**图搜索BFS算法及存储优化实验报告**

1. **实验目的**

通过实践，加深对邻接多重表和BFS算法的理解，提高编程能力。

1. **实验内容**

1.从data.txt文件获取无向图中顶点的和边的数据。

2.设计邻接多重表的数据结构和相应的函数，

3.用第2步中设计的邻接多重表来存储一个无向图。

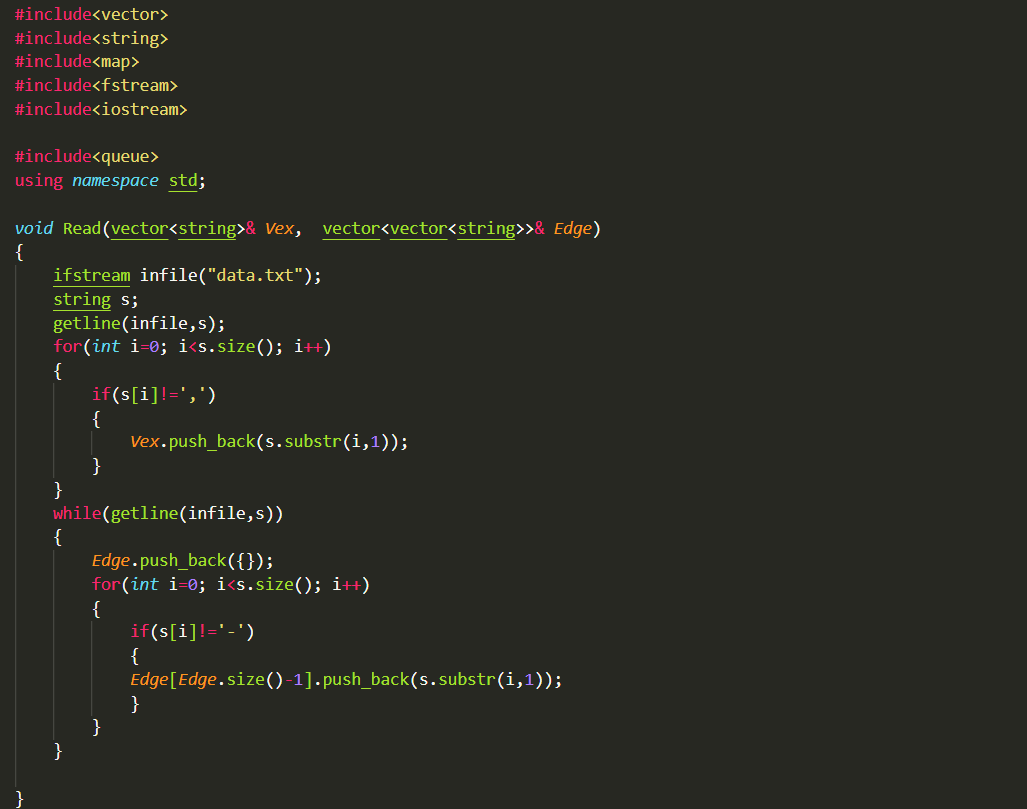
4.用BFS算法遍历第3步中存储的无向图。

1. **实验步骤**

**1.读取数据**

读取数据方法与前几次实验类似，通过getline函数对.txt文件按行读取，得到无向图的顶点和边，将顶点存放在一个一维数组中，将所有的边存放在一个二维数组中。但这次实验过程中我将IO相关的函数单独放在IO.cpp这个源文件中，并让另一个源文件来包含它，使得程序结构更加清晰，也提高了代码复用性。

IO相关的部分源程序如下所示：



**2.设计邻接多重表的数据结构**

邻接多重表由顶点结点和边结点构成。

其中，边结点中包括以下几个域：visited域、ivex域、jvex域、iLink域和jLink域，visited域用于在后续遍历图时标记这条边是否已被访问，ivex域和jvex域分别表示与这条边相连的两个顶点，iLink域指向与ivex顶点相连的下一条边，jLink域指向与jvex顶点相连的下一条边。

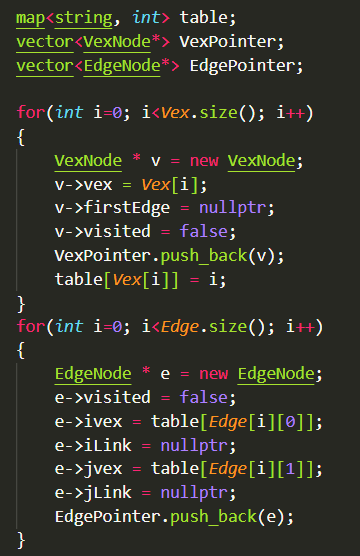
顶点结点包括以下几个域：visited域、vex域和firstEdge域，顶点结点中的visited域与边结点中的visited域作用相同，vex域表示本顶点对应的字符，firstEdge域指向与本顶点相连的第一条边。

顶点结点和边结点的数据结构如下所示：



**3.存储无向图**

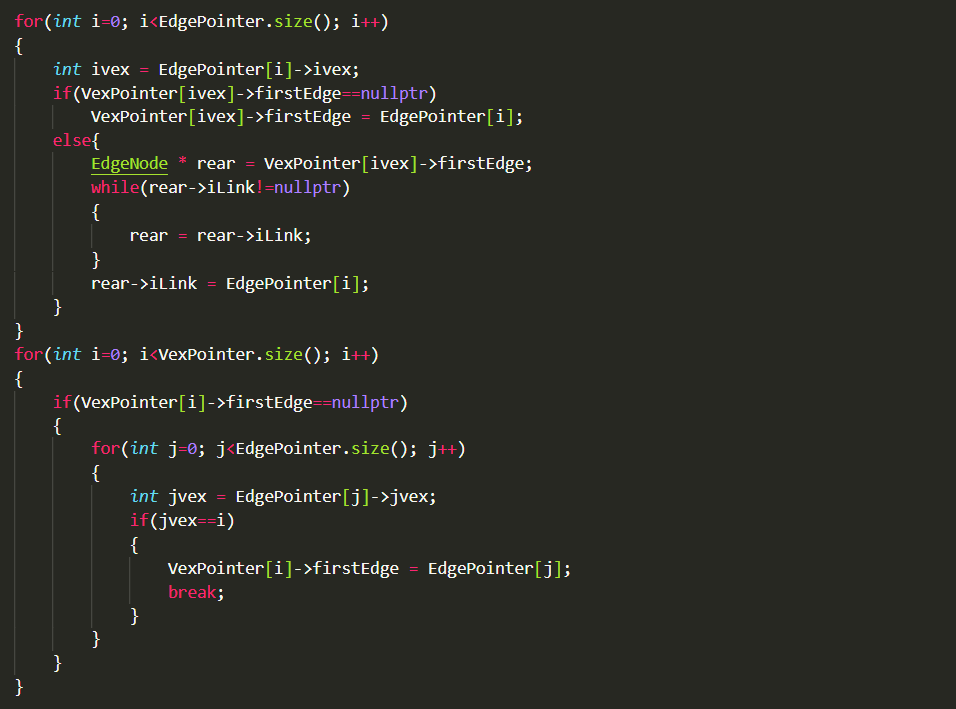
首先根据从.txt文件中读取的顶点和边的信息生成相应的顶点结点和边结点，并各自做相应的初始化，然后分别将指向结点的指针存储在VexPointer(一维数组)和EdgePointer(二维数组)中，如下所示:



接下来要做的就是将每条边结点“加入”到图中，遍历EdgePointer数组，对遍历到的每条边，尝试将其链接到ivex顶点和jvex顶点所对应的边结点链表中(ivex顶点相连的边都会通过iLink域链接到一条链表中，jevex顶点类似)，同时还要更新VexPointer数组中存储的各个顶点的firstEdge域，即如果是第一次往顶点所对应的链表(此时为空链表)中加入一个边结点，此时更新顶点的firstEdge域。另外，还会存在某个连通分量中只有单个节点的情况，此时这个顶点的firstEdge域为空。

在编码过程中，我是分三步实现上述过程的：先更新边结点的iLink域和部分顶点结点的firstEdge域，再更新剩余顶点结点的firstEdge域，然后更新边结点的jLink域，再更新jLink域时先将与jvex=0相连的边结点链到jvex对应的链表中，再将与jvex=1相连的边结点链到jvex对应的链表中……

部分代码如下所示：



**4.采用BFS遍历无向图**

遍历VexPointer数组，如果遇到一个未被访问过的顶点q，将其加入到访问队列中，然后进入while循环，直到包含顶点q的连通分量中的所有顶点都被遍历，退出while循环，继续遍历VexPointer数组，直到无向图中所有的顶点都被访问完毕。在while循环中，如果访问队列不为空，取出队首节点p，访问它，遍历与这个顶点p相连的边结点，如果某个边结点未被访问，并且这个边结点的jvex域对应的顶点也未被访问，那么将这个顶点放到队列中，将边结点的visited域置为true，直到与p相连的边结点都被访问完，进入下一次while循环…



1. **实验结果**

采用BFS遍历上述步骤中存储的无向图得到下面的结果：



1. **实验心得**

最后一次实验报告，想写一些与代码无关的实验心得：这次代码验收过程中，,在给班桑复述我的思路时，脑子一片混乱。其实之前再复述的过程中，我也出现过类似的情况，只是没有这次这么明显。8次实验我一直忽略了这个问题：很多时候，代码我可以写出来，但是当需要我向别人复述我的思路时，我又描述得很混乱。这其实是个很大的问题，希望我以后在写完代码以后，都能够做到自己复盘一下，自己给自己描述，2022，希望我可以变得越来越好。