

《C++面向对象程序设计》

课程设计报告

**评分：**

**评语：**

**专业年级：** 2022级软件工程

**学生姓名:**  邓海涛 **学 号:** 20224324

**课设题目：** 学生成绩管理系统

**指导教师:**  蒋湘涛

**日 期：** 2024.5.18

一、设计题目及要求

问题描述：设学生信息包括：学号、姓名、学期、每门课程的成绩(每学期的课程门数是不一样的) ，对学生的成绩信息进行管理。

实现要求：

实现：学生信息的录入；修改；删除和查询，按学期、学号、成绩不及格等查询。

⑴输入学生的成绩信息，包含学号、姓名、性别等基本信息和各课成绩

⑵显示全部学生各科成绩信息；

⑶对各科成绩统计分析（总分、平均分、最高分、最低分、及格率等）；

⑷统计各科各分数段人数；

⑸按学号或姓名查找并显示某个学生的各科成绩；

⑹按课程成绩或总分由高到低排序显示；

⑺更新某个学生的基本信息或课程成绩；

⑻设计一个字符菜单界面，具有上述规定的操作要求、退出系统等最基本的功能。

二、总体分析与设计

（一）、将代码分为主函数main、管理类StudentManager、学生类Student

1、主函数负责直接使用管理类

2、学生类负责定义数据：

数据：学号、姓名、性别、学期、课程数量、成绩

接口：用于对外公开以及设置的set、get函数接口

3、管理类对学生类进行管理：

数据成员：有一个vector的容器，用来对存储学生、还有一个函数指针、用来调用接口

接口：退出、录入、显示、统计学生成绩、统计单科成绩、查找、排序、更新、菜单

（二）、管理类接口的实现思想

1、排序接口：利用了algorithm中的sort算法、其中的谓词需要自己实现、可以选择升序或者降序

2、查找接口：使用vector容器的迭代器用for循环进行遍历，若满足条件即插入到temp容器，最终进行返回

3、文件保存及录入：利用fstream头文件，将数据保存到指定文件中，在下一次启动时在构造函数中就可以重新读取其中的数据

4、System接口：利用函数指针，可以简便的对各个接口进行调用，其中需要注意的是对于各指针的赋值，例如ptr[0]=&StudentManager::Exit;

三、详细设计

**main.cpp**

#include"StudentManager.h"

int main() {

StudentManager manager;

manager.System();

return 0;

}

**Student.h**

#pragma once

#include<string>

#include<map>

using namespace std;

class Student {

public:

Student(map<string, int> grades);

Student(int id, string name, string sex, int term, int courseCount, map<string, int> grades);

int GetId();

void SetId(int id);

string GetName();

void SetName(string name);

void SetSex(string sex);

string GetSex();

int GetTerm();

void SetTerm(int term);

int GetCourseCount();

void SetCourseCount(int courseCount);

//int\* GetGrades();

void SetGrades(map<string, int> grades);

map<string, int> GetGrades();

private:

int id; //学号

string name; //姓名

string sex; //性别

int term; //学期

int courseCount; //课程数量

map<string, int> grades;

static int stuCount;

};

**Student.cpp**

#include"Student.h"

Student::Student(map<string, int> grades) {

this->grades = grades;

}

Student::Student(int id, string name, string sex, int term, int courseCount, map<string, int> grades)

{

this->id = id;

this->name = name;

this->sex = sex;

this->term = term;

this->courseCount = courseCount;

this->grades = grades;

stuCount++;

}

int Student::stuCount = 0;

int Student::GetId() {

return id;

}

void Student::SetId(int id) {

this->id = id;

}

string Student::GetName() {

return name;

}

void Student::SetName(string name) {

this->name = name;

}

void Student::SetSex(string sex) {

this->sex = sex;

}

string Student::GetSex() {

return this->sex;

}

int Student::GetTerm() {

return term;

}

void Student::SetTerm(int term) {

this->term = term;

}

int Student::GetCourseCount() {

return courseCount;

}

void Student::SetCourseCount(int courseCount) {

this->courseCount = courseCount;

}

void Student::SetGrades(map<string, int> grades) {

this->grades = grades;

}

map<string, int> Student::GetGrades() {

return this->grades;

}

**StudentManager.h**

#pragma once

#include<iostream>

#include"Student.h"

#include<vector>

#include<fstream>

#include<algorithm>

using namespace std;

#define STUDENTFILE "student\_file.txt"

class StudentManager

{

public:

StudentManager();

void System(); //管理接口

//菜单

void Menu();

//退出

void Exit();

//录入学生信息

void AddInfo();

//更新学生信息

void UpdateStu();

//删除

void DeleteInfo();

//列出所有学生信息

void DisplayAllInfo();

//显示指定学生信息

void DisplayTargetInfo();

//查询

vector<Student> AcquireStu\_Name(string name); //by Term, Id, Failing grades

vector<Student> AcquireStu\_Id(int id);

vector<Student> AcquireStu\_Term(int term);

vector<Student> AcquireStu\_Fail(string course);

//打印

void CoutInfo(vector<Student>::iterator it);

//统计成绩（总分、平均分、最高分、最低分、及格率）

void StatsGrade();

//统计一科分数段人数

void StateCountInGradeRange(string course);

void StateCount();

//成绩排序

void Sort();

//对总分进行排序

void SortByTotal();

//对某一课程进行排序

void SortByCur(string course);

//保存信息

void SaveInfo(vector<Student> stus);

//读取信息

void ReadInfo();

private:

vector<Student> stus;

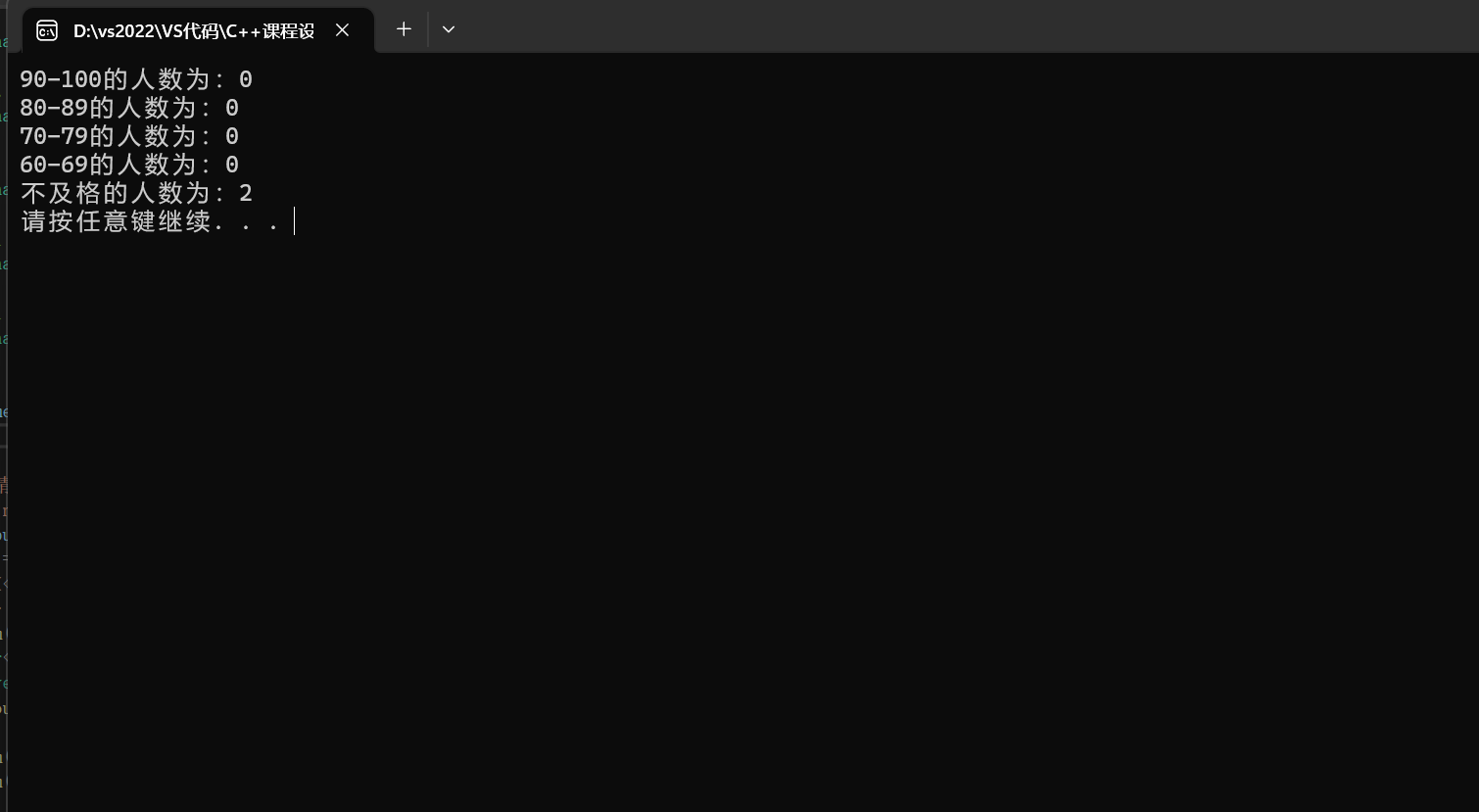
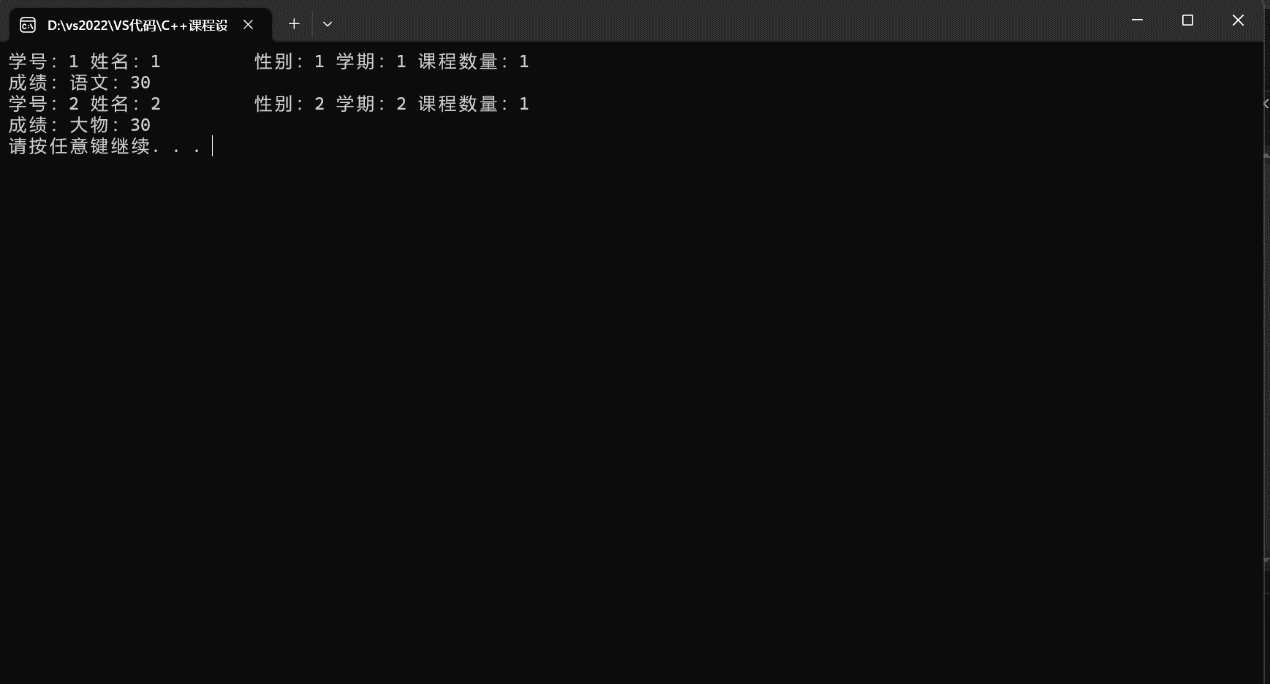
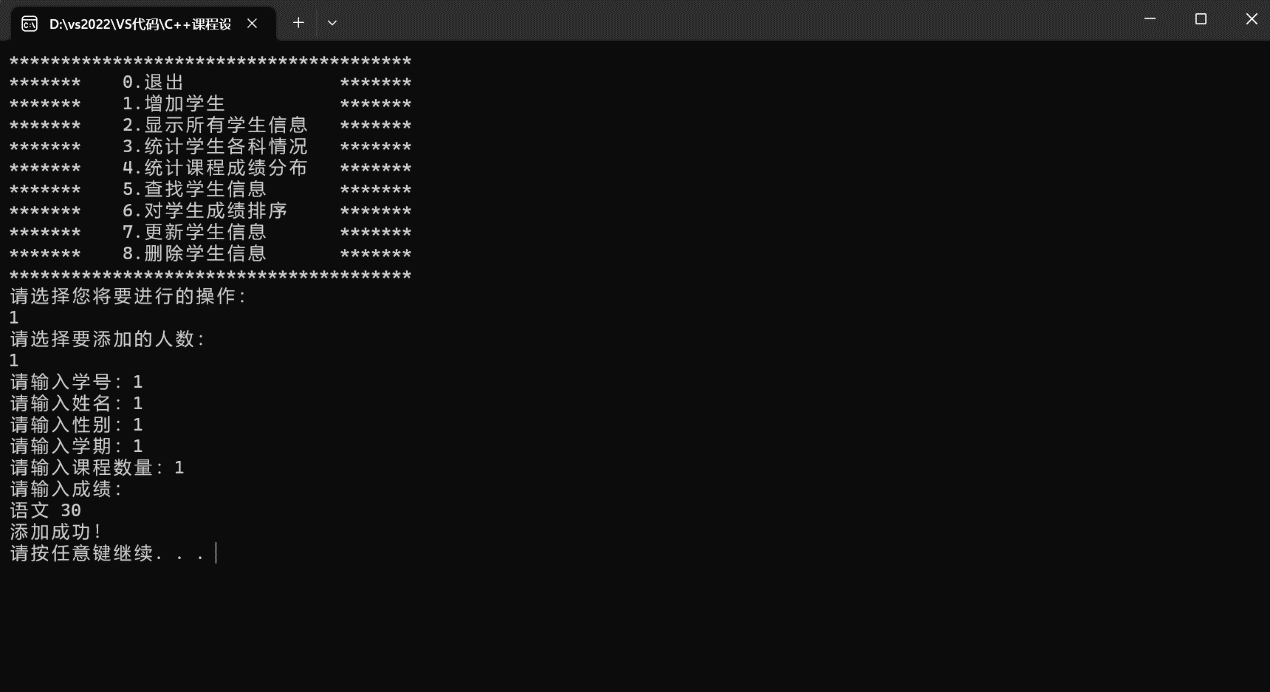
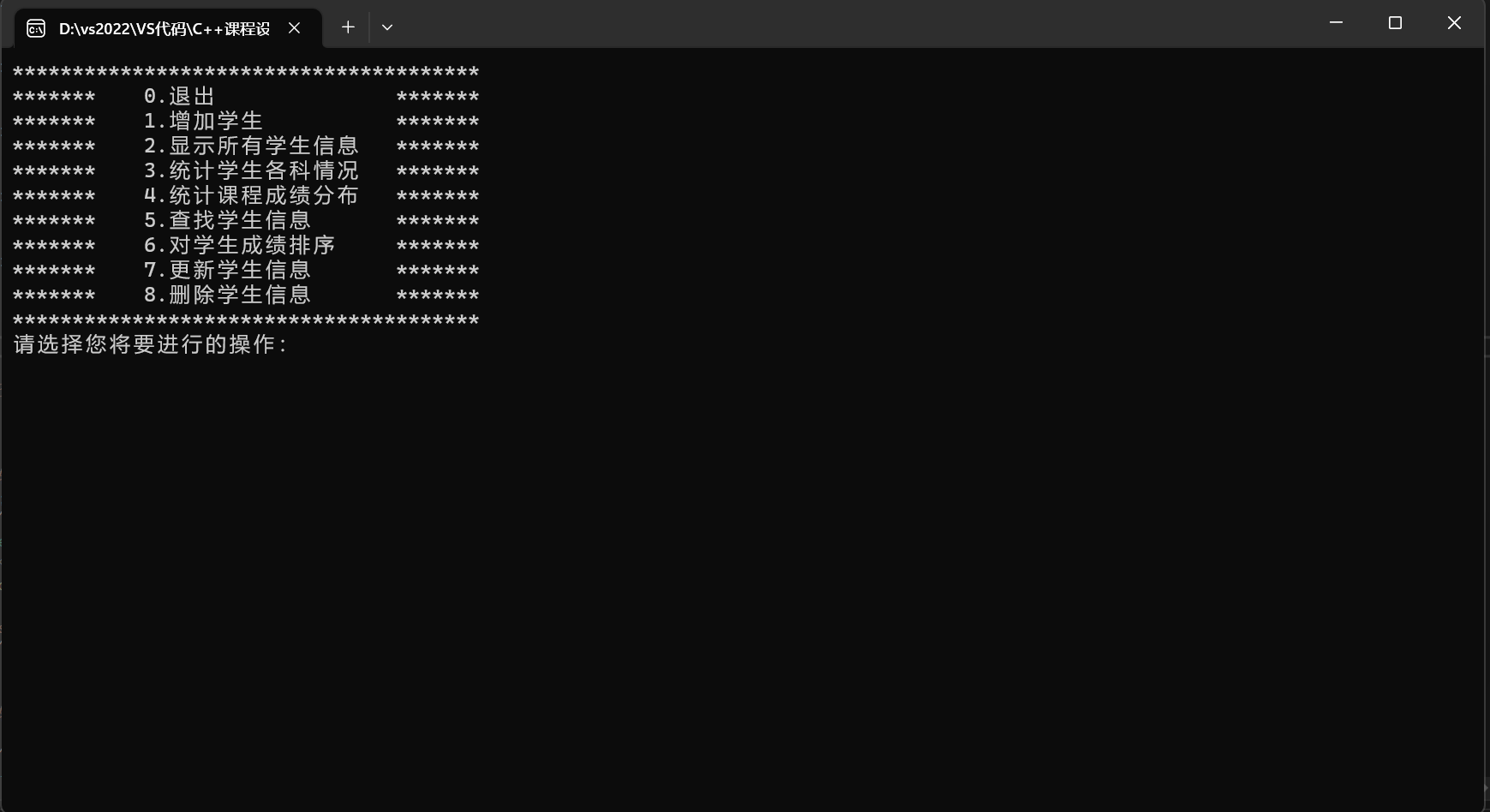
void (StudentManager::\*ptr[9])();

};

**StudentManager.cpp**

1. #include "StudentManager.h"
3. //排序的指定科目名称
4. **static** string targetCour;
6. //构造函数
7. StudentManager::StudentManager()
8. {
9. // 初始化函数指针数组，将每个指针指向一个成员函数
10. ptr[0] = &StudentManager::Exit;
11. ptr[1] = &StudentManager::AddInfo;
12. ptr[2] = &StudentManager::DisplayAllInfo;
13. ptr[3] = &StudentManager::StatsGrade;
14. ptr[4] = &StudentManager::StateCount;
15. ptr[5] = &StudentManager::DisplayTargetInfo;
16. ptr[6] = &StudentManager::Sort;
17. ptr[7] = &StudentManager::UpdateStu;
18. ptr[8] = &StudentManager::DeleteInfo;
19. // 读取学生信息
20. ReadInfo();
21. }
23. // 管理接口
24. **void** StudentManager::System()
25. {
26. **int** input = 1;
27. **while** (input) {
28. // 显示菜单
29. Menu();
30. **int** select;
31. cout << "请选择您将要进行的操作：" << endl;
32. cin >> select;
33. // 检查用户输入是否在有效范围内
34. **if** (select < 0 || select > 8) {
35. cout << "选择错误，请重新选择：" << endl;
36. system("pause");
37. system("cls");
38. **continue**;
39. }
41. // 调用用户选择的操作对应的成员函数
42. (**this**->\*ptr[select])();
43. }
44. }
46. //菜单
47. **void** StudentManager::Menu()
48. {
49. //打印出一个菜单，列出用户可以选择的操作
50. cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl
51. << "\*\*\*\*\*\*\*    0.退出           \*\*\*\*\*\*\*" << endl
52. << "\*\*\*\*\*\*\*    1.增加学生     \*\*\*\*\*\*\*" << endl
53. << "\*\*\*\*\*\*\*    2.显示所有学生信息 \*\*\*\*\*\*\*" << endl
54. << "\*\*\*\*\*\*\*    3.统计学生各科情况 \*\*\*\*\*\*\*" << endl
55. << "\*\*\*\*\*\*\*    4.统计课程成绩分布 \*\*\*\*\*\*\*" << endl
56. << "\*\*\*\*\*\*\*    5.查找学生信息   \*\*\*\*\*\*\*" << endl
57. << "\*\*\*\*\*\*\*    6.对学生成绩排序  \*\*\*\*\*\*\*" << endl
58. << "\*\*\*\*\*\*\*    7.更新学生信息   \*\*\*\*\*\*\*" << endl
59. << "\*\*\*\*\*\*\*    8.删除学生信息   \*\*\*\*\*\*\*" << endl
60. << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
61. }
63. //退出
64. **void** StudentManager::Exit()
65. {
66. //退出程序
67. cout << "退出程序，欢迎下次使用！" << endl;
68. exit(0);
69. }
71. // 添加学生信息
72. **void** StudentManager::AddInfo()
73. {
74. **int** num;
75. cout << "请选择要添加的人数：" << endl;
76. cin >> num;
78. **for** (**int** i = 0; i < num; i++) {
79. **int** id;
80. string name;
81. string sex;
82. **int** term;
83. **int** courseCount;
84. map<string, **int**> grades;
86. cout << "请输入学号：";
87. cin >> id;
89. cout << "请输入姓名：";
90. cin >> name;
92. cout << "请输入性别：";
93. cin >> sex;
95. cout << "请输入学期：";
96. cin >> term;
98. cout << "请输入课程数量：";
99. cin >> courseCount;
101. **for** (**int** j = 0; j < courseCount; j++) {
102. string course;
103. **int** grade;
104. cout << "请输入第 " << (j + 1) << " 门课的成绩：";
105. cin >> course >> grade;
107. // 检查成绩是否在有效范围内
108. **while** (grade < 0 || grade > 100) {
109. cout << "输入错误，请重新输入成绩（0-100）：";
110. cin >> grade;
111. }
113. grades.insert(make\_pair(course, grade));
114. }
116. // 创建新的学生对象并添加到列表
117. Student stu(id, name, sex, term, courseCount, grades);
118. stus.push\_back(stu);
120. // 保存学生信息
121. SaveInfo(stus);
123. cout << "学生信息添加成功！" << endl;
124. }
126. // 清屏
127. system("cls");
128. }
130. //更新学生信息
131. **void** StudentManager::UpdateStu()
132. {
133. **int** id;
134. cout << "请输入学生id：";
135. cin >> id;
137. **int** id1;
138. string name;
139. string sex;
140. //int term;
141. **int** courseCount;
142. map<string, **int**> grades;
144. **for** (vector<Student>::iterator it = stus.begin(); it != stus.end(); it++) {
145. **if** ((\*it).GetId() == id) {
146. CoutInfo(it);
148. cout << "请输入新的学号：";
149. cin >> id1;
150. cout << "请输入新的姓名：";
151. cin >> name;
152. cout << "请输入性别：";
153. cin >> sex;
154. //cout << "请输入学期：";
155. //cin >> term;
156. cout << "请输入课程数量：";
157. cin >> courseCount;
158. cout << "请输入成绩：" << endl;
159. **for** (**int** j = 0; j < courseCount; j++) {
160. string course;
161. **int** grade;
162. cin >> course;
163. **while** (1) {
164. cin >> grade;
165. **if** (0 <= grade && grade <= 100)
166. **break**;
167. **else**
168. cout << "输入错误，请重新输入：";
169. }
170. grades.insert(make\_pair(course, grade));
171. }
172. it->SetId(id1);
173. it->SetName(name);
174. it->SetSex(sex);
175. it->SetCourseCount(courseCount);
176. it->SetGrades(grades);
178. cout << "更新学号为:"<<id1<<"的学生信息成功！" << endl;
179. system("pause");
180. system("cls");
181. }
182. }
183. SaveInfo(stus);
184. }
186. //删除
187. **void** StudentManager::DeleteInfo()
188. {
189. **int** id;
190. cout << "请输入需要删除学生的id：";
191. cin >> id;
192. **int** pos = 0;
193. **for** (vector<Student>::iterator it = stus.begin(); it != stus.end(); it++) {       //遍历学生信息
195. **if** ((\*it).GetId() == id) {
196. stus.erase(stus.begin() + pos);
197. cout << "删除学号为:"<<id<<"的学生信息成功！" << endl;
198. system("pause");
199. system("cls");
200. **break**;
201. }
202. pos++;
203. }
204. SaveInfo(stus);
205. }
207. //显示所有学生
208. **void** StudentManager::DisplayAllInfo()
209. {
210. system("cls");
211. **for** (vector<Student>::iterator it = stus.begin(); it != stus.end(); it++) {
212. CoutInfo(it);
213. }
214. system("pause");
215. system("cls");
216. }
218. //显示指定学生
219. **void** StudentManager::DisplayTargetInfo()
220. {
221. **int** id;
222. string name;
223. **int** input;
224. **int** term;
225. cout << "请选择查找方式：" << endl
226. << "1.name        2.id        3.term" << endl;
227. cin >> input;
228. **if** (input == 1) {
229. cout << "请输入要查找学生的姓名：";
230. cin >> name;
231. system("cls");
232. vector<Student> stusTemp = AcquireStu\_Name(name);
233. **for** (vector<Student>::iterator it = stusTemp.begin(); it != stusTemp.end(); it++) {
234. CoutInfo(it);
235. }
236. system("pause;");
237. system("cls");
238. }
239. **if** (input == 2) {
240. cout << "请输入要查找学生的学号：";
241. cin >> id;
242. system("cls");
243. vector<Student> stusTemp = AcquireStu\_Id(id);
244. **for** (vector<Student>::iterator it = stusTemp.begin(); it != stusTemp.end(); it++) {
245. CoutInfo(it);
246. }
247. system("pause;");
248. system("cls");
249. }
250. **if** (input == 3) {
251. cout << "请输入要查找的学期：";
252. cin >> term;
253. system("cls");
254. vector<Student> stusTemp = AcquireStu\_Term(term);
255. **for** (vector<Student>::iterator it = stusTemp.begin(); it != stusTemp.end(); it++) {
256. CoutInfo(it);
257. }
258. system("pause;");
259. system("cls");
260. }
261. }
263. //获取指定学生
264. //按姓名
265. vector<Student> StudentManager::AcquireStu\_Name(string name)
266. {
267. vector<Student> temp;
268. **for** (vector<Student>::iterator it = stus.begin(); it != stus.end(); it++) {
269. **if** ((\*it).GetName() == name) {
270. temp.push\_back(\*it);
271. }
272. }
273. **return** temp;
274. }
275. //按学号
276. vector<Student> StudentManager::AcquireStu\_Id(**int** id)
277. {
278. vector<Student> temp;
279. **for** (vector<Student>::iterator it = stus.begin(); it != stus.end(); it++) {
280. **if** ((\*it).GetId() == id) {
281. temp.push\_back(\*it);
282. }
283. }
284. **return** temp;
285. }
287. //按学期
288. vector<Student> StudentManager::AcquireStu\_Term(**int** term) {
289. vector<Student> temp;
290. **for** (vector<Student>::iterator it = stus.begin(); it != stus.end(); it++) {
291. **if** (it->GetTerm() == term) {
292. temp.push\_back(\*it);
293. }
294. }
295. **return** temp;
296. }
297. //按是否及格
298. vector<Student> StudentManager::AcquireStu\_Fail(string course) {
299. vector<Student> temp;
300. **for** (vector<Student>::iterator it = stus.begin(); it != stus.end(); it++) {
301. **if** (it->GetGrades().find(course)->second < 60) {
302. temp.push\_back(\*it);
303. }
304. }
305. **return** temp;
306. }
308. //打印学生信息
309. **void** StudentManager::CoutInfo(vector<Student>::iterator it)
310. {
311. map<string, **int**> grades = it->GetGrades();     //没有此行会出现越界访问，就是指针已经销毁任然访问
312. cout << "学号：" << it->GetId()
313. << "\t姓名：" << it->GetName()
314. << " \t性别：" << it->GetSex()
315. << "\t学期：" << it->GetTerm()
316. << "\t课程数量：" << it->GetCourseCount() << endl;
317. //for (map<string, int>::iterator it1 = it->GetGrades().begin(); it1 != it->GetGrades().end(); it1++) {
318. cout << "成绩：";
319. **for** (map<string, **int**>::iterator it1 = grades.begin(); it1 != grades.end(); it1++) {
320. cout << it1->first << "：" << it1->second << " ";
321. }
322. cout << endl;
323. }
325. //统计学生成绩
326. **void** StudentManager::StatsGrade()
327. {
328. **int** id;
329. cout << "请输入学生id：";
330. cin >> id;
331. vector<Student> s = AcquireStu\_Id(id);
332. **if** (s.size() == 0) {
333. cout << "查无此人！" << endl;
334. system("pause");
335. system("cls");
336. **return**;
337. }
338. map<string, **int**> grades = s[0].GetGrades();
340. **int** total = 0;
341. **int** average = 0;
342. **int** maxGrade = 0;
343. **int** minGrade = 0;
344. **float** passRate = 0;
345. **int** failCount = 0;
346. **int** courseCount = s[0].GetCourseCount();
348. map<string, **int**>::iterator it;    //迭代器
349. **for** (it = grades.begin(); it != grades.end(); it++) {
350. **int** grade = it->second;
351. total += grade;
353. maxGrade = maxGrade > grade ? maxGrade : grade;
354. minGrade = minGrade < grade ? minGrade : grade;
356. **if** (grade < 60)
357. failCount++;
358. }
359. average = total / courseCount; //平均分
360. passRate = (courseCount - failCount) / courseCount; //及格率
361. system("cls");
362. cout << "总分为：" << total
363. << "  最高分为：" << maxGrade
364. << "  最低分为：" << minGrade
365. << "  及格率为：" << passRate << endl;
366. system("pause");
367. system("cls");
368. }
370. //统计学科成绩
371. **void** StudentManager::StateCountInGradeRange(string course) {
372. **int** gradeRange[5] = { 0, 0, 0, 0, 0 };  //不及格 60-69 70-79 80-89 90-100
373. **for** (vector<Student>::iterator it = stus.begin(); it!= stus.end(); it++) {
374. map<string, **int**> grades = it->GetGrades(); //没有此行会出现越界访问，就是指针已经销毁任然访问
375. **int** grade = grades[course]; //获取指定课程的成绩
376. **if** (grade >= 90)
377. gradeRange[0]++;
378. **else** **if** (80 <= grade && grade < 90)
379. gradeRange[1]++;
380. **else** **if** (70 <= grade && grade < 80)
381. gradeRange[2]++;
382. **else** **if** (60 <= grade && grade < 70)
383. gradeRange[3]++;
384. **else**
385. gradeRange[4]++;
386. }
387. system("cls");
388. cout << "90-100的人数为：" << gradeRange[0] << endl
389. << "80-89的人数为：" << gradeRange[1] << endl
390. << "70-79的人数为：" << gradeRange[2] << endl
391. << "60-69的人数为：" << gradeRange[3] << endl
392. << "不及格的人数为：" << gradeRange[4] << endl;
393. system("pause");
394. system("cls");
395. }
396. **void** StudentManager::StateCount() { //统计课程成绩分布
397. string course;
398. cout << "请输入需要统计的科目：";
399. cin >> course;
400. StateCountInGradeRange(course);
401. system("cls");
402. }
404. //保存信息
405. **void** StudentManager::SaveInfo(vector<Student> stus)
406. {
407. ofstream ofs;   //输出文件流
408. ofs.open(STUDENTFILE, ios::out);    //打开文件
409. **for** (vector<Student>::iterator it = stus.begin(); it != stus.end(); it++) {   //遍历学生信息
410. **int** count = it->GetCourseCount();    //获取课程数量
411. map<string, **int**> grades = it->GetGrades(); //获取成绩
413. ofs << it->GetId() << " "    //写入文件
414. << it->GetName() << " "
415. << it->GetSex() << " "
416. << count << " ";
418. //for (int i = 0; i < count; i++) {
419. //  ofs << grades[i] << " ";
420. //}
421. **for** (map<string, **int**>::iterator it1 = grades.begin(); it1 != grades.end(); it1++) {   //遍历成绩
422. ofs << it1->first << " " << it1->second << " ";
423. }
424. ofs << it->GetTerm() << endl;
425. }
427. ofs.close();
428. }
430. //降序排序
431. **bool** TotalDown(Student s1, Student s2) {
432. map<string, **int**> grades1 = s1.GetGrades();
433. map<string, **int**> grades2 = s2.GetGrades();
434. **int** total1 = 0;
435. **for** (map<string, **int**>::iterator it = grades1.begin(); it != grades1.end(); it++) {
436. total1 += it->second;
437. }
438. **int** total2 = 0;
439. **for** (map<string, **int**>::iterator it = grades2.begin(); it != grades2.end(); it++) {
440. total2 += it->second;
441. }
442. **return** total1 > total2;
443. }
444. **bool** CourDown(Student s1, Student s2) {
445. map<string, **int**> grades1 = s1.GetGrades();
446. map<string, **int**> grades2 = s2.GetGrades();
447. **return** grades1[targetCour] > grades2[targetCour];
448. }
449. //排序
450. **void** StudentManager::Sort()
451. {
452. **int** input;
453. cout << "请选择排序的方式：" << endl;
454. cout << "1.总分  2.单科课程" << endl;
455. cin >> input;
456. **if** (input == 1) {
457. SortByTotal();
458. }
459. **else** **if** (input == 2) {
460. string course;
461. cout << "请输入课程名称：";
462. cin >> course;
463. SortByCur(course);
464. }
465. }
466. //按总分
467. **void** StudentManager::SortByTotal()
468. {
469. //按总分降序排序
470. sort(stus.begin(), stus.end(), TotalDown);
471. //cout << "排序结果为：" << endl;
472. DisplayAllInfo();
473. }
474. //按单科
475. **void** StudentManager::SortByCur(string course)
476. {
477. targetCour = course;
478. sort(stus.begin(), stus.end(), CourDown);
479. DisplayAllInfo();
480. }
482. //读取学生信息
483. **void** StudentManager::ReadInfo()
484. {
485. ifstream ifs;
486. ifs.open(STUDENTFILE, ios::in);
488. **if** (!ifs.is\_open()) {
489. cout << "文件打开失败！" << endl;
490. ifs.close();
491. **return**;
492. }
493. **if** (ifs.get() == EOF) {
494. //cout << "文件数据为空！" << endl;
495. **return**;
496. }
497. ifs.close();
499. //打开成功
500. ifs.open(STUDENTFILE, ios::in);
501. **while** (1) {
502. **int** id;
503. string name;
504. string sex;
505. **int** term;
506. **int** courseCount;
507. map<string, **int**> grades;
508. ifs >> id >> name >> sex >> courseCount;
509. **for** (**int** i = 0; i < courseCount; i++) {
510. string course;
511. **int** grade;
512. ifs >> course >> grade;
513. grades.insert(pair<string, **int**>(course, grade));
514. }
515. ifs >> term;
516. Student s(id, name, sex, term, courseCount, grades);
517. stus.push\_back(s);
518. **if** (ifs.get() == EOF)
519. **break**;
520. }
521. stus.pop\_back();
522. ifs.close();
523. }

四、运行结果



五、总结

很高兴分享我们在开发一个全面的学生成绩管理系统方面的经验，这是C++面向对象编程课程的一部分。这个项目是一个具有挑战性但又有意义的旅程，它使我们能够展示我们的编程技能，并应用整个课程中所学到的概念。

学生成绩管理系统是一个控制台应用程序，旨在简化管理学生成绩和相关数据的过程。通过这个项目，我们努力创造一个高效和用户友好的解决方案，以满足导师或管理员的需求。

在整个开发过程中，我们一直专注于纳入基本功能，以确保有效的成绩管理。我们已经实现了一些功能，如添加、编辑和删除学生记录，以及记录、更新他们的成绩。此外，我们还开发了生成报告和进行统计分析的工具，实现了全面的数据评估。

我们特别自豪的一个方面是系统的面向对象设计。通过应用封装、继承和多态性等原则，我们实现了代码的可重用性、模块化和可维护性。这种设计方法使我们能够很容易地增加新的功能，并确保系统的可扩展性，以适应未来的增强。

基于控制台的用户界面是经过精心设计的，以提供一个无缝和直观的体验。我们创建了用户友好的菜单和提示，以促进毫不费力的导航和与系统的有效互动。此外，我们还实施了错误处理和输入验证机制，以保持数据的完整性并防止意外行为。

在开发过程中，我们利用了各种C++语言的特点和编程技术。类、对象、继承和模板等概念在创建一个强大的、可扩展的代码库中发挥了作用。此外，我们还利用了标准库和数据结构来优化性能和简化数据管理。

这个项目不仅加深了我们对面向对象编程的理解，还磨练了我们解决问题的能力。我们投入了大量时间进行测试和调试，确保了系统的可靠性和准确性。

总的来说，这个学生成绩管理系统的完成是一次宝贵的学习经历。它使我们能够展示我们在C++编程方面的熟练程度，并证明我们有能力将理论概念应用于现实世界的场景。