```
/* 图的邻接矩阵表示法 */

#define MaxVertexNum 100    /* 最大顶点数设为100 */
#define INFINITY 65535        /* ∞设为双字节无符号整数的最大值65535*/
typedef int Vertex;          /* 用顶点下标表示顶点,为整型 */
typedef int WeightType;         /* 边的权值设为整型 */
typedef char DataType;         /* 顶点存储的数据类型设为字符型 */

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

/* 图结点的定义 */
typedef struct GNode *PtrToGNode;
struct GNode{
    int Nv;  /* 顶点数 */
    int Ne;  /* 边数   */
    WeightType G[MaxVertexNum][MaxVertexNum]; /* 邻接矩阵 */
    DataType Data[MaxVertexNum];      /* 存顶点的数据 */
    /* 注意：很多情况下，顶点无数据，此时Data[]可以不用出现 */
};
typedef PtrToGNode MGraph; /* 以邻接矩阵存储的图类型 */



MGraph CreateGraph( int VertexNum )
{ /* 初始化一个有VertexNum个顶点但没有边的图 */
    Vertex V, W;
    MGraph Graph;

    Graph = (MGraph)malloc(sizeof(struct GNode)); /* 建立图 */
    Graph->Nv = VertexNum;
    Graph->Ne = 0;
    /* 初始化邻接矩阵 */
    /* 注意：这里默认顶点编号从0开始，到(Graph->Nv - 1) */
    for (V=0; V<Graph->Nv; V++)
        for (W=0; W<Graph->Nv; W++)
            Graph->G[V][W] = INFINITY;

    return Graph;
}

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

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

    scanf("%d", &Nv);   /* 读入顶点个数 */
    Graph = CreateGraph(Nv); /* 初始化有Nv个顶点但没有边的图 */

    scanf("%d", &(Graph->Ne));   /* 读入边数 */
    if ( Graph->Ne != 0 ) { /* 如果有边 */
        E = (Edge)malloc(sizeof(struct ENode)); /* 建立边结点 */
        /* 读入边，格式为"起点 终点 权重"，插入邻接矩阵 */
        for (i=0; i<Graph->Ne; i++) {
            scanf("%d %d %d", &E->V1, &E->V2, &E->Weight);
            /* 注意：如果权重不是整型，Weight的读入格式要改 */
            InsertEdge( Graph, E );
        }
    }

    /* 如果顶点有数据的话，读入数据 */
    for (V=0; V<Graph->Nv; V++)
        scanf(" %c", &(Graph->Data[V]));

    return Graph;
}
```