数据结构实验六_图基本操作的编程实现（C语言）

原创

老头儿ii 于 2019-04-24 14:10:07 发布 6480 收藏 37

分类专栏： 数据结构 文章标签： C语言 数据结构 图

本文链接：https://blog.csdn.net/qq_40833790/article/details/89491648

一、实验题目

图基本操作的编程实现

二、题目要求

图基本操作的编程实现，掌握图的建立、遍历、插入、删除等基本操作的编程实现，存储结构可以在顺序结构、链接结构、联合使用多种结构等中任选，也可以全部实现。也鼓励学生利用基本操作进行一些应用的程序设计。

三、运行结果

```
==================================
图的基本功能实验(存储结构邻接矩阵)
==================================
1.手工建立（有向）图
2.默认数据建立图
3.用邻接矩阵形式显示图
4.求图的结点数和边数
5.求某条边的权值
6.添加结点
7.删除结点
8.添加边
9.删除边
a.深度优先遍历图
s.广度优先遍历图
0.退出
==================================
请输入选项：2
================
调入默认数据成功!!!
当前图的坐标和结点如下:
    0    1    2    3    4    5    6
    a    b    c    d    e    f    g
当前图的邻接矩阵如下:
    0    1    1   ∞   ∞   ∞   ∞
    1    0   ∞    1   ∞    1   ∞
    1   ∞    0   ∞   ∞   ∞    1
   ∞    1   ∞    0   ∞    1   ∞
   ∞   ∞   ∞   ∞    0    1   ∞
   ∞    1   ∞    1    1    0   ∞
   ∞   ∞    1   ∞   ∞   ∞    0
================
请按任意键继续. . .
```



```
==================================
图的基本功能实验(存储结构邻接矩阵)
```

```
图的基本功能实验(存储结构邻接矩阵)
==================================
1.手工建立（有向）图
2.默认数据建立图
3.用邻接矩阵形式显示图
4.求图的结点数和边数
5.求某条边的权值
6.添加结点
7.删除结点
8.添加边
9.删除边
a.深度优先遍历图
s.广度优先遍历图
0.退出
==================================
请输入选项：
3
================
当前图的坐标和结点如下：
    0    1    2    3    4    5    6
    a    b    c    d    e    f    g
当前图的邻接矩阵如下：
    0    1    1   ∞   ∞   ∞   ∞
    1    0   ∞    1   ∞    1   ∞
    1   ∞    0   ∞   ∞   ∞    1
   ∞    1   ∞    0   ∞    1   ∞
   ∞   ∞   ∞   ∞    0    1   ∞
   ∞    1   ∞    1    1    0   ∞
   ∞   ∞    1   ∞   ∞   ∞    0
================
请按任意键继续. . .
```



```
==================================
图的基本功能实验(存储结构邻接矩阵)
==================================
1.手工建立（有向）图
2.默认数据建立图
3.用邻接矩阵形式显示图
4.求图的结点数和边数
5.求某条边的权值
6.添加结点
7.删除结点
8.添加边
9.删除边
a.深度优先遍历图
s.广度优先遍历图
0.退出
==================================
请输入选项：4
================
当前图的坐标和结点如下：
    0    1    2    3    4    5    6
    a    b    c    d    e    f    g
当前图的邻接矩阵如下：
    0    1    1   ∞   ∞   ∞   ∞
    1    0   ∞    1   ∞    1   ∞
    1   ∞    0   ∞   ∞   ∞    1
   ∞    1   ∞    0   ∞    1   ∞
   ∞   ∞   ∞   ∞    0    1   ∞
   ∞    1   ∞    1    1    0   ∞
   ∞   ∞    1   ∞   ∞   ∞    0
当前结点个数为 :7
当前图的边数为 :14
================
请按任意键继续. . .
```


四、程序基本功能

1、深度优先搜索遍历

函数名：`breadthfirstsearch(const int startpoint,int visited[],void visit(char item))`

描述：深度优先搜索遍历它类似于树的先根遍历，是树的先根遍历的推广

入口参数：`const int startpoint int visited[] void visit(char item)`

入口参数：`const int startpoint,int visited[],void visit(char item)`

出口参数：无

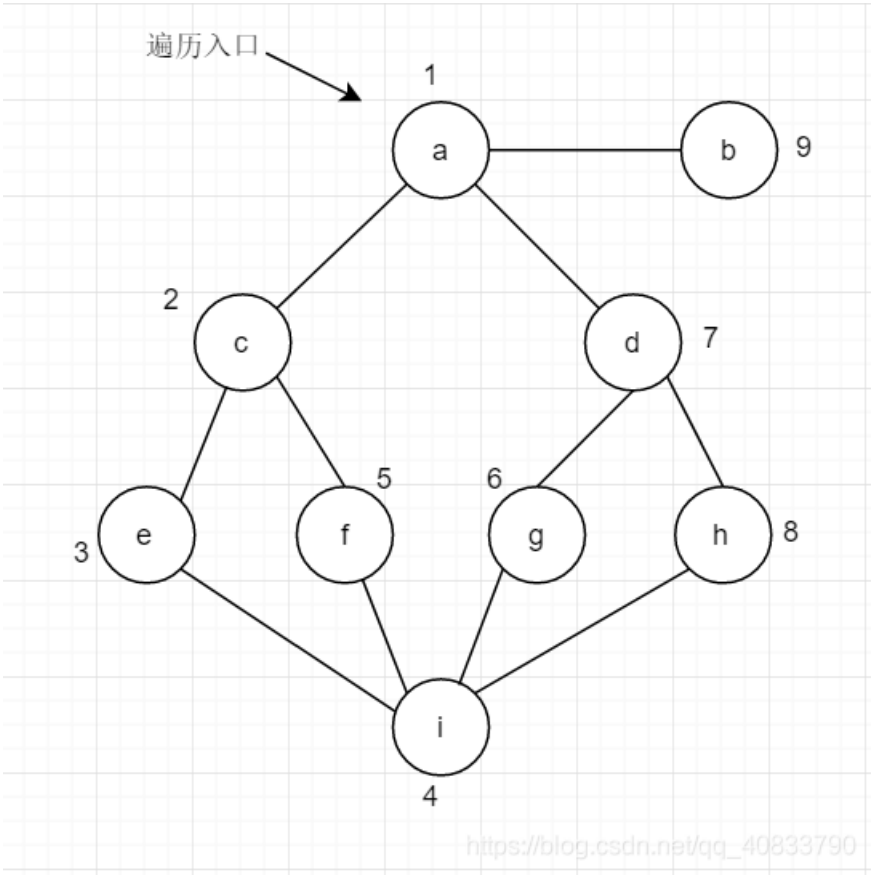
说明：假设初始状态是图中所有顶点未曾被访问，则深度优先搜索可从图中某个顶点v出发。访问次顶点，然后依次从v的未被访问的邻接点出发深度优先遍历图，直至图中所有和v有路径相通的顶点都被访问到；若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作为起始点，重复上述过程，直至图中所有丁点都被访问到为止。

（1）从某个点出发，访问之，并且把他的访问标志从0变成1，表示已经访问过此结点。

（2）从该结点选择第一个对面结点没有被访问的路径，到达该结点，访问之，把他的访问标志从0变成1，并且重复（2），直到该结点通路涉及的所有结点标志已经翻转为1，就进入步骤（3）。

（3）从刚才的结点开始回溯。退回到进入该结点经过的上一个结点，继续步骤（2）。

（4）回溯的过程一直要到进入结点，此时如果通路对面涉及的所有结点标志位都已经被翻转为1时，算法结束。




```
==================================
图的基本功能实验(存储结构邻接矩阵)
==================================
1.手工建立（有向）图
2.默认数据建立图
3.用邻接矩阵形式显示图
4.求图的结点数和边数
5.求某条边的权值
6.添加结点
7.删除结点
8.添加边
9.删除边
a.深度优先遍历图
s.广度优先遍历图
0.退出
==================================
请输入选项：a
================
当前图的坐标和结点如下:
    0    1    2    3    4    5    6
    a    b    c    d    e    f    g
当前图的邻接矩阵如下:
    0    1    1   ∞   ∞   ∞   ∞
    1    0   ∞    1   ∞    1   ∞
    1   ∞    0   ∞   ∞   ∞    1
   ∞    1   ∞    0   ∞    1   ∞
   ∞   ∞   ∞   ∞    0    1   ∞
```

```
   ∞     1    ∞     1     1     0    ∞
   ∞     ∞     1    ∞     ∞     ∞     0
此处约定从第一个结点开始遍历。
深度优先遍历序列为 :
     a     b     d     f     e     c     g
================
请按任意键继续. . .
```


2、广度优先搜索遍历

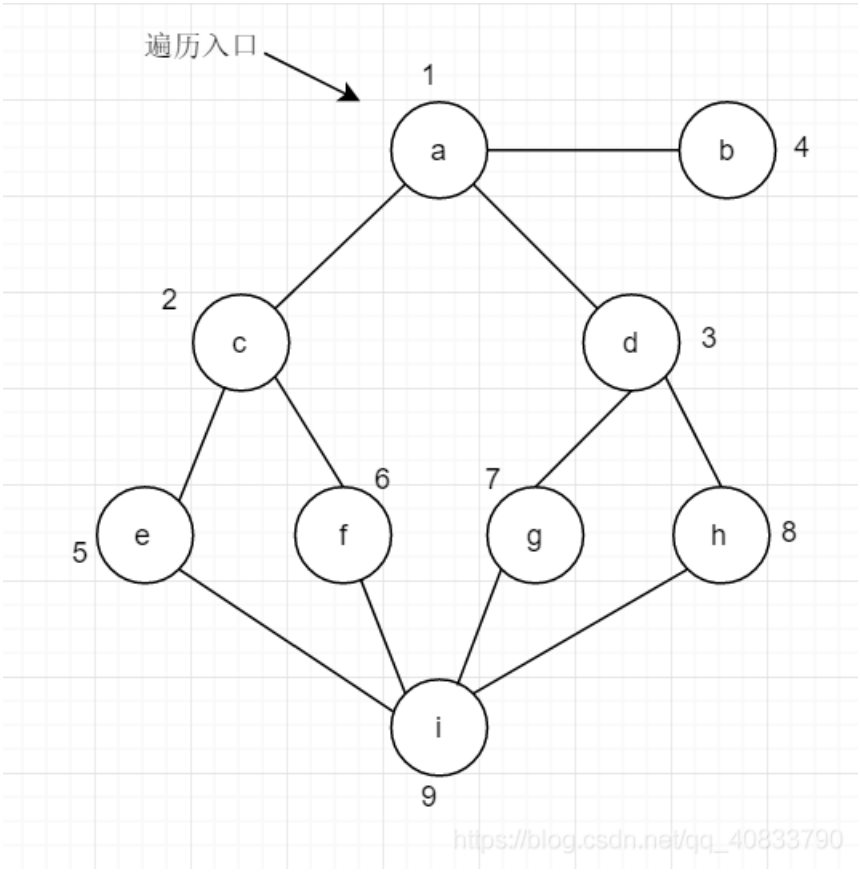
函数名：`breadthfirstsearch(const int startpoint,int visited[],void visit(char item))`

描述：深度优先搜索遍历它类似于树的先根遍历，是树的先根遍历的推广

入口参数：`const int startpoint,int visited[],void visit(char item)`

出口参数：无

说明：假设从图中某顶点v出发，在访问了v之后依次访问v的各个未曾访问过的邻接点，然后分别从这些邻接点依次出发访问他们的邻接点，并使先被访问的顶点的邻接点先于后被访问的顶点的邻接点被访问，直至图中所有已被访问的顶点的邻接点都被访问到。若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作为起始点，重复上述过程，直至图中所有都被到为止。换句话说，广度优先搜索遍历图中的过程中以v为起始点，由近至远，依次访问和v有路径相通的顶点。




```
==================================
图的基本功能实验(存储结构邻接矩阵)
==================================
1.手工建立（有向）图
2.默认数据建立图
3.用邻接矩阵形式显示图
4.求图的结点数和边数
5.求某条边的权值
6.添加结点
7.删除结点
8.添加边
9.删除边
a.深度优先遍历图
s.广度优先遍历图
0.退出
==================================
```

```
请输入选项：s
================
当前图的坐标和结点如下：
    0    1    2    3    4    5    6
    a    b    c    d    e    f    g
当前图的邻接矩阵如下：
    0    1    1   ∞   ∞   ∞   ∞
    1    0   ∞    1   ∞    1   ∞
    1   ∞    0   ∞   ∞   ∞    1
   ∞    1   ∞    0   ∞    1   ∞
   ∞   ∞   ∞   ∞    0    1   ∞
   ∞    1   ∞    1    1    0   ∞
   ∞   ∞    1   ∞   ∞   ∞    0
此处约定从第一个结点开始遍历。
广度优先遍历序列为 ：
    a    b    c    d    f    g    e
================
请按任意键继续. . .
```


五、部分源码

这次实验有点难度，这是老师提供的源码

```
/*功能描述：
图的相关功能实现，存储结构使用邻接矩阵
注意：程序基准为有向图，如果用输入两条边来达成无向图，内部的标志位没有跟上，会出现错误
*/

#include<iostream>
#include<iomanip>
#include<fstream>
#include <windows.h>
using namespace std;
enum returninfo {success,fail,overflow,underflow,nolchild,norchild,nofather,havesonl,havesonr,
                 haveason,havetwosons,range_error,quit
                }; //定义返回信息清单
int build;//用于控制是否是重建图，为0时表示第一次建立图，为1时表示重建图
/*************************************************************/
//顺序表类 把图中结点用顺序表来存储解决图中结点的动态插删问题
const int MaxListSize=26;//约定本顺序表最多只存26个字母
class SeqList {
 private:
  int size;
  int node[MaxListSize];
  /*这里必须考虑实现存储结点名的数据结构的存储结构，细节有
  类型为字符型，数组名可以到下面程序语句里去找，倒推出来。
  另外就是记录实际结点个数的变量。
  类型自己想，名字也到下面找吧。
    */

 public:
  SeqList();
  ~SeqList();
  int ListSize()const;
  int ListEmpty()const;
  int Find(char &item)const;
  char Getdata(int pos)const;
  void Insert(const char&item,int pos);
  int Delete(const int pos);
  void ClearList();
};
SeqList::SeqList() { //顺序表的构造函数
 size=0;
```

```
}
SeqList::~SeqList() {} //顺序表的析构函数
int SeqList::ListSize()const { //求顺序表的结点个数的函数
 return size;
}
int SeqList::ListEmpty()const { //判断顺序表是否为空的函数
 if(size==0)
  return 1;
 else
  return 0;
}
int SeqList::Find(char &item)const { //查找某个结点的函数
 if(size==0) {
  return -1;
 }
 int i=0;
 while(i<size && item!=node[i]) {
  i++;
 }
 if(i<size) {
  return i;
 } else return -1;
}
char SeqList::Getdata(int pos)const { //读取某个位置结点的函数
 if(pos<0||pos>size-1) {
  cout<<"对不起!位置参数越界出错!"<<endl;
  return(false);
 }
 return node[pos];
}
```

全部源码下载地址：https://download.csdn.net/download/qq_40833790/11141326