```
/*
Họ và tên     : Nguyễn Văn Tèo
Lớp           : 12DHTHxy
Buổi/Tiết     : ST6/12345
*/

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

#define VC 99
#define MAXE 100
#define MAXV 11
#define TRUE 1
#define FALSE 0

//=====================================================================
struct Graph
{
    int flag;  //0: ĐT vô hướng, 1: ĐT có hướng, -1: ĐT không hợp lệ
    int n;     //Số đỉnh
    int w[MAXV][MAXV];
};
//=====================================================================
void initGraph(Graph &g)
{
    g.flag = -1;
    g.n = 0;
    for (int i = 0; i <= MAXV; i++)
    {
        for (int j = 0; j <= MAXV; j++)
            g.w[i][j] = VC;
    }
}
//=====================================================================
void showGraph(Graph g)
{
    if (g.flag == -1)
    {
        printf("\nKhong phai do thi.");
        getch();
        return;
    }
    printf("\nSo dinh: %d", g.n);
    if (g.flag == 0)
        printf("\nDo thi vo huong");
    else if (g.flag == 1)
        printf("\nDo thi co huong");
    printf("\nDANH SACH CANH:\n");

    printf("     ");
    for (int i = 1; i <= g.n; i++)
```

```
        printf("%5d", i);
    printf("\n");

    for (int i = 1; i <= g.n; i++)
    {
        printf("%5d", i);
        for (int j = 1; j <= g.n; j++)
        {
            printf("%5d", g.w[i][j]);
        }
        printf("\n"); //Xuống dòng
    }
}
//=============================================================
bool read_Adjacent_Matrix_Data(char fileName[], Graph &g)
{ //Đọc Ma trận kề Hoặc Ma trận trọng số
    initGraph(g);
    FILE *fi = fopen(fileName, "rt"); //Read File input - fi
    if (fi == NULL)
    {
        printf("Khong the mo duoc file!");
        getch();
        return false;
    }
    fscanf(fi, "%d\n", &g.flag); //Đọc loại Đồ thị
    fscanf(fi, "%d\n", &g.n); //Đọc số đỉnh
    for (int i = 1; i <= g.n; i++)
    {
        for (int j = 1; j <= g.n; j++)
            fscanf(fi, "%d", &g.w[i][j]); //Đọc số cạnh
    }
    fclose(fi);
    return true;
}
//=============================================================
void input_Start_End(Graph g, int &start, int &end)
{
    int a, b;
    a = b = 0;
    printf("Cac dinh danh so tu 1 den %d.\n", g.n);
    do
    {
        printf("Nhap dinh bat dau: ");
        scanf("%d", &a);
        if (a < 1 || a > g.n)
            printf("Khong hop le! \nNhap lai dinh bat dau: ");
    } while (a < 1 || a > g.n);

    do
    {
        printf("Nhap dinh ket thuc: ");
        scanf("%d", &b);
        if (b < 1 || b > g.n)
            printf("Khong hop le! \nNhap lai dinh ket thuc: ");
```

```
    } while (b < 1 || b > g.n);
    start = a;
    end = b;
}
//=====================================================================
int Floyd(Graph g, int P[][MAXV], int start, int end)
{
    int a = start, b = end;
    int A[MAXV][MAXV];

    for (int i = 1; i <= g.n; i++)
    {
        for (int j = 1; j <= g.n; j++)
        {
            if (g.w[i][j])
                A[i][j] = g.w[i][j];
            else
                A[i][j] = VC;
            P[i][j] = -1;
        }
    }

    for (int k = 1; k <= g.n; k++)
    {
        for (int i = 1; i <= g.n; i++)
        {
            for (int j = 1; j <= g.n; j++)
            {
                if (A[i][j] > A[i][k] + A[k][j])
                {
                    A[i][j] = A[i][k] + A[k][j];
                    P[i][j] = k;
                }
            }
        }
    }
    return A[a][b];
}
void truyVet(Graph g, int P[][MAXV], int start, int end)
{
    //truy vet
    stack <int> S1;
    stack <int> S2;
    S1.push(start); //danh sach nap cac dinh vao
    S1.push(end); //danh sach xuat cac dinh ra
    int dich, tg;
    while (!S1.empty())
    {
        dich = S1.top();  //dich = phan tu dau tien
        S1.pop();         // dua phan tu do ra
        S2.push(dich);    //cho vao danh sach xuat
        if (!S1.empty())  //trong khi S1 ko rong thi tiep tuc tim cac dinh
        {
            tg = S1.top();
```

```
            while (P[tg][dich] != -1) //tim cac dinh di tu tg den dich
            {
                S1.push(P[tg][dich]);
                tg = S1.top();
            }
        }
    }

    printf("%d", S2.top());
    S2.pop();

    while (!S2.empty())
    {
        printf(" --> %d ", S2.top());
        S2.pop();
    }
}
//=====================================================================
void main()
{
    Graph G1, G2;
    int P[MAXV][MAXV];
    int start, end;

    //Ma trận kề (Ma trận trọng số)
    read_Adjacent_Matrix_Data("input_mtk_01.txt", G1);
    showGraph(G1);

    input_Start_End(G1, start, end);
    int len = Floyd(G1, P, start, end);
    // in ket qua
    printf("\nDo dai ngan nhat cua duong di tu %d den %d la: %d \n", start, end, len);
    printf("Qua trinh duong di: ");
    truyVet(G1, P, start, end);

    getch();
}
```