```
#define MaxVertexNum 1000
typedef int WeightType;
typedef int DataType;
typedef int Vertex;  /* 用顶点下标表示顶点，为整型 */
typedef struct GNode *PtrToGNode;
struct GNode
{
    int Nv;  /* 边数 */
    int Ne;  /* 顶点数 */
    WeightType G[MaxVertexNum];
    DataType Data[MaxVertexNum];  /* 存顶点的数据 */
};
typedef PtrToGNode MGraph;  /* 以邻接矩阵存储的图类型 */

typedef struct ENode *PtrToENode;
struct ENode
{
    Vertex V1,V2;  /* 有向边<V1,V2> */
    WeightType Weight;  /* 权重 */
};
typedef PtrToENode Edge;


MGraph CreateGraph(int VertexNum)
{
    Vertex V,W;
    MGraph Graph;

    Graph=(MGraph)malloc(sizeof(struct GNode));
    Graph->Nv=VertexNum;
    Graph->Ne=0;

    /* 注意：这里默认顶点编号从零开始，到(Graph->Nv-1) */
    for(V=0;V<Graph->Nv;V++)
    {
        for(W=0;W<Graph->Nv;W++)
        {
            Graph->G[V][W]=0;  /* or infinity */
        }
    }
    return Graph;
}

void InsertEdge(MGraph Graph,Edge E)
{
    /* 插入边<V1,V2> */
    Graph->G[E->V1][E->V2]=E->Weight;
    /* 若为无向图，还要插入边<V1,V2> */
    Graph->G[E->V2][E->V1]=E->Weight;
}

MGraph BuildGraph()
{
    MGraph Graph;
    Edge E;
    Vertex V;
    int Nv,i;

    scanf("%d",&Nv);
    Graph=CreateGraph(Nv);
    scanf("%d",&(Graph->Ne));
    if(Graph->Ne!=0)
    {
        E=(Edge)malloc(sizeof(struct ENode));
        for(i=0;i<Graph->Ne;i++)
        {
            scanf("%d %d %d",&E->V1,&E->V2,&E->Weight);
            InsertEdge(Graph,E);
        }
    }
    /* 如果顶点有数据，读入数据 */
    for(V=0;V<Graph->Nv;V++)
    {
        scanf("%d",&(Graph->Data[V]));
    }
    return Graph;
}
```

如果不要这么麻烦

```c
//以下为简化版本
int G[MAXN][MAXN],Nv,Ne;
void BuildGraph()
{
    int i,j,v1,v2,w;

    scanf("%d",&Nv);
    /* CreateGraph */
    for(i=0;i<Nv;i++)
    {
        for(j=0;j<Nv;j++)
        {
            G[i][j]=0;  /* or infinity */
        }
    }
    scanf("%d",&Ne);
    for(i=0;i<Ne;i++)
    {
        scanf("%d %d %d",&v1,&v2,&w);
        /* InsertEdge */
        G[v1][v2]=w;
        G[v2][v1]=w;
    }
}
```

用邻接表表示图

```c
#define MaxVertexNum 1000
#define MAXN 1000
typedef int WeightType;
typedef int DataType;
typedef int Vertex;  /* 用顶点下标表示顶点，为整型 */

typedef struct ENode *PtrToENode;
struct ENode
{
    Vertex V1,V2;  /* 有向边<V1,V2> */
    WeightType Weight;  /* 权重 */
};
typedef PtrToENode Edge;

typedef struct AdjVNode *PtrToAdjVNode;
struct AdjVNode
{
    Vertex AdjV;  /* 邻接点下标 */
    WeightType Weight;  /* 边权重 */
    PtrToAdjVNode Next;
};

typedef struct VNode
{
    PtrToAdjVNode FirstEdge;
    DataType Data;  /* 存顶点数据 */
}AdjList[MaxVertexNum];
/* AdjList是邻接表类型 */

typedef struct GNode *PtrToGNode;
struct GNode
{
    int Nv;  /* 顶点数 */
    int Ne;  /* 边数 */
    AdjList G;  /* 邻接表 */
};
typedef PtrToGNode LGraph;
/* 以邻接表方式存储图类型 */

LGraph CreateGraph(int VertexNum)
{
    Vertex V,W;
    LGraph Graph;

    Graph=(LGraph)malloc(sizeof(struct GNode));
    Graph->Nv=VertexNum;
    Graph->Ne=0;

    for(V=0;V<Graph->Nv;V++)
    {
        Graph->G[V].FirstEdge=NULL;
    }

    return Graph;
}

void InsertEdge(LGraph Graph,Edge E)
{
    PtrToAdjVNode NewNode;
    /* 插入边V1,V2 */
    /* 为V2建立新的邻接点 */
    NewNode=(PtrToAdjVNode)malloc(sizeof(struct AdjVNode));
    NewNode->AdjV=E->V2;
    NewNode->Weight=E->Weight;
    /* 将V2插入V1的表头 */
    NewNode->Next=Graph->G[E->V1].FirstEdge;
    Graph->G[E->V1].FirstEdge=NewNode;
    /* 若是无向图，还要插入边V2,V1 */
    /* 为V1建立新的邻接点 */
    NewNode=(PtrToAdjVNode)malloc(sizeof(struct AdjVNode));
    NewNode->AdjV=E->V1;
    NewNode->Weight=E->Weight;
    /* 将V1插入V2的表头 */
    NewNode->Next=Graph->G[E->V2].FirstEdge;
    Graph->G[E->V2].FirstEdge=NewNode;
}
```