```cpp
#include <stdio.h>
#include <stdlib.h>

enum Status
{
    OVERFLOW = -1,
    ERROR    =  0,
    OK       =  1
};

////////////////////////////////////////
//图的邻接表存储表示
#define MAX_VERTEXT_NUM 20
#define INFINITY        4294967296
enum GraphKind{DG, DN, UDG, UDN};
typedef int  VertexType;

typedef struct ArcNode
{
    VertexType  adjVex;
    ArcNode    *nextArc;
    int         weight;
} ArcNode;

typedef struct VNode
{
    VertexType  data;
    ArcNode    *firstArc;
}VNode, AdjList[MAX_VERTEXT_NUM + 1];

typedef struct
{
    AdjList   vertices;
    int       vexNum;
    int       arcNum;
    GraphKind kind;
} ALGraph;

////////////////////////////////////////

////////////////////////////////////////
//函数声明
Status GreateGraph(ALGraph& G);
Status GreateUDN(ALGraph& G);
void TraverseGraph(ALGraph& G, bool* visited, VertexType vex);
void DFSSearch(ALGraph& G, bool* visited, VertexType vex);
void PrintVexData(VNode* vnode);
Status DestroyGraph(ALGraph& G);

////////////////////////////////////////
//主函数
int main()
{
```

```cpp
    ALGraph G;
    bool visited[MAX_VERTEXT_NUM];
    bool w = false;         //检测图是否被建立
    char chos;

    while(1)
    {
        printf("1.Great a graph|network\n2.DFSTraverse\n3.Exit\n");
        printf("Choose what you want to do: ");
        scanf_s("%c", &chos);
        while(chos == 10)
        {
            scanf_s("%c", &chos);
        }

        switch(chos)
        {
        case '1':
            {
                if (w == true)
                {
                    printf("You have created one. You will creat another.\n");
                    DestroyGraph(G);
                }

                GreateGraph(G);

                w = true;
            }break;
        case '2':
            {
                if (w == false)
                {
                    printf("You have to create a graph or network first!\n");
                    break;
                }

                TraverseGraph(G, visited, 1);
            }break;
        case '3':
            {
                DestroyGraph(G);

                exit(0);
            }break;
        default:
            {
                printf("CHOS ERROR!\n");
            }break;
        }

        putchar(10);
    }
```

```
107
108     system("pause");
109 }
110
111 ////////////////////////////////////
112 //函数定义
113 Status GreateGraph(ALGraph& G)
114 {
115     char chos;
116     GT:
117     printf("1.DiGraph  2.UdiGraph  3.DiNet  4.UdiNet\n");
118     printf("Please choose what kind of graph you want to create: \n");
119     scanf_s("%c", &chos);
120     while (chos == 10)
121     {
122         scanf_s("%c", &chos);
123     }
124
125     switch (chos)
126     {
127         case '1': printf("NOT IMPLEMENT\n"); putchar(10); goto GT; break;
128         case '2': printf("NOT IMPLEMENT\n"); putchar(10); goto GT; break;
129         case '3': printf("NOT IMPLEMENT\n"); putchar(10); goto GT; break;
130         case '4': GreateUDN(G); break;
131         default : printf("CHOS ERROR!\n"); putchar(10); goto GT;
132     }
133
134     return OK;
135 }
136
137 Status GreateUDN  (ALGraph& G)
138 {
139     VertexType v1;
140     VertexType v2;
141     int        weight;
142
143
144     printf("You will create a undirected network soon.\n");
145     printf("Please input the information of his network.\n");
146     printf("vexNum: "); scanf_s("%d", &G.vexNum);
147     printf("arcNum: "); scanf_s("%d", &G.arcNum);
148
149     for (int i=1; i<=G.vexNum; ++i)     //初始化表头结点向量
150     {
151         G.vertices[i].data = i;
152         G.vertices[i].firstArc = NULL;
153     }
154
155     for (int i=1; i<=G.arcNum; ++i)     //构造邻接表
156     {
157         printf("Please input the information of No.%d edge & its weight.\n",
              i);
158         printf("vex1: ");   scanf_s("%d", &v1);
```

```
159         printf("vex2: ");   scanf_s("%d", &v2);
160         printf("weight: "); scanf_s("%d", &weight);
161 
162         ArcNode *arcNode1 = (ArcNode*) malloc(sizeof(ArcNode));
163         ArcNode *arcNode2 = (ArcNode*) malloc(sizeof(ArcNode));
164         if ((arcNode1 == NULL) || (arcNode2 == NULL))
165         {
166             return OVERFLOW;
167         }
168 
169         arcNode1->nextArc = G.vertices[v1].firstArc;
170         arcNode1->adjVex  = v2;
171         arcNode1->weight  = weight;
172         G.vertices[v1].firstArc = arcNode1;
173 
174         arcNode2->nextArc = G.vertices[v2].firstArc;
175         arcNode2->adjVex  = v1;
176         arcNode2->weight  = weight;
177         G.vertices[v2].firstArc = arcNode2;
178     }
179 
180     printf("Congratulations! The undirected network has been created!\n");
181     return OK;
182 }
183 
184 Status DestroyGraph(ALGraph& G)
185 {
186     ArcNode *p1;
187     ArcNode *p2;
188 
189     for (int i=1; i<=G.vexNum; ++i)
190     {
191         if (G.vertices[i].firstArc == NULL)
192         {
193             continue;
194         }
195 
196         p1 = G.vertices[i].firstArc;
197         p2 = p1->nextArc;
198 
199         while(p2 != NULL)
200         {
201             free(p1);
202             p1 = p2;
203             p2 = p2->nextArc;
204         }
205         free(p1);
206     }
207 
208     printf("The graph has been destroyed.\n");
209     return OK;
210 }
211 
```

```cpp
void TraverseGraph(ALGraph& G, bool* visited, VertexType vex)
{
    printf("The DFS sequence of this graph is:\n");
    for (int i=1; i<=G.vexNum; ++i)
    {
        *(visited + i) = false;
    }

    DFSSearch(G, visited, vex);
    putchar(10);

    for (int i=1; i<=G.vexNum; ++i)
    {
        if (visited[i] == false)
        {
            DFSSearch(G, visited, G.vertices[i].data);
            putchar(10);
        }
    }

    putchar(10);
}

void DFSSearch(ALGraph& G, bool* visited, VertexType vex)
{
    PrintVexData(&G.vertices[vex]);
    visited[vex] = true;

    ArcNode* p;
    p = G.vertices[vex].firstArc;

    while(p != NULL)
    {
        if (visited[p->adjVex] == false)
        {
            printf(" ->(%d) ", p->weight);
            DFSSearch(G, visited, p->adjVex);

            //该路径走到尽头，换行输出下一路径
            printf("*(end:v%d)\n    ", p->adjVex);
        }
        p = p->nextArc;
    }
}

void PrintVexData(VNode* vnode)
{
    printf("v%d ", vnode->data);
}
```