì ma trận vuông được nhập trong case 1 trong hàm, nên đối n trong hàm XuLyMenu ta dùng tham chiếu.

```
// Input : menu = Số thứ tự menu do người chọn,
//                ma tran vuong a
//                số nguyên dương n
// Output: Không có.
void XuLyMenu(int menu, MaTranVuong a, int  &n)
{
	// Khai báo biến

	switch (menu)
	{

		case 0:
			system("CLS");
			cout << endl<<"\n0. Thoat khoi chuong trinh\n";
			break;
		case 1:
			system("CLS");
			cout << endl << "1. Nhap ma tran vuong";
			//Bo sung sau
			break;
		case 2:
			system("CLS");
			cout << endl << "2. Xem ma tran vuong";
			//Bo sung sau
			break;
		case 3:
			system("CLS");
			cout << endl << "3. Tính tổng các phần tử của ma trận ";
			//Bo sung sau
			break;
		case 4:
			system("CLS");
			cout << endl << "4. Xuat cac phan tu thuoc duong cheo chinh";
			//Bo sung sau
			break;
```

```
		case 5:
			system("CLS");
			cout << endl << "5. Xuat cac phan tu thuoc duong cheo phu";
			//Bo sung sau
			break;
		case 6:
			system("CLS");
			cout << endl << "6. Tinh tong cac phan tu nam phia tren duong cheo chinh";
			//Bo sung sau
			break;
		case 7:
			system("CLS");
			cout << endl << "7. Tinh tich cac phan tu nam phia duoi duong cheo phu ";
			//Bo sung sau
			break;
		case 8:
			system("CLS");
			cout << endl << "7. Tinh tich cac phan tu nam phia duoi duong cheo phu ";
			cout << endl << "8.Xuat cac duong cheo // duong cheo chinh ";
			//Bo sung sau
			break;
	}
	_getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```
void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, MaTranVuong  a, int & n);
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

- Trong tập tin ***program.cpp*** :
  Hàm ***ChayChuongTrinh*** ta cập nhật lại như sau :

```
void ChayChuongTrinh()
{
	int soMenu = 8, //lưu số các chức năng
		menu, // lưu số thứ tự chức năng người dùng chọn
		n=0; //kích thước mảng và giá trị khởi tạo
	MaTranVuong  a;
	do
	{
		menu = ChonMenu(soMenu);
		XuLyMenu(menu,a,n);
	} while (menu > 0);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra sự vận hành của hệ thống menu, đặc biệt chức năng 0 - thoát khỏi chương trình.

Bước 4 :

Trong bước 4 này, ta làm công việc sau :

- Trong tập tin ***thuvien.h*** , soạn thảo các hàm nhập, xuất ma trận vuông
- Trong tập tin ***menu.h*** bổ sung xử lý chức năng xem dữ liệu trong hàm ***XuLyMenu.***

**- Trong tập tin *thuvien.h* :**

**Bổ sung các hàm nhập xuất :**

4.1 Hàm nhập dữ liệu ma trận vuông cấp n từ bàn phím

```
void NhapMaTran(MaTranVuong a, int n)
{
	int i, j;
	for (i = 0; i < n; i++) // hang i
	for (j = 0; j < n; j++)  //cot j
	{
		cout << "\na[" << i << "][" << j << "]= ";
		cin >> a[i][j];
	}
}
```

4.2 Hàm xuất dữ liệu ma trận vuông ra màn hình

```
void XuatMaTran(MaTranVuong a, int n)
{
	int i, j;
	for (i = 0; i < n; i++)
	{
		cout << '\n';
		for (j = 0; j < n; j++)
			cout << a[i][j] << TAB;
	}
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.3 Bổ sung nguyên mẫu các hàm :

```
 void NhapMaTran(MaTranVuong a, int n);
void XuatMaTran(MaTranVuong a, int n);
```

**- Trong tập tin *menu.h* :**

**Bổ sung xử lý chức năng nhập dữ liệu vào case 1, xem dữ liệu vào case 2 (Các case từ 3 đến 8 giữ nguyên)**

```
void XuLyMenu(int menu, MaTranVuong a, int &n)
{
	// Khai báo biến
	switch (menu)
	{

		case 0:
			system("CLS");
			cout <<endl<< "\n0. Thoat khoi chuong trinh\n";
			break;
		case 1:
			system("CLS");
			cout << endl << "1. Nhap ma tran vuong";
			cout << "\nNhap cap ma tran vuong : n = ";
```

```
                cin >> n;
                NhapMaTran(a, n);
                cout << "\nMa tran vuong vua nhap:\n";
                XuatMaTran(a, n);
                break;
        case 2:
                system("CLS");
                cout << endl << "2. Xem ma tran vuong";
                cout << "\nMa tran vuong hien hanh:\n";
                XuatMaTran(a, n);
                break;
        // bo sung sau
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 1, 2.

Trong các bước tiếp theo, ta lần lượt bổ sung các chức năng khác vào chương .

- Lần lượt soạn thảo từng hàm chức năng trong tập tin ***thuvien.h***,
- Lần lượt bổ sung xử lý chức năng trong hàm ***XuLyMenu*** của ***menu.h***,

***Bước 5:*** Bổ sung chức năng 3 (tính tổng các phần tử của ma trận) chương trình..

- Trong ***thuvien.h***,:

5.1 Định nghĩa hàm kiểm tra mảng a có chứa phần tử x?

```
// Input :   ma trận vuông a, kích thước n
// Output: sum = tổng giá trị các phần tử của a.
int TinhTong_MaTran(MaTranVuong a, int n)
{
    int i, j, sum = 0;
    for (i = 0; i < n; i++) // hang i
    for (j = 0; j < n; j++)  //cot j
            sum += a[i][j];
    return sum;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

5.2 - Khai báo nguyên mẫu hàm :

```
int TinhTong_MaTran(MaTranVuong a, int n);
```

- Trong ***menu.h*** :

Nội dung hàm XuLyMenu bổ sung khai báo biến sum kiểu int (để lưu tổng các phần tử ma trận) bổ sung chức năng 3 vào case 3, các case khác giữ nguyên.

```
void XuLyMenu(int menu, MaTranVuong  a, int &n)
{
    // Khai báo biến
    int sum;
    switch (menu)
    {
        //. . .
        case 3:
                system("CLS");
                cout << endl << "3. Tinh tong cac phan tu cua ma tran ";
```

```
                cout << "\nMa tran hien hanh :\n";
                XuatMaTran(a, n);
                sum = TinhTong_MaTran(a, n);
                cout << "\nTong cac phan tu cua ma tran : sum = " << sum;
                break;
        //. . .
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 3.

***Bước 6:*** Bổ sung chức năng 4 (xuất các phần tử thuộc đường chéo chính) vào chương trình..

- Trong ***thuvien.h***,:

6.1 Định nghĩa hàm xuất các phần tử thuộc đường chéo chính

```
// Input :   ma trận vuông a, kích thước n
// Output: không có
void XuatDuongCheoChinh(MaTranVuong a, int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout << a[i][i] << TAB;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

6.2 - Khai báo nguyên mẫu hàm :

```
void XuatDuongCheoChinh(MaTranVuong a, int n);
```

- Trong ***menu.h*** :

Trong hàm XuLyMenu bổ sung chức năng 4 vào case 4, các case khác giữ nguyên.

```
void XuLyMenu(int menu, MaTranVuong  a, int &n)
{
    // Khai báo biến
    int sum;
    switch (menu)
    {
        //. . .
        case 4:
            system("CLS");
            cout << endl << "4. Xuat cac phan tu thuoc duong cheo chinh";
            cout << "\nMa tran hien hanh :\n";
            XuatMaTran(a, n);
            cout << "\nCac phan tu thuoc duong cheo chinh cua a:\n";
            XuatDuongCheoChinh( a, n);
            break;
        //. . .
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 4.

***Bước 7:*** Bổ sung chức năng 5 (xuất các phần tử thuộc đường chéo phụ) vào chương trình..

- Trong ***thuvien.h***,:

7.1 Định nghĩa hàm xuất các phần tử thuộc đường chéo chính

```
// Input :   ma trận vuông a, kích thước n
// Output: không có
void XuatDuongCheoPhu(MaTranVuong a, int n)
{
	int i;
	for (i = 0; i < n; i++)
		cout << a[i][n-i-1] << TAB;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

7.2 - Khai báo nguyên mẫu hàm :

- Trong ***menu.h*** :

Trong hàm XuLyMenu bổ sung chức năng 5 vào case 5, các case khác giữ nguyên.

```
void XuLyMenu(int menu, MaTranVuong  a, int &n)
{
	// Khai báo biến
	int sum;
	switch (menu)
	{
		//. . .
		case 5:
			system("CLS");
			cout << endl << "5. Xuat cac phan tu thuoc duong cheo phu";
			cout << "\nMa tran hien hanh :\n";
			XuatMaTran(a, n);
			cout << "\nCac phan tu thuoc duong cheo chinh cua a:\n";
			XuatDuongCheoPhu(a, n);
			break;
		//. . .
	}
	_getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 5.

***Bước 8:*** Bổ sung chức năng 6 (tính tổng các phần tử phía trên đường chéo chính) vào chương trình..

- Trong ***thuvien.h***,:

8.1 Định nghĩa hàm tính tổng các phần tử phía trên đường chéo chính

```
// Input :   ma trận vuông a, kích thước n
// Output: sum = tổng các phần tử phía trên đường chéo chính
int int TinhTong_Tren_CheoChinh(MaTranVuong a, int n)
{
	int i, j, sum = 0;
	for (i = 0; i < n-1; i++) // hang i
	for (j = i+1; j < n; j++)  //cot j
		sum += a[i][j];
	return sum;
```

```
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

8.2 - Khai báo nguyên mẫu hàm :

```
int TinhTong_Tren_CheoChinh (MaTranVuong a, int n);
```

- Trong ***menu.h*** :

Trong hàm XuLyMenu bổ sung chức năng 6 vào case 6, các case khác giữ nguyên.

```
void XuLyMenu(int menu, MaTranVuong  a, int &n)
{
	// Khai báo biến
	int sum;
	switch (menu)
	{
		//. . .
		case 6:
			system("CLS");
			cout << endl << "6. Tinh tong cac phan tu nam phia tren duong cheo chinh";
			cout << "\nMa tran hien hanh :\n";
			XuatMaTran(a, n);
			sum =  TinhTong_Tren_CheoChinh (a, n);
			cout << "\nTong cac phan tu phia tren duong cheo chinh: sum = " << sum;
			break;
		//. . .
	}
	_getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 6.

***Bước 9:*** Bổ sung chức năng 7 (tính tích các phần tử phía dưới đường chéo phụ) vào chương trình..

- Trong ***thuvien.h***,:

9.1 Định nghĩa hàm tính tính tích các phần tử phía dưới đường chéo phụ

```
// Input :   ma trận vuông a, kích thước n
// Output: p = tích các phần tử phía dưới đường chéo phụ
//Tinh tch cac phan tu nam phia duoi duong cheo phu cua ma tran
int TinhTich_duoi_CheoPhu(MaTranVuong a, int n)
{
	int i, j, p = 1;
	for (i = 1; i < n ; i++) // hang i
	for (j = n-i; j < n; j++)  //cot j
		p *= a[i][j];
	return p;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

9.2 - Khai báo nguyên mẫu hàm :

```
int TinhTich_duoi_CheoPhu(MaTranVuong a, int n);
```

- Trong ***menu.h*** :

Trong hàm XuLyMenu, khai báo thêm biến p kiểu int để lưu trữ kết quả tích các phần tử phía dưới đường chéo phụ, bổ sung chức năng 7 vào case 7, các case khác giữ nguyên.

```
void XuLyMenu(int menu, MaTranVuong  a, int &n)
{
	// Khai báo biến
	int sum, p;
	switch (menu)
```

```
	{
		//. . .
		case 7:
			system("CLS");
			cout << endl << "7. Tinh tich cac phan tu nam phia duoi duong cheo phu";
			cout << "\nMa tran hien hanh :\n";
			XuatMaTran(a, n);
			p = TinhTich_duoi_CheoPhu(a, n);
			cout << "\nTich cac phan tu nam phia duoi duong cheo phu: p = " << p;
			break;
		//. . .
	}
	_getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 7.

***Bước 10:*** Bổ sung chức năng 8 (xuat cac duong cheo song song duong cheo chinh) vào chương trình..

- Trong ***thuvien.h***,:

10.1 Định nghĩa hàm xuất các đường chéo song song đường cheo chính

```
// Input :   ma trận vuông a, kích thước n
// Output: không có
void Xuat_DuongCheo_SS_DCChinh(MaTranVuong a, int n)
{
	int i, //hang
		j,//cot
		k;//cac duong cheo
	cout << "\n\nCac duong cheo phia tren duong cheo chinh:\n";
	for (k = n - 1; k >= 1; k--)
	{
		cout << "\nduong cheo thu " << k << ":\t";
		for (i = 0; i < n; i++)
		for (j = i + 1; j < n; j++)
		if (j - i == k)
			cout << a[i][j] << '\t';
	}

	cout << "\n\nCac duong cheo phia duoi duong cheo chinh:\n";
	for (k = n - 1; k >= 1; k--)
	{
		cout << "\nduong cheo thu " << k << ":\t";

		for (i = 1; i < n; i++)
		for (j = 0; j < i; j++)
		if (i - j == k)
			cout << a[i][j] << '\t';
	}
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

10.2 - Khai báo nguyên mẫu hàm :

```
void Xuat_DuongCheo_SS_DCChinh(MaTranVuong a, int n);
```

- Trong ***menu.h*** :

Trong hàm XuLyMenu bổ sung chức năng 8 vào case 8, các case khác giữ nguyên.

```
void XuLyMenu(int menu, MaTranVuong  a, int &n)
{
	// Khai báo biến
	int sum, p;
	switch (menu)
	{
		//. . .
		case 8:
			system("CLS");
			cout << endl << "8.Xuat cac duong cheo // duong cheo chinh ";
			cout << "\nMa tran hien hanh :\n";
			XuatMaTran(a, n);
			Xuat_DuongCheo_SS_DCChinh(a, n);
			break;
		//. . .
	}
	_getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 8

***Bước 11:*** Bổ sung chức năng 9 (xuat cac duong cheo song song duong cheo phu) vào chương trình..

- Trong ***thuvien.h***,:

11.1 Định nghĩa hàm xuất các đường chéo song song đường chéo phụ

```
// Input :   ma trận vuông a, kích thước n
// Output: không có

void Xuat_DuongCheo_SS_DCPhu(MaTranVuong a, int n)
{
	int i, //hang
		j,//cot
		k;//cac duong cheo
	cout << "\n\nCac duong cheo nam ben trai duong cheo phu:\n";
	for (k = 0; k < n - 1; k++)
	{
		cout << "\nduong cheo thu " << k + 1 << ":";
		cout << endl;
		for (i = 0; i < n - 1; i++)
		for (j = 0; j < n - 1 - i; j++)
		if (j + i == k)
			cout << a[i][j] << '\t';
	}

	cout << "\n\nCac duong cheo nam ben phai duong cheo phu:\n";
	for (k = n; k < 2 * n - 1; k++)
	{
		cout << "\nduong cheo thu " << k - n + 1 << ":";
		cout << endl;
		for (i = 1; i < n; i++)
		for (j = n - i; j < n; j++)
		if (j + i == k)
			cout << a[i][j] << '\t';
```

```
	}
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

11.2 - Khai báo nguyên mẫu hàm :

```
void Xuat_DuongCheo_SS_DCPhu(MaTranVuong a, int n);
```

- Trong ***menu.h*** :

Trong hàm XuLyMenu bổ sung chức năng 9 vào case 9, các case khác giữ nguyên.

```
void XuLyMenu(int menu, MaTranVuong  a, int &n)
{
	// Khai báo biến
	int sum, p;
	switch (menu)
	{
		//. . .
		case 9:
			system("CLS");
			cout << endl << "8.Xuat cac duong cheo // duong cheo chinh ";
			cout << "\nMa tran hien hanh :\n";
			XuatMaTran(a, n);
			Xuat_DuongCheo_SS_DCPhu(a, n);
			break;
		//. . .
	}
	_getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra kết quả thực hiện chức năng 9.

Kiểm tra tất cả các chức năng chương trình - Kết thúc chương trình

Lưu ý:

- Mỗi lần chạy chương trình, cần chọn chức năng 1 đầu tiên để nhập dữ liệu, khi đó mới có dữ liệu để các chức năng khác thực hiện. Trong quá trình lặp tùy chọn thực hiện các chức năng, nếu cần thay đổi bộ dữ liệu khác, chỉ cần chọn lại chức năng 1 để nhập lại dữ liệu.
- Trong trường hợp lần đầu người dùng không chọn chức năng 1 để tạo dữ liệu trước mà chọn các chức năng khác, ta nên xử lý như thế nào?

**Bài 2: Tính toán trên ma trận**

Viết chương trình thực hiện các thao tác trên ma trận cấp ***mxn*** với các phần tử là số nguyên. Chương trình cho phép người dùng chọn chức năng từ các menu sau:

- Tính giá trị lớn nhất của ma trận
- Tính giá trị lớn nhất hang i
- Tính Tổng các phần tử thuộc hàng ***i***.
- Tính giá trị nhỏ nhất cột j
- Tính Tích các phần tử thuộc cột ***j***.
- Xuất ra màn hình các phần tử $a_{ij}$ thỏa mãn: $a_{ij}$ là phần tử lớn nhất hàng ***i*** và nhỏ nhất cột ***j***.

Bước 1: Tạo dự án Win32 Console Application mới. Đặt tên là **Lab06_D_Bai1**

Buốc 2: Tạo cấu trúc cho chương trình như bài trên (bài 1, từ 1- 8 để có cấu trúc nội dung tối thiểu chạy được chương trình).
Bước 3: Tổ chức hệ thống menu và kiểm tra sự vận hành của nó.

- **Trong tập tin *thuvien.h* :**
  Vì thực hiện trên mảng 2 chiều nên ta bổ sung định nghia hằng là kích thước khai báo của mảng.
  Ngoài ra, bài này ta sử dụng trực tiếp biến ma trận mà không định nghĩa một kiểu ma trận.

```
//Dinh nghia hang
#define SIZE 5//kich thuoc khai bao mang 2chieu
#define   TAB  '\t'

//Dinh nghia kieu du lieu moi:

//Khai bao nguyen mau cac ham xu ly, nhap xuat
        //bổ sung sau

//Dinh nghia cac ham xu ly, nhap xuat
        //bổ sung sau
```

- **Trong tập tin *menu.h* :**
  **ta viết lại như sau (cấu trúc giống như bước 9 mục 2 lab 4, chỉ thay đổi nội dung theo yêu cầu bài toán) :**

```
// Khai báo nguyên mẫu các hàm xử lý menu
        //bổ sung sau
 // Định nghĩa các hàm xử lý menu

3.1 Định nghĩa hàm xuất danh sách chức năng ra màn hình
// Định nghĩa hàm xuất danh sách chức năng ra màn hình
//Ngoai cac chuc nang cua bai toan, ta them chuc nang xem du lieu day so
// Input : Không có
// Output: Không có
void XuatMenu()
{
        cout << endl << "============== CHON CHUC NANG =============";
        cout << endl << "0. Thoat khoi chuong trinh";
        cout << endl << "1. Nhap ma tran chu nhat m x n";
        cout << endl << "2. Xem ma tran chu nhat";
        cout << endl << "3. Tinh gia tri lon nhat cua ma tran";
        cout << endl << "4. Tinh gia tri lon nhat hang i ";
        cout << endl << "5. Tinh tong cacphan tu hang i ";
        cout << endl << "6. Tinh gia tri nho nhat cot  j ";
        cout << endl << "7. Tinh tich cac phan tu cot  j ";
        cout << endl << "8. Xuat aij : lon nhat hang I van ho nhat cot j ";
        cout << endl << "=========================================";
}
Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.2 Định nghĩa hàm chọn một menu trong danh sách
// Input : soMenu = Số lượng menu có thể chọn.
// Output: Số thứ tự menu do người dùng nhập vào.
int ChonMenu(int soMenu)
```

```
{
	int stt;
	for (;;)
	{
		system("CLS");
		XuatMenu();
		cout<<"\nNhap 1 so khong khoang [0,.." << soMenu << "] de chon chuc nang, stt = ";
		cin >> stt;
		if (0 <= stt && stt <= soMenu)
			break;
	}
	return stt;

}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.3 Định nghĩa hàm xử lý menu :

Các thao tác đều thực hiện trên cùng một đầu vào là ma trận mxn, nên ta bổ sung thêm biến ma trận chư nhạt a với kich thước khai báo SIZE x SIZE, m hang và n cột làm đối của hàm **XuLyMenu**, ngoài tham số đã có là tham số menu. Vì ma trận được nhập trong case 1 của hàm, nên đối m, n trong hàm XuLyMenu ta dùng tham chiếu.

```
// Input : menu = Số thứ tự menu do người chọn,
//                ma tran chu nhat a
//                số nguyên dươngm, n
// Output: Không có.
void XuLyMenu(int menu, int  a[SIZE][SIZE], int &m,  int  &n)
{
	// Khai báo biến

	switch (menu)
	{

		case 0:
			system("CLS");
			cout << endl<<"\n0. Thoat khoi chuong trinh\n";
			break;
		case 1:
			system("CLS");
			cout << endl << "1. Nhap ma tran chu nhat";
			//Bo sung sau
			break;
		case 2:
			system("CLS");
			cout << endl << "2. Xem ma tran chu nhat";
			//Bo sung sau
			break;
		case 3:
			system("CLS");
			cout << endl << "3. Tinh gia tri lon nhat cua ma tran";
			//Bo sung sau
			break;
		case 4:
			system("CLS");
```

```
                cout << endl << "4. Tinh gia tri lon nhat hang i ";
                //Bo sung sau
                break;
        case 5:
                system("CLS");
                cout << endl << "5. Tinh tong cac phan tu  hang i ";
                //Bo sung sau
                break;

        case 6:
                system("CLS");
                cout << endl << "6. Tinh gia tri nho nhat cot  j ";
                //Bo sung sau
                break;
        case 7:
                system("CLS");
                cout << endl << "7. Tinh tich cac phan tu cot  j ";
                //Bo sung sau
                break;
        case 8:
                system("CLS");
                cout << endl << "8. Xuat aij : lon nhat hang i va nho nhat cot j ";
                //Bo sung sau
                break;
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

3.4 Bổ sung khai báo nguyên mẫu các hàm tổ chức menu trong phần khai báo nguyên mẫu hàm

```
void XuatMenu();
int ChonMenu(int soMenu);
void XuLyMenu(int menu, int  a[SIZE][SIZE], int &m,  int  &n);
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có

- Trong tập tin ***program.cpp*** :

  Hàm ***ChayChuongTrinh*** ta cập nhật lại như sau :

```
void ChayChuongTrinh()
{
    int soMenu = 8, //lưu số các chức năng
        menu, // lưu số thứ tự chức năng người dùng chọn
        m = 0,// kích thước hang và giá trị khởi tạo
        n=0; //kích thước cot và giá trị khởi tạo
    int  a[SIZE][SIZE];
    do
    {
        menu = ChonMenu(soMenu);
        XuLyMenu(menu,a,m,n);
    } while (menu > 0);
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

Kiểm tra sự vận hành của hệ thống menu.
Kiểm tra chức năng thoát khỏi chương trình (chọn 0)

Bước 4 : (tổ chức nhập xuất dữ liệu)
Trong bước 4, ta làm công việc sau :
- Trong ***program.cpp*** ta bổ sung thêm các thư viện cần thiết.
- Trong ***thuvien.h*** , soạn thảo các hàm nhập, xuất ma trận
- Trong ***menu.h*** bổ sung xử lý chức năng nhập ma trận, xem ma trận trong hàm ***XuLyMenu.***

**- Trong tập tin *program.cpp*** bổ sung thêm các thư viện chứa các hàm xử lý số ngẫu nhiên :

```
#include <time.h>
#include <stdlib.h>
```

**- Trong tập tin *thuvien.h* :**

**Bổ sung các hàm nhập xuất :**

4.1 Hàm nhập dữ liệu ma trận m x n tự động:

```
void NhapTuDong_MaTran(int a[SIZE][SIZE], int m, int n)
{
	int i, j;
	srand((unsigned)time(NULL));
	for (i = 0; i < m; i++)
	for (j = 0; j < n; j++)
		a[i][j] = (rand() % (m*n)) - (m*n) / 2;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.2 Hàm xuất dữ liệu ma trận m x n ra màn hình

```
void XuatMaTran(int a[SIZE][SIZE], int m, int n)
{
	int i, j;
	for (i = 0; i < m; i++)
	{
		cout << endl;
		for (j = 0; j < n; j++)
			cout << a[i][j] << TAB;
	}
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

4.3 Bổ sung nguyên mẫu các hàm :

```
void NhapTuDong_MaTran(int a[SIZE][SIZE], int m, int n);
void XuatMaTran(int a[SIZE][SIZE], int m, int n);
```

**- Trong tập tin *menu.h* :**

Bổ sung xử lý chức năng nhập dữ liệu vào case 1, xem dữ liệu vào case 2 (Các case từ 3 đến 7 giữ nguyên)

```
void XuLyMenu(int menu, int  a[SIZE][SIZE], int &m,  int  &n)
{
	// Khai báo biến

	switch (menu)
	{
```

```
                //…
                case 1:
                        system("CLS");
                        cout << endl << "1. Nhap ma tran chu nhat";
                        cout << "\nNhap so hang : m = ";
                        cin >> m;
                        cout << "\nNhap so cot : n = ";
                        cin >> n;
                        NhapTuDong_MaTran(a,m, n);
                        cout << "\nMa tran vua nhap:\n";
                        XuatMaTran(a, m, n);
                        break;
                case 2:
                        system("CLS");
                        cout << endl << "2. Xem ma tran chu nhat";
                        cout << "\nMa tran hien hanh:\n";
                        XuatMaTran(a, m, n);
                        break;
                //. . .
        }
        _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện các chức năng 1, 2.

Trong các bước tiếp theo, ta bổ sung từng chức năng vào chương trình :

- Lần lượt soạn thảo từng hàm chức năng trong tập tin ***thuvien.h***,
- Lần lượt bổ sung xử lý chức năng trong hàm ***XuLyMenu*** của ***menu.h***,

***Bước 5:*** Bổ sung chức năng 3 (tính giá trị lớn nhất của ma trận) chương trình..

**- Trong tập tin *thuvien.h* :**

5.1 Định nghĩa hàm tính gia trị lớn nhất của ma trận :

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

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

5.2 Bổ sung nguyên mẫu các hàm :

```
int Tinh_Max_MaTran(int a[SIZE][SIZE], int m, int n);
```

**- Trong tập tin *menu.h* :**

Khai báo thêm biến kq kiểu int để lưu trử kết quả, bổ sung xử lý chức năng 3 vào case 3:

```
void XuLyMenu(int menu, int  a[SIZE][SIZE], int &m,  int  &n)
{
```

```
	// Khai báo biến
	int  kq;
	switch (menu)
	{
		//…
		case 3:
			system("CLS");
			cout << endl << "3. Tinh gia tri lon nhat cua ma tran";
			kq = Tinh_Max_MaTran(a, m, n);
			cout << "\nMa tran hien hanh:\n";
			XuatMaTran(a, m, n);
			cout << "\nMax(a) = " << kq;
			break;
		//. . .
	}
	_getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện  chức năng 3..

***Bước 6:*** *Bổ sung chức năng 4 (tính giá trị lớn nhất hàng i ) vào chương trình.*

**- Trong tập tin *thuvien.h* :**

6.1 Định nghĩa hàm tính gia trị lớn nhất hàng i :

```
int Tinh_Max_hang_i(int a[SIZE][SIZE], int i, int n)
{
	int maxi,j;
	maxi = a[i][0];//hang i cot 0
	for (j = 1; j < n; j++)
	if (maxi < a[i][j])
		maxi = a[i][j];
	return maxi;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

6.2 Bổ sung nguyên mẫu các hàm :

```
int Tinh_Max_hang_i(int a[SIZE][SIZE], int i, int n);
```

**- Trong tập tin *menu.h* :**

Trong hàm XuLyMenu, khai báo thêm i để lưu trử giá trị hàng nhập tư bàn phím, bổ sung xử lý chức năng 4 vào case 4:

```
void XuLyMenu(int menu, int  a[SIZE][SIZE], int &m,  int  &n)
{
	// Khai báo biến
	int  kq, i;
	switch (menu)
	{
		//…
		case 4:
			system("CLS");
```

```
                        cout << endl << "4. Tinh gia tri lon nhat hang i ";
                        do
                        {
                                cout << "\nChon hang i (0 <= i <= "<<m-1<<") : i = ";
                                cin >> i;
                        } while (i < 0 || i > m - 1);
                        kq = Tinh_Max_hang_i(a, i, n);
                        cout << "\nMa tran hien hanh:\n";
                        XuatMaTran(a, m, n);
                        cout << "\nMax(hang "<<i<<") = " << kq;
                        break;
                //. . .
        }
        _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 4..

***Bước 7:*** *Bổ sung chức năng 5 (tính tổng hàng i ) vào chương trình.*

**- Trong tập tin *thuvien.h* :**

7.1 Định nghĩa hàm tính tổng hàng i :

```
int Tinh_Tong_Hang_i(int a[SIZE][SIZE], int i, int n)
{
        int j, sum;
        sum = 0;
        for (j = 0; j < n; j++)
                sum += a[i][j];
        return sum;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

7.2 Bổ sung nguyên mẫu các hàm :

```
int Tinh_Tong_Hang_i(int a[SIZE][SIZE], int i, int n);
```

**- Trong tập tin *menu.h* :**

Trong hàm XuLyMenu bổ sung xử lý chức năng 5 vào case 5:

```
void XuLyMenu(int menu, int  a[SIZE][SIZE], int &m,  int  &n)
{
        // Khai báo biến
        int  kq, i;
        switch (menu)
        {
                //…
                case 5:
                        system("CLS");
                        cout << endl << "5. Tinh gia tri lon nhat hang i ";
                        do
                        {
                                cout << "\nChon hang i (0 <= i <= "<<m-1<<") : i = ";
```

```
                cin >> i;
            } while (i < 0 || i > m - 1);
            kq = Tinh_Tong_hang_i(a, i, n);
            cout << "\nMa tran hien hanh:\n";
            XuatMaTran(a, m, n);
            cout << "\nTong (hang "<<i<<") = " << kq;
            break;
        //. . .
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 5.

***Bước 8:*** *Bổ sung chức năng 6 (tính giá trị nhỏ nhất cột j ) vào chương trình..*

**- Trong tập tin *thuvien.h* :**

8.1 Định nghĩa hàm tính gia trị nhỏ nhất cột j :

```
int Tinh_Min_Cot_j(int a[SIZE][SIZE], int m, int j)
{
    int minj, i;
    minj = a[0][j];//hang i cot 0
    for (i = 1; i < m; i++)
    if (minj > a[i][j])
        minj = a[i][j];
    return minj;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

8.2 Bổ sung nguyên mẫu các hàm :

```
int Tinh_Min_Cot_j(int a[SIZE][SIZE], int m, int j);
```

**- Trong tập tin *menu.h* :**

Trong hàm XuLyMenu, khai báo thêm j để lưu trử giá trị cột nhập tư bàn phím, bổ sung xử lý chức năng 6 vào case 6:

```
void XuLyMenu(int menu, int  a[SIZE][SIZE], int &m,  int  &n)
{
    // Khai báo biến
    int  kq, i,j;
    switch (menu)
    {
        //…
        case 6:
            system("CLS");
            cout << endl << "6. Tinh gia tri nho nhat cot  j ";
            do
            {
                cout << "\nChon cot j (0 <= j <= " << n - 1 << ") : j = ";
                cin >> j;
            } while (j < 0 || j > n - 1);
            kq = Tinh_Min_Cot_j(a, m, j);
            cout << "\nMa tran hien hanh:\n";
```

```
                XuatMaTran(a, m, n);
                cout << "\nMax(cot " << j << ") = " << kq;
                break;
        //. . .
    }
    _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 6.

***Bước 9:*** *Bổ sung chức năng 7 (tính tích cột j ) vào chương trình..*

**- Trong tập tin *thuvien.h* :**

9.1 Định nghĩa hàm tính tích cột j :

```
int Tinh_Tich_Cot_j(int a[SIZE][SIZE], int m, int j)
{
    int p, i;
    p = 1;
    for (i = 0; i < m; i++)
        p *= a[i][j];
    return p;
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

9.2 Bổ sung nguyên mẫu các hàm :

```
int Tinh_Tich_Cot_j(int a[SIZE][SIZE], int m, int j);
```

**- Trong tập tin *menu.h* :**

Trong hàm XuLyMenu bổ sung xử lý chức năng 7 vào case 7:

```
void XuLyMenu(int menu, int  a[SIZE][SIZE], int &m,  int  &n)
{
    // Khai báo biến
    int  kq, i,j;
    switch (menu)
    {
        //…
        case 7:
            system("CLS");
            cout << endl << "7. Tinh tich cot  j ";
            do
            {
                cout << "\nChon cot j (0 <= j <= " << n - 1 << ") : j = ";
                cin >> j;
            } while (j < 0 || j > n - 1);
            kq = Tinh_Tich_Cot_j(a, m, j);
            cout << "\nMa tran hien hanh:\n";
            XuatMaTran(a, m, n);
            cout << "\nTich(cot " << j << ") = " << kq;
            break;
        //. . .
    }
```

```
	_getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 7.

***Bước 10:*** *Bổ sung chức năng 8 (*Xuat aij : lon nhat hang i va nho nhat cot j *) vào chương trình..*

**- Trong tập tin *thuvien.h* :**

10.1 Định nghĩa hàm Xuat aij : lon nhat hang i va nho nhat cot j :

```
void MaxHang_MinCot(int a[SIZE][SIZE], int m, int n)
{
	int i, j;
	int dau = 0;
	for (i = 0; i < m; i++)
	for (j = 0; j < n; j++)
	if (a[i][j] == Tinh_Max_Hang_i(a, i, n) && a[i][j] == Tinh_Min_Cot_j(a, m, j))
	{
		dau = 1;
		break;
	}
	if (!dau)
		cout << "\nKhong co phan tu nao thoa man dieu kien bai toan";
	else
	{
		cout << "\nCac phan tu a[i][j] thoa : Max hang i va Min cot j:\n";
		for (i = 0; i < m; i++)
		for (j = 0; j < n; j++)
		if (a[i][j] == Tinh_Max_Hang_i(a, i, n) && a[i][j] == Tinh_Min_Cot_j(a, m, j))
			cout << "\na[" << i << "][" << j << "] = " << a[i][j]
		<<" : Max hang "<<i<<" va Min cot "<<j;
	}
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.

10.2 Bổ sung nguyên mẫu các hàm :

```
void MaxHang_MinCot(int a[SIZE][SIZE], int m, int n);
```

**- Trong tập tin *menu.h* :**

Trong hàm XuLyMenu bổ sung xử lý chức năng 8 vào case 8:

```
void XuLyMenu(int menu, int  a[SIZE][SIZE], int &m,  int  &n)
{
	// Khai báo biến
	int  kq, i,j;
	switch (menu)
	{
		//…
		case 8:
			system("CLS");
			cout << endl << "8. Xuat aij : lon nhat hang i va nho nhat cot j";
			cout << "\nMa tran hien hanh:\n";
			XuatMaTran(a, m, n);
			MaxHang_MinCot(a, m, n);
```

```
                break;
            //. . .
        }
        _getch();
}
```

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi nếu có.
Kiểm tra kết quả thực hiện chức năng 8.

***Kiểm tra tất cả các chức năng – kết thúc chương trình.***

**Bài 3: Xoắn ốc**

Viết chương trình cho phép người dùng nhập vào số nguyên ***n***. Sau đó, xuất ra màn hình ma trận vuông cấp ***n*** sau khi đã điền các số từ ***1*** đến $n^2$ theo chiều xoắn ốc như hình dưới đây (với n = 5)

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

***Mấu chốt của chương trình là viết hàm tạo được ma trận xoắn ốc gồm $n^2$ số nguyên dương đầu tiên như hình vẽ. sau đó xuất ma trận đã tạo được.***

***Chương trình sẽ tổ chức như mục 1, phần C, lab 4, chỉ có 2 tập tin : program.cpp và thuvien.h ( không có tập tin menu.h vì bài toán không yêu cầu tổ chức tùy chọn menu)***

Bước 1: Tạo dự án Win32 Console Application mới. Đặt tên là **Lab06_D_Bai3**
Bước 2: Tạo cấu trúc chương trình như đã hướng dẫn trong **mục 1 phần C lab 4** (từ bước 1 đến bước 7).
Tức là ta đã có phần chương trình cốt lõi sau :

- Trong tập tin ***program.cpp*** ta có nội dung sau :

```
#include <iostream>
#include <conio.h>

using namespace std;

#include "thuvien.h"
void ChayChuongTrinh();
int main()
{
        ChayChuongTrinh();
        return 1;
}
void ChayChuongTrinh()
{
        _getch();
}
```

- Tập tin ***thuvien.h*** là rỗng.

Nhấn Ctrl+F5 để chạy chương trình, sửa lỗi neus có

Bước 3: Bổ xung thao tác xuất ma trận

- Trong tập tin ***program.cpp*** :
  Bổ sung thư viện ***<iomaip> //do xuất có định dạng***

- Trong tập tin ***thuvien.h*** bổ sung các nội dung:

```
//Định nghĩa hằng
#define MAX 10 //kích thước tối đa ma trận vuông
//Khai báo nguyên mẫu các hàm xử lý

        //. . . (bổ sung sau)

//Định nghĩa các hàm xử lý
```

### *3.1 Hàm xuất ma trận vuông*

```
void XuatMaTran(int a[MAX][MAX], int n)
{
        Int i, j;
        for (i = 0; i<n; i++)
        {
                cout << "\n\n\n";
                for ( j = 0; j < n; j++)
                        cout << setw(5) << a[i][j];
        }
}
```

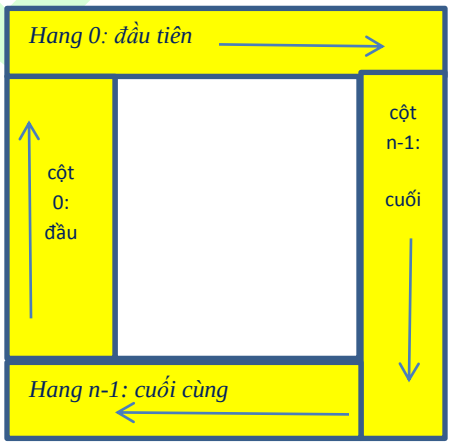
Nhấn Ctrl+F5 chạy chương trình, sửa lỗi nếu có.

### *3.2 Khai bao nguyên mẫu :*

```
void XuatMaTran(int a[MAX][MAX], int n);
```

Bước 4: Bổ xung thao tác tạo ma trận xoắn ốc

***Vấn đề chính là : ∀value ∈[1,$n^2$ ], tìm hàng, cột ∈[0,n-1] : a[hang,cột] = value sao cho ma trận tạo được là ma trận xoắn (các value tăng dần theo vòng xoắn].***




***Thuật toán thực hiện theo nhiều vòng xoắn:***

***- Vòng xoắn đầu tiên :***

***+ Gán n value từ 1 đến n cho hang 0 (đầu tiên).***

*(tổng quát, ta gọi là hang trên)*

*a[hangTren][cột] : cột từ 0 đến n-1; hangTren = 0;*

***Để chuẩn bị cho vòng xoắn sau, hàng tăng lên 1 đơn vị (hangTren++)***

*+ Theo chiều xoắn, Gán n-1 value kế tiếp cho cột n-1(cột cuối) kể từ hàng 1 (bỏ đivị trí hàng đâu cột cuối là a[0][n-1] đã gán rồi).*

*(Tổng quat, ta gọi là cột phải)*

*a[hàng][cotPhai] : hàng từ 1 đến n-1; cotPhai = n-1;*

***Để chuẩn bị cho vòng xoắn sau, cột phải giảm 1 đơn vị (cotPhai--)***

+ Theo chiều xoắn, ta gán ngược n-1 value kế tiếp cho hàng n-1 ( cuối cùng ), do a[n-1][n-1] đã được gán rồi.

(Tổng quát, ta gọi là hàng dưới)

*a[hangDuoi][cột] : cột n-2 đến 0; hangDuoi = n-1;*

***Để chuẩn bị cho vòng xoắn sau, hangDuoi giảm 1 đơn vị (hangDuoi--)***

+ Theo chiều xoắn, ta gán ngược n-2 value kế tiếp cho cột 0 (đâu tiên ), do a[n-1][0] và a[0][0] đã được gán rồi.
(Tổng quát, ta gọi là cột trái)
*a[hàng][cotTrai] : hàng n-2 đến 1; cotTraii =0;*

***Để chuẩn bị cho vòng xoắn sau, cotTrai tăng 1 đơn vị (cotTrai)***

***Nhận xét :***

- Kết thúc vòng xoắn đầu tiên, các value đã dùng từ 1 đến 4(n-1).
- Ma trận sử dụng vòng xoắn đầu cấp n, vòng kế tiếp cấp n-2 (giảm 2 hàng, 2 cột), vòng kê tiếp cấp n-4, . . .
- Vòng xoắn từ ngoài vào trong càng nhỏ dần (hàng trên tăng, hàng dưới giảm; cột trái tăng, cột phải giảm) , dừng khi mọi vòng xoắn đề đã gan value. (mọi vị trí của ma trận đã được gán value). Nên cách đơn giản điều khiển vòng lặp dừng là kiểm tra value đã sử dụng đến trị $n^2$.

***- Các vòng xoắn kế tiếp : . . .***

Ta có thể viết hàm tạo ma trận xoắn ốc ( sắp ma trận $n^2$ số nguyên dương đầu tiên tăng dần theo chiều xoắn ốc như sau :
Input : n
Output : ma trận xoắn ốc gồm $n^2$ số nguyên dương đầu tiên

```
void Tao_MaTran_XoanOc(int a[MAX][MAX], int n)
{
	int value, hangTren , hangDuoi , cotTrai, cotPhai ;
	int i, j;
	//Khoi tao gia tri cho hangTren, cotPhai,hangDuoi, cotTrai
	hangTren = 0;//hang dau tien
	cotPhai = n - 1;//cot cuoi cung
	hangDuoi = n - 1;//hang cuoi cung
	cotTrai = 0; //cot dau tien
	value = 1;

	while (value <= n*n)
	{
		//gan gia tri value cho hang tren
		for (j = cotTrai; (j <= cotPhai) && (value <= n*n); j++) //hang tren
		{
			a[hangTren][j] = value;
			value++;
		}
		if (value > n*n)
			break;
		hangTren++; //chuan bi cho hang ke tiep

		//gan gia tri value cho cot phai
		for (i = hangTren; (i <= hangDuoi) && (value <= n*n); i++) //Cot phai
		{
			a[i][cotPhai] = value;
```

```
                value++;
        }

        if (value > n*n)
                break;
        cotPhai--; //chuan bi cho cot ke truoc

        //Gan gia tri value cho hang duoi
        for (j = cotPhai; (j >= cotTrai) && (value <= n*n); j--) //Hang duoi
        {
                a[hangDuoi][j] = value;
                value++;
        }
        if (value > n*n)
                break;
        hangDuoi--; // chuan bi cho hang ke tren

        //Gan gia tri value cho cot trai
        for (i = hangDuoi; (i >= hangTren) && (value <= n*n); i--) //Cot trai
        {
                a[i][cotTrai] = value;
                value++;
        }
        if (value > n*n)
                break;
        cotTrai++; //chuan bi cho cot ke tiep
    }
    //Sau 1 lần xoắn gồm 4 vòng for ma tran bot di 2 hang 2 cot (value tăng theo vòng xoắn)
}
```

Nhấn Ctrl+F5 chạy chương trình, sửa lỗi nếu có.

### *4.2 Khai bao nguyên mẫu :*

```
void Tao_MaTran_XoanOc(int a[MAX][MAX], int n);
```

Bước 5:

Trong ***program.cpp***, cập nhập lại hàm ChayChuongTrinh, gọi các hàm tạo ma trận xoắn , xuất ma trận xoắn và điều khiển việc lặp thực hiện chương trình.

```
void ChayChuongTrinh()
{
    char kt;
    int a[MAX][MAX], n;
    do
    {
        system("CLS");
        cout << "\nNhap n = ";
        cin >> n;
        Tao_MaTran_XoanOc(a, n);
        system("CLS");
        cout << "\nMa tran xoan oc cap "<<n<<":";
        XuatMaTran(a, n);
        _getch();
        cout << "\n\nNua khong, nhan ESC neu khong!\n";
```

```
            kt = _getch();
        } while (kt != 27);
}
```

Nhấn Ctrl+F5 chạy chương trình, sửa lỗi nếu có.

Thực hiện chương trình với n từ 5 đến 10. Kết thúc chương trình.

***Bài 4.*** **Các phép toán ma trận**

Viết chương trình thực hiện các phép toán sau trên các ma trận:

- Tổng hai ma trận cùng cấp
- Hiệu hai ma trận cùng cấp
- Tích hai ma trận

Tạo dự án Win32 Console Application mới. Đặt tên là **Lab06_D_Bai4.**

**Chương trình tổ chức như Lab06_D_Bai1. (sinh viên tự viết).**

Tham khảo tập tin ***thuvien.h*** sau :

```
//Dinh nghia hang
#define SIZE 10
//Dinh nghia kieu du lieu moi
typedef  int  MaTranVuong[SIZE][SIZE];

//Khai bao nguyen mau cac ham:
void NhapMaTran(MaTranVuong a, int n);
void XuatMaTran(MaTranVuong a, int n);
void TinhTong_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n);
void TinhHieu_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n);
void TinhTich_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n);

//=====================================================
//Dinh nghia cac ham

void NhapMaTran(MaTranVuong a, int n,char kt)
{
        int i, j;
        for (i = 0; i < n; i++) // hang i
        for (j = 0; j < n; j++)  //cot j
        {
                cout << endl<<kt<<"[" << i << "][" << j << "]= ";
                cin >> a[i][j];
        }
}

void XuatMaTran(MaTranVuong a, int n)
{
        int i, j;
        for (i = 0; i < n; i++)
        {
                cout << endl << endl;
                for (j = 0; j < n; j++)
                        cout << setw(4) << a[i][j];
        }
}
```

```
//c[i][j] = a[i][j] + b[i][j] ; ∀i,j ∈{0,..,n-1}
void TinhTong_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n)
{
	int i, j;
	for (i = 0; i < n; i++)
	for (j = 0; j < n; j++)
		c[i][j] = a[i][j] + b[i][j];
}

//c[i][j] = a[i][j] - b[i][j] ; ∀i,j ∈{0,..,n-1}
void TinhHieu_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n)
{
	int i, j;
	for (i = 0; i < n; i++)
	for (j = 0; j < n; j++)
		c[i][j] = a[i][j] - b[i][j];
}

//c[i][j] = a[i][0] b[0][j] + a[i][1] b[1][j] +…+  a[i][n-1] b[n-1][j] ;    ∀i,j ∈{0,..,n-1}
void TinhTich_2_MaTran(MaTranVuong a, MaTranVuong b, MaTranVuong c, int n)
{
	int i, j, k;
	for (i = 0; i < n; i++)
	for (j = 0; j < n; j++)
	{
		c[i][j] = 0;
		for (k = 0; k < n; k++)
			c[i][j] += a[i][k] - b[k][j];
	}
}
```

## E. Bài tập bắt buộc

**Tất cả các bài tập phải được tổ chức dưới dạng thư viện hàm .**

**Bài 1. Âm - Dương**

Viết chương trình nhập vào một ma trận vuông cấp ***n*** với ***n*** là số nguyên dương, các phần tử của ma trận là số thực. Xuất ra màn hình ma trận đã nhập, tính và in ra màn hình các giá trị **S-T**, trong đó:

- $S = \sum_{i=1}^{n} h_i$ với $h_i$ là số dương nhỏ nhất của hàng thứ i.
- $T = \sum_{j=1}^{n} v_j$ với $v_j$ là số âm lớn nhất của cột thứ j.

**Bài 2. Kiểm tra tính chất ma trận vuông**

Viết chương trình thực hiện các thao tác trên các ma trận vuông cấp ***n***. Yêu cầu của chương trình:

- Kiểm tra ma trận có phải là ma trận đối xứng.
- Kiểm tra ma trận có phải là ma trận tam giác trên.
- Kiểm tra ma trận có phải là ma trận tam giác dưới.
- Kiểm tra ma trận có phải là ma trận chéo
- Kiểm tra ma trận có phải là ma trận đơn vị

**Bài 3. Ma trận vuông**

Viết chương trình thực hiện các thao tác trên các ma trận vuông cấp ***n***. Yêu cầu của chương trình:

- Hoán vị hai cột j và h của ma trận, xuất kết quả.
- Hoán vị hai hàng i và k của ma trận, in kết quả.
- Tìm ma trận hoán vị

**Bài 4. Lịch**

Viết chương trình in ra lịch của tháng ***M*** trong năm ***N*** cho trước theo một trong 2 mẫu trong hình dưới đây. *(xem lại Lab ,1 Lab 3 để biết cách tính thứ trong tuần và số ngày trong một tháng*.)

Ví dụ: Lịch của tháng 1 năm 2015.

| Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 | 31 | |

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |

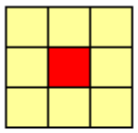
**Bài 5. Đổi đơn vị tiền tệ**

Viết chương trình cho phép nhập vào một số tiền M thuộc một trong các loại tiền tệ sau: Đô-la Mỹ, Yên Nhật, Bảng Anh, Euro, Baht Thái, VN Đồng, Nhân dân tệ (Trung Quốc), Won (Hàn Quốc), Ruble (Nga) và rupee (Ấn Độ). Sau đó, xuất ra kết quả đổi M sang các loại tiền tệ khác.
Xem bảng tỷ giá giữa các loại tiền tệ tại: http://www.xe.com/.

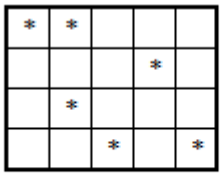
## F. Bài tập làm thêm

### 1. Bãi mìn

Xem bãi mìn như một lưới (ma trận) các ô. Hãy viết chương trình cho phép người dùng nhập vào 3 giá trị ***m***, ***n*** và ***p***, trong đó, ***m***, ***n*** là kích thước của bãi mìn, ***p*** là số quả mìn. Sau đó, đặt ngẫu nhiên ***p*** quả mìn vào các ô trong lưới. Với các ô còn lại, hãy điền vào số quả mìn ở 8 ô xung quanh nó. Xuất ra màn hình bãi mìn đã tạo, ký hiệu quả mìn là dấu *.
Ví dụ: với m = 4, n = 5, p = 6, hình dưới đây cho thấy một kết quả được tạo ra từ chương trình.



ô hiện tại và 8 ô lân cận



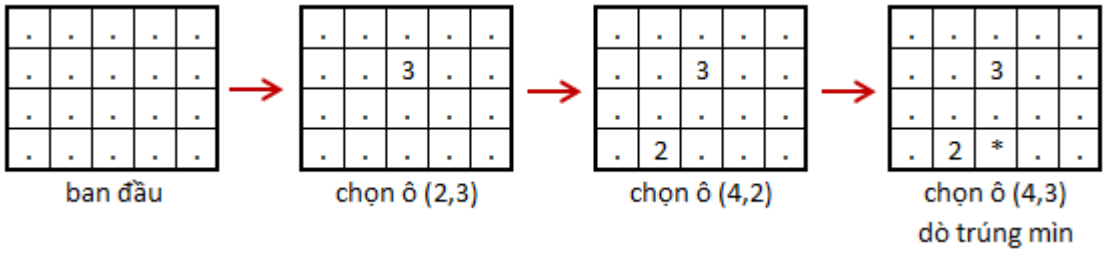
Sau khi đặt mìn

| * | * | 2 | 1 | 1 |
|---|---|---|---|---|
| 3 | 3 | 3 | * | 1 |
| 1 | * | 3 | 3 | 2 |
| 1 | 2 | * | 2 | * |

Sau khi tính số mìn xung quanh

### 2. Trò chơi dò mìn

Viết chương trình tạo ra một bãi mìn như trong bài tập 1 ở trên. Ban đầu, chương trình hiển thị bãi mìn chỉ là lưới các dấu chấm. Sau đó, chương trình cho phép người chơi nhập vị trí (hàng, cột) của các ô muốn dò mìn. Nếu người chơi mở trúng ô không có mìn thì hiển thị số lượng mìn ở các ô xung quanh tại vị trí được chọn. Chương trình dừng khi người chơi dò trúng ô có mìn hoặc tất cả các ô không chứa mìn đã được mở và thông báo kết quả.




Cải tiến chương trình trò chơi trên bằng cách cho người chơi ***k*** mạng (***k*** << ***p***). Nghĩa là người chơi được phép dò trúng ***k*** quả mìn trước khi chết (chương trình dừng).

### 3. X-O-Empty

Cho một ma trận vuông cấp ***n*** trong đó, mỗi ô chứa một trong hai giá trị X, O hoặc là ô trống. Viết chương trình tìm ra dãy liên tiếp các ô (theo chiều ngang, chiều dọc, chéo) dài nhất chứa giá trị X.

### 4. Sudoku (Số độc)

Sudoku là một loại trò chơi logic và cách chơi là điền các số từ 1 tới 9 vào những ô trống (trong một lưới 9x9 ô như hình bên trái) sao cho mỗi cột dọc, mỗi hàng ngang, mỗi vùng nhỏ 3x3 có đủ các số từ 1 tới 9 mà không được lặp lại. Hình bên phải là một cách điền (lời giải) như vậy.

Cho một lưới 9x9 ô chứa các số từ 1 tới 9. Hãy viết chương trình kiểm tra xem đó có phải là một lời giải của câu đó Sudoku hay không.

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

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

Cải tiến chương trình trên để người dùng có thể chơi Sudoku.

### 5. Ma phương lẻ

Ma phương là một ma trận vuông cấp ***n*** được tạo ra từ một dãy các số nguyên liên tiếp từ 1 tới $n^2$ trong đó, tổng các số trên mỗi hàng bằng tổng các số trên mỗi cột và bằng tổng các số trên các đường chéo. Giá trị tổng này gọi là hằng số ma phương.

| | | | |
|---|---|---|---|
| 8 | 1 | 6 | 15 |
| 3 | 5 | 7 | 15 |
| 4 | 9 | 2 | 15 |
| 15 | 15 | 15 | 15 |

| | | | | | |
|---|---|---|---|---|---|
| 17 | 24 | 1 | 8 | 15 | 65 |
| 23 | 5 | 7 | 14 | 16 | 65 |
| 4 | 6 | 13 | 20 | 22 | 65 |
| 10 | 12 | 19 | 21 | 3 | 65 |
| 11 | 18 | 25 | 2 | 9 | 65 |
| 65 | 65 | 65 | 65 | 65 | 65 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 30 | 39 | 48 | 1 | 10 | 19 | 28 | 175 |
| 38 | 47 | 7 | 9 | 18 | 27 | 29 | 175 |
| 46 | 6 | 8 | 17 | 26 | 35 | 37 | 175 |
| 5 | 14 | 16 | 25 | 34 | 36 | 45 | 175 |
| 13 | 15 | 24 | 33 | 42 | 44 | 4 | 175 |
| 21 | 23 | 32 | 41 | 43 | 3 | 12 | 175 |
| 22 | 31 | 40 | 49 | 2 | 11 | 20 | 175 |
| 175 | 175 | 175 | 175 | 175 | 175 | 175 | 175 |

Viết chương trình xuất ra ma phương bậc ***n*** với ***n*** lẻ được nhập từ bàn phím.

***Hướng dẫn cách lập ma phương lẻ:***

- Số 1 luôn viết ở ô chính giữa của dòng 1. Bắt đầu từ ô này, luôn di chuyển theo hướng Đông-Bắc để đến ô tiếp theo.
- Khi mũi tên đi ra ngoài một cạnh thì sẽ đi vào cạnh đối diện với cạnh đó (ví dụ ô 1->2, 3->4) ở cột (hàng) sau (trước).
- Khi mũi tên gặp góc trên cùng bên phải (ở đây là ô chứa số 15) thì sẽ chạy xuống ô nằm ngay bên dưới.
- Khi mũi tên gặp một ô đã có số trong đó rồi thì sẽ đi xuống ô nằm ngay bên dưới (5->6, 20->21). Cứ như thế cho đến khi đi hết các ô trong ma phương.