卫星搜索树

卫星成像技术可以识别每一棵树的种类，输入观测到的每棵树的英文名称，请编写程序输出每种树的数量和所占比例。

实验目的

1. 巩固树的相关知识，懂得如何使用树来解决生活中的问题。
2. 学习如何创建一颗树，如何遍历和搜索一颗树。

实验内容

用户输入树的名称，最后由程序统计每一颗树出现的数量和所占的比例，模拟卫星成像技术。

实验输入和输出说明

用户先输入树的总数，再分别输入每一颗树的名字，最后程序输出每一颗树的数量以及所占的比例。

示例输入:

```
Please enter the number of all trees:
8
🌵
🎄
🌺
🌵
🍁
🍁
🍁
🌵
```

示例输出:

```
The number of 🌵 tree is 3, Its proportion is 37.50%
The number of 🌺 tree is 1, Its proportion is 12.50%
The number of 🍁 tree is 3, Its proportion is 37.50%
The number of 🎄 tree is 1, Its proportion is 12.50%
```

解题思路

这一道题比较简单，只是一个有关树的创建和树的遍历有关的题目，在这里，在声明树的结构体的时候，在结构体里面可以包含树的种类名称$name$、树的个数$count$、左子树$lchild$和右子树$rchild$，如果遇到相同名字的树可以直接递增$count$,如果是新的树，就新建一个子树，并将该种类树的个数$count$初始化为1即可。

在输入方面，我们声明一个变量n来储存用户输入的所有树的个数，接着循环n次，依次输入树的名字，在通过判断该种类树是否已经存在来添加树的个数。这里有一点想要注意，如果输入的树的名字不包含空格，那么可以直接使用

```
int  scanf(const char * __restrict, ...) __scanflike(1, 2);
```

来进行输入，但是有些树的名字包含空格，我们可以使用

```
char  *fgets(char * __restrict, int, FILE *);
```

来输入我们树的名字。

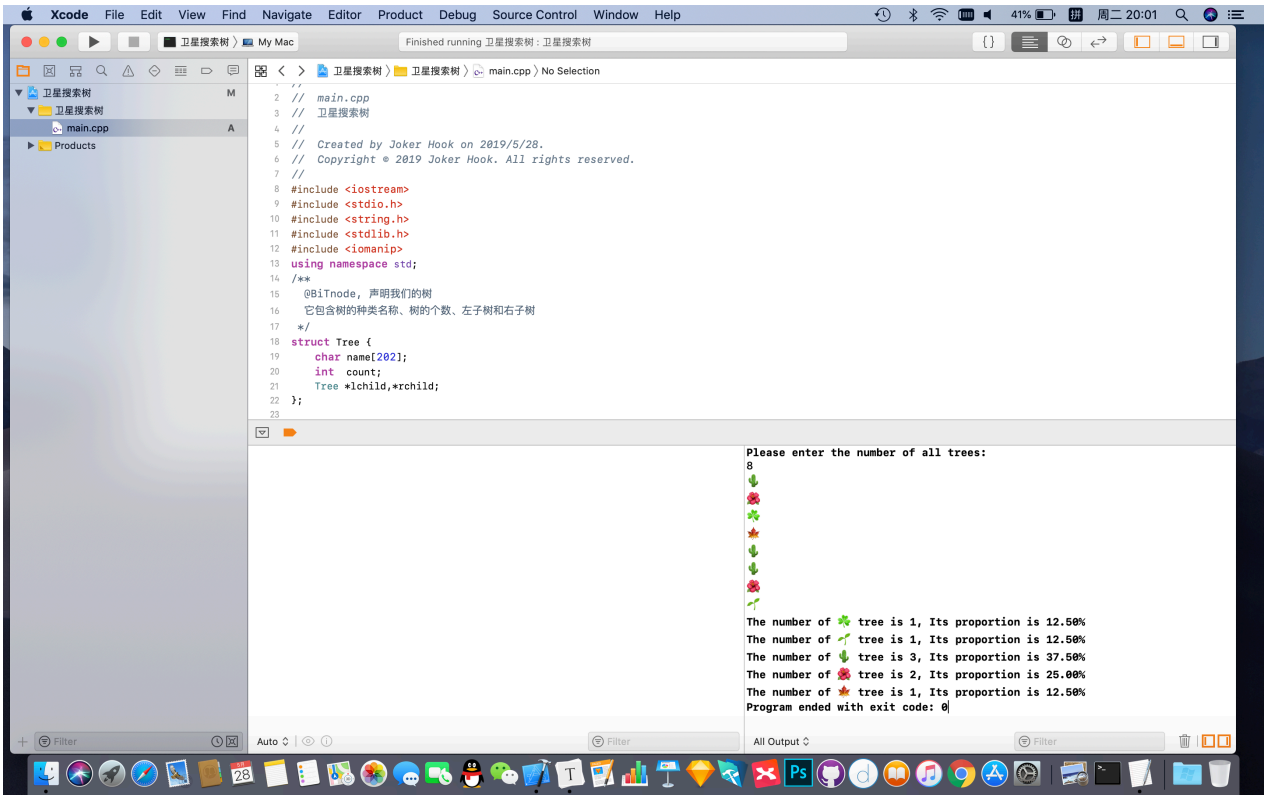
实验代码及注释

```
//
//  main.cpp
```

```
//  卫星搜索树
//
//  Created by Joker Hook on 2019/5/28.
//  Copyright © 2019 Joker Hook. All rights reserved.
//
#include <iostream>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iomanip>
using namespace std;
/**
  @BiTnode, 声明我们的树
  它包含树的种类名称、树的个数、左子树和右子树
 */
struct Tree {
    char name[202];
    int  count;
    Tree *lchild,*rchild;
};

// n 代表用户输入的树的个数
int n;

/**
  @CreatBitree(BiTnode*,char*)
  创建我们的树，如果新添加的树在已有的种类里面找不到，就重新添加一个子树
 */
Tree * CreatTree(Tree * root,char *s) {
    if(!root) {
        root = new Tree;
        root->lchild=NULL;
        root->rchild=NULL;
        strcpy(root->name,s);
        root->count=1;
    } else {
        int cmp=strcmp(root->name,s);
        if(cmp>0) root->lchild=CreatTree(root->lchild,s);
        else if(cmp<0) root->rchild=CreatTree(root->rchild,s);
        else root->count++;
    }
    return root;
}

/**
  @InOrder(BiTnode*)
  树的中序遍历，采用递归的方法完成
 */
void InOrder(Tree *root) {
```

```
    if(root) {
        InOrder(root->lchild);
        printf("The number of %s tree is %d, Its proportion is
%.2lf%c\n",root->name,root->count,root->count*100.0/n,'%');
        InOrder(root->rchild);
    }
}

int main() {
    // str数组用来存储用户写的树的名字
    char str[202];
    Tree *root;
    root = NULL;
    printf("Please enter the number of all trees: \n");
    // 用户写下树的总个数
    scanf("%d\n",&n);
    for(int i=0;i<n;i++) {
        /**
         此处使用scanf()来输入树的名字
         如果输入的名字有包含空格，可以使用fgets(str, 202, stdin);
         */
        scanf("%s",str);
        // 循环是为了将树名字统一转换为小写
        for(int j=0;str[j];j++)
            if(str[j]>='A' && str[j]<='Z')
                str[j]+=32;
        root=CreatTree(root,str);
    }
    InOrder(root);
    return 0;
}
```

实验运行结果截图



实验收获总结

1. 二叉树支持动态的插入和查找，保证操作在$O(height)$时间，这就是完成了哈希表不便完成的工作，动态性。但是二叉树有可能出现最糟糕的情况，但是如果输入序列已经排序，则时间复杂度为$O(N)$。给定值的比较次数等于给定值节点在二叉排序树中的层数。如果二叉排序树是平衡的，则n个节点的二叉排序树的高度为$\log 2n+1$,其查找效率为$O(\log 2n)$，近似于折半查找。如果二叉排序树完全不平衡，则其深度可达到n，查找效率为$O(n)$，退化为顺序查找。一般的，二叉排序树的查找性能在$O(\log 2n)$到$O(n)$之间。因此，为了获得较好的查找性能，就要构造一棵平衡的二叉排序树。
2. 在输入方面可以结合情况选择$scanf()$或者$fgets()$。
3. 实不相瞒，本次实验课我了解了如何让C语言输出特殊字符如🍎🍐🍋🍊等😅。