```cpp
#include"MGraph.h"

//辅助数组结构
typedef struct{
    VertexType adjvex;
    VRType lowcost;
}Closedge;

//定义辅助数组的全局变量
Closedge closedge[MAX_VERTEX_NUM];

//求出当前顶点集中权值最小的边
int minimum(int num){
    int i;
    VRType minp=0;
    for(i=0; i<num; i++)
        if(closedge[i].lowcost != 0){
            minp=i;
            break;
        }

    for(i=0; i<num; i++)
        if(closedge[i].lowcost<closedge[minp].lowcost && closedge[i].lowcost!=0)
            minp = i;

    return minp;
}

//用prim算法构造输出G的最小生成树
void MiniSpanTree_PRIM(MGraph G, VertexType u){
    int k=LocateVex(G,u);

    //初始化辅助数组
    for(int i=0; i<G.vexnum; i++)
        if(i!=k)
            closedge[i] = {u, G.arcs[k][i].adj};    //{adjvex, lowcost}
    closedge[k].lowcost = 0;          //初始U = {u}

    for(int i=1; i<G.vexnum; i++){       //选择其余G.vexnum-1个顶点
        k=minimum(G.vexnum);
        //输出生成树的边
        cout << closedge[k].adjvex << "__" << G.vexs[k] << "\t" ;

        closedge[k].lowcost = 0;          //第k顶点并入U集
        for(int j=0; j<G.vexnum; j++)
            if(G.arcs[k][j].adj < closedge[j].lowcost)  //新顶点并入U后重新选择最小边
                closedge[j] = {G.vexs[k], G.arcs[k][j].adj};
    }
    cout << endl;
}

int main(){
    MGraph G;
    CreateGraph(G);
    PrintAdjMatrix(G);
    MiniSpanTree_PRIM(G, 1);
    return 0;
}
```