Score Processing

2021-03-28

混合1902

余丛杉

3190103165

01. Introduction

Write a program to process students score data.

- **Input Specification**
  - Student's name and student id, as `<student id>, <name>`, and
  - Score for one student of one course, as `<student id>, <course name>, <marks>`.
  - Examples:

```
3190101234, Zhang San
3190101111, Linear Algebra, 89.5
```

- **Output Specification**
  - The first line of the output is the table head, consists fields like this:

```
student id, name, <course name 1>, <course name 2>, ..., average
```

    where the course names are all the courses read, in alphabet order. There should be one space after each comma.
  - Then each line of the output is data for one student, in the ascended order of their student id, with score of each course, like:

```
3190101234, Zhang San, 85.0, , 89.5, , , 87.3
```

  - And the last line of the output is a summary line for average score of every course, like:

```
, , 76.2, 87.4, , , 76.8
```

    All the number output, including the averages have one decimal place.
- **Preliminary idea**
  - Create a `Student` class to store student information.
  - Use map to tie ID and student information together.
  - Record all IDs, subjects, and the average value of each course.

02.Algorithm Specification

2.0 Student.h

- **Data Structure**
  - Define the structure for storing personal information `struct Student`
    - `` `name `` record name
    - `lec_level` bound courses and grades
    - `aver` record the student's average grade
- Define read-in, reset, average calculation, and output functions.

```
# ifndef STUDENT_H
# define STUDENT_H

#include<iostream>
```

```
#include<string>
#include<map>
#include<set>
#include<vector>
#include<iomanip>
using namespace std;

// 定义学生结构
struct Student
{
    string name;
    map<string,double> lec_level;
    double aver;
};

// 读入函数
void Read( map<string, Student> &id_stu , set<string> &All_lec , set<string>
&All_ID );

// 重置学生没有的科目成绩为-1
void Reset( map<string, Student> &info , const set<string> lec , const
set<string> id );

// 计算学生成绩均值及科目均值
void SetAve( map<string, Student> &info , const set<string> lec , const
set<string> id , vector<double> &grade);

// 输出函数
void Print(const set<string> lec, const map<string, Student> id_stu, const
set<string> id, vector<double> grade);

#endif
```

2.1 `read()`

- Loop until the return value of getline is 0, that is, the end of reading.
- Read one line of information at a time.
  - Use a comma as a sign to split a line of information.
    - If there is a comma, it is the input of the first case.
    - If there are two commas, it is the input of the second case.
  - Use `set All_ID` to determine whether the added information belongs to an existing student.
    - If it is a new student ID, create a new `Student`.
    - If it is an existing student ID, just add the information.

```
void Read( map<string, Student> &id_stu , set<string> &All_lec , set<string>
&All_ID )
{
    string str;

    while( (bool)getline(cin,str) != 0 ){   // 读入一行并判断读入是否结束
        bool IsNew = true;  // 判断是否为新学生
        int start = 0;      // 字符串起始位置
        int end;        // 字符串终止位置
        string s[3];    // 储存被切分的信息
```

```
        int index = 0;  // 记录需切分的个数
        while( (end = str.find(',')) != -1 ){   // 直到找不到','为止
            s[index++] = str.substr(start, end-start ); // 切分信息
            str.erase(end, 1);
            start = ++end;  // 逗号不保存
        }
        s[index] = str.substr(start);   // 处理最后一个','后面的信息

        if( id_stu.find(s[0]) != id_stu.end() ){    // 判断是否为新学生
            IsNew = false;
            All_ID.insert(s[0]);
        }

        if( index == 2 ){       // 存储学号、学科、成绩
            double grade = atof(s[2].c_str());  // 字符串转成double类型
            All_lec.insert(s[1]);

            if( IsNew ){    // 若为新学生，则新建一个Student结构
                Student temp;
                temp.lec_level[s[1]] = grade;
                id_stu[s[0]] = temp;
            }else{  // 不为新学生，添加信息即可
                id_stu[s[0]].lec_level[s[1]] = grade;
            }
        }else{  // 存储学号和姓名
            if( IsNew ){    // 若为新学生，则新建一个Student结构
                Student temp;
                temp.name = s[1];
                id_stu[s[0]] = temp;
            }else{   // 不为新学生，添加信息即可
                id_stu[s[0]].name = s[1];
            }
        }
    }
}
```

2.2 `Reset()`

- Reset the grade of subjects that the student does not have to -1.
  - Loop through each student.
  - The inner loop is to traverse each course.
    - If the grade of the corresponding subject is not found, it will be assigned a value of -1.

```
// 重置学生没有的科目成绩为-1
void Reset( map<string, Student> &info , const set<string> lec , const
set<string> id )
{
    // 遍历每个学生
    for( set<string>::iterator it_id = id.begin() ; it_id != id.end() ; it_id++)
{
        // 遍历每门课
        for( set<string>::iterator it_lec = lec.begin() ; it_lec != lec.end() ;
it_lec++){
            // 找不到成绩则赋值为-1
```

```
            if( info[*it_id].lec_level.find(*it_lec) ==
info[*it_id].lec_level.end() ){
                info[*it_id].lec_level[*it_lec] = -1;
            }
        }
    }
}
```

2.3 `SetAve()`

- Calculate the average value of student achievement and the average value of subjects.
  - Traverse each student and find the average grade of each student.
    - Sum if there are results.
    - Record the number of subjects.
    - Find the mean.
  - Iterate through each course and find the average value of each course.
    - Sum if there are results
    - Record the number of students in each course.
    - Find the mean.

```
// 计算学生成绩均值及科目均值
void SetAve( map<string, Student> &info , const set<string> lec , const
set<string> id , vector<double> &grade)
{
    // 遍历每个学生,求每个学生的平均成绩
    for( set<string>::iterator it_id = id.begin() ; it_id != id.end() ; it_id++)
{
        int count = 0;  // 记录学科数目
        double sum = 0; // 记录总成绩
        // 遍历每门课
        for( set<string>::iterator it_lec = lec.begin() ; it_lec != lec.end() ;
it_lec++){
            // 若有成绩则求和
            if( (int)info[*it_id].lec_level[*it_lec] != -1 ){
                sum += info[*it_id].lec_level[*it_lec];
                count++;
            }
        }
        info[*it_id].aver = 1.0 * sum / count;
    }

    // 遍历每门课，求每门课的平均值
    for( set<string>::iterator it_lec = lec.begin() ; it_lec != lec.end() ;
it_lec++ ){
        int count = 0;  // 记录每门课的学生人数
        double sum = 0; // 记录总成绩
        // 遍历每个学生
        for( set<string>::iterator it_id = id.begin() ; it_id != id.end() ;
it_id++){
            // 若有成绩则求和
            if( (int)info[*it_id].lec_level[*it_lec] != -1 ){
                sum += info[*it_id].lec_level[*it_lec];
                count++;
            }
        }
```

```
        double ave = 1.0 * sum / count;
        grade.push_back(ave);
    }

}
```

2.3 `Print()`

- Output title.
- Output each classmate's information.
- Output the average value of each course.

```
// 输出
void Print(const set<string> lec, const map<string, Student> id_stu, const
set<string> id, vector<double> grade)
{
    // 输出title
    cout << "student id, name, ";
    for( set<string>::iterator it = lec.begin() ; it != lec.end() ; it++ )
        cout << *it << ", ";
    cout << "average" << endl;

    // 输出每个同学的信息
    map<string, Student> info = id_stu;
    for( set<string>::iterator it = id.begin() ; it != id.end() ; it++){
        cout << *it  << ", " << info[*it].name << ", ";
        for(set<string>::iterator it_lec = lec.begin() ; it_lec != lec.end() ;
it_lec++ ){
            if( (int)info[*it].lec_level[*it_lec] != -1 ){
                cout <<  fixed << setprecision(1) <<
info[*it].lec_level[*it_lec] << ", " ;
            }else{
                cout << ", ";
            }
        }
        cout << info[*it].aver << endl;
    }

    // 输出每门课的平均值
    double sum = 0;
    int count = 0;
    cout << ", " << ", ";
    for( vector<double>::iterator it = grade.begin() ; it != grade.end() ; it++
){
        cout << *it << ", ";
        sum += *it;
        count++;
    }
    cout << fixed << setprecision(1) <<  1.0*sum/count << endl;
}
```

03.Testing Results

提交结果 ×

提交时间	状态	分数	题目	编译器	耗时	用户
2021/03/26 18:48:31	答案正确	10	编程题	C++ (g++)	17 ms	余丛杉

测试点	结果	分数	耗时	内存
0	答案正确	2	8 ms	448 KB
1	答案正确	2	17 ms	444 KB
2	答案正确	3	10 ms	412 KB
3	答案正确	3	11 ms	568 KB

04. Analysis and Comments

- **Analysis**
  - To deal with variable-length structures, use vector to deal with statistics traversing all elements of a data set, use set to deal with key-value pairs, use map.
  - The complexity of inserting and deleting set and map are both O(lgn) (actually smaller) and automatic sorting.
- **Comments**
  - Have a new understanding of the use of const and reference for c++ function parameters.
  - Wrote a .h file for the first time, and learned how to write a .h file.
  - More familiar with set, vector and map in stl library.
  - Encapsulate functions as much as possible.

Source Code

```
# ifndef STUDENT_H
# define STUDENT_H

#include<iostream>
#include<string>
#include<map>
#include<set>
#include<vector>
#include<iomanip>
using namespace std;

// 定义学生结构
struct Student
{
    string name;
    map<string,double> lec_level;
    double aver;
};
```

```
// 读入函数
void Read( map<string, Student> &id_stu , set<string> &All_lec , set<string>
&All_ID );

// 重置学生没有的科目成绩为-1
void Reset( map<string, Student> &info , const set<string> lec , const
set<string> id );

// 计算学生成绩均值及科目均值
void SetAve( map<string, Student> &info , const set<string> lec , const
set<string> id , vector<double> &grade);

// 输出函数
void Print(const set<string> lec, const map<string, Student> id_stu, const
set<string> id, vector<double> grade);

#endif
```

```
#include "student.h"

int main()
{
    // 把ID和学生信息绑在一起
    map<string, Student> id_stu;
    // 记录所有的ID
    set<string> All_ID;
    // 记录所有的科目
    set<string> All_lec;
    // 记录每门课程的均值
    vector<double> grade;

    Read( id_stu, All_lec, All_ID );

    Reset( id_stu , All_lec , All_ID);
    SetAve( id_stu , All_lec, All_ID , grade );

    Print( All_lec , id_stu , All_ID, grade);
}

// 读取信息
void Read( map<string, Student> &id_stu , set<string> &All_lec , set<string>
&All_ID )
{
    string str;

    while( (bool)getline(cin,str) != 0 ){   // 读入一行并判断读入是否结束
        bool IsNew = true;  // 判断是否为新学生
        int start = 0;      // 字符串起始位置
        int end;        // 字符串终止位置
        string s[3];    // 储存被切分的信息
        int index = 0;  // 记录需切分的个数
        while( (end = str.find(',')) != -1 ){   // 直到找不到','为止
            s[index++] = str.substr(start, end-start ); // 切分信息
            str.erase(end, 1);
            start = ++end;  // 逗号不保存
        }
```

```
        s[index] = str.substr(start);    // 处理最后一个','后面的信息

        if( id_stu.find(s[0]) != id_stu.end() ){    // 判断是否为新学生
            IsNew = false;
            All_ID.insert(s[0]);
        }

        if( index == 2 ){       // 存储学号、学科、成绩
            double grade = atof(s[2].c_str());  // 字符串转成double类型
            All_lec.insert(s[1]);

            if( IsNew ){    // 若为新学生，则新建一个Student结构
                Student temp;
                temp.lec_level[s[1]] = grade;
                id_stu[s[0]] = temp;
            }else{  // 不为新学生，添加信息即可
                id_stu[s[0]].lec_level[s[1]] = grade;
            }
        }else{  // 存储学号和姓名
            if( IsNew ){    // 若为新学生，则新建一个Student结构
                Student temp;
                temp.name = s[1];
                id_stu[s[0]] = temp;
            }else{   // 不为新学生，添加信息即可
                id_stu[s[0]].name = s[1];
            }
        }
    }
}

// 计算学生成绩均值及科目均值
void SetAve( map<string, Student> &info , const set<string> lec , const
set<string> id , vector<double> &grade)
{
    // 遍历每个学生,求每个学生的平均成绩
    for( set<string>::iterator it_id = id.begin() ; it_id != id.end() ; it_id++)
{
        int count = 0;  // 记录学科数目
        double sum = 0; // 记录总成绩
        // 遍历每门课
        for( set<string>::iterator it_lec = lec.begin() ; it_lec != lec.end() ;
it_lec++){
            // 若有成绩则求和
            if( (int)info[*it_id].lec_level[*it_lec] != -1 ){
                sum += info[*it_id].lec_level[*it_lec];
                count++;
            }
        }
        info[*it_id].aver = 1.0 * sum / count;
    }

    // 遍历每门课，求每门课的平均值
    for( set<string>::iterator it_lec = lec.begin() ; it_lec != lec.end() ;
it_lec++ ){
        int count = 0;  // 记录每门课的学生人数
        double sum = 0; // 记录总成绩
        // 遍历每个学生
```

```
        for( set<string>::iterator it_id = id.begin() ; it_id != id.end() ;
it_id++){
            // 若有成绩则求和
            if( (int)info[*it_id].lec_level[*it_lec] != -1 ){
                sum += info[*it_id].lec_level[*it_lec];
                count++;
            }
        }
        double ave = 1.0 * sum / count;
        grade.push_back(ave);
    }

}

// 重置学生没有的科目成绩为-1
void Reset( map<string, Student> &info , const set<string> lec , const
set<string> id )
{
    // 遍历每个学生
    for( set<string>::iterator it_id = id.begin() ; it_id != id.end() ; it_id++)
{
        // 遍历每门课
        for( set<string>::iterator it_lec = lec.begin() ; it_lec != lec.end() ;
it_lec++){
            // 找不到成绩则赋值为-1
            if( info[*it_id].lec_level.find(*it_lec) ==
info[*it_id].lec_level.end() ){
                info[*it_id].lec_level[*it_lec] = -1;
            }
        }
    }
}

// 输出
void Print(const set<string> lec, const map<string, Student> id_stu, const
set<string> id, vector<double> grade)
{
    // 输出title
    cout << "student id, name, ";
    for( set<string>::iterator it = lec.begin() ; it != lec.end() ; it++ )
        cout << *it << ", ";
    cout << "average" << endl;

    // 输出每个同学的信息
    map<string, Student> info = id_stu;
    for( set<string>::iterator it = id.begin() ; it != id.end() ; it++){
        cout << *it  << ", " << info[*it].name << ", ";
        for(set<string>::iterator it_lec = lec.begin() ; it_lec != lec.end() ;
it_lec++ ){
            if( (int)info[*it].lec_level[*it_lec] != -1 ){
                cout <<  fixed << setprecision(1) <<
info[*it].lec_level[*it_lec] << ", " ;
            }else{
                cout << ", ";
            }
        }
        cout << info[*it].aver << endl;
    }
```

```
    // 输出每门课的平均值
    double sum = 0;
    int count = 0;
    cout << ", " << ", ";
    for( vector<double>::iterator it = grade.begin() ; it != grade.end() ; it++
){
        cout << *it << ", ";
        sum += *it;
        count++;
    }
    cout << fixed << setprecision(1) <<  1.0*sum/count << endl;
}
```

Declaration

I hereby declare that all the work done in this project titled ' The World's Richest ' is of my independ ent effort .