**一、【实验标题】**

**实验5-查找（字符统计程序）**

**二、【实验目的】**

1.掌握查找定义。

2.掌握二叉排序树的构造。

3.应用二叉排序树进行查找。

4.使用C/C++语言和查找实现“字符串统计程序”专题。

**三、【实验环境】**

硬件: Intel®Core™i5-6500 CPU3.20GHz,8GB内存

软件: Windows 7 64位,Microsoft Visual Studio 2010

**四、【实验内容】**

编写一个控制台的程序，可以对用户任意输入的字符串中的字符进行统计，统计出各种字符、标点、数字出现的次数。根据用户输入的字符，构建一个二叉排序树的动态查找表，key为字符的ASCLII码，count值为该字符出现的次数。

1. 字符串的输入
2. 构建二叉排序树。
3. 删除空格字符。
4. 二叉树结构输出。
5. 二叉树的遍历。

**五、【实验代码】**

//该程序实现的功能:

//1.画图

//2.排序二叉树输入显示

//3.删除结点（必须为带孩子结点，要考虑删除根节点的情况）

//4.排序二叉树输出显示

//5.字符统计功能

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N=1e5+5,M=2e4+5,inf=0x3f3f3f3f,mod=1e9+7;

#define mst(a,b) memset(a,b,sizeof a)

#define PII pair<int,int>

#define fi first

#define se second

#define pb push\_back

struct node{ //结点结构体

char ch; //字符

int v; //用于统计字符个数

node \*l,\*r; //结点的左右儿子

};

node\* search(node \*x,node \*fa,char key){ //查找值为key的结点是否在树中数出现过

if(!x){

return NULL;

}

else if(x->ch==key) return x;

else if(x->ch<key) return search(x->l,x,key);

else return search(x->r,x,key);

}

node\* build(node \*x,char key){ //创建二叉排序树

node \*tmp =search(x,NULL,key);

if(tmp==NULL){ //为空则新建一个结点

if(!x){

x=new node();

x->ch=key;

x->v=1;

x->l=x->r=NULL;

}

else if(key<=x->ch) x->l=build(x->l,key); //向左子树插入

else x->r=build(x->r,key); //向右子树插入

}

else { //如果出现过直接次数+1

tmp->v++;

}

return x;

}

node \* del(node \*x,char key){ //删除一个结点

node \*p,\*fa;

p=x,fa=NULL;

while(p){

if(p->ch==key) break; //如果找到了直接跳出

fa=p;

if(p->ch>=key) p=p->l;

else p=p->r;

}

if(p==NULL) return x;

if(p->l==NULL){

if(fa==NULL) x=p->r;

else if(fa->l==p) fa->l=p->r;

else fa->r=p->r;

free(p);

}

else {

node \*q=p;

node \*s=p->l;//此时将p赋值为q，s为p的左子树，则q为s的双亲

while(s->r)//查找p的左子树中查找最右下结点——最大结点

{

q=s;

s=s->r;//s为左子树最大结点，q为左子树最大节点的父母结点

}

if(q==p)//用来判断p结点的左子树是否有右子树的

q->l=s->l;//s为p左子树的最右子树，所以它没有右子树了

else q->r=s->l;

p->ch=s->ch;

free(s);

}

return x;

}

void P(int f){ //用于判断是左儿子还是右儿子

if(f==-1) putchar('L');

else if(f==1) putchar('R');

}

void fun(node \*x,int sp,int f){ //打印二叉树的结构

if(!x) return;

for(int i=0;i<sp;i++) printf(" ");

P(f);printf("[%c:%d]\n",x->ch,x->v);

fun(x->l,sp+2,-1);

fun(x->r,sp+2,1);

}

void in(node \*x){ //中序遍历

if(!x) return;

in(x->l);

printf("[%c,%d]\n",x->ch,x->v);

in(x->r);

}

void pre(node \*x){ //前序遍历

if(!x) return;

printf("[%c,%d]\n",x->ch,x->v);

in(x->l);

in(x->r);

}

void post(node \*x){ //后序遍历

if(!x) return;

in(x->l);

in(x->r);

printf("[%c,%d]\n",x->ch,x->v);

}

int main(){

printf("统计输入文本中各种字符出现的次数\n");

printf("--------------------------------------\n");

node\* rt=NULL;

char ch;

while((ch=getchar())!='\n'){

rt=build(rt,ch);

}

printf("--------------------------------------\n");

rt=del(rt,'y');

fun(rt,0,0);

printf("\n---------\n");

printf("1、前序遍历输出\n");

printf("2、中序遍历输出\n");

printf("3、后序遍历输出\n");

printf("--------------------------------------\n");

printf("请选择菜单项: ");

int op;scanf("%d",&op);

if(op==1){

pre(rt);

}else if(op==2) in(rt);

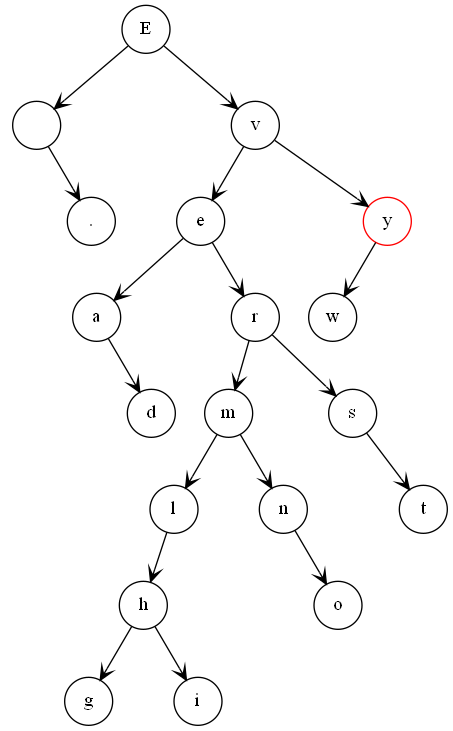
else post(rt);

return 0;

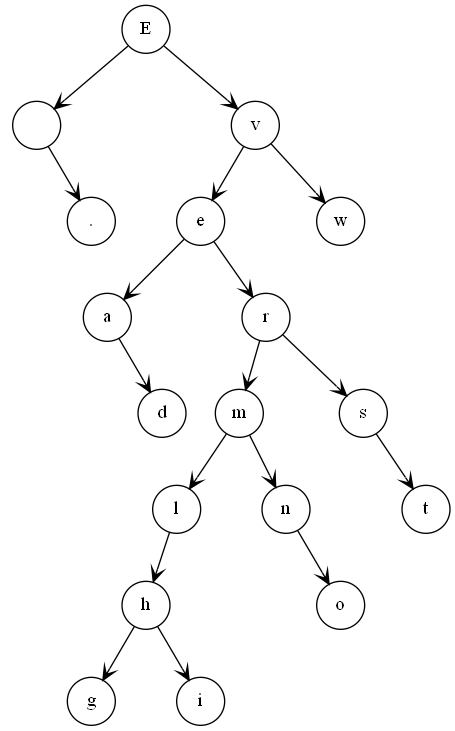
}

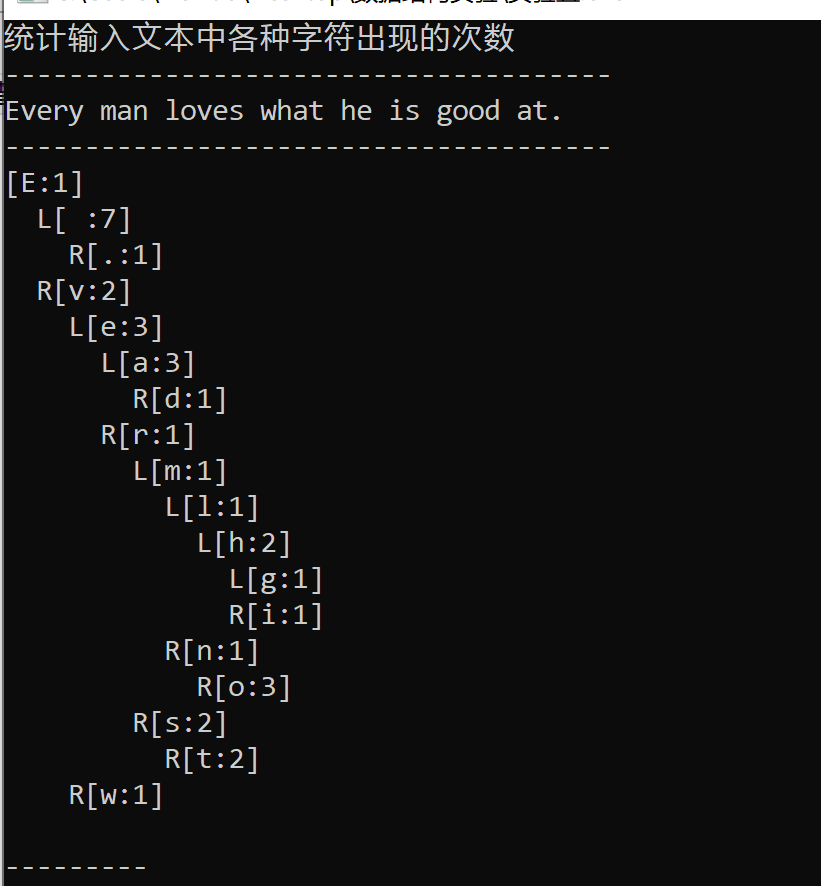
**六、【实验运行结果】**

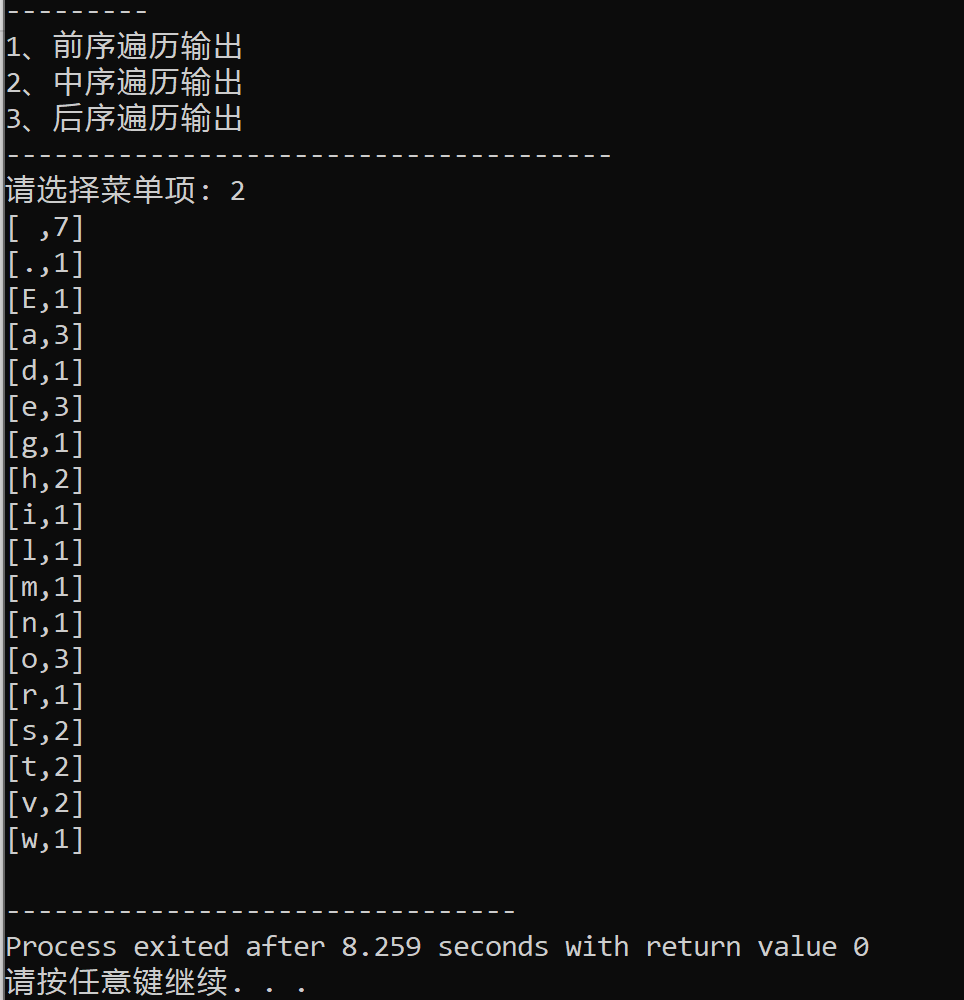
**1.删除y结点前的二叉排序树**



**2.删除y结点后的二叉树**







**七、【实验小结】**

**本次实验我学会了**查找的定义和具体内容，熟练地掌握了二叉排序树的构造，学会了如何判断一个结点是否在二叉树中出现过，并深刻地理解了如何利用二叉排序树进行查找需要的结点。掌握了如何使用递归和指针来实现二叉树的相关功能。