长　沙 　学　 院

课程设计报告

课程设计名称　数据结构与算法课程设计

系 （部）　 数学与计算机科学系

专 业 　 软件工程

班 级 　 15级软件5班

2016年 12月 13日

**目 录**

[一、引言 3](#_Toc469499729)

[1.1编写目的 3](#_Toc469499730)

[1.2课程设计任务 3](#_Toc469499731)

[**二、需求分析** 9](#_Toc469499732)

[2.1问题描述 9](#_Toc469499733)

[**2.2功能需求** 9](#_Toc469499734)

[**2.3界面需求** 9](#_Toc469499735)

[**2.4数据测试** 9](#_Toc469499736)

[**三、概要设计** 10](#_Toc469499737)

[3.1基本算法与分析设计 10](#_Toc469499738)

[3.2程序框架设计 10](#_Toc469499739)

[3.3界面设计 11](#_Toc469499740)

[**四、详细设计** 12](#_Toc469499741)

[4.1数据结构设计 12](#_Toc469499742)

[4.2功能模块设计 12](#_Toc469499743)

[4.2.1.初始数据插入模块 12](#_Toc469499744)

[4.2.2.二叉树生成模块 14](#_Toc469499745)

[4.2.3.插入数据模块 15](#_Toc469499746)

[4.2.4.二叉树遍历模块 16](#_Toc469499747)

[4.2.5.删除数据模块 16](#_Toc469499748)

[**五、软件测试** 19](#_Toc469499749)

[5.1功能测试 19](#_Toc469499750)

[**六、总结** 22](#_Toc469499751)

[**附录一：程序源代码** 24](#_Toc469499752)

[**附录二：程序源代码** 25](#_Toc469499753)

# 一、引言

## 1.1编写目的

1、巩固与加深对课程内容的理解。通过应用程序的设计过程，使学生加深对课程内容的理解，巩固所学的专业基础知识。

2、熟悉开发工具的使用。通过编制与上机调试程序，使学生逐步积累编制与调试程序的经验，促进实践能力提高。

3、培养与增强学生的综合运用能力。通过有一定实用价值的应用程序的开发，激发学生的学习兴趣，促进学生综合运用能力的提高。

4、培养与增强学生的自学能力和利用文献的能力。

## 1.2课程设计任务

**课程名称：**数据结构与算法课程设计

**课程编号：**JX020357

**主 笔 人：**何可可

**主 审 人：**陈治平

**1.课程设计的目的**

1 巩固与加深对课程内容的理解。通过应用程序的设计过程，使学生加深对课程内容的理解，巩固所学的专业基础知识。

2 熟悉开发工具的使用。通过编制与上机调试程序，使学生逐步积累编制与调试程序的经验，促进实践能力提高。

3 培养与增强学生的综合运用能力。通过有一定实用价值的应用程序的开发，激发学生的学习兴趣，促进学生综合运用能力的提高。

4培养与增强学生的自学能力和利用文献的能力。

**2.课程设计的题目**

为了充分调动学生的学习积极性与主动性，适应不同兴趣、不同程度的学生对课程设计的要求，本课程设计提供四个任选题。每个学生可以根据本人的兴趣及能力选择教师指定的选题，也可以自定其他的选题。

1、一元多项式计算问题

2、迷宫问题

3、利用二叉排序树对顺序表进行排序

4、交通咨询系统

5、内部排序算法的比较

**3.设计内容**

**1、一元多项式计算问题**

**问题描述：**

设计一个稀疏多项式简单计算器

**基本要求：**

(1) 输入并分别建立多项式A和B

(2) 输入输出多项式，输出形式为整数序列：n,c1,e1,c2,e2……，其中n是多项式的项数，ci和ei是第i项的系数和指数，序列按指数降序排列

(3) 完成两个多项式的相加、相减，并将结果输出；

**测试数据：**

(1) A+B A= 3x14-8x8+6x2+2 B=2x10+4x8+-6x2

(2) A-B A=11x14+3x10+2x8+10x6+5 B=2x14+3x8+5x6+7

(3) A+B A=x3+x1 B=-x3-x1

(4) A+B A=0 B=x7+x5+x3+x1

(5) A-B A=100x100+50x50+20x20+x B=10x100+10x50+10x20+x

**选作内容：**

(1) 多项式在x=1时的运算结果

(2) 求多项式A和B的乘积

**2、迷宫问题**

**问题描述：**

以一个m\*n的长方阵表示迷宫，0和1分别表示迷宫中的通路和障碍。迷宫问题要求求出从入口（1，1）到出口（m,n）的一条通路，或得出没有通路的结论。

**基本要求：**

首先实现一个以链表作存储结构的栈类型，然后编写一个求迷宫问题的非递归程序，求得的通路以三元组（i，j，d）的形式输出，其中：（i，j）指示迷宫中的一个坐标， d表示走到下一坐标的方向。

**测试数据：**

左上角（1，1）为入口，右下角（m,n）为出口。

**选作内容：**

(1) 编写递归形式的算法，求得迷宫中的所有可能的通路

(2) 以方阵的形式输出迷宫及其通路迷宫中的所有可能的通路

**3、利用二叉排序树对顺序表进行排序**

**问题描述：**

利用二叉排序树对顺序表进行排序。

**基本要求：**

(1) 生成一个顺序表L;

(2) 对所生成的顺序表L构造二叉排序树;

(3) 利用栈结构实现中序遍历二叉排序树;

(4) 中序遍历所构造的二叉排序树将记录由小到大输出。

**测试数据：**

用伪随机数产生程序产生，表长不小于20。

**选作内容：**

用实现二叉排序树的插入和删除操作。

**4、交通咨询系统**

**问题描述：**

设计一个交通咨询系统，为自驾游旅行者客咨询从任一个城市到另一个城市之间的最短路径问题。设计分三个部分，一是建立交通网络图的存储结构；二是解决单源最短路径问题；最后再实现两个城市顶点之间的最短路径问题。

**基本要求：**

(1) 对城市信息(城市名、城市间的里程)进行编辑：具备添加、修改、删除功能；

(2) 咨询以用户和计算机对话方式进行，要注意人机交互的屏幕界面。由用户选择输入起点、终点，输出信息：旅行者从起点、终点经过的每一座城市。

(3) 主程序可以有系统界面、菜单；也可用命令提示方式；选择功能模块执行，要求在程序运行过程中可以反复操作。

**测试数据：**

参考《数据结构（C语言版）》（严蔚敏 吴伟民编著）7.6节图7.33的交通图。

测试数据：北京到乌鲁木齐；北京到昆明；广州到哈尔滨；乌鲁木齐到南昌；沈阳到昆明。

**5、内部排序算法的比较**

**问题描述：**

通过随机数据比较各内部排序算法的关键字比较次数和关键字移动的次数，以取得直观感受。

**基本要求：**

(1) 待排序表的表长不小于100；

(2) 至少要用5组不同的输入数据作比较；

(3) 排序算法不少于5种；

(4) 最后要对结果作简单的分析。

**测试数据：**

用伪随机数产生程序产生。

**选作内容：**

对不同的表长做试验分析两个指标相对于表长变化关系。

**4.设计步骤**

**1、问题分析和任务定义**

根据设计题目的要求，充分地分析和理解问题，明确问题要求做什么？（而不是怎么做？）限制条件是什么？

**2、逻辑设计**

对问题描述中涉及的操作对象定义相应的数据类型，并按照以数据结构为中心的原则划分模块，定义主程序模块和各抽象数据类型。逻辑设计的结果应写出每个抽象数据类型的定义(包括数据结构的描述和每个基本操作的功能说明)，各个主要模块的算法，并画出模块之间的调用关系图；

**3、详细设计**

定义相应的存储结构并写出各函数的伪码算法。在这个过程中，要综合考虑系统功能，使得系统结构清晰、合理、简单和易于调试，抽象数据类型的实现尽可能做到数据封装，基本操作的规格说明尽可能明确具体。详细设计的结果是对数据结构和基本操作作出进一步的求精，写出数据存储结构的类型定义，写出函数形式的算法框架；

**4、程序编码**

把详细设计的结果进一步求精为程序设计语言程序。同时加入一些注解和断言，使程序中逻辑概念清楚；

**5、程序调试与测试**

采用自底向上，分模块进行，即先调试低层函数。能够熟练掌握调试工具的各种功能，设计测试数据确定疑点，通过修改程序来证实它或绕过它。调试正确后，认真整理源程序及其注释，形成格式和风格良好的源程序清单和结果；

1. **结果分析**

程序运行结果包括正确的输入及其输出结果和含有错误的输入及其输出结果。算法的时间、空间复杂性分析；

**7、撰写课程设计报告**

**5.设计要点**

1、问题分析和功能定义准确

2、数据结构定义合理

1. 关键算法描述清楚
2. 代码编写力求规范
3. 测试时注意边缘条件的测试

6、课程设计报告书按规范编写

**6.设计进度安排**

见课程设计工作计划表。

**7.考核标准**

**1、考核方式**

课程设计结束时，在机房当场验收。

(1) 教师检查运行结果是否正确。

(2) 学生回答教师提出的问题。

(3) 学生提交课程设计文档（A4纸打印）

**2、评分方法**

(1) 课程设计的成绩分为：优、良、中、及格、不及格五个等级。

(2) 评分标准：独立完成课程设计、并有所创新，作品有实用价值，评为优；独立完成课程设计、个性化特色明显，课程设计报告完成较好，评为良；按规定完成课程设计并提交成果，课程设计报告一般，评为中；按规定完成课程设计并提交成果，课程设计报告较差，评为及格；未提交报告者不及格。

**8.注意事项**

1、学生独立完成课程设计任务，杜绝拷贝抄袭

2、任务书下发后，学生不能无故更换课题

3、没有通过上机检查的学生其成绩直接记录不及格

4、答辩结束后给出最终成绩

**二、需求分析**

## 2.1问题描述

随机产生一串数据构造二叉排序树进行排序可以插入和删除，最后遍历输出。

**2.2功能需求**

分为三个功能：

1.对初始的随机数进行二叉排序

2.对二叉树进行插入操作

3.对二叉树进行删除操作

**2.3界面需求**

最开始要有插入随机数个数的界面，然后要有选择插入和删除的界面进行插入和删除的操作。

**2.4数据测试**

用伪随机数产生程序产生，表长不小于20。

**三、概要设计**

## 3.1基本算法与分析设计

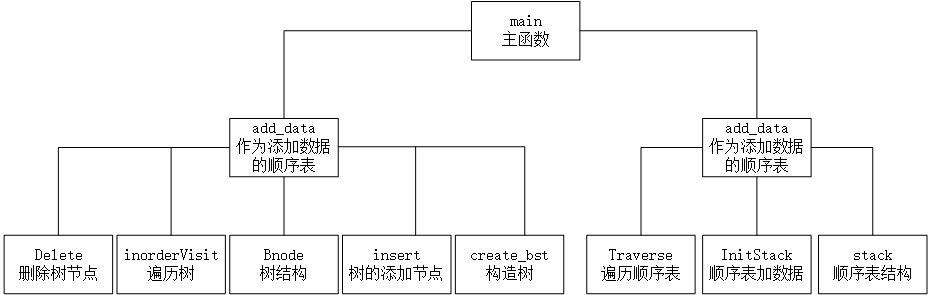
数据导入的时候只要输入数据个数，数据就会通过随机数生成来生成数据，生成的第一个随机数作为顺序表的头结点，然后每产生一个数据就执行一次顺序表的插入操作。最后用遍历的方法查看顺序表中的数据。

新建一个树的数据类型，定义一个T为树的根，然后遍历顺序表，每个数据先创建一个新的空间给空间里面的变量赋初始值，然后再插入这个空间在树之中，通过递归的方法实现。

插入节点的设计是先判断当前节点是否为空，是的话就直接插入到当前，如果不是就看要插入的数据和当前数据的大小关系，如果要插入的数据比当前数据小的话就继续往左遍历，如果要插入的数据比当前数据大的话就继续往右遍历，一直找到空为止。

删除节点的设计是先遍历找到这个数据和其父节点，找不到就返回空，找到后判断是不是只有右子树，也就是没有左子树，是的话就直接把当前的右孩子代替这个位置。在有左子树的情况下，如果左子树无左右孩子就把左子树代替当前节点，如果有孩子就循环找到其左子树中最大的数字用最大的数据代替要删除的数据。

## 3.2程序框架设计

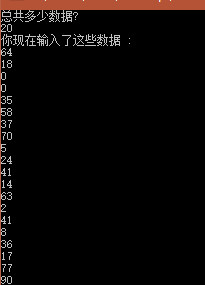


建立一个顺序表，然后建立一个二叉树。主要分顺序表模块和二叉树模块，顺序表用来储存初始的数据，二叉树用来排序和插入和删除和输出，用到的算法主要是中序遍历。

## 3.3界面设计

主要包括以下界面：

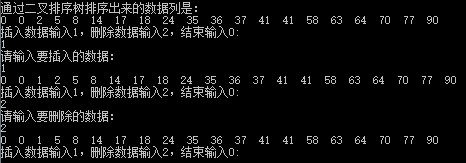
**1.初始数据插入界面**



**2.选择插入删除界面**



**3.插入和删除界面**



**四、详细设计**

## 4.1数据结构设计

**1.顺序表**

开始要确定顺序表要有多少个数据，然后根据数据的个数新建一个长度为length的空间然后通过随机数生成产生length个100以内的随机数然后没生成一个随机数就调用一次InitStack函数把数据加入到顺序表中最后调用一次Traverse函数查看当前的初始值有哪些。

**2.二叉树**

建立一个新的空间S用来储存要加入的节点，然后把数据赋值给S的data然后S的儿子都是空，然后调用二叉树的插入函数insert把S插进去。

## 4.2功能模块设计

### 4.2.1.初始数据插入模块

//建立顺序表的结构，在结构体中定义一个data用来储存数据

typedef struct stack{

int data;

}\*sqstack;

sqstack head = NULL,tail = NULL;

//向顺序表中添加数据把data传入进来尾指针向前移动一位然后把data加入到链表中

void InitStack(int data){

tail++;

tail -> data = data;

}

//输出顺序表中的数据定义一个p指针用来遍历顺序表从头开始通过for循环从头到尾部循环一次就输出一次

void Traverse(){

sqstack p = head;

for(;p != tail+1;p++){

printf("%d \n",p -> data);

}

}

//输入数据生成顺序表然后遍历查看

//开始要确定顺序表要有多少个数据，然后根据数据的个数新建一个长度为length的空间然后通过随机数生成产生length个100以内的随机数然后没生成一个随机数就调用一次InitStack函数把数据加入到顺序表中最后调用一次Traverse函数查看当前的初始值有哪些。

void add\_data()

{

srand((unsigned) time(NULL));//用时间做种，每次产生随机数不一样

int length,data;

int i;

printf("总共多少数据?\n");

scanf("%d",&length);//确定好顺序表初始长度

head = (sqstack)malloc(length\*sizeof(struct stack));//新建一个长度为length的空间

tail = head;

// printf("输入 %d 个数据:\n", length);

//scanf("%d",&data);

data = rand() % 101; //产生0-100的随机数

head -> data = data;//先给他加第一个数据就进去

for (i = 1; i < length; i++)

{

data = rand() % 101; //产生0-100的随机数

//scanf("%d",&data);

InitStack(data);

}

printf("你现在输入了这些数据 :\n");

Traverse();

}

### 4.2.2.二叉树生成模块

//建立一个树的结构体包括数据data左子树的地址lchild和右子树的地址rchild

typedef struct Bnode{

int data;

struct Bnode \*lchild;

struct Bnode \*rchild;

}Bnode,\*Tree;

//遍历线性表为生成树做准备

//定义一个p用来遍历顺序表然后简历使用for函数来先取出线性表中的一个数据然后建立一个新的空间S用来储存要加入的节点，然后把数据赋值给S的data然后S的儿子都是空，然后调用二叉树的插入函数insert把S插进去。

void create\_bst(Tree &T){

sqstack p = head;//遍历线性表

for( ; p!=tail+1 ; p++ ){

Tree S = (Tree)malloc(sizeof(struct Bnode));//新建一个空间用来储存树的一个节点

S -> data = p -> data;//补充好节点的数据

S -> lchild = NULL;

S -> rchild = NULL;

insert(T,S);

//printf("\*\*%d\*\*",T -> data);

}

}

### 4.2.3.插入数据模块

//节点插入

//先判断当前的节点是否为空的，如果是空的就把要加入的节点代替当前找到的节点，如果当前找到的节点比要加入的节点要大就递归当前函数去向它的左节点移动。如果当前找到的节点比要加入的节点要小就递归当前函数去向它的右节点移动。一直递归到找到的节点为空的时候再替代。

void insert(Tree &T,Tree S){//T为当前找到的根节点//S为要加入的节点

if(T == NULL)//如果当前节点是空的话S就变成当前叶子节点

{

T = S;

//printf("0-%d-0",T -> data);

}

else if(S -> data < T -> data)//要加的数据比当前节点数据小的话就继续向左遍历

{

insert(T -> lchild,S);

//printf("L");

}

else

{

insert(T -> rchild,S);

//printf("R");

}

}

### 4.2.4.二叉树遍历模块

//由于这是二叉排序树所以就要按照中序遍历来输出二叉树，首先把根节点T引入来然后先找T的左节点一直找到空然后找右节点

void inorderVisit(Tree T){ //中序遍历

if(T != NULL){

inorderVisit(T->lchild);

printf("%d ",T->data);

inorderVisit(T->rchild);

}

}

### 4.2.5.删除数据模块

//引入跟节点T和要删除的节点x，首先用一个while循环去找到data为x的节点，如果比当前找到的大就向左边移动如果小的话就向右边移动，在最后为空或者找到的时候就退出，如果为空就说明没有找到然后就退出函数，如果找到了就用p记录找到的地方，f记录p的双亲。然后就开始把找到的删除，首先如果要删除的节点没有左节点就是只有右节点的就直接把p的右孩子代替p就可以了，如果在有左节点的情况下，定义一个q为p的左子树中最大的节点，s为q的双亲，然后判断如果p的左子树无左右孩子的话，就用q来代替p然后释放之前的p，如果p有左右孩子就用p的左孩子中最大的q来代替p，然后释放p就完成删除了

void Delete(Tree T,int x){ //删除节点

Tree f,p = T;

while(p && p->data !=x){ //去找到data为x的节点 //在为空或者找到就退出

if(p->data > x){

f = p; //f记录的是p的双亲 //p为找到的节点

p = p->lchild;

}

else{

f = p;

p = p->rchild;

}

}

if(p == NULL) return; //表示没找到

if(p->lchild == NULL){ //只有右子树的话

if(f->lchild == p) //直接把p的右孩子代替p

f->lchild = p->rchild;

else

f->rchild = p->rchild;

}

else{ //结点有左子树

Tree q = p->lchild;

Tree s = q;

while(q->rchild != NULL){ //q为p的左子树中最大的//s为q的双亲

s = q;

q = q->rchild;

}

if(s == p->lchild){//p的左子树无左右孩子//如果上面循环没有动s也是p的左子树

p->lchild = s->lchild;

p->data = s -> data;

free(s);

}

else{

p->data = q->data;

s->rchild = q->lchild;

free(q);

}

}

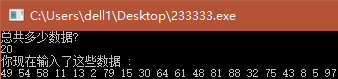
}

**五、软件测试**

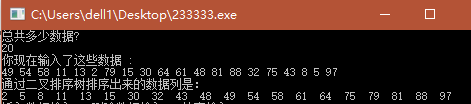
## 5.1功能测试

1.生成初始线性表功能

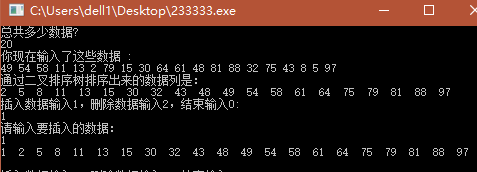
生成20个随机数并输出

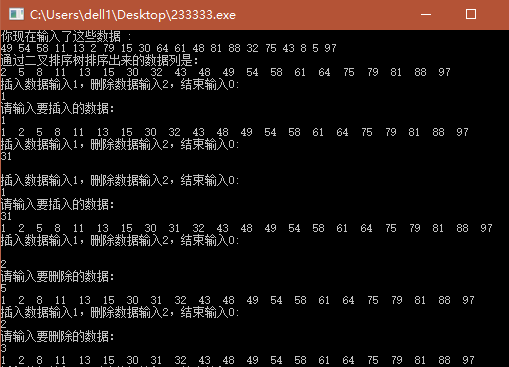


2.对初始的随机数进行二叉排序

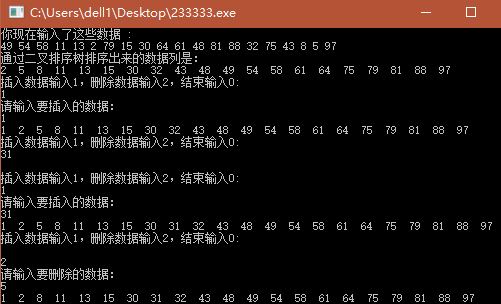


3.对二叉树进行插入操作

插入一个1到二叉树中并排序好

插入一个31到二叉树中并排序好

4.对二叉树进行删除操作

删除数据5并输出

删除数据3，在二叉树中没有3这个数据所以就不做操作就输出

**六、总结**

这几天做出的一个简单的课程设计。我想这对于自己以后的学习会有很大的帮助。在这次设计中遇到了很多实际性的问题，以前的学习的时候感觉老师讲了就懂了，最后没想到还是不懂所以学习程序还是要多动手自己写写，书本上理论性的东西与在实际运用中的还是有一定的出入的，所以有些问题不但要深入地理解，而且要不断地更正以前的错误思维。

对于我写的这个这个二叉排序树，其程序是比较简单的，用到的知识有顺序表和树的知识。

**1、写顺序表的时候出现的问题：**

在之前写顺序表的时候一直都出现了内存错误的问题，之前的题目都没有太在意，但是这次又出现了，于是我准备找到问题到底是出在哪里，自己重新写了一遍顺序表的发现内存还是错了，然后准备一个地方一个地方的检查，开始我把顺序表的遍历给注释掉，发现问题还在，然后随便插入几个数据输出看看，发现问题还在但是有数据输出，这说明我定义结构体没有错，最后就只剩创建空间加数据了最后发现我输入两三个数据就是对的，输入多了就出现内存错误，所以说加入数据的时候没有错，当我没看出错误去问大神的时候他一眼就发现了错误，原来是申请内存的时候错了，之前一直以为顺序表只要新建一个空间添加的时候用尾部向后移一个就可以了，但是实际上你开始要确定你顺序表的初始空间大小在你定义的空间之内才能往后移动尾节点如果你还想多加空间的话你需要用到realloc函数加。

**2、写二叉排序数的时候出现的问题：**

在写二叉树的时候把插入和遍历写完，运行发现没有报错，但是就是什么也没输出，于是就判断出不是二叉树插入的时候错了就是遍历的时候错了，两个之中遍历简单一些就先检查遍历，自己手动制作一个简单的树然后遍历发现遍历是没问题的，所以只能是插入的时候错了，于是在插入的时候在插入成功的地方输出当前数据，在向左边遍历是输出一个L在向右变量时输出一个R最后发现是变量的赋值的时候错了。

**附录一：程序源代码**

《数据结构》（C语言版） 严蔚敏 清华大学出版社

**附录二：程序源代码**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

typedef struct Bnode{

int data;

struct Bnode \*lchild;

struct Bnode \*rchild;

}Bnode,\*Tree;

typedef struct stack{

int data;

}\*sqstack;

sqstack head = NULL,tail = NULL;

//向顺序表中添加数据

void InitStack(int data){

tail++;

tail -> data = data;

}

//输出顺序表中的数据

void Traverse(){

sqstack p = head;

for(;p != tail+1;p++){

printf("%d \n",p -> data);

}

}

//输入数据生成顺序表然后遍历查看

//之前一直以为顺序表只要新建一个空间添加的时候用尾部向后移一个就可以了，

//但是实际上你开始要确定你顺序表的初始空间大小在你定义的空间之内才能往后移动尾节点

//如果你还想多加空间的话你需要用到realloc函数加

void add\_data()

{

srand((unsigned) time(NULL));//用时间做种，每次产生随机数不一样

int length,data;

int i;

printf("总共多少数据?\n");

scanf("%d",&length);//确定好顺序表初始长度

head = (sqstack)malloc(length\*sizeof(struct stack));//新建一个长度为length的空间

tail = head;

printf("输入 %d 个数据:\n", length);

//scanf("%d",&data);

data = rand() % 101; //产生0-100的随机数

head -> data = data;//先给他加第一个数据就进去

for (i = 1; i < length; i++)

{

data = rand() % 101; //产生0-100的随机数

//scanf("%d",&data);

InitStack(data);

}

printf("你现在输入了这些数据 :\n");

Traverse();

}

//树的生成

void insert(Tree &T,Tree S){//T为当前找到的根节点//S为要加入的节点

if(T == NULL)//如果当前节点是空的话S就变成当前叶子节点

{

T = S;

//printf("0-%d-0",T -> data);

}

else if(S -> data < T -> data)//要加的数据比当前节点数据小的话就继续向左遍历

{

insert(T -> lchild,S);

//printf("L");

}

else

{

insert(T -> rchild,S);

//printf("R");

}

}

//遍历线性表为生成树做准备

void create\_bst(Tree &T){

sqstack p = head;//遍历线性表

for( ; p!=tail+1 ; p++ ){

Tree S = (Tree)malloc(sizeof(struct Bnode));//新建一个空间用来储存树的一个节点

S -> data = p -> data;//补充好节点的数据

S -> lchild = NULL;

S -> rchild = NULL;

insert(T,S);

//printf("\*\*%d\*\*",T -> data);

}

}

void inorderVisit(Tree T){ //中序遍历

if(T != NULL){

inorderVisit(T->lchild);

printf("%d ",T->data);

inorderVisit(T->rchild);

}

}

void Delete(Tree T,int x){ //删除节点

Tree f,p = T;

while(p && p->data !=x){ //去找到data为x的节点 //在为空或者找到就退出

if(p->data > x){

f = p; //f记录的是p的双亲 //p为找到的节点

p = p->lchild;

}

else{

f = p;

p = p->rchild;

}

}

if(p == NULL) return; //表示没找到

if(p->lchild == NULL){ //只有右子树的话

if(f->lchild == p) //直接把p的右孩子代替p

f->lchild = p->rchild;

else

f->rchild = p->rchild;

}

else{ //结点有左子树

Tree q = p->lchild;

Tree s = q;

while(q->rchild != NULL){ //q为p的左子树中最大的//s为q的双亲

s = q;

q = q->rchild;

}

if(s == p->lchild){//p的左子树无左右孩子//如果上面循环没有动s也是p的左子树

p->lchild = s->lchild;

p->data = s -> data;

free(s);

}

else{

p->data = q->data;

s->rchild = q->lchild;

free(q);

}

}

}

int main()

{

int i = -1,data;

Tree T = NULL;//T为最上面的根节点开始为空

add\_data();//用顺序表储存数据

create\_bst(T);//建立二叉排序树

//printf("!!!");

printf("通过二叉排序树排序出来的数据列是：\n");

inorderVisit(T);//中序遍历二叉排序树，从小到大输

for(; i!=0 ;){

printf("\n插入数据输入1，删除数据输入2，结束输入0:\n");

scanf("%d",&i);

if(i == 1){

printf("请输入要插入的数据：\n");

scanf("%d",&data);

Tree S = (Tree)malloc(sizeof(struct Bnode));

S -> data = data;

S -> lchild = NULL;

S -> rchild = NULL;

insert(T,S);

inorderVisit(T);

} else if (i == 2){

printf("请输入要删除的数据：\n");

scanf("%d",&data);

Delete(T,data);

inorderVisit(T);

}

}

return 0;

}