《数据结构与算法》实验报告

题目：二叉树与表达式

分工：二叉树与表达式

姓名：张富利

学号：1507020329

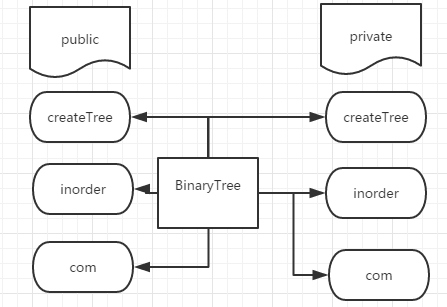
团队成员：张富利、杨强、陈子枫、王浩竣

2017年4月

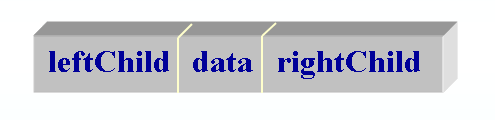
1. **分工任务描述**

整个实验分为建立二叉树、二叉树的中序遍历、计算求值等3个模块，其中，我具体负责全部模块，模块主要功能为通过二叉链表实现的表达式二叉树进行输出，同时计算出结果。

1. **逻辑结构设计**

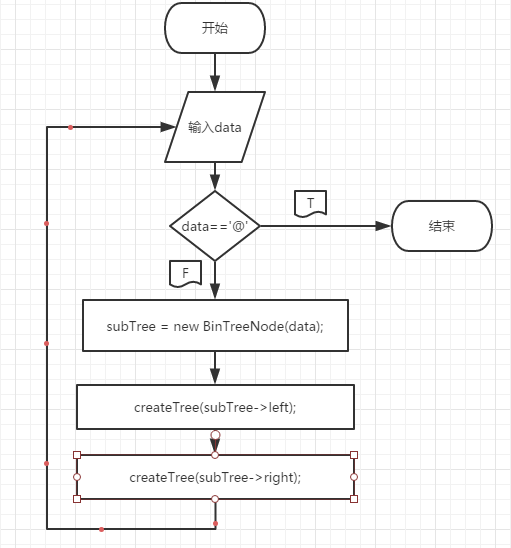


1. **物理结构设计**

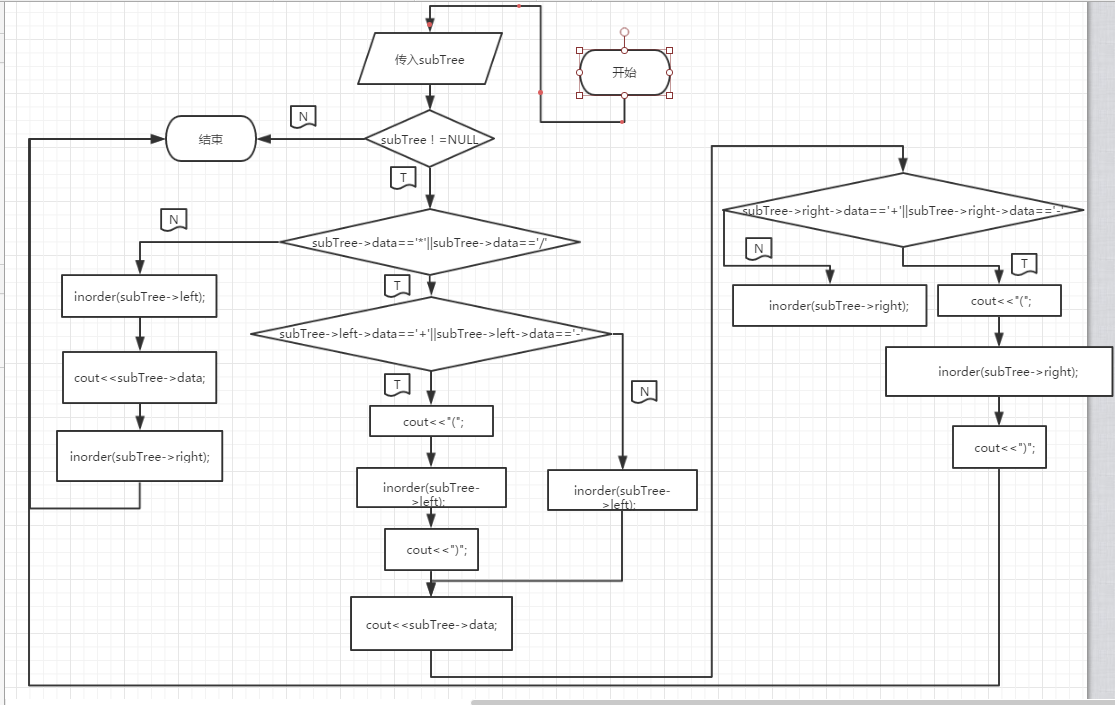


1. **程序流程图**

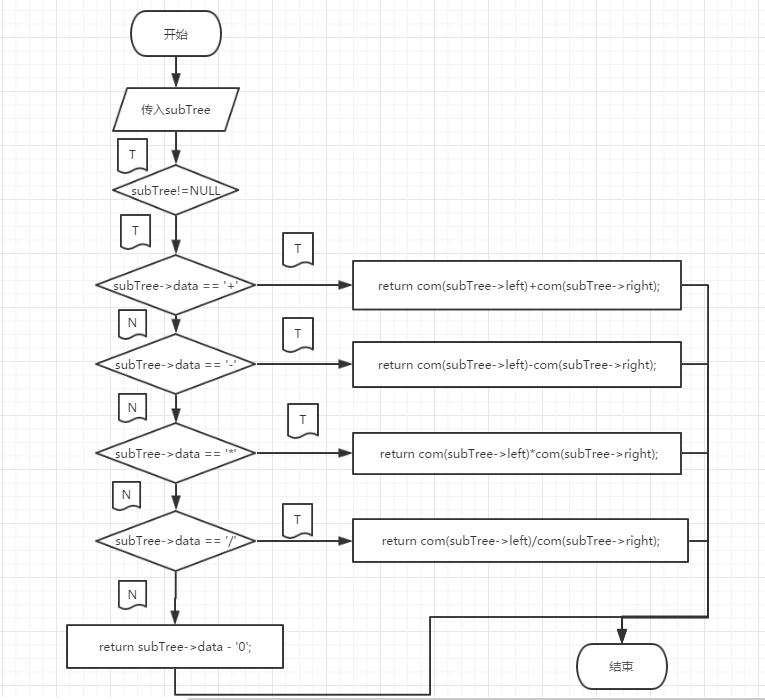
1.createTree

****

2.中序遍历



3.计算



1. **算法复杂度分析**
2. 建立二叉树

O(n) = nlogn

1. 中序遍历

O(n) = nlogn

1. 计算

O(n) = nlogn

1. **心得体会**

通过这次实验，初步掌握了二叉树的基本操作。尤其巩固了递归方面的知识，总结了递归方面的方法步骤，收获颇丰。

1. **附录**

#include <iostream>

using namespace std;

class BinaryTree;

class BinTreeNode{

friend class BinaryTree;

private:

char data;

BinTreeNode \*left;

BinTreeNode \*right;

public:

BinTreeNode(){left = NULL;right = NULL;}

BinTreeNode(char d,BinTreeNode \*l=NULL,BinTreeNode \*r=NULL){

data = d;

left = l;

right = r;

}

};

class BinaryTree{

public:

BinaryTree(){root = NULL;endTag='@';maxsize = 0;}

BinaryTree(char d){data = d;root = NULL;endTag='@';maxsize = 0;}

void createTree(){createTree(root);}

void inorder(){inorder(root);}

int com(){return com(root);}

private:

BinTreeNode \*root;

char endTag;

int data;

char kuo[10];

int maxsize;

void createTree(BinTreeNode \*&root);

void inorder(BinTreeNode \*root);

int com(BinTreeNode \*root);

};

void BinaryTree::createTree(BinTreeNode \*&subTree){

char data;

cin>>data;

if(data!=endTag){

subTree = new BinTreeNode(data);

createTree(subTree->left);

createTree(subTree->right);

}

}

void BinaryTree::inorder(BinTreeNode \*subTree){

if(subTree!=NULL){

if(subTree->data=='\*'||subTree->data=='/'){

if(subTree->left->data=='+'||subTree->left->data=='-'){

cout<<"(";

inorder(subTree->left);

cout<<")";

}else{

inorder(subTree->left);

}

cout<<subTree->data;

if(subTree->right->data=='+'||subTree->right->data=='-'){

cout<<"(";

inorder(subTree->right);

cout<<")";

}else{

inorder(subTree->right);

}

}else{

inorder(subTree->left);

cout<<subTree->data;

inorder(subTree->right);

}

}

}

int BinaryTree::com(BinTreeNode \*subTree){

if(subTree!=NULL){

if(subTree->data == '+'){

return com(subTree->left)+com(subTree->right);

}

else if(subTree->data == '-'){

return com(subTree->left)-com(subTree->right);

}

else if(subTree->data == '\*'){

return com(subTree->left)\*com(subTree->right);

}

else if(subTree->data == '/'){

return com(subTree->left)/com(subTree->right);

}

else{

return subTree->data - '0';

}

}

}

int main()

{

BinaryTree bt;

bt.createTree();

bt.inorder();

cout<<"="<<bt.com()<<endl;

return 0;

}