# 学生信息管理系统（控制台应用）

目录

[学生信息管理系统（控制台应用） 1](#_Toc1785)

[一、 学生信息存储 2](#_Toc23671)

[1.1初始数据存储 2](#_Toc23676)

[1.2内存数据存储 2](#_Toc10531)

[二、 学生信息管理 6](#_Toc12922)

[2.1插入 6](#_Toc2113)

[1. 新增学生（学生ID + 姓名） 6](#_Toc12920)

[2. 新增课程（课程ID + 课程名） 6](#_Toc28251)

[3. 新增选课记录（学生ID+课程ID+成绩） 6](#_Toc5005)

[2.2删除 6](#_Toc16410)

[1.删除学生（ID） 6](#_Toc12796)

[2.删除课程（ID） 6](#_Toc20968)

[3.删除选课记录（学生ID+课程ID） 7](#_Toc15576)

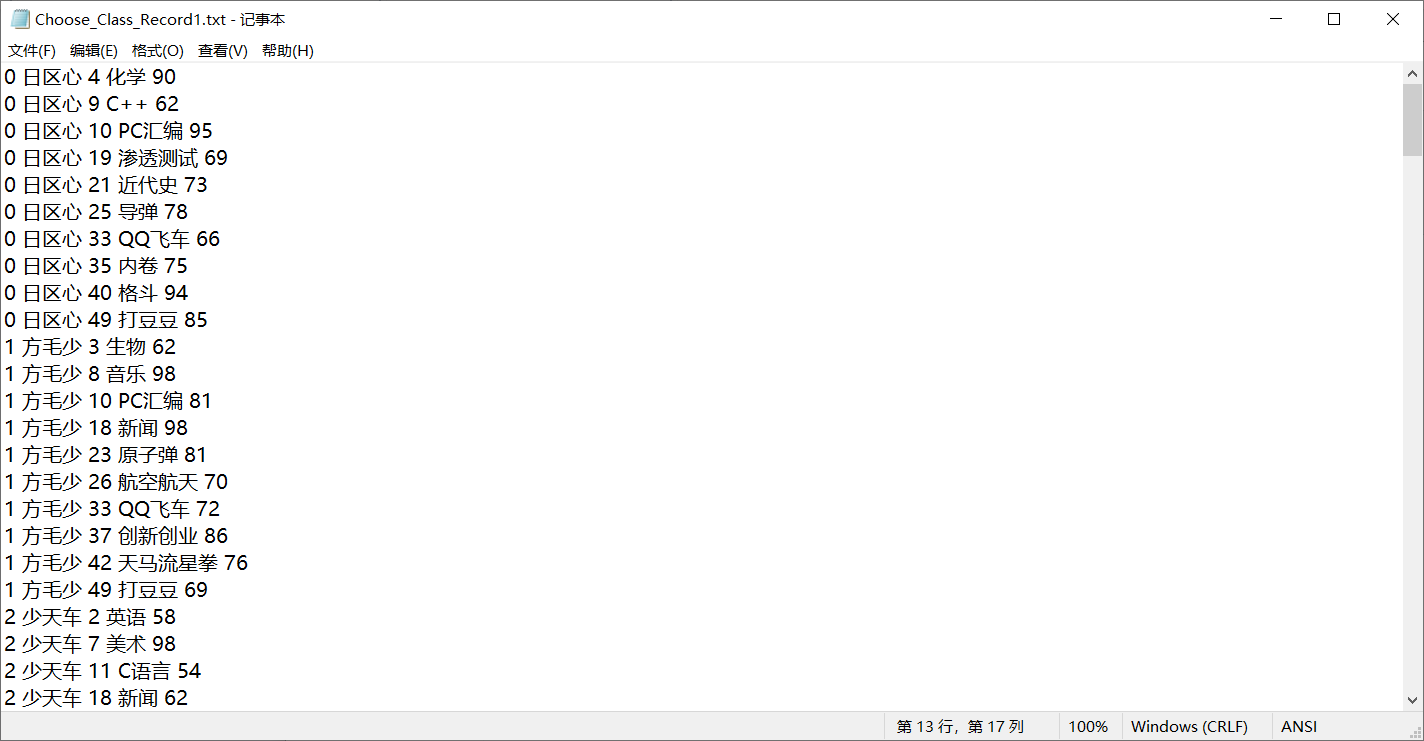
[2.3查询 7](#_Toc22941)

(代码总数：1200行)

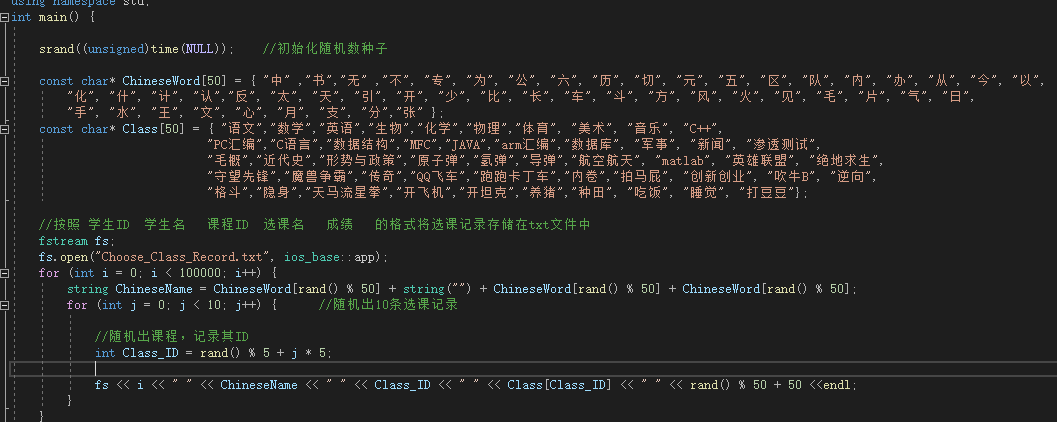
# 学生信息存储

### 1.1初始数据存储

初始数据使用硬盘存储，用“学生ID 学生姓名 课程ID 课程名 成绩”的格式存储在txt文件里。



初始数据中的学生ID、姓名、课程ID、课程名、成绩是通过代码生成的

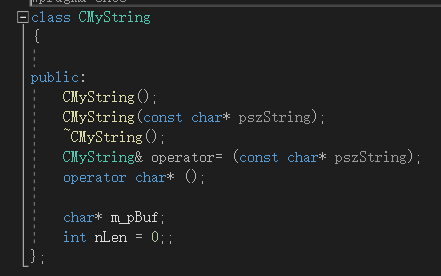


### 1.2内存数据存储

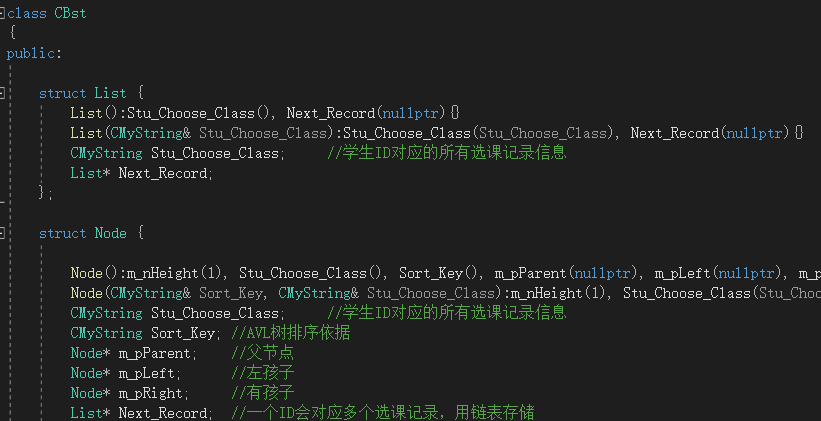
为了对学生信息进行高效率管理，需要将硬盘数据读取到内存中，使用文件流的方式读取。

读取完之后，将这些存储到平衡二叉树中。

首先设计了一个变长字符串存储类CMystring，用来存储学生选课信息；



然后设计了一个CBst平衡二叉树类，实例化4课平衡二叉树，用来存储学生、课程信息：





第一棵：CAvlTree\_StuID

CAvlTree\_StuID以学生ID为排序标准，进行建树，一个ID如果有多个选课记录，则以链表形式附加到节点后面，如图1-2-1所示：

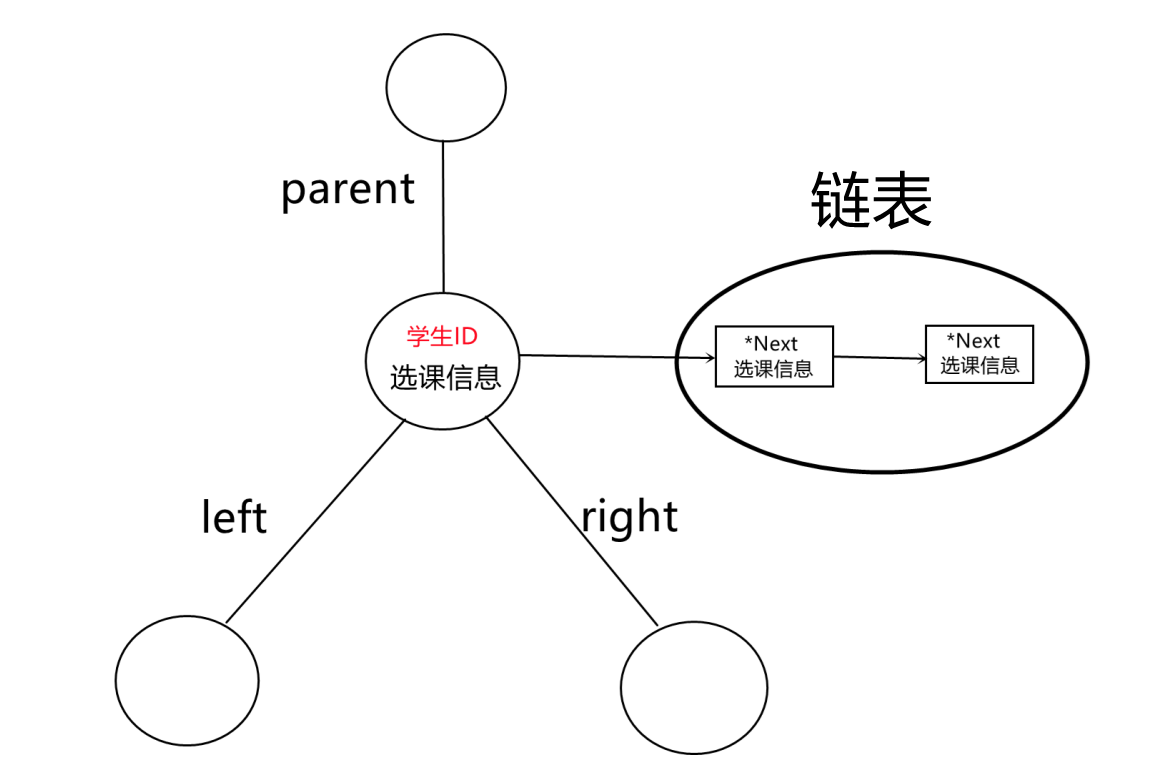


图1-2-1

第二棵：CAvlTree\_StuName

CAvlTree\_StuName以学生姓名为排序标准，进行建树，一个姓名如果对应多个ID（同名），则以链表形式附加到节点后面，如图1-2-2所示：

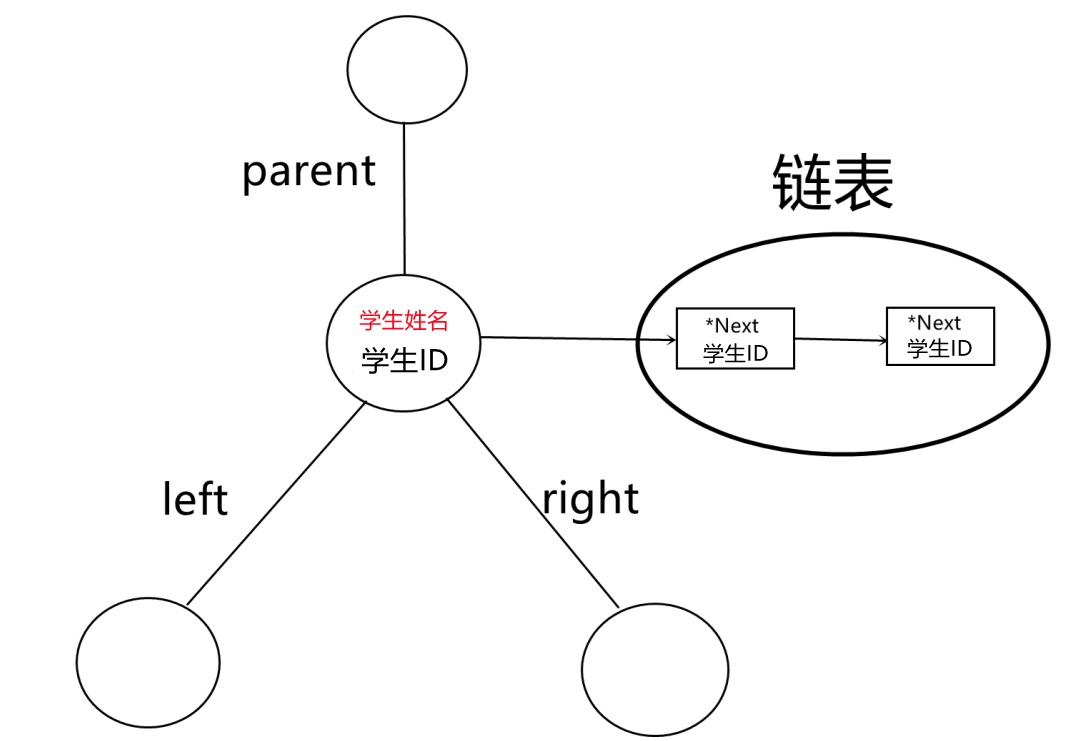


图1-2-2

第三棵：CAvlTree\_ClassID

CAvlTree\_ClassID以课程ID为排序标准，进行建树，一个课程如果有多个学生选，则以链表形式附加到节点后面，如图1-2-3所示：

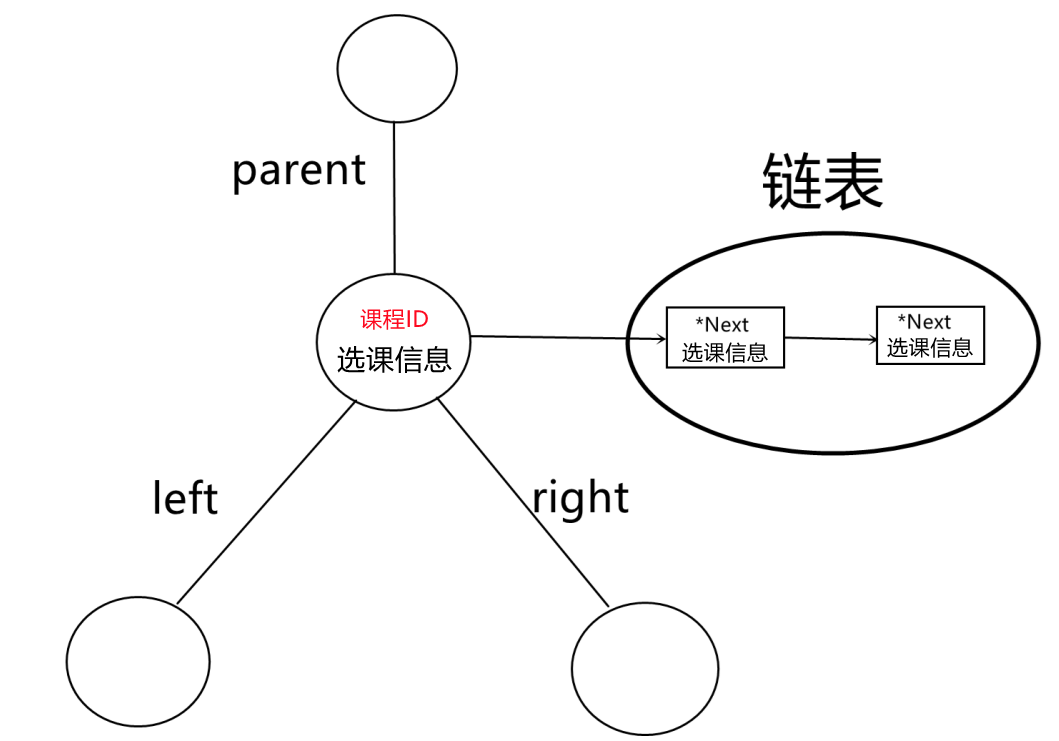


图1-2-3

第四棵：CAvlTree\_ClassName

CAvlTree\_ClassName以课程名为排序标准，进行建树，一个课程名对应唯一课程ID，因此不需要附加链表，如图1-2-4所示：

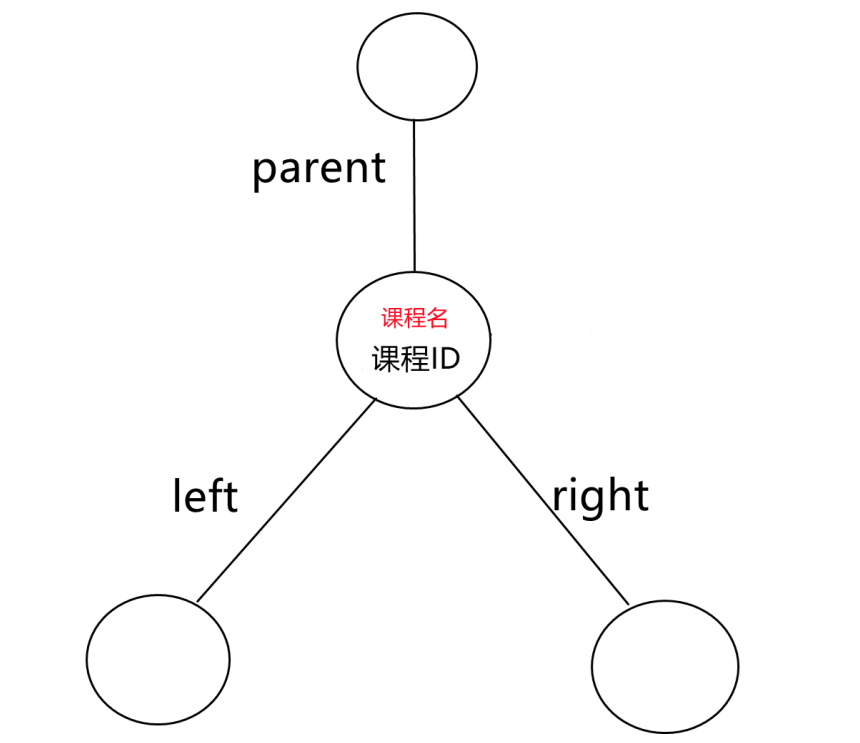


图1-2-4

# 学生信息管理

### 2.1插入

插入分3种：

1. 新增学生（学生ID + 姓名）

需要对CAvlTree\_StuID树和CAvlTree\_StuName树进行操作

CAvlTree\_StuID树：通过学生ID进行排序，新增节点

CAvlTree\_StuName树：通过学生姓名进行排序，新增节点

1. 新增课程（课程ID + 课程名）

需要对CAvlTree\_ClassID树和CAvlTree\_ClassName树进行操作

CAvlTree\_ClassID树：通过课程ID进行排序，新增节点

CAvlTree\_ClassName树：通过课程名进行排序，新增节点

1. 新增选课记录（学生ID+课程ID+成绩）

需要对CAvlTree\_StuID树、CAvlTree\_ClassID树进行操作

CAvlTree\_StuID树：首先通过学生ID找到相应节点，然后将新的选课记录存储在该节点的附加链表里面

CAvlTree\_ClassID树：首先通过课程ID找到相应节点，然后将新的选课记录存储在该节点的附加链表里面

最后，每插入一条3.选课记录，需要将该记录同步更新到txt文件中，直接通过文件流插入到文件末尾即可。

### 2.2删除

删除分3种：

1.删除学生（ID）

需要对CAvlTree\_StuID树、CAvlTree\_StuName树、CAvlTree\_ClassID树进行操作

CAvlTree\_StuID树：通过学生ID找到对应节点，记录下学生姓名、课程ID，删掉该节点

CAvlTree\_StuName树：通过记录的学生姓名找到对应节点，将节点及其附加链表的学生ID进行比对，删掉与学生ID匹配的数据，避免同名问题。

CAvlTree\_ClassID树：通过记录的课程ID找到对应节点，如果节点及其附加链表里包含相匹配的学生ID、学生姓名的数据，删掉该数据

2.删除课程（ID）

需要对CAvlTree\_StuID树、CAvlTree\_ClassID树、CAvlTree\_ClassName树进行操作

CAvlTree\_ClassID树：通过课程ID找到对应节点，记录下课程名、学生ID，删掉该节点

CAvlTree\_ClassName树：通过记录下的课程名找到对应节点，删除该节点（不用考虑课程同名问题）

CAvlTree\_StuID树:通过记录下的学生ID找到对应节点，如果节点及其附加链表包含相匹配的课程ID和课程名，删除相应数据

3.删除选课记录（学生ID+课程ID）

需要对CAvlTree\_StuID树、CAvlTree\_ClassID树进行操作

先从这两棵树中找到学生ID对应的姓名，课程ID对应的课程名，记录下来。

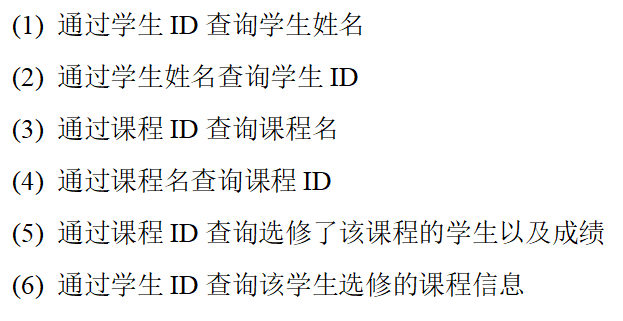
CAvlTree\_StuID树：首先通过学生ID找到相应节点，如果节点及其附加链表包含相匹配的课程ID和课程名，删除相应数据

CAvlTree\_ClassID树：首先通过课程ID找到相应节点，如果节点及其附加链表里包含相匹配的学生ID、学生姓名的数据，删掉该数据

最后，每删除一条记录，首先要将原txt数据全部拷贝到新的txt文件中（除了要删除的数据），然后再将新的txt文件内容写入原txt文件，达到删除效果。

### 2.3查询

查询分6种：



但实际上相当于4种，通过学生ID、学生姓名、课程ID、课程名4种条件进行查询

1. （6）实现过程：通过学生ID，从CAvlTree\_StuID树找到相应节点，将里面的信息打印出来
2. 实现过程：通过学生姓名，从CAvlTree\_StuName树找到相应节点，将里面的信息打印出来
3. （5）实现过程：通过课程ID，从CAvlTree\_ClassID树找到相应节点，将里面的信息打印出来
4. 实现过程：通过课程名，从CAvlTree\_ClassName树找到相应节点，将里面的信息打印出来