《数据结构》课程实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 专业名称 | 计算机 | 年级 | 2017 | 班级 | 2 |
| 学生姓名 | 王汝芸 | 学号 | 201711010202 | 指导教师 | 郑志华 |
| 实验题目 | 实验1 顺序表的建立和运算 | | | 提交时间 | 2018.5.12 |

一、实验目的和要求

（1）熟悉图和其邻接表

（2）掌握深度优先遍历或广度优先遍历方法

二、实验环境

Visual Studio2017

Windows 10

三、实验内容及实施

采用数组作为表头，链节作为表尾；宏定义N为节点长度，方便后期修改。

**实验6：深度优先遍历**

实验要求：

1、用邻接表表示图；

2、深度或广度优先遍历图；

**模块图**

建立n个节点的图

输出邻接表

输出广度优先遍历结果

NULL

NULL

1、用邻接表表示图；

2、深度或广度优先遍历图；

**【源程序】**

/\*

实验6：深度优先遍历（或广度优先遍历）一个具有n个顶点的无向图。

//广度优先

https://blog.csdn.net/wqc\_csdn/article/details/52673664

\*/

#include<stdio.h>

#include<stdlib.h>

#define N 8 //节点个数

//邻接表表示法 表头结点

typedef struct VNode

{

int data;//顶点信息

VNode\* firstarc;//指向链表中第一个节点

}VNode;

//创建图

VNode\* Creat(VNode\*a)

{

int i = 0;

for (i = 0; i < N; i++)//初始化

{

a[i].data = i;

}

VNode\* p0, \*q0;

p0 = (VNode\*)malloc(sizeof(VNode));

a[0].firstarc = p0;

p0->data = 1;

q0 = (VNode\*)malloc(sizeof(VNode));

p0->firstarc = q0;

q0->data = 2;

q0->firstarc = NULL;

VNode\* p1, \*q1, \*r1;

p1 = (VNode\*)malloc(sizeof(VNode));

p1->data = 0;

a[1].firstarc = p1;

q1 = (VNode\*)malloc(sizeof(VNode));

q1->data = 3;

p1->firstarc = q1;

r1 = (VNode\*)malloc(sizeof(VNode));

r1->data = 4;

r1->firstarc = NULL;

q1->firstarc = r1;

VNode\* p2, \*q2,\*r2;

p2 = (VNode\*)malloc(sizeof(VNode));

p2->data = 0;

a[2].firstarc = p2;

q2 = (VNode\*)malloc(sizeof(VNode));

q2->data = 5;

p2->firstarc = q2;

r2 = (VNode\*)malloc(sizeof(VNode));

r2->data = 6;

r2->firstarc = NULL;

q2->firstarc = r2;

VNode\* p3,\*q3;

p3 = (VNode\*)malloc(sizeof(VNode));

p3->data = 1;

a[3].firstarc = p3;

q3 = (VNode\*)malloc(sizeof(VNode));

q3->data = 7;

q3->firstarc = NULL;

p3->firstarc = q3;

VNode\* p4,\*q4;

p4 = (VNode\*)malloc(sizeof(VNode));

p4->data = 1;

a[4].firstarc = p4;

q4 = (VNode\*)malloc(sizeof(VNode));

q4->data = 7;

q4->firstarc = NULL;

p4->firstarc = q4;

VNode\* p5;

p5 = (VNode\*)malloc(sizeof(VNode));

p5->data = 2;

p5->firstarc = NULL;

a[5].firstarc = p5;

VNode \*p6;

p6 = (VNode\*)malloc(sizeof(VNode));

p6->data = 2;

p6->firstarc = NULL;

a[6].firstarc = p6;

VNode\*p7,\*q7;

p7 = (VNode\*)malloc(sizeof(VNode));

p7->data = 3;

q7 = (VNode\*)malloc(sizeof(VNode));

q7->data = 4;

p7->firstarc = q7;

q7->firstarc = NULL;

a[7].firstarc = p7;

return a;

}

//打印图的邻接表

void Print(VNode\*a)

{

VNode\* p;

printf("该图的邻接表如下：\n");

for (int i = 0; i < N; i++)//输出图

{

printf("v%d→", a[i].data);

if (a[i].firstarc)//如果a有尾巴,打印尾巴

{

p = a[i].firstarc;

while (1)

{

printf("%d→", p->data);

if (p->firstarc == NULL)

{

printf("NULL\n");

break;

}

p = p->firstarc;

}

}

else

{

printf("NULL\n");

}

}

}

//广度优先遍历图

void WScan(VNode\* a)

{

printf("广度优先遍历如下：\n");

int visited[N] = { 0 };//标记

visited[0] = 1;

printf("v0 ");

VNode \*p;

//广度优先遍历图

for (int i = 0; i < N; i++)

{

if (a[i].firstarc)//如果a有尾巴,开始访问尾巴

{

p = a[i].firstarc;//初始化p在头结点

while(1)

{

if (visited[p->data] == 0 )//如果p所指结点未被访问，则访问

{

printf("v%d ", p->data);

visited[p->data] = 1;

}

if (p->firstarc == NULL)//当邻接表结束，跳出循环，开始下一行邻接表遍历

{

break;

}

p = p->firstarc;

}

}

else//如果a没尾巴，开始新的循环

{

continue;

}

}

}

int main()

{

struct VNode a[N];//表头数组

Creat(a);//创建邻接表

Print(a);//打印图的邻接表

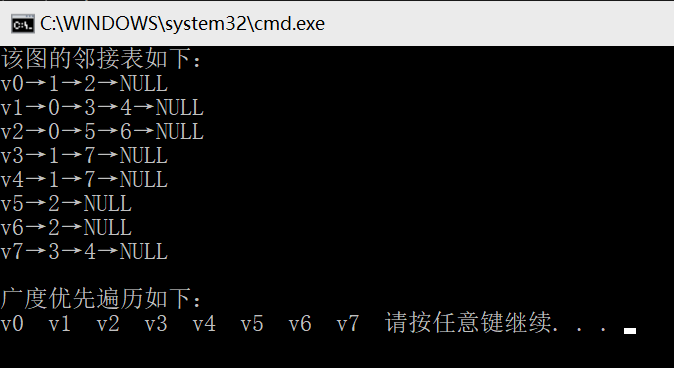
printf("\n");

WScan(a);//广度优先遍历

return 0;

}

**四、实验结果 (程序的执行结果)**



**五、实验讨论（可选）**

无