## 数据结构上机测试

**学号： 姓名：**

**命题人：周波**

**测试时间：2020年6月12日**

**地点：在线**

**注意事项：**

1. **测试须独立完成，若发现作弊（抄袭、交头接耳、程序代码雷同等均视为作弊），上机测试成绩作0分处理。**
2. **测试时长为2个半小时。**
3. **提交文件以 学号 姓名 命名。**
4. **不得查阅任何网页。**

**（1）编写算法，将二个升序链表在原表空间上合并成一个升序链表，须自行写出测试样例进行测试，截图为证。**

**解答：**

1. 源代码

#pragma once

#include <iostream>

using namespace *std*;

typedef struct node

{

int data;

node\* next;

}List;

class linkedlist

{

public:

linkedlist();

~linkedlist();

void setlist();

void createlist(int len);

void hebinglist(linkedlist& a, linkedlist& b);

void printlist();

private:

node\* head;

};

#include "linkedlist.h"

linkedlist::linkedlist()

{

head = *NULL*;

}

linkedlist::~linkedlist()

{

}

void linkedlist::setlist()

{

head = new List;

if (!head)

{

*cout* << "构造链表失败" << *endl*;

}

else

{

head->next = *NULL*;

}

}

void linkedlist::createlist(int len)

{

setlist();

List\* tail = head;

*cout* << "输入"<<len<<"个元素构建链表" << *endl*;

for (int i=1;i<=len;i++)

{

int num;

List\* tmp = new List;

*cin* >> num;

tmp->*data* = num;

tmp->*next* =tail->*next*;

tail->*next* = tmp;

tail = tmp;

}

}

void linkedlist::hebinglist(linkedlist& a, linkedlist& b)

{

List\* L1, \* L2, \* tmp,\*nL1;

L1 = a.head;

L2 = b.head;

nL1 = a.head;//new L1 list

while (L1 != *NULL* && L2 != *NULL*)

{

if (L1->data==L2->data)//L1保留动l2

{

L2 = L2->next;

}

else if(L1->data<L2->data)//l1保留

{

nL1 = L1;

L1 = L1->next;

}

else

{

tmp = L2->next;

L2->next = nL1->next;//插 l2 到 l1

nL1->next = L2;

nL1 = nL1->next;

L2 = tmp;

}

}

while (L2!=*NULL*)//如果有剩下全扔l1

{

nL1->next = L2;

nL1 = L2;

L2 = L2->next;

}

*cout* << "合并后";

a.printlist();

}

void linkedlist::printlist()

{

if (head->next==*NULL*)

{

return;

}

else

{

List\* tmp = head->next;

while (tmp)

{

*cout* << tmp->*data* << " ";

tmp = tmp->*next*;

}

}

*cout* << *endl*;

}

#include <iostream>

#include "linkedlist.h"

int main()

{

linkedlist L1,L2;

int len1,len2;

*cout* << "构造两个升序链表并且合并为一个" << *endl*;

*cout* << "第1个表长";

*cin* >> len1;

L1.createlist(len1);

L1.printlist();

*cout* << "第2个表长";

*cin* >> len2;

L2.createlist(len2);

L2.printlist();

L1.hebinglist(L1, L2);

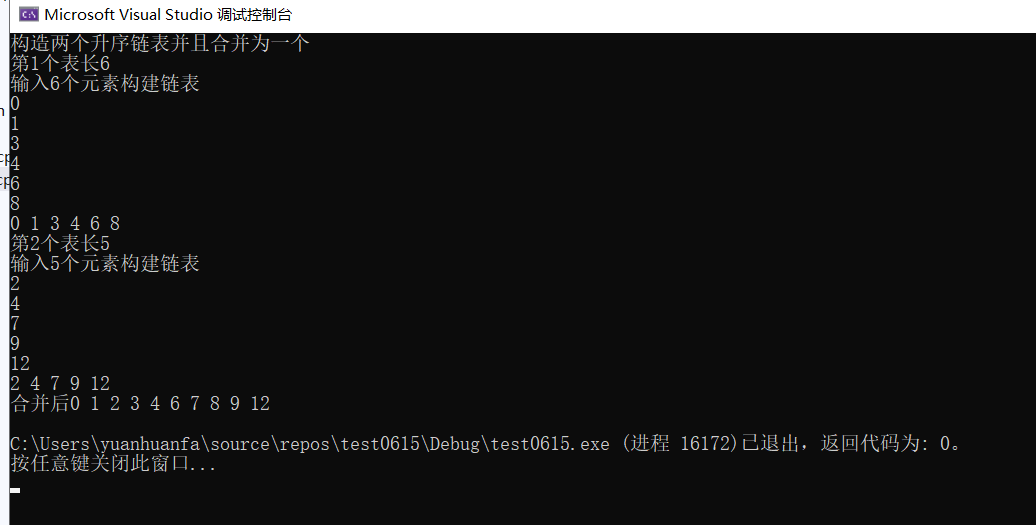
//cout << "合并";

//L1.printlist();

return 0;

}

（2）程序运行结果截图与说明



**（2）利用顺序存储结构存储如下二叉树，并在顺序结构的基础上（利用链式结构完成给予零分）完成如下功能：**

**1）完成后序遍历，输出后序序列；**

**2）编写算法输出所有度为1的结点及数量；**

|  |
| --- |
|  |
| 图1 二叉树结构 |

**解答：**

1. 源代码

#pragma once

#include <iostream>

using namespace *std*;

typedef struct tree

{

char data;

struct tree\* lchild, \* rchild;

}Atree;

class Tree

{

public:

Tree();

~Tree();

void sontree(Atree\* T, int x);

void creatree(Atree\* &T);

void lasttravel(Atree\* T);

};

#include "Tree.h"

Tree::Tree()

{

}

Tree::~Tree()

{

}

void Tree::creatree(Atree\*& T)

{

Atree\* p;

char x;

*cin* >> x;

if (x=='/')

{

return;

}

T = new Atree;

T->data =x;

T->lchild = *NULL*;

T->rchild = *NULL*;

p = T;

sontree(p, 1);

sontree(p, 2);

}

void Tree::sontree(Atree\* T, int i)

{

char x;

Atree\* tmp;

*cin* >> x;

if (x != '/')

{

tmp = new Atree;

tmp->data = x;

tmp->lchild = *NULL*;

tmp->rchild = *NULL*;

if (i == 1)

{

T->lchild = tmp;

}

if (i == 2)

{

T->rchild = tmp;

}

sontree(tmp, 1);

sontree(tmp, 2);

}

}

void Tree::lasttravel(Atree\* T)

{

if (T)

{

lasttravel(T->lchild);

lasttravel(T->rchild);

*cout* << T->data;

}

}

#include "Tree.h"

#include <iostream>

int main()

{

Tree Z;

Atree\* T;

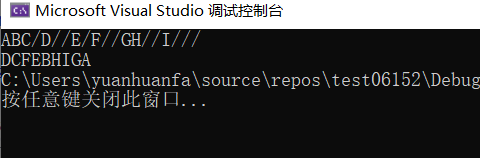
Z.creatree(T);

Z.lasttravel(T);

return 0;

}

（2）程序运行结果截图与说明



**（3）接上一题（此题单独评分）：**

**1）求出该二叉树的深度**

**2）输出图中所有叶子结点到根节点的路径**

**3）输出最长路径及长度**

**解答：**

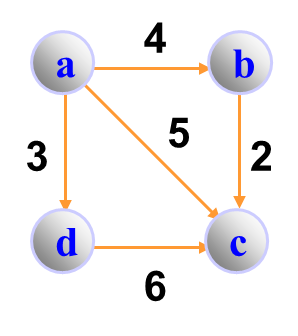
（1）源代码

（2）程序运行结果截图与说明

**（4）利用邻接矩阵存储如下图结构，并完成如下功能：**

**1）读取邻接矩阵中存储的数据，将其转化为邻接表存储，并将邻接表输出；**

**2）利用邻接表统计每个顶点的出度、入度；**



**解答：**

（1）源代码

（2）程序运行结果截图与说明