```
/* 图的邻接表表示法 */

#define MaxVertexNum 100    /* 最大顶点数设为100 */
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;

/* 邻接点的定义 */
typedef struct AdjVNode *PtrToAdjVNode;
struct AdjVNode{
    Vertex AdjV;        /* 邻接点下标 */
    WeightType Weight;  /* 边权重 */
    PtrToAdjVNode Next;    /* 指向下一个邻接点的指针 */
};

/* 顶点表头结点的定义 */
typedef struct Vnode{
    PtrToAdjVNode FirstEdge;/* 边表头指针 */
    DataType Data;            /* 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 */
} AdjList[MaxVertexNum];    /* AdjList是邻接表类型 */

/* 图结点的定义 */
typedef struct GNode *PtrToGNode;
struct GNode{
    int Nv;     /* 顶点数 */
    int Ne;     /* 边数   */
    AdjList G;  /* 邻接表 */
};
typedef PtrToGNode LGraph; /* 以邻接表方式存储的图类型 */



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

    Graph = (LGraph)malloc( sizeof(struct GNode) ); /* 建立图 */
    Graph->Nv = VertexNum;
    Graph->Ne = 0;
    /* 初始化邻接表头指针 */
    /* 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++)
        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;
}

LGraph BuildGraph()
```

```
{
    LGraph 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->G[V].Data));

    return Graph;
}
```