# C++实例代码

信1703 姚鋆 20173417

/\*

·File name: file.h

·Description: 所有和文件读写相关的操作

\*/

#pragma once

#include<iostream>

#include<fstream>

#include<string>

#include"student.h"

/\*

函数:read\_record()

功能:从record.txt读入记录，得到现有的记录条数

参数:学生信息记录条数&m, 成绩条数&n，

返回值:1

\*/

int read\_record(int &m, int &n)

{

std::ifstream fin("record.txt");

fin >> m;

fin >> n;

return 1;

}

/\*

函数:write\_record()

功能:写入记录到record.txt，更新记录条数

参数:学生信息记录条数m, 成绩条数n，

返回值: 0

\*/

int write\_record(int m, int n)

{

std::ofstream fout("record.txt");

fout << m;

fout << " ";

fout << n;

return 1;

}

/\*

函数:save\_score()

功能:保存文件到score.dat

参数: 学生成绩指针StudentScore\*, 学生成绩条数n

\*/

void save\_score(StudentScore\* stu\_score,int n)

{

using namespace std;

ofstream fout("score.dat", ios::binary);

if (!fout)

{

cout << "创建文件失败\n";

return;

}

int i = 0;

for (int i = 0; i < n; i++)

{

fout << stu\_score[i].ID << " " << stu\_score[i].lesson\_number << " "

<< stu\_score[i].lesson\_name << " " << stu\_score[i].credit<< " "

<< stu\_score[i].daily\_score << " " << stu\_score[i].exp\_score<< " "

<< stu\_score[i].exam\_score << " " << stu\_score[i].total\_score<< " "

<< stu\_score[i].actual\_score << "\n";

}

fout.close();

}

/\*

函数:read\_to\_stu\_score()

功能:从文件score.dat读入分数到数组

参数:学生成绩数组StudentScore\*, 学生成绩条数&n

\*/

int read\_to\_stu\_score(StudentScore\* stu\_score, int &n)

{

using namespace std;

int i = 0;

ifstream fin("score.dat", ios::binary);

if (!fin)

{

cout << "读取成绩文件score.dat失败" << endl;

return 0;

}

for (int i = 0; i < n; i++)

{

fin >> stu\_score[i].ID >> stu\_score[i].lesson\_number

>> stu\_score[i].lesson\_name >> stu\_score[i].credit

>> stu\_score[i].daily\_score >> stu\_score[i].exp\_score

>> stu\_score[i].exam\_score >> stu\_score[i].total\_score

>> stu\_score[i].actual\_score;

}

fin.close();

return i;

}

/\*

函数:read\_to\_stu()

功能:从文件student.txt读入学生信息到数组

参数:学生数组Student\*, 学生信息条数m

\*/

int read\_to\_stu(Student\* stu, int &m)

{

using namespace std;

int i = 0;

ifstream fin("student.txt");

if (!fin)

{

cout << "读取学生基本文件student.txt失败" << endl;

return 0;

}

string temp;//临时string，读入第一行的5个字符串

for (int i = 0; i < 5; i++)

{

fin >> temp;

}

for (int i = 0; i < m; i++)

{

fin >> stu[i].ID >> stu[i].name >> stu[i].sex >> stu[i].dor\_number

>> stu[i].phone;

}

fin.close();

return 1;

}

/\*

函数:save\_stu

功能:写入学生信息到student.txt

参数:学生数组Student\*,学生信息条数m

\*/

void save\_stu(Student\* stu, int m)

{

using namespace std;

ofstream fout("student.txt");

if (!fout)

{

cout << "创建文件失败\n";

return;

}

fout << "学号 姓名 性别 宿舍号码 电话号码\n";

for (int i = 0; i < m; i++)

{

fout << stu[i].ID << " " << stu[i].name << " " << stu[i].sex << " "

<< stu[i].dor\_number << " " << stu[i].phone<<"\n";

}

fout.close();

}

/\*

·File name: func.h

·Description: 实现学生管理系统的相关函数的声明

\*/

#pragma once

#include<string>

#include"student.h"

int read\_data\_and\_calcu(StudentScore\* stu\_score, int & n);

int find(Student \* stu, StudentScore\* stu\_score,int m, int n);

void find\_baseinfo(Student\* stu, int n);

void find\_baseinfo\_by\_id\_name(Student\* stu, int m);

void find\_baseinfo\_by\_dor(Student\* stu, int m);

void find\_score(Student\* stu, StudentScore \* stu\_score,int m, int n);

int sort\_score(StudentScore \* stu\_score, int n);

int delete\_by\_ID(Student \* stu, StudentScore \* stu\_score, int &m, int &n);

/\*

·File name: menu.h

·Description: 菜单相关函数的声明

\*/

#pragma once

#include"student.h"

void show\_menu();

bool choose\_func(char choice, bool & active, StudentScore \* stu\_score, Student \* stu,

int & m, int& n);

/\*

·File name: student.h

·Description: 学生类和学生成绩类

\*/

#pragma once

#include<iostream>

#include<iomanip>

#include<string>

using std::string;

// StudentScore 存储学生成绩

class StudentScore

{

public:

int ID; //学号

string lesson\_number; //课程编号

string lesson\_name; //课程名称

int credit; //学分

int daily\_score; //平时成绩

int exp\_score; //实验成绩

int exam\_score; //卷面成绩

double total\_score; //综合成绩, 计算得到

double actual\_score; // 实际得分， 计算得到

public:

StudentScore() ;

StudentScore(int ID, string lesson\_number,string lesson\_name,int credit,

int daily\_score,int exp\_score,int exam\_score);

~StudentScore() ;

friend std::ostream& operator<<(std::ostream& os, const StudentScore& score);

};

//Student类，存储学生信息

class Student

{

public:

int ID; //学号

string name; //姓名

string sex; //性别

int dor\_number; //宿舍号码

int phone; //电话

public:

Student();

~Student();

friend std::ostream & operator<<(std::ostream & os, const Student& stu);

};

/\*

·File name: func.cpp

·Description: 实现菜单的4项功能：1.读入数据; 2.查找; 3.删除; 4.成绩排序

\*/

#include<iostream>

#include<iomanip>

#include<string>

#include<fstream>

#include"student.h"

using std::string;

using std::cin;

using std::cout;

//下面这4个函数被find调用，故先声明。

void find\_baseinfo(Student\* stu, int m);

void find\_baseinfo\_by\_id\_name(Student\* stu, int m);

void find\_baseinfo\_by\_dor(Student\* stu, int m);

void find\_score(Student\* stu, StudentScore \* stu\_score,int m,int n);

/\*

函数:read\_data\_and\_calcu()

功能:从键盘读入成绩相关的7项数据，并根据公式计算出综合成绩和实得学分。

参数：StudentScore\*（分数数组地址）， &n（已有的成绩条数）

返回值：写入的条数

\*/

int read\_data\_and\_calcu(StudentScore\* stu\_score, int & n)

{

int Max\_read = 1; //每次最多读入的条数

bool flag\_input = true;// 用来判断是否正常输入

int id = 0; //学号

string lesson\_number;//课程编号

string lesson\_name; //课程名称

int credit; //学分

int daily\_score; //平时成绩

int exp\_score; //实验成绩

int exam\_score; //卷面成绩

int i = 0;

for( i = n; i< n+Max\_read && flag\_input; i++)

{

cout << "开始输入学生数据：\n";

cout << "学号,"<< " 课程编号, " << " 课程名称: \n";

cin >> id;

cin >> lesson\_number;

cin >> lesson\_name;

cout << "学分, " << " 平时成绩, " << " 实验成绩," << " 卷面成绩:\n";

cin >> credit>> daily\_score >>exp\_score>> exam\_score;

stu\_score[i] = StudentScore(id, lesson\_number, lesson\_name, credit, daily\_score,

exp\_score, exam\_score);

cout << "录入该学生数据成功";

}

cout << "\n";

n += i;

return i;

}

/\*

函数:find()

功能:查询（总）,通过用户选择查询类型调用find\_baseinfo()或find\_score()，显示对应的信息。

参数：Student\*(学生数组地址), StudentScore\*（分数数组地址）,int m（学生信息条数）, int n（分数条数）

返回值：0

\*/

int find(Student \* stu,StudentScore\* stu\_score,int m, int n)

{

cout << "输入查询类型：1.学生基本情况查询； 2.成绩查询：";

char find\_kind;

cin >> find\_kind;

switch (find\_kind)

{

case '1':

//学生基本情况查询

find\_baseinfo( stu, m);

break;

case '2':

//成绩查询

find\_score(stu, stu\_score,m, n);

break;

default:

cout<<"无效选择\n";

break;

}

cout << "\n";

return 0;

};

/\*

函数:find\_baseinfo()

功能:查询\_学生基本信息,被find()调用，显示对应的学生信息

参数：Student\*, m

\*/

void find\_baseinfo(Student\* stu, int m)

{

cout << "输入查询方式：1.按学号或姓名查询 2.按宿舍查询: ";

int choice;

cin >> choice;

switch (choice)

{

case 1:

find\_baseinfo\_by\_id\_name(stu, m);

break;

case 2:

find\_baseinfo\_by\_dor(stu, m);

break;

}

}

/\*

函数:find\_baseinfo\_by\_id\_name()

功能:查询\_学生基本信息\_通过\_id或name,被find\_baseinfo()调用

参数:Student\*, m

\*/

void find\_baseinfo\_by\_id\_name(Student\* stu, int m)

{

cout << "选择：1.使用学号查询 2.使用姓名: ";

int choice\_2;

cin >> choice\_2;

switch (choice\_2)

{

case 1:

{

//按学号查询，打印结果

cout << "输入学号: ";

int id;

cin >> id;

bool flag\_find\_id = false;

for (int i = 0; i < m; i++)

{

if (id == stu[i].ID)

{

flag\_find\_id = true;

cout << "学号 姓名 性别 宿舍号码 电话号码\n";

cout << stu[i];

break;

}

if (i == m - 1 && !flag\_find\_id )

cout << " 此ID记录不存在\n";

}

}

break;

case 2:

{

//使用姓名查询，打印结果

cout << "输入姓名: ";

string name;

cin >> name;

bool flag\_find\_name = false;

for (int i = 0; i < m; i++)

{

if (name == stu[i].name)

{

flag\_find\_name = true;

cout << "学号 姓名 性别 宿舍号码 电话号码\n";

cout << stu[i];

break;

}

if (i == m - 1 && !flag\_find\_name)

cout << " 此姓名记录不存在\n";

}

break;

}

}

}

/\*

函数:find\_baseinfo\_by\_dor()

功能:查询\_学生基本信息\_通过\_宿舍,被find\_baseinfo()调用

参数:Stuent\*, m

\*/

void find\_baseinfo\_by\_dor(Student\* stu, int m)

{

//按宿舍查询，打印结果

cout << "输入宿舍号: ";

int dor\_number;

cin >> dor\_number;

cout << "学号 姓名 性别 宿舍号码 电话号码\n";

bool flag\_find\_dor = false;

for (int i = 0; i < m; i++)

{

if (dor\_number == stu[i].dor\_number)

{

flag\_find\_dor = true;

cout << stu[i];

cout << "\n";

}

if (i == m - 1 && !flag\_find\_dor )

cout << " 宿舍记录不存在\n";

}

}

/\*

函数:find\_score()

功能:查询\_分数,被find()调用，显示对应的分数信息

参数：Student\*, StudentScore\*,m(学生信息条数),n(分数信息条数)

\*/

void find\_score(Student\* stu, StudentScore\* stu\_score, int m, int n)

{

cout << "输入该学生的学号:";

int id;

cin >> id;

bool flag\_find\_id = false;

for (int i = 0; i < m; i++)

{

if (id == stu[i].ID)

{

flag\_find\_id = true;

cout << "\t\t\t\t学号: "<<std::setiosflags(std::ios::left)

<<std::setw(10)<< stu[i].ID << "姓名: "

<<std::setw(10)<< stu[i].name;

break;

}

if (i == m - 1 && !flag\_find\_id)

{

cout << "\n\t\t此ID学生信息记录不存在\n";

}

}

bool flag\_find\_score = false;

int lesson\_n = 0;

double all\_credit = 0;

for (int i = 0; i < n; i++)

{

if (id == stu\_score[i].ID)

{

flag\_find\_score = true;

cout << "\n" << stu\_score[i];

lesson\_n++;

all\_credit += stu\_score[i].actual\_score;

}

if (i == n - 1 && !flag\_find\_score)

{

cout << "\n\t\t此ID成绩记录不存在\n";

}

}

cout << "\n\t\t共修: " << lesson\_n << "科，实得总学分为: " << all\_credit;

}

/\*

函数:delete\_by\_ID()

功能:删除和ID相关的信息

返回值:0

\*/

int delete\_by\_ID(Student \* stu, StudentScore \* stu\_score, int &m, int &n)

{

cout << "删除该学号的所有相关信息：\n输入学号：";

int ID;

cin >> ID;

//删除ID相关的学生基本信息

for (int i = 0; i < m; i++)

{

if (stu[i].ID == ID)

{

int k = i;

if (k == m - 1)

{

stu[i] = Student(); //使其为空

m--;

}

else

{

for (int j = k+1; j < m; j++)

{

stu[j-1] = stu[j];

}

m--;

}

}

}

//删除ID相关的成绩信息

for (int i = n-1; i >= 0; i--)

{

if (stu\_score[i].ID == ID)

{

int k = i;

if (k == n - 1)

{

stu\_score[i] = StudentScore(); //使其为空

n--;

}

else

{

for (int j = k + 1; j < n; j++)

{

stu\_score[j - 1] = stu\_score[j];

}

n--;

}

}

}

cout << "该学生所有相关信息删除成功\n";

return 0;

};

/\*

函数:sort\_score()

功能:按分数排序,(降序)。

参数：StudentScore\*, n(成绩条数)

返回值:n(成绩条数)

\*/

int sort\_score(StudentScore \* stu\_score, int n)

{

int choice;

cout << "选择：1.按综合成绩排序 2.按实得学分排序";

cin >> choice;

switch (choice)

{

case 1:

{

StudentScore temp;

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (stu\_score[j].total\_score < stu\_score[j + 1].total\_score)

{

temp = stu\_score[j];

stu\_score[j] = stu\_score[j+1];

stu\_score[j + 1] = temp;

}

}

}

break;

}

case 2:

{

StudentScore temp;

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - i - 1; j++)

{

if (stu\_score[j].actual\_score < stu\_score[j + 1].actual\_score)

{

temp = stu\_score[j];

stu\_score[j] = stu\_score[j + 1];

stu\_score[j + 1] = temp;

}

}

}

break;

}

}

for (int i = 0; i < n; i++)

{

cout<<"学号: "<<std::setiosflags(std::ios::left)<<std::setw(4)

<<stu\_score[i].ID<<stu\_score[i] << "\n";

}

return n;

};

/\*

·File name: menu.cpp

·Description: 所有和菜单相关的操作。

\*/

#include<iostream>

#include"student.h"

#include"func.h"

using std::cout;

using std::cin;

/\*

函数:show\_menu()

功能:显示主菜单

\*/

void show\_menu()

{

cout<<"############################\n";

cout<<" 学生信息管理系统\n"

<<" 选择功能\n"

<<"1.数据的录入和计算\n"

<<"2.查询功能\n"

<<"3.删除功能\n"

<<"4.排序功能\n"

<<"q.退出程序。\n";

}

/\*

函数:choose\_func

功能:读入选择，执行功能

返回:程序状态(运行ture，结束false)

\*/

bool choose\_func(char choice, bool & active,StudentScore \* stu\_score,Student \* stu,

int & m, int& n )

{

switch(choice)

{

case '1' :

n = read\_data\_and\_calcu(stu\_score,n);

break;

case '2':

find( stu, stu\_score,m, n);

break;

case '3':

delete\_by\_ID(stu, stu\_score, m, n);

break;

case '4':

sort\_score(stu\_score, n);

break;

case 'q':

cout<<"quit~\n";

active = false;

break;

default:

cout << "无效输入\n";

}

return active;

}

/\*

·File name: student.cpp

·Description: 学生类和成绩类的相关函数。

\*/

#include"student.h"

/\*

计算综合成绩，返回综合成绩。

传入参数: exp\_score, exam\_score, daily\_score

返回值: total\_score

\*/

double calcu\_total\_score(int exp\_score, int exam\_score, int daily\_score);

/\*

计算实得学分，返回实得学分

传入参数: total\_score, credit

返回值: actual\_score

\*/

double calcu\_actual\_score(double total\_score, int credit);

//分数的默认构造函数，初始化ID = 0，分数为负数，表示无效成绩;

StudentScore::StudentScore() : ID(0),lesson\_number("None"),lesson\_name("None"),credit(-1),

daily\_score(-1),exp\_score(-1),exam\_score(-1)

{

}

//分数的构造函数2，传入参数进行初始化，自动调用calcu\_total\_score，calcu\_actual\_score计算 综合成绩和实得学分

StudentScore::StudentScore(int m\_ID, string m\_lesson\_number, string m\_lesson\_name, int m\_credit,

int m\_daily\_score, int m\_exp\_score, int m\_exam\_score) : ID(m\_ID),lesson\_number(m\_lesson\_number),

lesson\_name(m\_lesson\_name),credit(m\_credit),daily\_score(m\_daily\_score),

exp\_score(m\_exp\_score),exam\_score(m\_exam\_score)

{

total\_score = calcu\_total\_score( m\_exp\_score, m\_exam\_score, m\_daily\_score);

actual\_score = calcu\_actual\_score( total\_score, m\_credit);

}

//分数的析构函数，暂无工作要做

StudentScore::~StudentScore()

{

}

//分数的友元函数<<，用于显示

std::ostream& operator<<(std::ostream& os, const StudentScore& stu\_score)

{

os << std::setiosflags(std::ios::left)

<<"课程编号: " <<std::setw(8)<< stu\_score.lesson\_number

<< "课程名称: "<<std::setw(12) << stu\_score.lesson\_name

<< "综合成绩: "<<std::setw(8) << stu\_score.total\_score

<< "实得学分: " << stu\_score.actual\_score;

return os;

}

//学生的默认构造函数，初始化0

Student::Student(): ID(0), name("None"), sex("None"), dor\_number(0), phone(0)

{

}

//学生的析构函数，暂无工作要做

Student::~Student()

{

}

//学生的友元函数<< 用于打印输出.

std::ostream & operator<<(std::ostream & os, const Student& stu)

{

os <<std::setiosflags(std::ios::left)

<<std::setw(5)<< stu.ID

<<std::setw(8)<< stu.name

<< stu.sex << " " << stu.dor\_number << " "

<< stu.phone;

return os;

}

// 计算实得学分

double calcu\_actual\_score(double total\_score, int credit)

{

double actual\_score;

int flag\_total =total\_score/10;

switch (flag\_total)

{

case 10: case 9: actual\_score = credit;

break;

case 8: actual\_score = credit \* 0.8;

break;

case 7: actual\_score = credit \* 0.75;

break;

case 6: actual\_score = credit \* 0.6;

break;

default: actual\_score = 0;

break;

}

return actual\_score ;

}

//计算综合成绩

double calcu\_total\_score(int exp\_score, int exam\_score, int daily\_score)

{

double total\_score;

if(exp\_score == -1)

total\_score = daily\_score \* 0.3 + exam\_score \* 0.7;

else

total\_score = daily\_score \* 0.15 + exp\_score \* 0.15 + exam\_score \* 0.7;

return total\_score;

}

/\*

# File name: main.cpp

# Author: 李金峰

# Date: 2019/6/14

# Description: 学生信息管理系统的主函数

\*/

#include<iostream>

#include"menu.h"

#include"student.h"

#include"file.h"

//主函数main,程序入口

int main()

{

char choice = 0; //choice，让用户输入选择

bool active = true; //active,用来判断程序运行状态，true代表运行中，false代表结束

int m = 0; //学生基本信息 数量

int n = 0; //成绩信息 数量

read\_record(m, n);//得到学生基本信息、成绩信息条数

Student \* stu = new Student[m+10]; //学生数组stu，存储学生信息

StudentScore \* stu\_score = new StudentScore[n+400];//分数数组stu\_score，存储成绩信息

read\_to\_stu(stu, m); //从student.txt读入数据到数组中

read\_to\_stu\_score(stu\_score, n); //从score.dat读入数据到数组中

//显示主菜单

show\_menu();

//进入循环，用户选择菜单上的功能。直到active为false，程序即将结束时，退出循环

while(active)

{

std::cin>>choice;

choose\_func(choice,active,stu\_score,stu,m,n);

if(active)

show\_menu();

}

/\*

程序即将结束，将学生信息写入student.txt;将分数信息写入score.dat;

写入信息数量到record.txt中;

\*/

save\_stu(stu, m);

save\_score(stu\_score, n);

write\_record(m, n);

//归还使用的内存

delete[] stu;

delete[] stu\_score;

system("pause"); //暂停

return 0;

}