哈尔滨工业大学计算机科学与技术学院

实验报告

课程名称： 数据结构与算法

课程类型：必修

实验项目名称：数据结构与算法实验

实验题目：BST 存储结构建立（插入）、删除、查找算法的实现及应用

班级：1403106
学号：1140310606
姓名：张茗帅

设计成绩	报告成绩	指导老师

一、实验目的

本实验要求编写程序实现 BST 存储结构的建立（插入）、删除、查找及应用。

二、实验要求及实验环境

实验要求

1.设计 BST 的左右链存储结构；

2.实现 BST 左右链存储结构上的插入（建立）、删除、查找和排序算法。

3.利用 BST 结构和相应的操作算法，实现班级学习成绩管理（单科成绩管理，排名；加权绩点管理与排名等）

4.学生的基础成绩信息以文件形式保存；学生基础成绩信息和排名信息以文件

形式存储；并能显示到屏幕。

实验环境

CodeBlocks

三、设计思想（本程序中的用到的所有数据类型的定义，主程序的流程图及各程序模块之间的调用关系）

逻辑设计

查找：

通过用户输入学生学号来查找学生的成绩，我采用对树进行遍历，当找到一个结点的学号信息与所输入的匹配时，即查找成功。

插入：

首先找到该学生在已经存在的二叉排序树中的合理位置，然后进行插入。第一个插入的数据为树的根。

删除：

先找到删除的结点位置，然后根据结点 p 的三种情况进行分析（p 没有孩子，p 有左孩子或右孩子，p 有左孩子和右孩子），找出相应的插入方法。

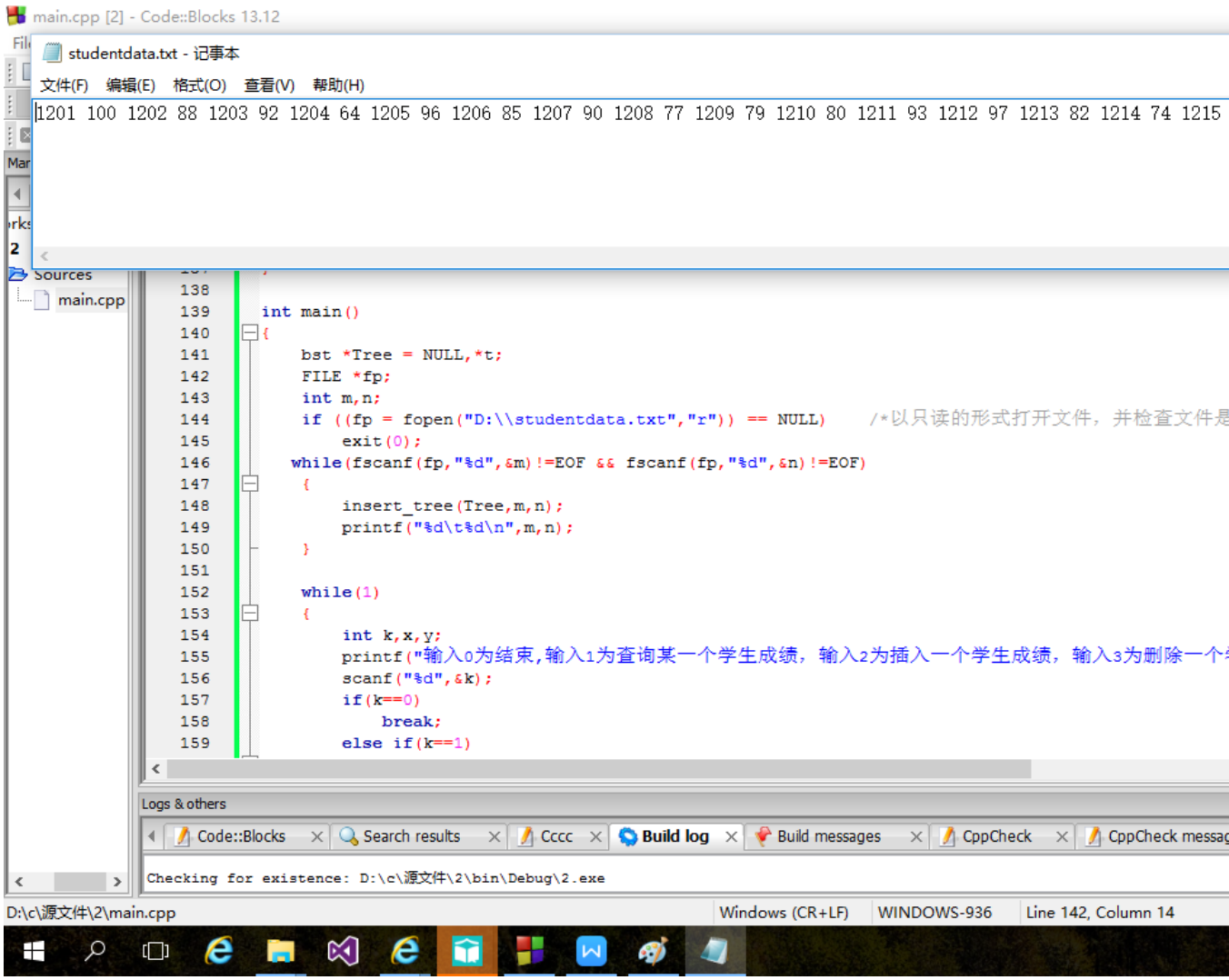
。

物理设计

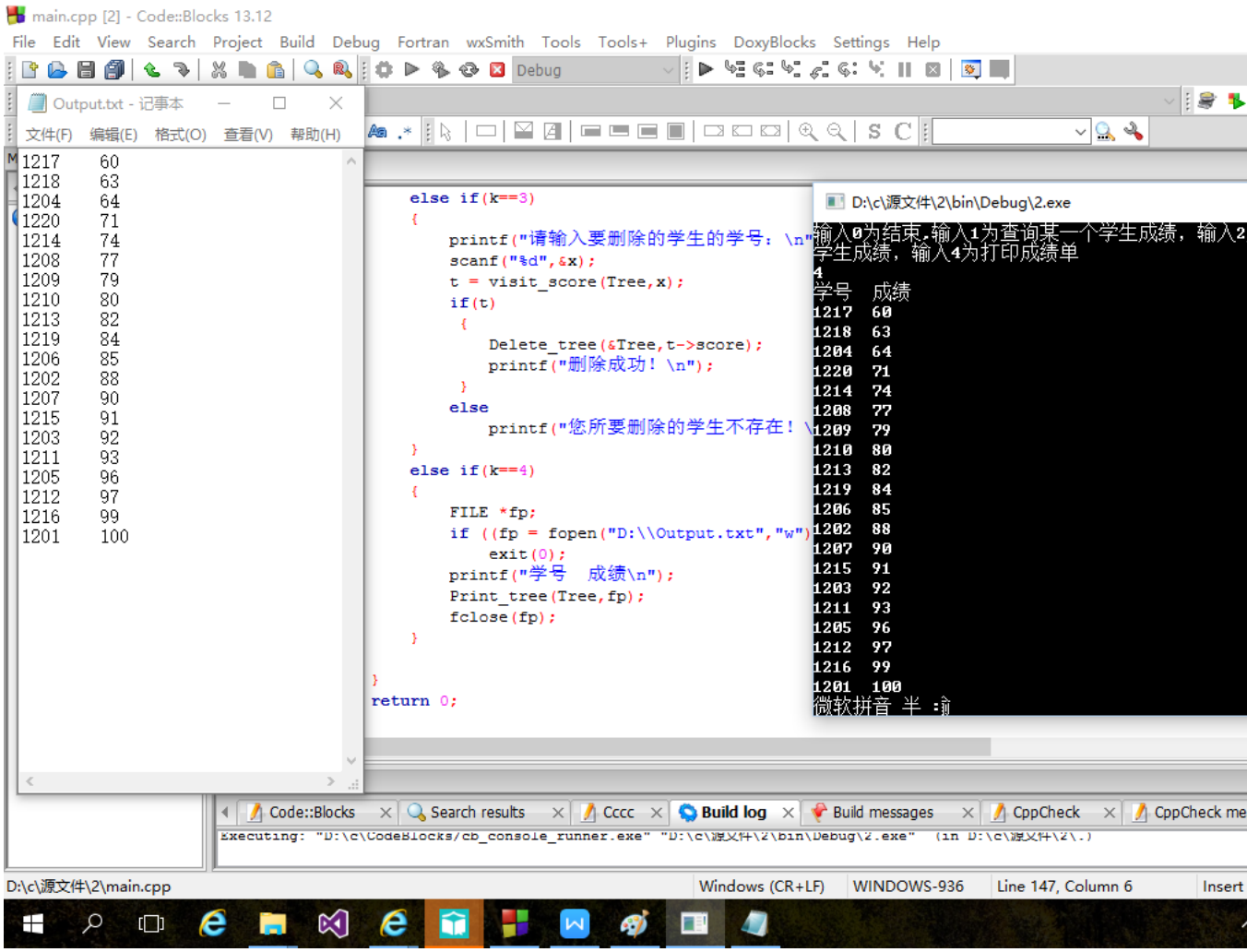
学生的信息保存在树的每一个结点中，每一个学生的信息包括学号和成绩。　本次实验以学生的成绩为键值存储在排序二叉树中

四、测试结果

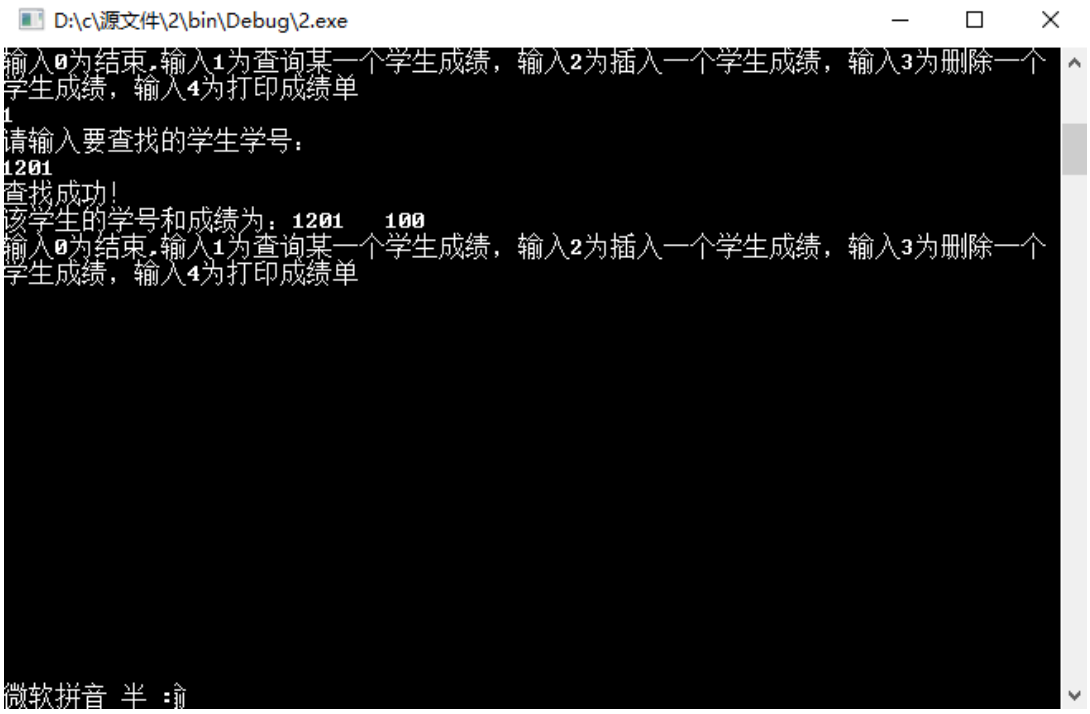
从文件读取数据如下图：



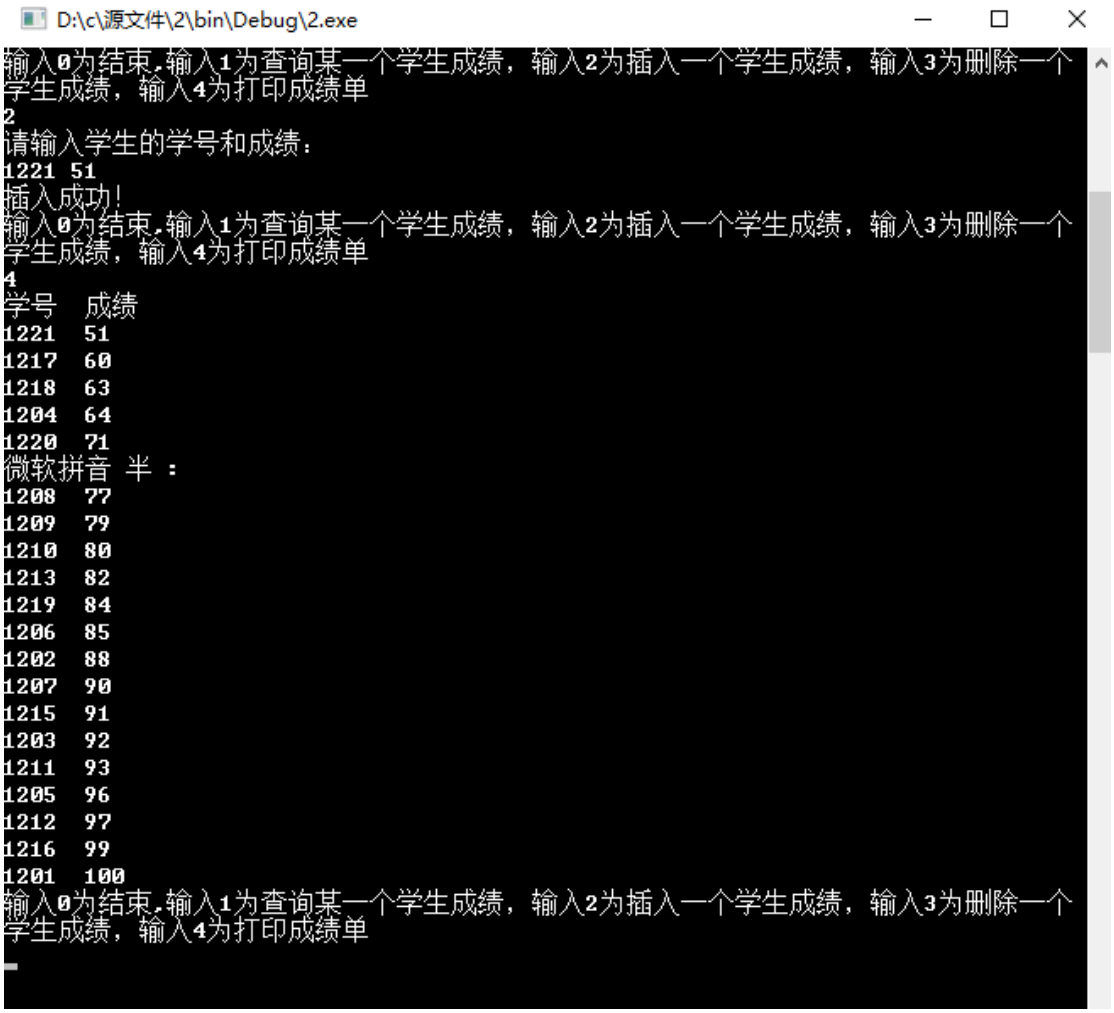
打印的成绩单和保存在相应文件如下图：


main.cpp [2] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Output.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
1217 60
1218 63
1204 64
1220 71
1214 74
1208 77
1209 79
1210 80
1213 82
1219 84
1206 85
1202 88
1207 90
1215 91
1203 92
1211 93
1205 96
1212 97
1216 99
1201 100
else if(k==3)
{
printf("请输入要删除的学生的学号：\n");
scanf("%d",&x);
t = visit_score(Tree,x);
if(t)
{
Delete_tree(&Tree,t->score);
printf("删除成功！\n");
}
else
printf("您所要删除的学生不存在！\
}
else if(k==4)
{
FILE *fp;
if ((fp = fopen("D:\\Output.txt","w")
exit(0);
printf("学号 成绩\n");
Print_tree(Tree,fp);
fclose(fp);
}
}
return 0;
D:\c\源文件\2\bin\Debug\2.exe
输入0为结束,输入1为查询某一个学生成绩，输入2
学生成绩，输入4为打印成绩单
4
学号 成绩
1217 60
1218 63
1204 64
1220 71
1214 74
1208 77
1209 79
1210 80
1213 82
1219 84
1206 85
1202 88
1207 90
1215 91
1203 92
1211 93
1205 96
1212 97
1216 99
1201 100
微软拼音 半 :
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck
CppCheck me
Executing: "D:\c\CodeBlocks/cb_console_runner.exe" "D:\c\源文件\2\bin\Debug\2.exe" (in D:\c\源文件\2\.)
D:\c\源文件\2\main.cpp
Windows (CR+LF)
WINDOWS-936
Line 147, Column 6
Insert

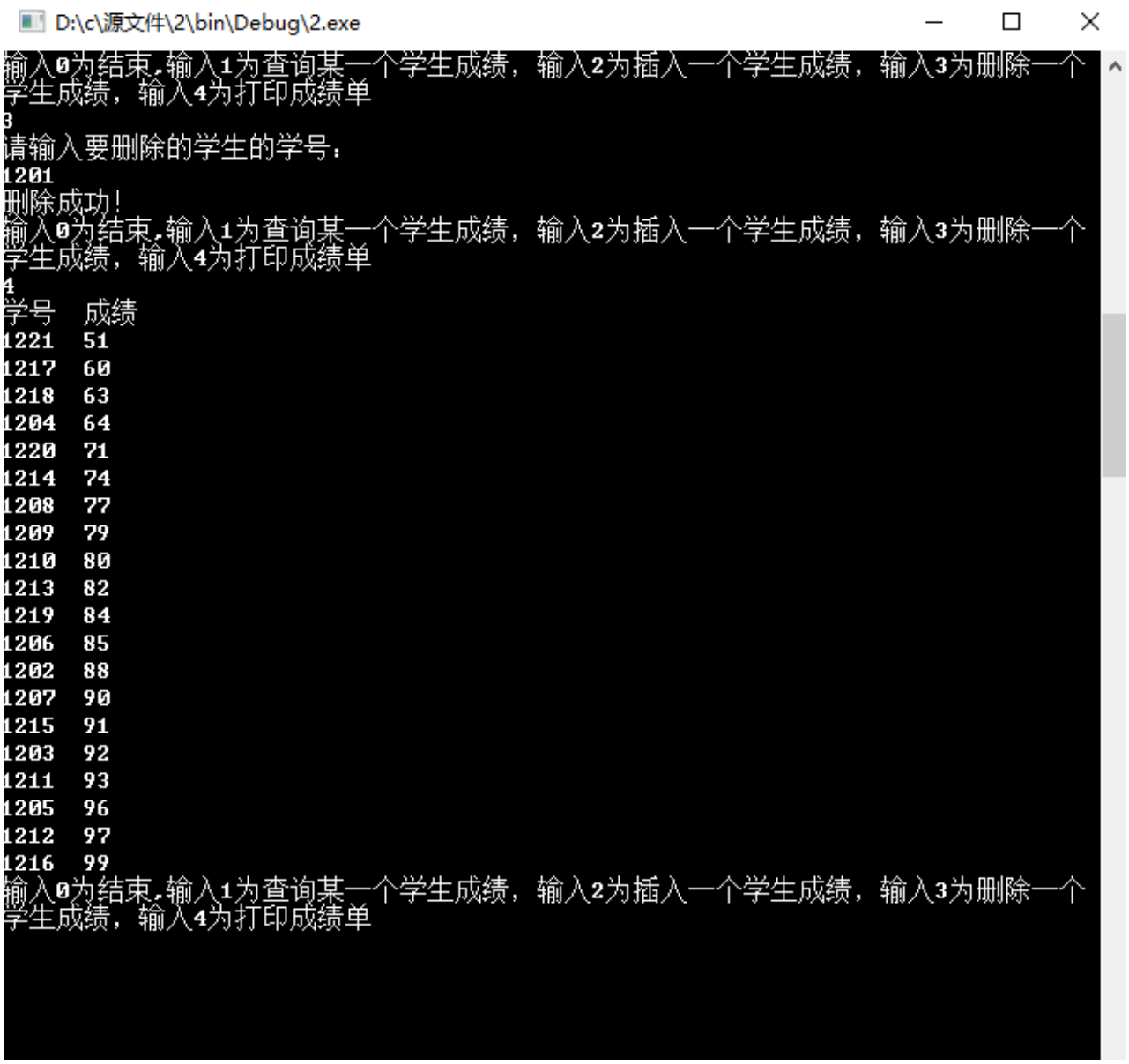
查找操作如下图：


D:\c\源文件\2\bin\Debug\2.exe
输入0为结束,输入1为查询某一个学生成绩，输入2为插入一个学生成绩，输入3为删除一个学生成绩，输入4为打印成绩单
1
请输入要查找的学生学号：
1201
查找成功!
该学生的学号和成绩为：1201 100
输入0为结束,输入1为查询某一个学生成绩，输入2为插入一个学生成绩，输入3为删除一个学生成绩，输入4为打印成绩单
微软拼音 半 :

插入操作：


D:\c\源文件\2\bin\Debug\2.exe
输入0为结束,输入1为查询某一个学生成绩，输入2为插入一个学生成绩，输入3为删除一个学生成绩，输入4为打印成绩单
2
请输入学生的学号和成绩：
1221 51
插入成功!
输入0为结束,输入1为查询某一个学生成绩，输入2为插入一个学生成绩，输入3为删除一个学生成绩，输入4为打印成绩单
4
学号 成绩
1221 51
1217 60
1218 63
1204 64
1220 71
微软拼音 半 :
1208 77
1209 79
1210 80
1213 82
1219 84
1206 85
1202 88
1207 90
1215 91
1203 92
1211 93
1205 96
1212 97
1216 99
1201 100
输入0为结束,输入1为查询某一个学生成绩，输入2为插入一个学生成绩，输入3为删除一个学生成绩，输入4为打印成绩单

删除操作：



五、系统不足与经验体会

系统不足：

对于查找算法，我是通过遍历整个树来寻找的，时间复杂度较大，而且寻找速度相对较慢，需要改进此算法。

经验体会：

深刻掌握了 BST 存储结构的查找，插入，删除算法，进一步增强了自己的思维能力和代码实力。

六、附录：源代码（带注释）

```
#include <stdio.h>
#include <stdlib.h>

typedef struct tree
{
   int student_number;
   int score;
   struct tree *lchild;
   struct tree *rchild;
}bst;


int search_tree(bst *t, bst **p, bst**q, int key)
{
    int flag = 0;
    *q = t;
    while(*q)
    {
        if(key > (*q)->score)
        {
            *p = *q;
            *q = (*q)->rchild;
        }
        else if(key < (*q)->score)
        {
            *p = *q;
            *q = (*q)->lchild;
        }
        else
        {
            flag = 1;
            break;
        }
    }
    return flag;
}

int insert_tree(bst *&t,int number,int key)
{
    bst *p = t,*q,*s;
    int flag = 0;
    if (search_tree(t,&p,&q,key) == 0)       //寻找此键值对应的应该插入
的结点的位置
    {
```

```
        flag = 1;
        s = (bst *)malloc(sizeof(bst));
        s ->score = key;
        s ->student_number = number;
        s->lchild = NULL;
        s->rchild = NULL;
        if (p == NULL)
            t = s;
        else if(key > p->score)
            p->rchild =s;
        else
            p->lchild = s;
    }
    return flag;
}

int Delete_tree(bst **t, int key)
{
    bst *p = *t, *q,*s,**f;
    int flag = 0;
    if (search_tree(*t,&p,&q,key) == 1)        //寻找要删除的结点是否在
已知的二叉排序树中
    {
        flag = 1;
        if (p == q)
            f = &(*t);
        else
        {
            f = &(p->lchild);
            if (key > p->score)
                f = &(p->rchild);
        }
        if (!q->rchild)                        //对于q的三种情况进行判断
        {
            *f = q ->lchild;
        }
        else if (!q->lchild)
        {
            *f = q ->rchild;
        }
        else
        {
            p = q->rchild;
            s = p;
```

```
            while(p->lchild)
            {
                s = p;
                p = p->lchild;
            }
            *f = p;
            p->lchild = q->rchild;
            if (s!=p)
            {
                s->lchild = p->rchild;
                p->rchild = q->rchild;
            }

        }
        free(q);
    }
    return flag;
}

bst *visit_score(bst *p,int number)              //查找学号为number的
结点
{
    bst *q = NULL;
    if(!p)
        return NULL;
    if(p->student_number == number)
        return p;
    if(p->lchild)
    {
        q = visit_score(p->lchild,number);
        if(q)
            return q;
    }
    if(p->rchild)
    {
        q = visit_score(p->rchild,number);
        if(q)
            return q;
    }
    return NULL;
}

void Print_tree(bst *p,FILE *fp)
{
```

```
    if (p)
    {
        Print_tree(p->lchild,fp);
        printf("%d  %d\n",p->student_number,p->score);
        fprintf(fp,"%d\t%d\n",p->student_number,p->score);
        Print_tree(p->rchild,fp);
    }
}

int main()
{
    bst *Tree = NULL,*t;
    FILE *fp;
    int m,n;
    if ((fp = fopen("D:\\studentdata.txt","r")) == NULL)    /*以只读的
形式打开文件，并检查文件是否打开*/
        exit(0);
   while(fscanf(fp,"%d",&m)!=EOF && fscanf(fp,"%d",&n)!=EOF)
    {
        insert_tree(Tree,m,n);
    }

    while(1)
    {
        int k,x,y;
        printf("输入 0 为结束,输入 1 为查询某一个学生成绩，输入 2 为插入
一个学生成绩，输入 3 为删除一个学生成绩，输入 4 为打印成绩单\n");
        scanf("%d",&k);
        if(k==0)
            break;
        else if(k==1)
        {
            printf("请输入要查找的学生学号：\n");
            scanf("%d",&x);
            t = visit_score(Tree,x);
            if(t)
            {
                printf("查找成功！\n");
                printf("该学生的学号和成绩为：");
                printf("%d   %d\n",t->student_number,t->score);
            }
            else
                printf("未查找到该结点！\n");
```

```
        }
        else if(k==2)
        {
            printf("请输入学生的学号和成绩：\n");
            scanf("%d %d",&x,&y);
            if(insert_tree(Tree,x,y))
                printf("插入成功！\n");
            else
                printf("插入失败，该结点已经存在！\n");
        }
        else if(k==3)
        {
            printf("请输入要删除的学生的学号：\n");
            scanf("%d",&x);
            t = visit_score(Tree,x);
            if(t)
             {
                Delete_tree(&Tree,t->score);
                printf("删除成功！\n");
             }
            else
                printf("您所要删除的学生不存在！\n");
        }
        else if(k==4)
        {
            FILE *fp;
            if ((fp = fopen("D:\\Output.txt","w"))==NULL)          /*
以写的形式打开文件，并检查文件是否打开*/
                exit(0);
            printf("学号  成绩\n");
            Print_tree(Tree,fp);
            fclose(fp);
        }

    }
    return 0;
}
```