数据结构实验报告6

学号：2022114467 姓名：林泽宇 专业：软件工程

知识范畴：树与二叉树 完成日期：2023年 11月2日

实验题目：二叉链表的建立、遍历以及生成完全二叉树顺序存储结构

|  |  |  |  |
| --- | --- | --- | --- |
| 课程目标1（60%） | 课程目标2（40%） | 得分(100分)  期末成绩占比5% | 批阅人签字 |
| 程序代码及测试 | 写作、排版、代码注释等 |  |
|  |  |  |

实验内容及要求：

设二叉树结点数据域为字符类型，从键盘输入先序递归遍历字符序列（用#字符表示NULL指针域）建立二叉链表存储结构，然后输出其先、中、后序递归遍历结点访问次序以及层次遍历结点访问次序(层次遍历所需队列要求采用循环队列)；

建立该二叉树对应的完全二叉树顺序存储结构，输出完全二叉树顺序存储结构(哑元素采用@字符表示)。

如：二叉树如下所示，则输出的完全二叉树顺序结构为：A@B@@C@@@@@@D



实验目的：掌握二叉树二叉链表存储结构的建立、遍历等基本算法。

数据结构设计简要描述：

typedef char elemT;

typedef struct BtNode

{

    elemT data;

    struct BtNode \*lc, \*rc;

} BtNode, \*btptr;

采用二叉链表, 每个节点包括一个字符数据与和指向两个子树的指针域.

算法设计简要描述：

采用递归算法实现前中后序遍历, 采用循环链表实现层序遍历, 使用层序遍历的思想实现将链式存储结构转换为顺序存储结构.

输入/输出设计简要描述：

从键盘输入数据, 以字符#表示空指针域, 换行或者空格表示结束输入.

输出按前序, 中序, 后序, 层序, 完全二叉树顺序输出.

输入有文字提示.

编程语言说明：

使用Visual C++编程。 主要代码采用C语言实现 ；动态存储分配采用C++的new和delete操作符实现；输入与输出采用C++的cin和cout流；程序注释采用C/C++规范。

主要函数说明：

void initBt(btptr &root) //初始化二叉树

void assignBiTree(btptr &root, const string str) //通过前序str给二叉树赋值

void preOrderTraverse(btptr &root) //前序遍历

void inOrderTraverse(btptr &root) //中序遍历

void postOrderTraverse(btptr &root) //后序遍历

void layerOrderTraverse(btptr &root) //层序遍历

char \*formSeq(btptr &root, int pos, char seq[]) //返回完全二叉树顺序存储数组地址

int getEndFromSeq(char seq[]) // 通过完全二叉树顺序存储字符数组获得最后一个非空元素的位置

程序测试简要报告：

图形用户界面, 文本

中度可信度描述已自动生成

文本

低可信度描述已自动生成

文本

描述已自动生成

经过测试, 程序可以快速得到正确结果

源程序代码：

#include <iostream>

#include <iomanip>

#include <string>

#include <cstring>

#include <cmath>

#include "stack.h"

#include "queue.h"

using namespace std;

typedef char elemT;

typedef struct BtNode

{

    elemT data;

    struct BtNode \*lc, \*rc;

} BtNode, \*btptr;

void initBt(btptr &root) //初始化二叉树

{

    root->data = '\0';

    root->lc = root->rc = nullptr;

}

void assignBiTree(btptr &root, const string str) //通过前序str给二叉树赋值

{

    static int idx = 0;

    if (idx >= str.size())

        return;

    char ch = str[idx];

    if (ch == '#')

    {

        ++idx;

        root = new BtNode{0, nullptr, nullptr};

        root->data = '@';

        return;

    }

    else if (ch == '\n' || ch == ' ')

        return;

    else

    {

        root = new BtNode{0, nullptr, nullptr};

        root->data = ch, ++idx;

    }

    assignBiTree(root->lc, str);

    assignBiTree(root->rc, str);

}

void preOrderTraverse(btptr &root) //前序遍历

{

    if (!root || root->data == '@')

        return;

    printf("%c", root->data);

    preOrderTraverse(root->lc);

    preOrderTraverse(root->rc);

}

void inOrderTraverse(btptr &root) //中序遍历

{

    if (!root || root->data == '@')

        return;

    inOrderTraverse(root->lc);

    printf("%c", root->data);

    inOrderTraverse(root->rc);

}

void postOrderTraverse(btptr &root) //后序遍历

{

    if (!root || root->data == '@')

        return;

    postOrderTraverse(root->lc);

    postOrderTraverse(root->rc);

    printf("%c", root->data);

}

void layerOrderTraverse(btptr &root) //层序遍历

{

    if (!root)

        return;

    queue<btptr> q;

    q.enQueue(root);

    while (!q.isEmpty())

    {

        char e = q.deQueue()->data;

        if (e != '@')

            printf("%c", e);

        if (root->lc)

            layerOrderTraverse(root->lc);

        if (root->rc)

            layerOrderTraverse(root->rc);

    }

}

char \*formSeq(btptr &root, int pos, char seq[]) //返回完全二叉树顺序存储数组地址

{

    if (!root)

        return nullptr;

    queue<btptr> q;

    q.enQueue(root);

    while (!q.isEmpty())

    {

        char e = q.deQueue()->data;

        seq[pos - 1] = e;

        if (root->lc)

            formSeq(root->lc, 2 \* pos, seq);

        if (root->rc)

            formSeq(root->rc, 2 \* pos + 1, seq);

    }

    return seq;

}

int getEndFromSeq(char seq[]) // 通过完全二叉树顺序存储字符数组获得最后一个非空元素的位置

{

    int layer = 1, length = 1, idx = 0, pos = 0;

    while (1)

    {

        bool flag = 1;

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

            if (seq[idx++] != '@')

            {

                pos = idx;

                flag = 0;

            }

        if (flag)

            return pos;

        ++layer;

        idx = length;

        length = pow(2, layer) - 1;

    }

}

int main()

{

    string str;

    printf("请输入先序递归遍历字符序列:\n");

    cin >> str;

    btptr root = new BtNode{0, nullptr, nullptr};

    initBt(root);

    assignBiTree(root, str);

    printf("\nPreorder:\n");

    preOrderTraverse(root);

    printf("\nInorder:\n");

    inOrderTraverse(root);

    printf("\nPostorder:\n");

    postOrderTraverse(root);

    printf("\nLayerorder:\n");

    layerOrderTraverse(root);

    printf("\nSeqTraverse:\n");

    char seq[1000];

    memset(seq, '@', sizeof(seq));

    formSeq(root, 1, seq);

    int end = getEndFromSeq(seq);

    for (int i = 0; i < end; ++i)

        printf("%c", seq[i]);

    return 0;

}