## 4基于邻接表的图实现

## 4.1 问题描述

通过实验达到⑴加深对图的概念、基本运算的理解；⑵熟练掌握图的逻辑结构与物理结构的关系；⑶以邻接表作为物理结构，熟练掌握图基本运算的实现。

## 4.1.1 实现基本操作

依据最小完备性和常用性相结合的原则，以函数形式定义了创建图、销毁图、查找顶点、获得顶点值和顶点赋值等12种基本运算。

## 4.1.2 实现数据存储与读取

设计函数使得能够从文件中读取或者存储图。

## 4.1.3 实现多个图管理

能管理多个图。

## 4.2 系统设计

## 4.2.1 程序总体框架

该系统包括用户交互模块，操作管理模块，多图管理模块，文件I/O模块，单个图操作模块。其中文件I/O模块包括写入与读取功能；单个图操作模块包括：创建图，销毁图，查找顶点，顶点赋值，获得第一邻接点，获得下一邻接点，插入顶点，删除顶点，插入弧，删除弧，深度优先搜索遍历，广度优先搜索遍历12个功能，如图4-1所示。

操作管理模块

多图管理模块

用户交互模块

单个图操作模块

文件I/O模块

查找顶点

图初始化

……

顶点赋值

销毁图

图4-1系统总体框架

## 4.2.2 系统环境

系统：macOS Catalina

IDE：clion

## 4.2.3 自定义部分

#include <stdio.h>  
#include <stdlib.h>  
#include <iostream>  
#include <string.h>  
#include <queue>  
#include <vector>  
  
using namespace std;  
#ifndef DS4\_GRAPH\_H  
#define DS4\_GRAPH\_H  
  
#define TRUE 1  
#define FALSE 0  
#define OK 1  
#define ERROR 0  
#define INFEASTIBLE -1  
#define OVERFLOW -2  
#define GLIST\_SIZE 10  
#define MAX\_VERTEX\_NUM 50  
#define STRING\_LENGTH 50  
#endif //DS4\_GRAPH\_H  
  
//文件绝对路径  
char FILE\_NAME[100] =  
 "/Users/akunda/Downloads/DS+/DS4/output.txt";  
typedef enum {  
 DG, DN, UDG, UDN  
} GraphKind;  
typedef int status;  
typedef char ElemType[STRING\_LENGTH]; //数据元素类型定义  
//元素结构  
typedef struct elem {  
 int id;  
 ElemType data;  
} Elem;  
//弧结构  
typedef struct ArcNode {  
 //相邻顶点编号  
 int adjvex;  
 //下一个弧  
 struct ArcNode \*next\_arc;  
 ElemType data;//其他信息  
} ArcNode, \*ArcNode\_p;  
//顶点结构  
typedef struct VNode {  
 Elem elem;  
 //相邻第一个弧  
 ArcNode \*first\_arc;  
} VNode, \*VNode\_p, AdjList[MAX\_VERTEX\_NUM];  
typedef struct {  
 AdjList vertices;//头结点数组  
 //定点数，弧数  
 int vexnum, arcnum;  
 GraphKind kind;//本实验使用UDG无向图  
} ALGraph, \*ALGraph\_p;  
  
//顶点初始化数组，以END结尾  
ElemType V\_init[] = {"zero", "one", "two", "three",  
 "four", "five", "END"};  
//弧初始化数组，两两成对，表示有关联的两个顶点，以-1结尾，  
int VR\_init[] = {1, 2, 1, 3, 1, 0, 2, 4, 3, 4, -1};

## 4.2.4 系统说明

1. 使用了一些辅助函数使程序简洁。
2. 创建图的依据是自定义部分已经定义的两个数组V\_init和VR\_init，如果要修改初始化图的结构需要将这两个数组的内容更改。
3. 初始化一个顶点的函数已经给出（在插入顶点的时候会用到）
4. 大部分函数传入的参数都是图的指针，并且在函数开始时会首先判断图是否存在（图指针是否为空），后面不再赘述。
5. 关于函数参数是顶点的id还是顶点位序（数组下标），参考函数分析或者函数注释。

## 4.3 系统实现

## 4.3.1 菜单栏

菜单栏操作步骤如图4-2所示：

开始

选择操作

选择图

退出

单个图操作模块

文件I/O模块

存储图

读取图

……

销毁图

查找顶点

图初始化

图4-2菜单栏示意图

## 4.3.2 函数实现与分析

⑴创建图：函数名称是CreateCraph(G,V,VR)；初始条件是V是图的顶点集，VR是图的关系集；操作结果是按V和VR的定义构造图G。

实现：1.初始化图的种类。2.初始化顶点id和顶点数。3.初始化弧数。4.遍历两次顶点，加入弧节点（因为无向图每个弧出现两次）。

要点：使用了GetLastArc函数，初始化数组需要有END和-1结尾标志。

流程图：

开始

图不存在

是

否

返回error

定义图类型（默认UDG）

遍历顶点数组，初始化顶点id并且统计顶点数量

遍历弧数组，统计弧数量

遍历顶点（两遍），分配弧节点

返回OK

图4-3创建图流程图

⑵销毁图：函数名称是DestroyGraph(G)；初始条件图G已存在；操作结果是销毁图G。

实现：删除：第一层遍历：所有顶点，第二层遍历：每一个顶点的所有弧。最后删除整个图。

⑶查找顶点：函数名称是LocateVex(G,u)；初始条件是图G存在，u是和G中顶点关键字类型相同的给定值；操作结果是若u在图G中存在，返回关键字为u的顶点位置序号（简称位序），否则返回-1。

实现：遍历顶点，比较顶点id和u。

⑷顶点赋值：函数名称是PutVex (G,u,value)；初始条件是图G存在，u是和G中顶点关键字类型相同的给定值；操作结果是对关键字为u的顶点赋值value。

要点：调用LocateVex函数，ChangeElem函数。

⑸获得第一邻接点：函数名称是FirstAdjVex(G, u)；初始条件是图G存在，u是G中顶点的位序；操作结果是返回u对应顶点的第一个邻接顶点位序，如果u的顶点没有邻接顶点，返回-1。

⑹获得下一邻接点：函数名称是NextAdjVex(G, v, w)；初始条件是图G存在，v和w是G中两个顶点的位序，v对应G的一个顶点,w对应v的邻接顶点；操作结果是返回v的（相对于w）下一个邻接顶点的位序，如果w是最后一个邻接顶点，返回-1。

实现：遍历表查找目标值。

⑺插入顶点：函数名称是InsertVex(G,v)；初始条件是图G存在，v和G中的顶点具有相同特征；操作结果是在图G中增加新顶点v。（在这里也保持顶点关键字的唯一性）

实现：1.判断顶点列表是否已满，已满则返回error。2.在顶点数组里添加顶点，同时更新id和图顶点数。3.记录新加顶点的关联顶点，更新图弧数。4.更新添加原来顶点的弧节点。

要点：1.判断顶点列表是否已满。2.调用GetLastArc函数

⑻删除顶点：函数名称是DeleteVex(G,v)；初始条件是图G存在，v是和G中顶点关键字类型相同的给定值；操作结果是在图G中删除关键字v对应的顶点以及相关的弧。

实现：1.找到删除顶点的位序。2.删除顶点的弧节点，同时更改图弧数。3.删除顶点，之后的顶点依次前移。4.遍历所有顶点，删除相关弧。

要点：调用了DeleteArcInNode函数，用来删除一个顶点中的某个弧。

⑼插入弧：函数名称是InsertArc(G,v,w)；初始条件是图G存在，v、w是和G中顶点关键字类型相同的给定值；操作结果是在图G中增加弧<v,w>，如果图G是无向图，还需要增加<w,v>。

实现：1.定位v，w。2.检查v，w之间是否已经有弧。3.给v，w分别添加弧。4.图弧数+1.

⑽删除弧：函数名称是DeleteArc(G,v,w)；初始条件是图G存在，v、w是和G中顶点关键字类型相同的给定值；操作结果是在图G中删除弧<v,w>，如果图G是无向图，还需要删除<w,v>。

要点：调用DeleteArcInNode函数。

⑾深度优先搜索遍历：函数名称是DFSTraverse(G,visit())；初始条件是图G存在；操作结果是图G进行深度优先搜索遍历，依次对图中的每一个顶点使用函数visit访问一次，且仅访问一次。

要点：visit数组记录数据是否访问过；遍历完起始顶点之后，还要检查是否有其他顶点没有访问过（不连通）。

流程图：

开始

图不存在

是

否

返回error

初始化visit数组为0

返回OK

输入遍历初始顶点id

对起始顶点调用DFS函数

对visit数组中仍为0的顶点调用vsit函数

图4-4深度优先搜索遍历流程图

⑿广度优先搜索遍历：函数名称是BFSTraverse(G,visit())；初始条件是图G存在；操作结果是图G进行广度优先搜索遍历，依次对图中的每一个顶点使用函数visit访问一次，且仅访问一次。

要点：visit数组记录数据是否访问过；使用队列；遍历完起始顶点之后，还要检查是否有其他顶点没有访问过（不连通）。

⒀存储表：f\_write(G, file\_name)，操作结果是将G的数据存到file\_name文件里。

实现：打开文件，判断文件是否打开，写入数据，关闭文件。其中写入数据包括写入顶点数据和写入弧关系，顶点数据只需简单遍历；弧关系的写入要考虑到同一个弧不能写入两次。

⒁读取表：f\_read(G, file\_name)，操作结果是将file\_name的数据读入到G内。

实现：判断表是否已存在，打开文件，判断文件是否打开，读取数据，分别把顶点信息和顶点关系读入两个数组，然后调用CreateGraph函数，关闭文件。

**4.4 系统测试**

## 4.4.1 准备

使用Show函数用来展示图的邻接表结构，每进行一次操作都会展示当前图的结构。

## 4.4.2 测试

说明：

在选择操作前需要选择操作的图，所以实际上已经实现了对多个图进行管理。

菜单栏如下：

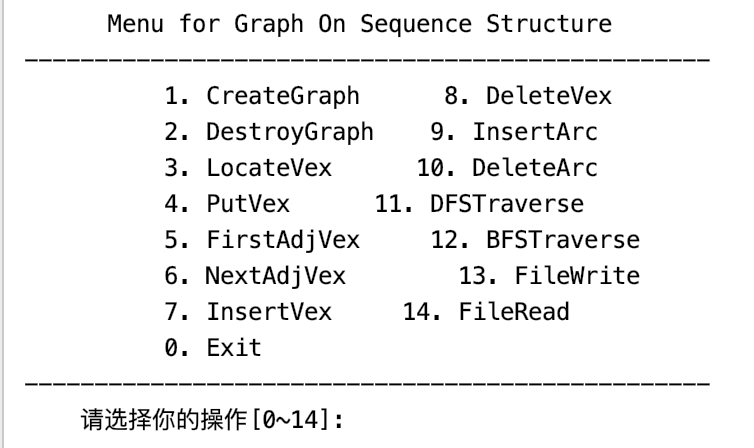


图4-5菜单栏

⑴创建图

操作结果：

表 4-1创建图结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 正常操作 | 截屏2019-12-17下午9.14.44  图4-6-a创建图结果 |
| 图已经存在 | 截屏2019-12-17下午9.14.55  图4-6-b创建图结果（图已存在） |

⑵销毁图

操作结果：

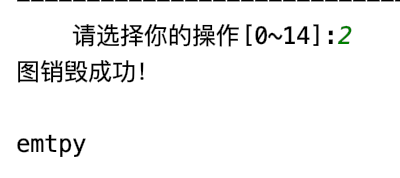


图4-7销毁图结果

⑶查找节点

操作结果：

表 4-2查找节点结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 查找成功 | 截屏2019-12-17下午9.26.21  图4-8-a查找节点结果（内容存在） |
| 查找失败 | 截屏2019-12-17下午9.22.06  图4-8-b查找节点结果（内容不存在） |

⑷顶点赋值

操作结果：

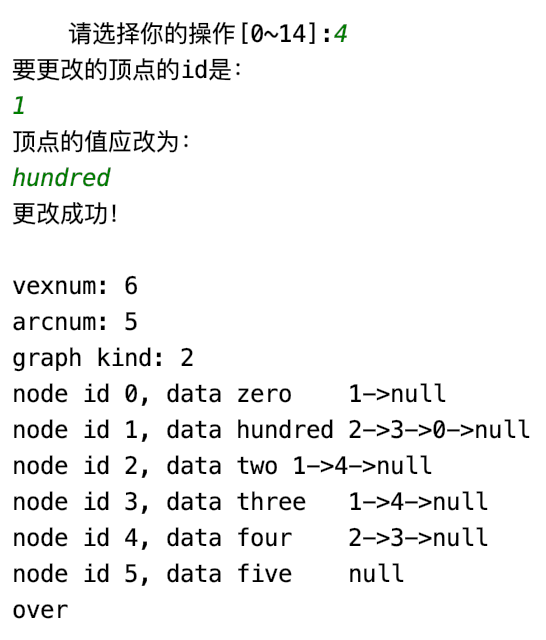


图4-9销毁图结果

⑸获得第一邻接点

操作结果：

表 4-3获得第一邻接点结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 获取成功 | 截屏2019-12-17下午9.29.52  图4-10-a获得第一邻接点结果（内容存在） |
| 获取失败 | 截屏2019-12-17下午9.30.18  图4-10-b获得第一邻接点结果（内容不存在） |

⑹获得下一邻接点

操作结果：

表 4-4获得下一邻接点结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 获取成功 | 截屏2019-12-17下午9.31.15  图4-11-a获得下一邻接点结果（内容存在） |
| 获取失败 | 截屏2019-12-17下午9.31.35  图4-11-b获得下一邻接点结果（内容不存在） |

⑺插入顶点

操作结果：

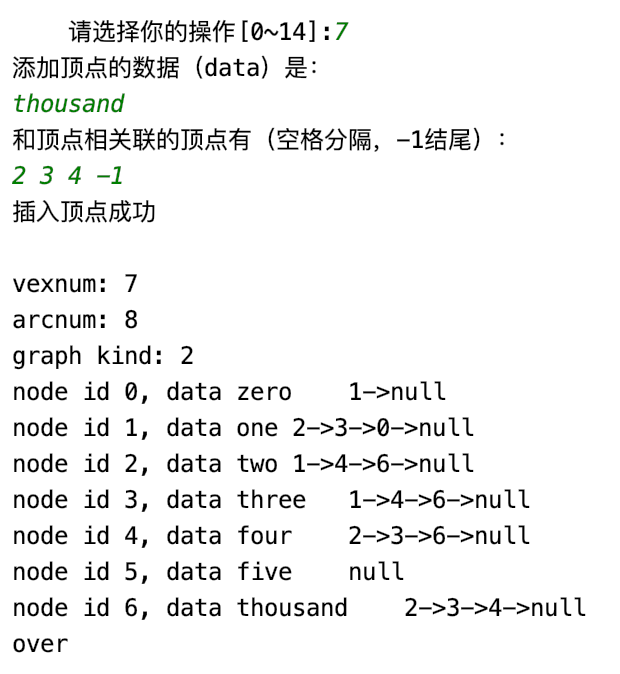


图4-12插入顶点结果

⑻删除顶点

操作结果：（删除了（7）里添加的顶点）

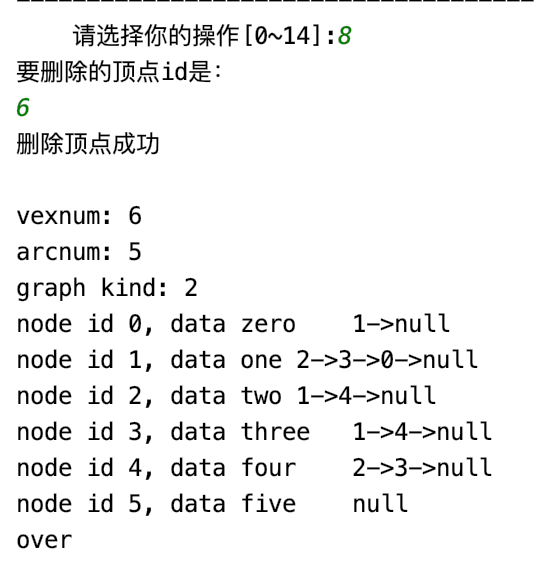


图4-13删除顶点结果

⑼插入弧

操作结果：

表 4-5插入弧结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 正常操作 | 截屏2019-12-17下午9.34.38  图4-14-a插入弧结果（正常） |
| 弧已存在 | 截屏2019-12-17下午9.43.44  图4-14-b插入弧结果（弧已存在） |

⑽删除弧

操作结果：

表 4-6删除弧结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 正常操作 | 截屏2019-12-17下午9.44.40  图4-15-a删除弧结果（正常） |
| 弧不存在 | 截屏2019-12-17下午9.44.55  图4-15-b删除弧结果（弧不存在） |

⑾深度优先搜索遍历

操作结果：

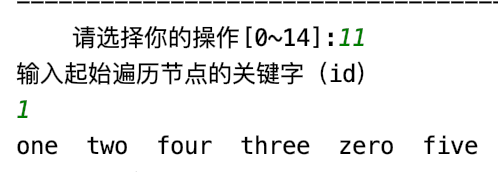


图4-16深度优先搜索遍历结果

⑿广度优先搜索遍历

操作结果：

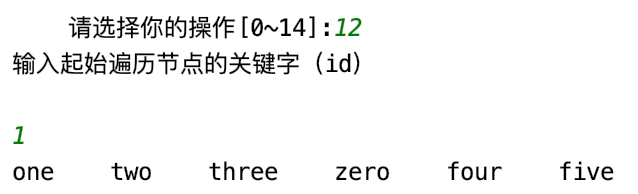


图4-17广度优先搜索遍历结果

文件操作：

文件内容：截屏2019-12-17下午9.47.40

图4-18文件内容

⒀存储表

操作结果：

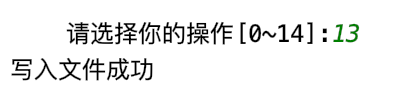


图4-19存储表结果

⒁读取表

操作结果：

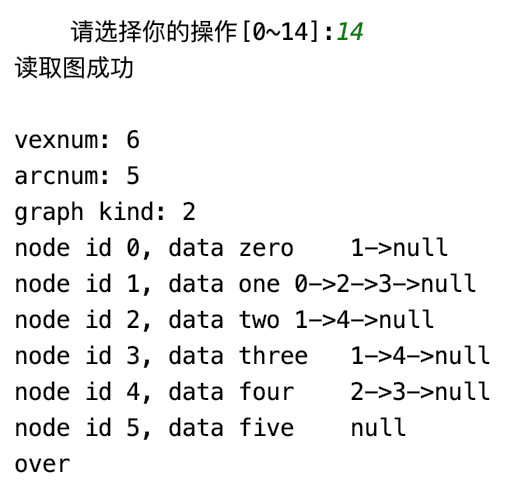


图4-20读取表结果

## 4.5 实验小结

1. 掌握了对图的使用加深了理解。
2. 借鉴二叉树实验的经验，先编写了展示图的函数，方便后续调试。
3. 对动态内存分配有了进一步的认识。
4. 经常会遇到函数里面多次使用同一功能的情况，需要将功能分离出来写成单独的函数。