山东大学 软件学院 学院

软件测试 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201800301202 | 姓名：李成 | | 班级： 2018级软件4班 |
| 实验题目：二叉树操作 | | | |
| 实验学时：4 | | 实验日期2019-11-26 | |
| 实验目的:  掌握二叉树的基本概念，二叉树的存储结构使用链表。 | | | |
| 硬件环境：  1、笔记本PC | | | |
| 软件环境：   1. Windows10 2. Eclipse 3. Java | | | |
| 实验步骤与内容：  实验内容：  1、输入一个完全二叉树的层次遍历字符串，创建这个二叉树，输出这个  二叉树的前序遍历字符串、中序遍历字符串、后序遍历字符串、结点 数目、二叉树高度(上述每一个结果独立一行显示)。  2、输入二叉树前序序列和中序序列(各元素各不相同)，创建这个二叉  树，输出该二叉树的后序序列、层次遍历  实验步骤：  1.先对二叉树的树节点进行定义  class treeNode{    public:    treeNode \*leftNode,\*rightNode;  char element;  treeNode(){    leftNode = NULL;  rightNode = NULL;  element=NULL;    }  treeNode(char a){    leftNode = NULL;  rightNode = NULL;  element=a;  }    treeNode(char a , treeNode left , treeNode right){    \*leftNode = left;  \*rightNode = right;  element = a;    }    treeNode& operator=(const treeNode &t ){    this->element = t.element;    this->leftNode = t.leftNode;    this->rightNode = t.rightNode;      }  };  2.为了完成层次遍历，我们还学要一个队列  class queue{    private:    treeNode \*\*q;    int capacity;    int begin;    int end;    public:    queue(int n){    q = new treeNode\* [n];  begin = 0;  end = 0;    capacity = n;    }    void push(treeNode \*T){    end = (end+1)%capacity;    q[end] = T;    }    treeNode\* read(){    return q[(begin+1)%capacity];    }    void pop(){    begin = (begin+1)%capacity;    }    void del(){    delete[] q;  }  };  3.然后就是对二叉树的类，在二叉树的类中其中最重要的就是三种遍历方法，他们可以用递归来是实现，层次遍历则可以通过队列来实现。  class binaryTree{    public:  int number=0;  int height=0;  treeNode\* root = NULL;  treeNode \*p;  int Middlei = 0;  int Posti = 0;  int count = 0;  treeNode\* p1;    void bT(treeNode \*a,char ch[],int n,int i){    a->leftNode = new treeNode(ch[2\*i+1]);    a->rightNode = new treeNode(ch[2\*i+2]);      }  binaryTree(char ch[],int n){    number = n;  height = log(n+1)/log(2);    queue q(n+1);    root = new treeNode(ch[0]);    q.push(root);    for(int i = 0 ; i<n/2 ; i++){    p = q.read();    bT(p,ch,n,i);    q.pop();    q.push(p->leftNode);  q.push(p->rightNode);  }    q.del();    }    binaryTree(char ch1[],char ch2[],int n) {    int x = 0;    int count = 1;    while(ch1[x]!=ch2[0]){    if(x==0){    this->root = new treeNode(ch1[0]);    p1 = root;    }  else{    p1->leftNode = new treeNode(ch1[x]);    p1 = p1->leftNode;    }    x++;    }  if(x==0){    this->root = new treeNode(ch1[x]);  }  else{    p1->leftNode = new treeNode(ch1[x]);    p1 = p1->leftNode;    }    x++;    int z = x;    while(count < n&&ch1[0]!=ch2[count]){    int y =0;  treeNode \*p2;  p2 = root;      while(y<x&&ch1[y]!=ch2[count]){    p2 = p2->leftNode;  y++;    }    if(y<x&&ch1[y]==ch2[count]){    p2->rightNode = new treeNode(ch2[count+1]);    z++;    }    count++;  }    int a = z;    if(n-z>0){    char cha1[n-z],cha2[n-z];    for(int i = 0; i < n-z ; i++){    cha1[i] = ch1[z];    cha2[i] = ch2[z];    z++;    }    binaryTree T(cha1,cha2,n-a);    root->rightNode = T.root;    }    }    void PreRead(treeNode \*a){    if(a->leftNode!=NULL){    cout<<","<<a->leftNode->element;    PreRead(a->leftNode);    }  if(a->rightNode!=NULL){    cout<<","<<a->rightNode->element;    PreRead(a->rightNode);  }  }    void preRead(){    cout<<root->element;    PreRead(root);    }    void MiddleRead(treeNode \*a){    if(a->leftNode!=NULL){    MiddleRead(a->leftNode);    cout<<","<<a->element;    MiddleRead(a->rightNode);    }  else{    if(Middlei == 0){    cout<<a->element;    Middlei++;  }  else    cout<<","<<a->element;    }  }    void middleRead(){  cout<<endl;  MiddleRead(root);    }    void PostRead(treeNode \*a){    if(a->leftNode!=NULL){    PostRead(a->leftNode);    if(a->rightNode!=NULL)  PostRead(a->rightNode);  cout<<","<<a->element;    }    else{    if(Posti==0){    cout<<a->element;  Posti++;    }else{      cout<<","<<a->element;    }  }  }    void postRead(){    cout<<endl;  PostRead(root);    }    int getNumber(){    return number;    }    int getHeight(){    return height;    }  void LevelRead(treeNode \*a){    if(a->leftNode!=NULL)  cout<<","<<a->leftNode->element;  if(a->rightNode!=NULL)  cout<<","<<a->rightNode->element;  if(a->leftNode!=NULL)  LevelRead(a->leftNode);  if(a->rightNode!=NULL)  LevelRead(a->rightNode);    }  void levelRead(){    cout<<root->element;    LevelRead(root);    }    };  4.main（）方法  int main(){  char ch[20];    int num = 0;  for(int i = 0 ; i < 20 ; i++) {    ch[i] = NULL;  }    cout<<"Input1"<<endl;    cin>>ch;    cout<<"Output1"<<endl;    while(ch[num]!=NULL){    num++;    }  binaryTree T(ch,num);    T.preRead();  T.middleRead();  T.postRead();    cout<<endl;  cout<<T.getNumber()<<endl;  cout<<T.getHeight()<<endl;  char ch1[20],ch2[20];    cout<<"Input2"<<endl;  cin>>ch1;  cin>>ch2;    int k = 0;    for(;ch1[k]!=NULL;k++);  cout<<"Output2";    binaryTree Te(ch1,ch2,k);    Te.postRead();  cout<<endl;  Te.levelRead();  cout<<endl;  cout<<"End";    return 0;  } | | | |
| 结论分析与体会：  通过这次实验，体会到了二叉树的各种操作，以及它的三种遍历方法，其中较为困难的就是二叉树的遍历方法，需要用到之前的知识相互结合才能得到最后的结果。 | | | |