## 数据结构上机测试

**学号：2019217819 姓名：付炎平**

**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

#include<iostream>

using namespace std;

struct node{

int data;

node\* next;

};

node\* create\_list(int \*a){

node\* ptr=new node;

node\* head=ptr;

ptr->next=NULL;

for(int i=0;a[i]!=0;i++){

node\* new\_node=new node;

new\_node->data=a[i];

new\_node->next=NULL;

ptr->next=new\_node;

ptr=ptr->next;

}

return head;

}

node\* Merge(int\* a,int \*b){

node\* head1=create\_list(a);

node\* head2=create\_list(b);

node\* ptr1=head1;

node\* ptr2=head2->next;

while(ptr2!=NULL&&ptr1->next!=NULL){

if(ptr2->data<ptr1->next->data){

node\* ptr=ptr2;

ptr2=ptr2->next;

ptr->next=ptr1->next;

ptr1->next=ptr;

}

else{

ptr1=ptr1->next;

}

}

if(ptr1->next==NULL){

ptr1->next=ptr2;

}

node \*p=head1->next;

return head1;

}

int main(){

int a[100]={1,3,6,8,10,15};

int b[100]={2,5,7,11,13,18};

node\* head=Merge(a,b);

cout<<"表1为：1,3,6,8,10,15"<<endl<<"表2为：2,5,7,11,13,18"<<endl<<"合并两表后：";

node\* ptr=head->next;

while(ptr!=NULL){

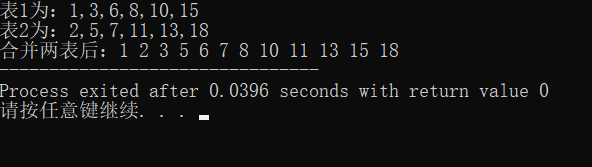
cout<<ptr->data<<" ";

ptr=ptr->next;

}

}

（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

#include<iostream>

using namespace std;

char Tree[100]={0};

int j=0;

void create\_tree(int i,char\* a){

if(a[j]!='#'){

Tree[i]=a[j];

j++;

create\_tree(2\*i,a);

create\_tree(2\*i+1,a);

}

else{

j++;

}

}

void back\_find(int i){

if(Tree[i]!=0){

back\_find(2\*i);

back\_find(2\*i+1);

cout<<Tree[i]<<" ";

}

}

void find\_one(int i,int &num){

if(Tree[i]!=0){

if(Tree[2\*i]==0&&Tree[2\*i+1]!=0||Tree[2\*i]!=0&&Tree[2\*i+1]==0){

cout<<Tree[i]<<" ";

num++;

}

find\_one(2\*i,num);

find\_one(2\*i+1,num);

}

}

int main(){

char a[]="ABC#D##E#F##GH##I##";

create\_tree(1,a);

cout<<"后序序列为：";

back\_find(1);

int num=0;

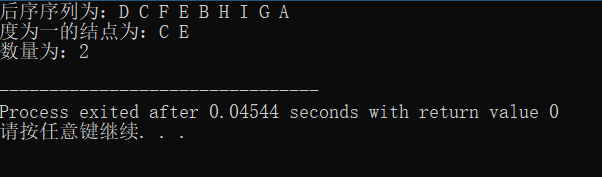
cout<<endl<<"度为一的结点为：";

find\_one(1,num);

cout<<endl<<"数量为："<<num<<endl;

}

（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

int get\_height(int i){

if(Tree[i]!=0){

int a=get\_height(2\*i);

int b=get\_height(2\*i+1);

if(a>b){

return a+1;

}

else{

return b+1;

}

}

return 0;

}

void output\_path(int i,char\* path,int j){

if(Tree[i]!=0){

path[j]=Tree[i];

output\_path(2\*i,path,j+1);

output\_path(2\*i+1,path,j+1);

if(Tree[2\*i]==0&&Tree[2\*i+1]==0){

for(int k=j;k>=0;k--){

cout<<path[k];

}

cout<<endl;

}

}

}

void max\_path(int i,char\* path,int j,int height){

if(Tree[i]!=0){

path[j]=Tree[i];

max\_path(2\*i,path,j+1,height);

max\_path(2\*i+1,path,j+1,height);

if(j==height-1){

cout<<"最长路径为："<<endl;

for(int k=j;k>=0;k--){

cout<<path[k];

}

cout<<endl;

}

}

}

int main(){

char a[]="ABC#D##E#F##GH##I##";

create\_tree(1,a);

int height=get\_height(1);

cout<<"树的深度为："<<height<<endl;

char path1[100]={0};

char path2[100]={0};

cout<<"各个叶子结点到根节点的路径为："<<endl;

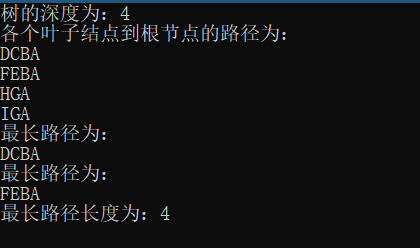
output\_path(1,path1,0);

max\_path(1,path2,0,height);

cout<<"最长路径长度为："<<height<<endl;

}

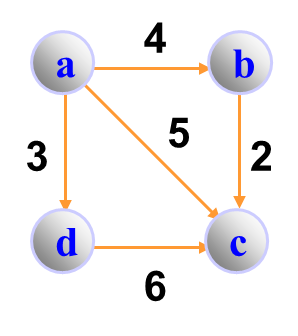
（2）程序运行结果截图与说明



**（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

#include<iostream>

using namespace std;

struct node{

char data;

node\* next;

};

int Graph[4][4]={

{0,4,5,3},

{0,0,2,0},

{0,0,0,0},

{0,0,6,0}

};

void transform(node\* a,char\* list){

for(int i=0;i<4;i++){

node\* ptr=a+i;

for(int j=0;j<4;j++){

if(Graph[i][j]!=0){

node\* new\_node=new node;

new\_node->data=list[j];

new\_node->next=NULL;

ptr->next=new\_node;

ptr=ptr->next;

}

}

}

}

void find\_node(node\* a,int \*in\_node,int \*out\_node,char\* list){

for(int i=0;i<4;i++){

node\* ptr=a[i].next;

while(ptr){

out\_node[i]++;

for(int k=0;k<4;k++){

if(a[k].data==ptr->data){

in\_node[k]++;

}

}

ptr=ptr->next;

}

}

}

int main(){

int in\_node[4]={0};

int out\_node[4]={0};

node\* a=new node[4];

char list[]="abcd";

for(int i=0;i<4;i++){

a[i].data=list[i];

a[i].next=NULL;

}

transform(a,list);

find\_node(a,in\_node,out\_node,list);

for(int i=0;i<4;i++){

cout<<a[i].data<<"的入度为"<<in\_node[i]<<" "<<endl;

}

for(int i=0;i<4;i++){

cout<<a[i].data<<"的出度为："<<out\_node[i]<<" "<<endl;

}

}

（2）程序运行结果截图与说明

