暨南大学本科实验报告专用纸

课程名称 高级语言程序设计 成绩评定

实验项目名称 大实验 指导教师 张鑫源

实验项目编号 102 实验项目类型 实验地点

学生姓名 陈俊文 学号 2019051113

学院 智能科学与工程 系 专业 信息安全

实验时间 2020 年 6 月 20 日 午～ 月 日 午 温度 ℃湿度

（一）实验目的

模拟关系数据库

（二）实验内容和要求

- **实验要求**

1. 设计合理的输出展示实验结果；
2. 三个表的大小做如下规定，表一至少存储1000条学生信息的记录，表二至少存储50条课程信息的记录，表三至少存储10000条学生课程成绩的记录。
3. 所有的增删改查都在内存中进行，最后才将内存中的数据导入硬盘。
4. 增：表一至表三中的数据按照学号/课程号/学号从小到大的顺序排列，增加记录不能破坏顺序；
5. 删：由于表之间不是独立的，例如删除表一中某一学生的信息，表三也要相应的修改；
6. 改：修改表一中每条记录的任意属性（第一列的属性除外）
7. 查：最基础的功能是按照学号/姓名查询学生的信息，其余查询功能（例如查询年龄在20岁以下的学生的学号）自行丰富；
8. 增、删、改、查的其余功能自行开发；
9. 分别统计增删改查的执行时间，自行优化增删改查的执行时间。

- **DDL**

1. 7.5号20点之前提交实验报告至学委（实验报告格式为pdf，附源代码）。
2. 学委将实验报告发送至邮箱zhangxy@jnu.edu.cn。

大实验 模拟关系数据库

➢ 实验内容

模拟关系数据库，实现增、删、改、查四项功能。数据库中的数据可以视为大量元组构成的一个集合。例如存储学生-课程的关系表由下面三个表构成（可以视为三份文件）：

表一. 学生信息

学号	姓名	性别	年龄	院系

表二. 课程信息

课程号	课程名	先行课	学分

表三. 学生-课程成绩信息

学号	课程号	成绩

（三）主要仪器设备

仪器：计算机

实验环境：DevC++

（四）源程序

（五）实验步骤与调试

1. 根据三个关系主体分别建立类，分别对应 Student,Course,Point(学生，课程，成绩）

2. 根据对三个表的功能要求，选择数据结构。这里我选择用 C++标准库中的 vector 类（动态数组）来储存数据。

3. 为了提高效率与掌控方向感，开始编程之前先将功能菜单列举。以学生信息为例：

   1) 查：按学号/名字/性别/院系匹配；

   2) 增：输入名字，院系，性别，年龄。(学号由系统分配）

   3) 改：输入学号获取对象后修改；

   4) 删：通过学号删除；返回主菜单

4. 由于三个表的功能实现算法雷同，因此下面以学生表的操作为例子作解释。

5. 首先是学生类的编写。除了学生的基本属性（学号，姓名，院系，性别，年龄）外还需要一个静态成员变量 count（对象通用），用来记录当前表中学生的数量。这个变量对实现学号回收再利用的功能很重要，后文再做解释。使用 static 变量时需要记得在类体外作一次声明。

```
public:
    static int count;
    int id;             //学号
    string name;        //姓名
    string faculty;     //院系
    string gender;      //性别
    int age;            //年龄
};
int Student::count = 0;
```

6. 然后我们需要常规地编写学生类的构造函数（别忘了复制构造函数）与析构函数。为了方便输出，通过友元函数，对 ostream 类的 <<（左移符）进行重载。

```
friend ostream & operator<<(ostream & os, const Student& stud){
    string str_id = to_string(stud.id);
    while(str_id.size() < 4){
        str_id = "0" + str_id;
    }
    os << "学号: "<<str_id<<endl
       << "姓名: "<<stud.name<<endl
       << "性别: "<<stud.gender<<endl
       << "年龄: "<<stud.age<<endl
       << "院系: "<<stud.faculty<<endl;
    return os;
}
```

7. 接下来介绍学号再利用的算法实现，首先声明一个 vector 数组 empty 用作储存闲置的学号（即被回收的），然后对 Student 类的有参数构造函数做修正，当 empty 数组中储存有学号时，构造函数给新对象分配的学号则为 empty 数组的尾部元素。并且同时将该元素从 empty 中删除。如果 empty 数组为空，则分配给新对象的学号为当前学生数量+1 所得数。

```
vector <int> empty;    //闲置的二手学号储存地
```

```
Student(string _name,string _faculty,string _gender,int _age):
        name(_name),faculty(_faculty),gender(_gender),age(_age){
        if(empty.size() == 0){
            id = ++count;
        }else{
            int size = empty.size();
            id = empty[size - 1];
            empty.erase(empty.begin() + size - 1);
        }
}
```

8. 新增学生的方法主要通过构造函数来实现，只要引导客户端通过输入流键入 姓名，院系，性别，年龄，通过构造函数构造对象后 push 进学生表即可。（此时是乱序）

9. 删除学生可以通过遍历学生表，找到对应的学号，将其对应的对象从表中删除之后，将这个学号存入 empty 表内以便重新利用。删除学生信息的同时，需要将该学生的成绩信息也删除掉（遍历得到对象删除）.

10. 修改学生信息同理，此处尝试使用指针操作，提高效率。

```
Student * queryStu_byID(int id){
    for(int i = 0;i < stu.size(); ++i){
        if(stu[i].id == id)
            return &stu[i];
    }
}
void del_Stu(int id){                    //通过id删除学生
void edit_Stu(){                         //通过id修改学生信息
    int id;
    cout<<"请输入待修改学生的学号: ";
    cin>>id;
    Student * p = queryStu_byID(id);

    cout<<"1.修改名字  2.修改院系"<<endl;
    int choice;
    string temp;
    cin>>choice;
    cout<<"你希望更改为: ";
    cin>>temp;
    switch(choice){
        case 1:
            p->name = temp;
            break;
        case 2:
            p->faculty = temp;
            break;
    }
}
```

11. 课程表与成绩表基本上与学生表雷同。另外，我在成绩表中设置了一个查看该课程按成绩排名的菜单选项。遍历成绩表，将所对应课程的所有成绩copy存到一个新的数组target之中。通过标准库中的 sort 方法，自编排序规则（分数从高到低）来实现。

```
bool scorelistRule(const Point &p1, const Point &p2){
    return p1.score > p2.score;
}
```

12. 下面介绍文件读写，保存与读取。这里以成绩表的读写作为例子。这里需先声明头文件 fstream，ifstream 对应输入，ofstream 对应输出。读取的流程是：先找到对象文件 score.txt，文件存在则开始读取，不存在则创建文件。该函数的形参为成绩表的行数。该数据也会有对应函数方法来保存和读取。值得注意的是，为了避免出现 3221225477（访问越界）和 3221225725（堆栈溢出）。需要编写当成绩表 size 为 0 时的读取方式。

```
int read_to_poi(int p){                                    //读取成绩档案
    ifstream inscore("score.txt");
    if(!inscore){
        cout<<"读取成绩信息失败"<<endl;
        return 0;
    }
    int size = poi.size();
    for(int i = 0;i < p;++i){
        if(size == 0){
            int id,num,score;
            inscore >>  id >> num >> score;
            Point test(id,num,score);
            poi.push_back(test);
        }else{
            inscore >>  poi[i].id >> poi[i].num >> poi[i].score
        }
    }

    inscore.close();
    cout<<"读取成绩信息完成"<<endl;
    return 1;
}
```

13. 写入成绩表的储存文件时，需要先对成绩表信息进行排序（学号优先，再是课程编号，从小到大），排序规则还是自编写。

```
bool scoreCpRule(const Point &p1,const Point &p2){
    if(p1.id == p2.id)
        return p2.num < p1.num;
    else
        return p1.id < p2.id;
}
```

```
int save_to_poi(int p){                                    //写入成绩档案
    sort(poi.begin(),poi.end(),scoreCpRule);
    cout<<"成绩排序成功"<<endl;
    ofstream fout("score.txt");  //创建文件
    if (!fout){
        cout << "保存学生成绩失败\n";
        return 0;
    }
    for (int i = 0; i < p; i++){
        fout << poi[i].id << " " << poi[i].num << " " << poi[i].score <<"\n";
    }
    cout<<"成功保存学生成绩"<<endl;
    fout.close();
    return 1;
}
```

14.接着需要编写读写基础设置如（学生表行数，课程表行数，成绩表行数，以及 empty 表和 empty_CourseNum 表的 size 与元素）的函数方法。

```
void output_setting(){                          //输出 空学号数组信息 空课程数组信息  学生数量
    cout<<"-------------"<<endl;
    ofstream setting("setting.txt");
    if(!setting)
        cout<<"创建配置失败"<<endl;

    setting <<"empty_size:"<< empty.size()<<endl;                          //空学号
    for(int i:empty){
        setting<<i<<" ";
    }
    setting<<endl;
    setting<<"empty_course_size:"<< empty_CourseNum.size()<<endl;          //空课程编号
    for(int j:empty_CourseNum){
        setting<<j<<" ";
    }

    setting<<endl<<endl;
    setting<< "Student_Count:" << stu.size() << endl <<
              "Course_Count:"<< cor.size() << endl <<
              "Score_num:"   << poi.size() << endl;        //学生数量,课程数量，成绩数量
    cout<<"保存设置成功"<<endl;
    cout<<"-------------"<<endl;
}
```

读写流程用 fstream 中的方法可以简单实现，不过多做解释。

读取的时候，可以先将字段用一个 char 数组读取了，再开始读取内容，存入变量，如：

```
int size1,size2;
char temp[20];
for(int i = 0;i<11;++i){
    setting >> temp[i];
}

setting >> size1;
```

```
for(int i =0;i<size1;++i){
    int index;
    setting >> index;
    empty.push_back(index);
    cout<<"empty["<<i<<"] = "<<empty[i]<<endl;
}
```

15.最后是随机数据初始化，需要引用 time.h 头文件。先初始化随机种子，再根据用户输入的各个表初始化数量进行随机初始化。这里用到 rand()%方法。成绩表初始化完成后需要做一次排序。

16.最后通过菜单索引的方式将各个功能整合在一起。

（六）实验结果与分析

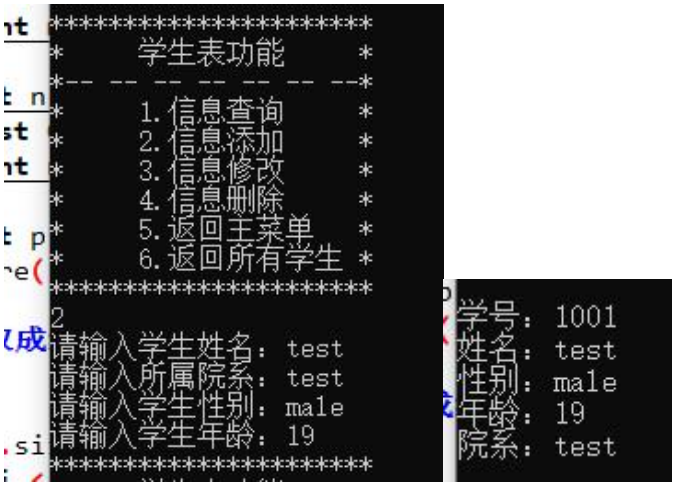
1. 初始化


*   5.保存修改   *
*   6.随机初始化   *
输入菜单索引：6
输入初始化的学生人数，课程数，成绩数.用空格隔开:1000 50 10000
* 学生列表初始化成功 *
* 课程列表初始化成功 *
* 成绩列表初始化成功 *
* 选择需要操作的表 *
*   P S   *
*   无本地数据的   *
* 可先随机初始化数据 *
有本地数据时可先读取
*   1.学生信息   *
*   2.课程信息   *
*   3.成绩信息   *
*   4.读取数据   *
*   5.保存修改   *
*   6.随机初始化   *
输入菜单索引：

2. 查询学生：

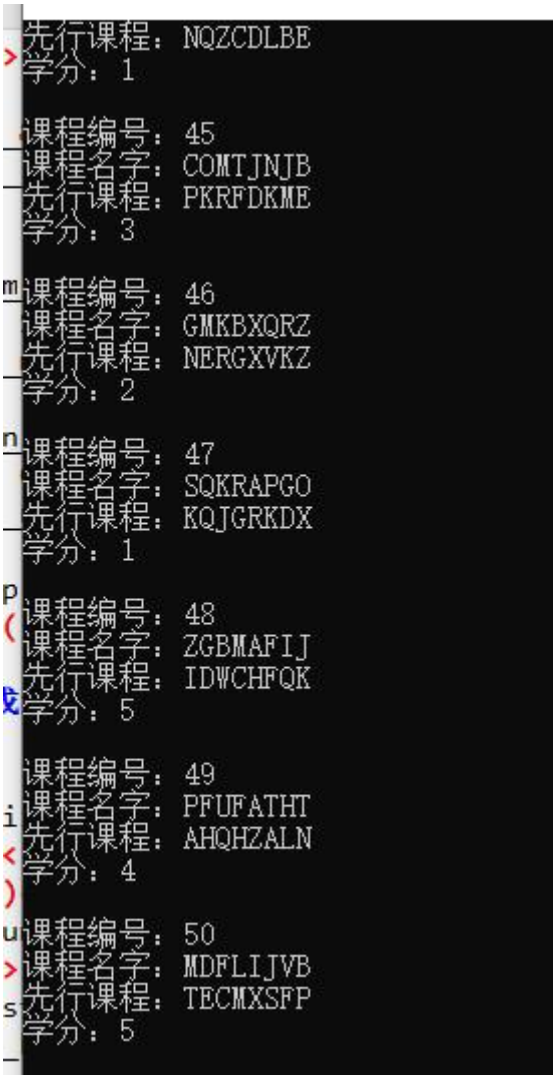

*   请选择查询条件   *
*   1.学号匹配   *
*   2.名字匹配   *
*   3.性别匹配   *
*   4.院系匹配   *
1
输入匹配值:1000
学号：1000
姓名：UjbxXLHO
性别：male
年龄：18
院系：APVXSFBQ

3. 添加学生：

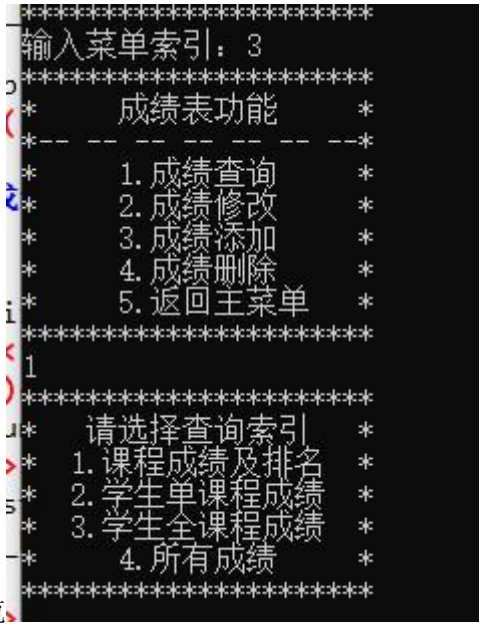

学生表功能
1.信息查询
2.信息添加
3.信息修改
4.信息删除
5.返回主菜单
6.返回所有学生
2
请输入学生姓名：test
请输入所属院系：test
请输入学生性别：male
请输入学生年龄：19
学号：1001
姓名：test
性别：male
年龄：19
院系：test

4.

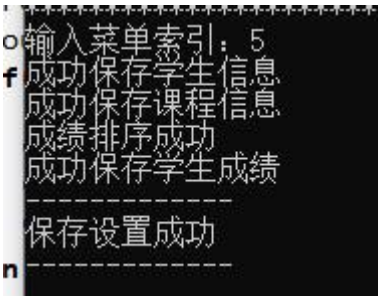
课程的随机化：


先行课程：NQZCDLBE
学分：1
课程编号：45
课程名字：COMTJNJB
先行课程：PKRFDKME
学分：3
课程编号：46
课程名字：GMKBXQRZ
先行课程：NERGXVKZ
学分：2
课程编号：47
课程名字：SQKRAPGO
先行课程：KQJGRKDX
学分：1
课程编号：48
课程名字：ZGBMAFIJ
先行课程：IDWCHFQK
学分：5
课程编号：49
课程名字：PFUFATHT
先行课程：AHQHZALN
学分：4
课程编号：50
课程名字：MDFLIJVB
先行课程：TECMXSFP
学分：5

成绩表功能表略览


输入菜单索引：3
成绩表功能
1.成绩查询
2.成绩修改
3.成绩添加
4.成绩删除
5.返回主菜单
1
请选择查询索引
1.课程成绩及排名
2.学生单课程成绩
3.学生全课程成绩
4.所有成绩

保存：


输入菜单索引：5
成功保存学生信息
成功保存课程信息
成绩排序成功
成功保存学生成绩
保存设置成功