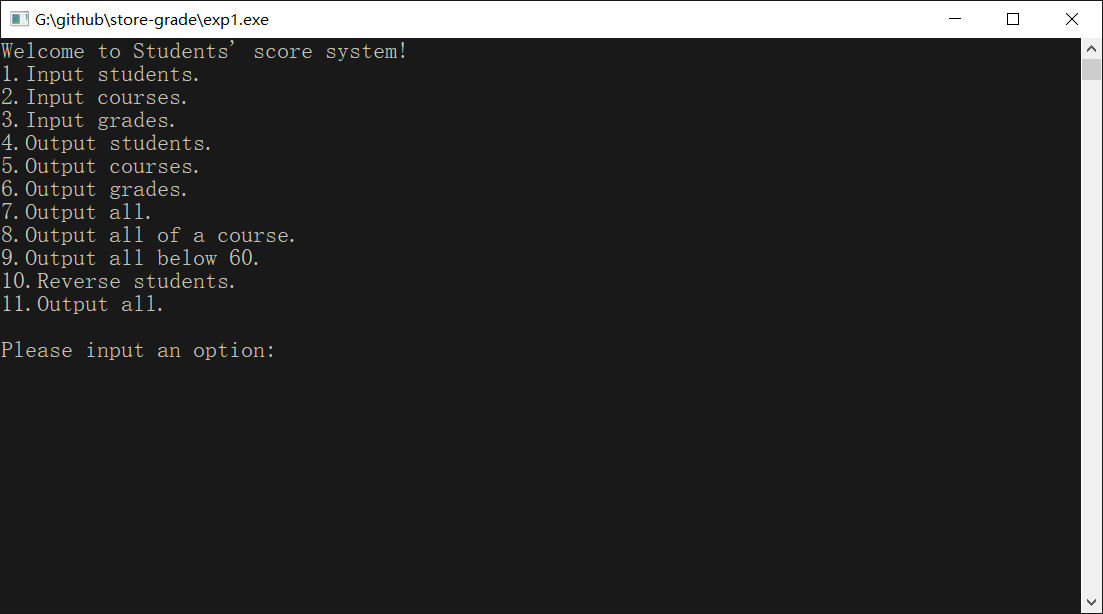
**实 验 报 告**

1. 实验名称：学生课程成绩查询程序
2. 实验学时：4学时
3. 实验目的：用链表和文件实现对学生课程成绩的查询和录入程序
4. 实验原理：链表、文件读入
5. 实验内容：
6. UI



其中包含了题目要求的十一项功能，通过一个while循环进行读取指令并且调用函数，输入-1结束程序。

相关代码：

（其中有一个switch，里面的case我省略了，将在后面每个功能的解释中以“main配套函数”的形式展示出来）

1. **int** main(){
2. **int** opt;
3. Student \*HEAD = NULL;
4. printf("Welcome to Students' score system!\n");
5. printf("1.Input students.\n");
6. printf("2.Input courses.\n");
7. printf("3.Input grades.\n");
8. printf("4.Output students.\n");
9. printf("5.Output courses.\n");
10. printf("6.Output grades.\n");
11. printf("7.Output all.\n");
12. printf("8.Output all of a course.\n");
13. printf("9.Output all below 60.\n");
14. printf("10.Reverse students.\n");
15. printf("11.Output all.\n");
17. **while**(1){
18. printf("\nPlease input an option:");
19. scanf("%d", &opt);
20. **if**(opt == -1) **break**;
21. **switch**(opt){
22. /\*
23. 此处省略，后面会进行详细介绍
24. \*/
25. }
26. }
28. }
30. }

相关定义以及基础函数：

定义学生、课程、成绩的结构体，并且定义一个文件读写的函数，方便后面直接调用，而不用再写一遍

1. **struct** Student{
2. **char** sno[12];
3. **char** sname[10];
4. **char** sex[4];
5. **char** major[20];
6. Student \*next; //指向后继数据结点
7. }\*StudentList;
9. **struct** Course{
10. **char** cno[10];
11. **char** cname[20];
12. **int** classHours;
13. Course \*next;
14. }\*CourseList;
16. **struct** Grade{
17. **char** sno[12];
18. **char** cno[10];
19. **int** score ;
20. Grade \*next;
21. }\*GradeList;
22. /\*
23. struct allinfo{


27. };
28. \*/
29. **FILE** \*Open\_File(**int** who, **int** rw){//打开哪个文件，打开方式如何
30. **FILE** \*p;
31. **switch**(who){
32. **case** 1:{
33. **if**(rw) p = fopen("student.dat","w");
34. **else** p = fopen("student.dat","r");
35. **break**;
36. }
37. **case** 2:{
38. **if**(rw) p = fopen("course.dat","w");
39. **else** p = fopen("course.dat","r");
40. **break**;
41. }
42. **case** 3:{
43. **if**(rw) p = fopen("courseGrade.dat","w");
44. **else** p = fopen("courseGrade.dat","r");
45. **break**;
46. }
48. }
49. **return** p;
50. }

**需要实现的功能：**

1. **输入10个学生记录，其中软件技术专业5人，人工智能专业5人，并存入文件student.dat中；**

**(2) 输入3门课程（数据库、数据结构、程序设计）信息记录，并存入文件course.dat中；**

**(3) 输入上述10位同学分别选修上述三门课程的考试成绩到文件courseGrade.dat中；**

以上三个功能都是对文件写入的一种操作，通过读入用户输入，加上之前写的文件操作函数，很容易实现对三个数据库文件的写入

1.写入学生信息

1. **void** Write\_Student(){
2. **FILE** \*StuFile = Open\_File(1, 1);//write
3. Student \*p = (Student \*) malloc(**sizeof**(Student));
4. **for**(**int** i = 1; i <= 10; i++){
5. printf("No.%2d:", i);
6. scanf("%s%s%s%s", p->sno, p->sname, p->sex, p->major);
7. fprintf(StuFile, "%s %s %s %s", p->sno, p->sname, p->sex, p->major);
8. **if**(i != 10) fprintf(StuFile, "\n");
9. }
10. free(p);
11. fclose(StuFile);
12. **return**;
13. }

2.写入课程信息

1. **void** Write\_Course(){
2. **FILE** \*CourseFile = Open\_File(2, 1);//write
3. Course \*p = (Course \*) malloc(**sizeof**(Course));
4. **for**(**int** i = 1; i <= 3; i++){
5. printf("No.%2d:", i);
6. scanf("%s%s%d", p->cno, p->cname, &p->classHours);
7. fprintf(CourseFile, "%s %s %d", p->cno, p->cname, p->classHours);
8. **if**(i != 3) fprintf(CourseFile, "\n");
9. }
10. free(p);
11. fclose(CourseFile);
12. **return**;
13. }

3.写入分数信息

1. **void** Write\_Grade(){
2. **FILE** \*GradeFile = Open\_File(3, 1);//write
3. Grade \*p = (Grade \*) malloc(**sizeof**(Grade));
4. **for**(**int** i = 1; i <= 10; i++){
5. printf("No.%2d:", i);
6. scanf("%s%s%d", p->sno, p->cno, &p->score);
7. fprintf(GradeFile, "%s %s %d", p->sno, p->cno, p->score);
8. **if**(i != 10) fprintf(GradeFile, "\n");
9. }
10. free(p);
11. fclose(GradeFile);
12. **return**;
13. }

**配套main函数调用的代码：**

1. **case** 1:{
2. Write\_Student();
3. printf("done.\n");
4. **break**;
5. }
7. **case** 2:{
8. Write\_Course();
9. printf("done.\n");
10. **break**;
11. }
13. **case** 3:{
14. Write\_Grade();
15. printf("done.\n");
16. **break**;
17. }

**(4) 从文件student.dat中读出学生信息，生成按照学号升序排列的单向链表，并在屏幕上显示输出；**

**(5) 从文件course.dat中读出课程信息，生成按照课程号升序排列的单向链表，并在屏幕上显示输出；**

**(6) 从文件courseGrade.dat中读出成绩信息，生成按照学号和课程号升序排列的单链表，并在屏幕上显示输出；**

以上三个功能的实现用到了文件和链表操作，先使用函数将文件信息全部读入链表中，再对这个链表进行冒泡排序，最后输出至屏幕

一下九个函数分别为对三种不同自定义结构体的读、排序、输出的操作

以BubbleSort开头的函数为冒泡排序函数，它会读入这个结构体所生成的一个链表，然后重开一个新链，将新链排序完成之后，覆盖掉以前那个旧链表

1. /\*
2. 以下是关于学生的代码
3. \*/
5. Student\* Read\_Student(){
6. **FILE** \*StuFile = Open\_File(1, 0);//read
7. Student \*head = (Student \*) malloc(**sizeof**(Student));
8. Student \*pre = head;
9. **while**(!feof(StuFile)){
10. Student \*p = (Student \*) malloc(**sizeof**(Student));
11. fscanf(StuFile, "%s%s%s%s", p->sno, p->sname, p->sex, p->major);
12. pre->next = p;
13. pre = p;
14. }
15. pre->next = NULL;
17. fclose(StuFile);
18. **return** head;
19. }
21. **void** BubbleSort\_Student(){
22. **for**(**int** i = 1; i <= 10; i++){
23. Student \*p = StudentList->next;
24. Student \*pre = StudentList;
25. **while**(p != NULL && p->next != NULL ){//存在两个可以比较的结点
26. Student \*pnext = p->next;
27. **if**(strcmp(p->sno, pnext->sno) > 0){
28. pre->next = pnext;
29. p->next = pnext->next;
30. pnext->next = p;
31. }
32. pre = p;
33. p = p->next;
34. }
35. }

38. }
40. **void** Print\_Student(){
41. Student \*p = StudentList->next;
42. printf("\nStudentList");
43. printf("\n---------------------\n");
44. **while**(p != NULL){
45. printf("%s %s %s %s", p->sno, p->sname, p->sex, p->major);
46. p = p->next;
47. **if**(p != NULL) printf("\n");
48. }
49. printf("\n---------------------\n\n");
50. }
52. /\*
53. 以下是关于课程的代码
54. \*/
56. Course\* Read\_Course(){
57. **FILE** \*CourseFile = Open\_File(2, 0);//read
58. Course \*head = (Course \*) malloc(**sizeof**(Course));
59. Course \*pre = head;
60. **while**(!feof(CourseFile)){
61. Course \*p = (Course \*) malloc(**sizeof**(Course));
62. fscanf(CourseFile, "%s%s%d", p->cno, p->cname, &p->classHours);
63. pre->next = p;
64. pre = p;
65. }
66. pre->next = NULL;
68. fclose(CourseFile);
69. **return** head;
70. }
72. **void** BubbleSort\_Course(){
73. **for**(**int** i = 1; i <= 10; i++){
74. Course \*p = CourseList->next;
75. Course \*pre = CourseList;
76. **while**(p != NULL && p->next != NULL ){//存在两个可以比较的结点
77. Course \*pnext = p->next;
78. **if**(strcmp(p->cno, pnext->cno) > 0){
79. pre->next = pnext;
80. p->next = pnext->next;
81. pnext->next = p;
82. }
83. pre = p;
84. p = p->next;
85. }
86. }

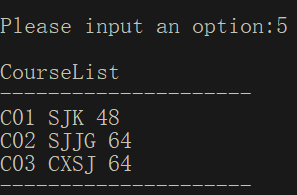
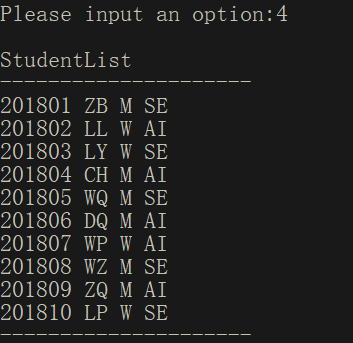
89. }
91. **void** Print\_Course(){
92. Course \*p = CourseList->next;
93. printf("\nCourseList");
94. printf("\n---------------------\n");
95. **while**(p != NULL){
96. printf("%s %s %d", p->cno, p->cname, p->classHours);
97. p = p->next;
98. **if**(p != NULL) printf("\n");
99. }
100. printf("\n---------------------\n\n");
101. }

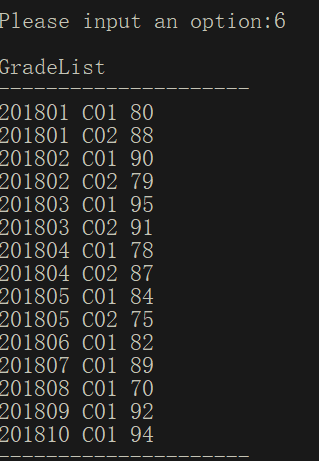
2. /\*
3. 以下是关于成绩的代码
4. \*/
6. Grade\* Read\_Grade(){
7. **FILE** \*GradeFile = Open\_File(3, 0);//read
8. Grade \*head = (Grade \*) malloc(**sizeof**(Grade));
9. Grade \*pre = head;
10. **while**(!feof(GradeFile)){
11. Grade \*p = (Grade \*) malloc(**sizeof**(Grade));
12. fscanf(GradeFile, "%s%s%d", p->sno, p->cno, &p->score);
13. pre->next = p;
14. pre = p;
15. }
16. pre->next = NULL;
18. fclose(GradeFile);
19. **return** head;
20. }
22. **void** BubbleSort\_Grade(){
23. **for**(**int** i = 1; i <= 10; i++){
24. Grade \*p = GradeList->next;
25. Grade \*pre = GradeList;
26. **while**(p != NULL && p->next != NULL ){//存在两个可以比较的结点
27. Grade \*pnext = p->next;
28. **if**(strcmp(p->sno, pnext->sno) > 0 || (strcmp(p->sno, pnext->sno) == 0 && strcmp(p->cno, pnext->cno) > 0) ){
29. pre->next = pnext;
30. p->next = pnext->next;
31. pnext->next = p;
32. }
33. pre = p;
34. p = p->next;
35. }
36. }
37. }
39. **void** Print\_Grade(){
40. Grade \*p = GradeList->next;
41. printf("\nGradeList");
42. printf("\n---------------------\n");
43. **while**(p != NULL){
44. printf("%s %s %d", p->sno, p->cno, p->score);
45. p = p->next;
46. **if**(p != NULL) printf("\n");
47. }
48. printf("\n---------------------\n\n");
49. }

52. **void** BubbleSort\_StudentsGrade(){
53. **for**(**int** i = 1; i <= 10; i++){
54. Grade \*p = GradeList->next;
55. Grade \*pre = GradeList;
56. **while**(p != NULL && p->next != NULL ){//存在两个可以比较的结点
57. Grade \*pnext = p->next;
58. **if**(p->score < pnext->score){
59. pre->next = pnext;
60. p->next = pnext->next;
61. pnext->next = p;
62. }
63. pre = p;
64. p = p->next;
65. }
66. }
67. }

**配套main函数调用的代码：**

1. **case** 4:{
2. StudentList = Read\_Student();
3. BubbleSort\_Student();
4. Print\_Student();
5. **break**;
6. }
8. **case** 5:{
9. CourseList = Read\_Course();
10. BubbleSort\_Course();
11. Print\_Course();
12. **break**;
13. }
15. **case** 6:{
16. GradeList = Read\_Grade();
17. BubbleSort\_Grade();
18. Print\_Grade();
19. **break**;
20. }





**(7) 查询所有学生所有课程的考试成绩，生成该课程的成绩单链表，要求包括学号、学生姓名、专业、课程名、考试成绩等信息，按照考试成绩降序排列，并将学生的该成绩信息输出到文件studentGrade.dat中，同时在屏幕上显示输出**

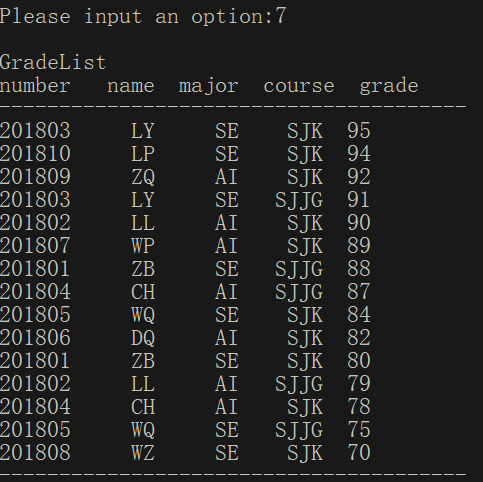
这个功能要运用一定的链表中元素查找的函数，因为通过成绩信息，只能获取到少部分学生、课程信息，对于具体信息的输出，必须依靠其他链表中所储存的内容

所以我编写了Find系列函数，旨在返回链表中某一个符合要求的结点，同时制作了错误信息：只要没找到，那么返回的信息自动为-1，那样的话查询的人也能很快定位出错的成绩信息

1. Student \*Find\_Student(**char** \*num){
2. Student \*p = StudentList->next;
3. **do**{
4. **if**(strcmp(p->sno, num) == 0) **return** p;
5. p = p->next;
6. }**while**(p != NULL);
8. p = (Student\*) malloc(**sizeof**(Student));
9. **char** temp[3] = "-1";
10. strcpy(p->major, temp);
11. strcpy(p->sex, temp);
12. strcpy(p->sname, temp);
13. strcpy(p->sno, temp);
14. p->next = NULL;
15. **return** p;//返回错误
16. }
18. Course \*Find\_Course(**char** \*num){
19. Course \*p = CourseList->next;
20. **do**{
21. **if**(strcmp(p->cno, num) == 0) **return** p;
22. p = p->next;
23. }**while**(p != NULL);
25. p = (Course\*) malloc(**sizeof**(Course));
26. **char** temp[3] = "-1";
27. strcpy(p->cname, temp);
28. strcpy(p->cno, temp);
29. p->classHours = -1;
30. p->next = NULL;
31. **return** p;//返回错误
32. }
33. **void** Print\_All(Grade \*p){
34. Student \*s = Find\_Student(p->sno);
35. Course \*c = Find\_Course(p->cno);
36. printf("%6s %6s %6s %6s %3d", s->sno, s->sname, s->major, c->cname, p->score);
37. }

**配套main函数调用的代码：**

1. **case** 7:{
2. StudentList = Read\_Student();
3. CourseList = Read\_Course();
4. GradeList = Read\_Grade();
5. BubbleSort\_StudentsGrade();//按照成绩排序
6. Print\_Seven();
8. **break**;
9. }



**(8) 在（7）的链表中，查询指定课程号的所有学生的考试成绩，生成该课程的成绩单链表，要求包括学号、学生姓名、专业、课程名、考试成绩等信息，按照考试成绩降序排列输出到屏幕上显示；**

只需要多一个步骤，把所有要输出的结点都加上判断，判断其是否属于该课程，不属于则不输出

1. **void** Print\_Seven(){//Task7
2. Grade \*p = GradeList->next;
3. printf("\nGradeList");
4. printf("\nnumber   name  major  course  grade");
5. printf("\n---------------------------------------\n");
6. **while**(p != NULL){
7. Print\_All(p);
8. p = p->next;
9. **if**(p != NULL) printf("\n");
10. }
11. printf("\n---------------------------------------\n\n");
12. }

配套main函数：

1. **case** 8:{
2. StudentList = Read\_Student();
3. CourseList = Read\_Course();
4. GradeList = Read\_Grade();
5. BubbleSort\_StudentsGrade();//按照成绩排序
6. **char** str[10];
7. printf("input course name:");
8. scanf("%s", str);
9. Print\_Eight(str);
11. **break**;
12. }

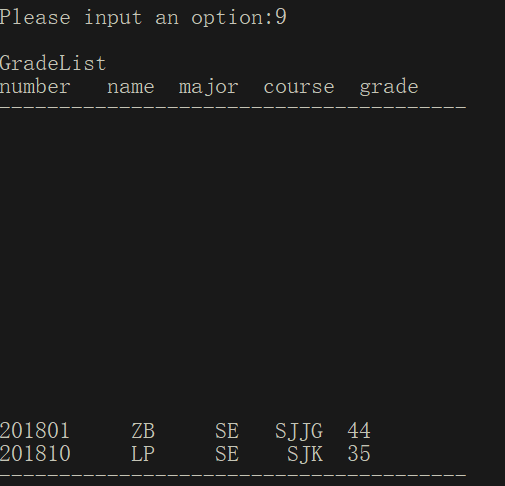
**(9) 在（7）的链表中，查询指定课程号的考试成绩小于60分的学生成绩信息，生成该课程的成绩链表，要求包括学号、学生姓名、专业、课程名、考试成绩等信息，并按照考试成绩降序排列在屏幕上显示输出；**

**类似操作，只需要判断一下成绩即可。**

1. **void** Print\_Eight(**char**\* num){//Task8 指定课程号
2. Grade \*p = GradeList->next;
3. printf("\nGradeList");
4. printf("\nnumber   name  major  course  grade");
5. printf("\n---------------------------------------\n");
6. **while**(p != NULL){
7. **if**(strcmp(p->cno, num) == 0) Print\_All(p);
8. p = p->next;
9. **if**(p != NULL) printf("\n");
10. }
11. printf("\n---------------------------------------\n\n");
12. }

**配套main函数：**

1. **case** 8:{
2. StudentList = Read\_Student();
3. CourseList = Read\_Course();
4. GradeList = Read\_Grade();
5. BubbleSort\_StudentsGrade();//按照成绩排序
6. **char** str[10];
7. printf("input course name:");
8. scanf("%s", str);
9. Print\_Eight(str);
11. **break**;
12. }



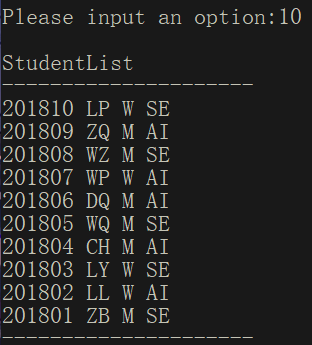
**(10) 使用栈实现将（4）的单链表中的学生信息逆序生存新的链表。**

**直接新开一个链表，将原链表依次插入，得到的自然是逆序链表**

1. **void** ReverseStudent(){
2. Student \*head = (Student\*) malloc(**sizeof**(Student));
3. Student \*p = StudentList->next;
4. head->next = NULL;
6. **while**(p != NULL){
7. Student \*t = p->next;//暂时保存p的下一个结点
8. p->next = head->next;
9. head->next = p;
10. p = t;
11. }
13. StudentList = head;
14. }

**配套main函数：**

1. **case** 10:{
2. StudentList = Read\_Student();
3. ReverseStudent();//逆序生成新的链表
4. Print\_Student();//输出该链表
5. **break**;
6. }



**(11) 设计使用链式队列完成问题（7）的要求，建议使用带头指针和尾指针链式队列。**

1. **void** Queue\_Push(Grade \*p, Grade \*&top, Grade \*&rear){
2. **if**(top->next == NULL){
3. top->next = p;
4. rear->next = p;
5. }
6. **else**{
7. p->next = top;
8. top = p;
9. }

12. }
14. Grade\* Queue\_Pop(Grade \*&top, Grade \*&rear){
15. **if**(top->next == rear->next){
16. top->next = rear->next = NULL;
17. **return** top->next;
18. }**else**{
19. rear->next = top->next;
20. **return** rear->next;
21. }

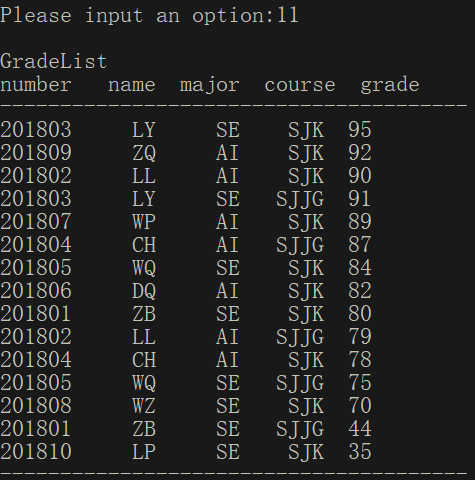
24. }


28. **void** Print\_Eleven(Grade \*top, Grade \*rear){
29. Grade \*p = Queue\_Pop(top, rear);
30. printf("\nGradeList");
31. printf("\nnumber   name  major  course  grade");
32. printf("\n---------------------------------------\n");
33. **while**(p != NULL){
34. Print\_All(p);
35. p = p->next;
36. **if**(p != NULL) printf("\n");
37. }
38. printf("\n---------------------------------------\n\n");
39. }

**配套main函数**

1. **case** 11:{
2. StudentList = Read\_Student();
3. CourseList = Read\_Course();
4. GradeList = Read\_Grade();
5. BubbleSort\_StudentsGrade();//按照成绩排序
7. Grade \*top = (Grade\*) malloc(**sizeof**(Grade));
8. Grade \*rear = (Grade\*) malloc(**sizeof**(Grade));
9. top->next = rear->next = NULL;
10. Grade \*p = GradeList->next;
11. **do**{
12. Queue\_Push(p, top, rear);
13. }**while**(p != NULL);
15. Print\_Eleven(top, rear);

18. **break**;
19. }



1. 总结及心得体会：

**链表的操作一定不要慌张，要对链表每个之间的关系有了一个全面的了解之后再开始写代码，不然会出现自己都还没想清楚就写了一堆运行错误的代码，时间得不偿失。**

1. 对本实验过程及方法、手段的改进建议：

实验10,11不必非要用栈和队列实现，直接进行链表的插入和修改会比用栈队列结构要更节约时间提升效率。

1. 附录（详细实现代码）
2. #include<cstdio>
3. #include<cstdlib>
4. #include<cstring>
6. **struct** Student{
7. **char** sno[12];
8. **char** sname[10];
9. **char** sex[4];
10. **char** major[20];
11. Student \*next; //指向后继数据结点
12. }\*StudentList;
14. **struct** Course{
15. **char** cno[10];
16. **char** cname[20];
17. **int** classHours;
18. Course \*next;
19. }\*CourseList;
21. **struct** Grade{
22. **char** sno[12];
23. **char** cno[10];
24. **int** score ;
25. Grade \*next;
26. }\*GradeList;
27. /\*
28. struct allinfo{


32. };
33. \*/
34. **FILE** \*Open\_File(**int** who, **int** rw){//打开哪个文件，打开方式如何
35. **FILE** \*p;
36. **switch**(who){
37. **case** 1:{
38. **if**(rw) p = fopen("student.dat","w");
39. **else** p = fopen("student.dat","r");
40. **break**;
41. }
42. **case** 2:{
43. **if**(rw) p = fopen("course.dat","w");
44. **else** p = fopen("course.dat","r");
45. **break**;
46. }
47. **case** 3:{
48. **if**(rw) p = fopen("courseGrade.dat","w");
49. **else** p = fopen("courseGrade.dat","r");
50. **break**;
51. }
53. }
54. **return** p;
55. }
57. /\*
58. 以下是关于学生的代码
59. \*/
61. Student\* Read\_Student(){
62. **FILE** \*StuFile = Open\_File(1, 0);//read
63. Student \*head = (Student \*) malloc(**sizeof**(Student));
64. Student \*pre = head;
65. **while**(!feof(StuFile)){
66. Student \*p = (Student \*) malloc(**sizeof**(Student));
67. fscanf(StuFile, "%s%s%s%s", p->sno, p->sname, p->sex, p->major);
68. pre->next = p;
69. pre = p;
70. }
71. pre->next = NULL;
73. fclose(StuFile);
74. **return** head;
75. }
77. **void** BubbleSort\_Student(){
78. **for**(**int** i = 1; i <= 10; i++){
79. Student \*p = StudentList->next;
80. Student \*pre = StudentList;
81. **while**(p != NULL && p->next != NULL ){//存在两个可以比较的结点
82. Student \*pnext = p->next;
83. **if**(strcmp(p->sno, pnext->sno) > 0){
84. pre->next = pnext;
85. p->next = pnext->next;
86. pnext->next = p;
87. }
88. pre = p;
89. p = p->next;
90. }
91. }

94. }
96. **void** Print\_Student(){
97. Student \*p = StudentList->next;
98. printf("\nStudentList");
99. printf("\n---------------------\n");
100. **while**(p != NULL){
101. printf("%s %s %s %s", p->sno, p->sname, p->sex, p->major);
102. p = p->next;
103. **if**(p != NULL) printf("\n");
104. }
105. printf("\n---------------------\n\n");
106. }
108. **void** Write\_Student(){
109. **FILE** \*StuFile = Open\_File(1, 1);//write
110. Student \*p = (Student \*) malloc(**sizeof**(Student));
111. **for**(**int** i = 1; i <= 10; i++){
112. printf("No.%2d:", i);
113. scanf("%s%s%s%s", p->sno, p->sname, p->sex, p->major);
114. fprintf(StuFile, "%s %s %s %s", p->sno, p->sname, p->sex, p->major);
115. **if**(i != 10) fprintf(StuFile, "\n");
116. }
117. free(p);
118. fclose(StuFile);
119. **return**;
120. }

123. /\*
124. 以下是关于课程的代码
125. \*/
127. Course\* Read\_Course(){
128. **FILE** \*CourseFile = Open\_File(2, 0);//read
129. Course \*head = (Course \*) malloc(**sizeof**(Course));
130. Course \*pre = head;
131. **while**(!feof(CourseFile)){
132. Course \*p = (Course \*) malloc(**sizeof**(Course));
133. fscanf(CourseFile, "%s%s%d", p->cno, p->cname, &p->classHours);
134. pre->next = p;
135. pre = p;
136. }
137. pre->next = NULL;
139. fclose(CourseFile);
140. **return** head;
141. }
143. **void** BubbleSort\_Course(){
144. **for**(**int** i = 1; i <= 10; i++){
145. Course \*p = CourseList->next;
146. Course \*pre = CourseList;
147. **while**(p != NULL && p->next != NULL ){//存在两个可以比较的结点
148. Course \*pnext = p->next;
149. **if**(strcmp(p->cno, pnext->cno) > 0){
150. pre->next = pnext;
151. p->next = pnext->next;
152. pnext->next = p;
153. }
154. pre = p;
155. p = p->next;
156. }
157. }

160. }
162. **void** Print\_Course(){
163. Course \*p = CourseList->next;
164. printf("\nCourseList");
165. printf("\n---------------------\n");
166. **while**(p != NULL){
167. printf("%s %s %d", p->cno, p->cname, p->classHours);
168. p = p->next;
169. **if**(p != NULL) printf("\n");
170. }
171. printf("\n---------------------\n\n");
172. }
174. **void** Write\_Course(){
175. **FILE** \*CourseFile = Open\_File(2, 1);//write
176. Course \*p = (Course \*) malloc(**sizeof**(Course));
177. **for**(**int** i = 1; i <= 3; i++){
178. printf("No.%2d:", i);
179. scanf("%s%s%d", p->cno, p->cname, &p->classHours);
180. fprintf(CourseFile, "%s %s %d", p->cno, p->cname, p->classHours);
181. **if**(i != 3) fprintf(CourseFile, "\n");
182. }
183. free(p);
184. fclose(CourseFile);
185. **return**;
186. }

189. /\*
190. 以下是关于成绩的代码
191. \*/
193. Grade\* Read\_Grade(){
194. **FILE** \*GradeFile = Open\_File(3, 0);//read
195. Grade \*head = (Grade \*) malloc(**sizeof**(Grade));
196. Grade \*pre = head;
197. **while**(!feof(GradeFile)){
198. Grade \*p = (Grade \*) malloc(**sizeof**(Grade));
199. fscanf(GradeFile, "%s%s%d", p->sno, p->cno, &p->score);
200. pre->next = p;
201. pre = p;
202. }
203. pre->next = NULL;
205. fclose(GradeFile);
206. **return** head;
207. }
209. **void** BubbleSort\_Grade(){
210. **for**(**int** i = 1; i <= 10; i++){
211. Grade \*p = GradeList->next;
212. Grade \*pre = GradeList;
213. **while**(p != NULL && p->next != NULL ){//存在两个可以比较的结点
214. Grade \*pnext = p->next;
215. **if**(strcmp(p->sno, pnext->sno) > 0 || (strcmp(p->sno, pnext->sno) == 0 && strcmp(p->cno, pnext->cno) > 0) ){
216. pre->next = pnext;
217. p->next = pnext->next;
218. pnext->next = p;
219. }
220. pre = p;
221. p = p->next;
222. }
223. }
224. }
226. **void** Print\_Grade(){
227. Grade \*p = GradeList->next;
228. printf("\nGradeList");
229. printf("\n---------------------\n");
230. **while**(p != NULL){
231. printf("%s %s %d", p->sno, p->cno, p->score);
232. p = p->next;
233. **if**(p != NULL) printf("\n");
234. }
235. printf("\n---------------------\n\n");
236. }
238. **void** Write\_Grade(){
239. **FILE** \*GradeFile = Open\_File(3, 1);//write
240. Grade \*p = (Grade \*) malloc(**sizeof**(Grade));
241. **for**(**int** i = 1; i <= 10; i++){
242. printf("No.%2d:", i);
243. scanf("%s%s%d", p->sno, p->cno, &p->score);
244. fprintf(GradeFile, "%s %s %d", p->sno, p->cno, p->score);
245. **if**(i != 10) fprintf(GradeFile, "\n");
246. }
247. free(p);
248. fclose(GradeFile);
249. **return**;
250. }
252. **void** BubbleSort\_StudentsGrade(){
253. **for**(**int** i = 1; i <= 10; i++){
254. Grade \*p = GradeList->next;
255. Grade \*pre = GradeList;
256. **while**(p != NULL && p->next != NULL ){//存在两个可以比较的结点
257. Grade \*pnext = p->next;
258. **if**(p->score < pnext->score){
259. pre->next = pnext;
260. p->next = pnext->next;
261. pnext->next = p;
262. }
263. pre = p;
264. p = p->next;
265. }
266. }
267. }
269. Student \*Find\_Student(**char** \*num){
270. Student \*p = StudentList->next;
271. **do**{
272. **if**(strcmp(p->sno, num) == 0) **return** p;
273. p = p->next;
274. }**while**(p != NULL);
276. p = (Student\*) malloc(**sizeof**(Student));
277. **char** temp[3] = "-1";
278. strcpy(p->major, temp);
279. strcpy(p->sex, temp);
280. strcpy(p->sname, temp);
281. strcpy(p->sno, temp);
282. p->next = NULL;
283. **return** p;//返回错误
284. }
286. Course \*Find\_Course(**char** \*num){
287. Course \*p = CourseList->next;
288. **do**{
289. **if**(strcmp(p->cno, num) == 0) **return** p;
290. p = p->next;
291. }**while**(p != NULL);
293. p = (Course\*) malloc(**sizeof**(Course));
294. **char** temp[3] = "-1";
295. strcpy(p->cname, temp);
296. strcpy(p->cno, temp);
297. p->classHours = -1;
298. p->next = NULL;
299. **return** p;//返回错误
300. }
302. **void** Print\_All(Grade \*p){
303. Student \*s = Find\_Student(p->sno);
304. Course \*c = Find\_Course(p->cno);
305. printf("%6s %6s %6s %6s %3d", s->sno, s->sname, s->major, c->cname, p->score);
306. }
308. **void** Print\_Seven(){//Task7
309. Grade \*p = GradeList->next;
310. printf("\nGradeList");
311. printf("\nnumber   name  major  course  grade");
312. printf("\n---------------------------------------\n");
313. **while**(p != NULL){
314. Print\_All(p);
315. p = p->next;
316. **if**(p != NULL) printf("\n");
317. }
318. printf("\n---------------------------------------\n\n");
319. }
321. **void** Print\_Eight(**char**\* num){//Task8 指定课程号
322. Grade \*p = GradeList->next;
323. printf("\nGradeList");
324. printf("\nnumber   name  major  course  grade");
325. printf("\n---------------------------------------\n");
326. **while**(p != NULL){
327. **if**(strcmp(p->cno, num) == 0) Print\_All(p);
328. p = p->next;
329. **if**(p != NULL) printf("\n");
330. }
331. printf("\n---------------------------------------\n\n");
332. }
334. **void** Print\_Nine(){//Task9
335. Grade \*p = GradeList->next;
336. printf("\nGradeList");
337. printf("\nnumber   name  major  course  grade");
338. printf("\n---------------------------------------\n");
339. **while**(p != NULL){
340. **if**(p->score < 60) Print\_All(p);
341. p = p->next;
342. **if**(p != NULL) printf("\n");
343. }
344. printf("\n---------------------------------------\n\n");
345. }
347. **void** ReverseStudent(){
348. Student \*head = (Student\*) malloc(**sizeof**(Student));
349. Student \*p = StudentList->next;
350. head->next = NULL;
352. **while**(p != NULL){
353. Student \*t = p->next;//暂时保存p的下一个结点
354. p->next = head->next;
355. head->next = p;
356. p = t;
357. }
359. StudentList = head;
360. }
362. **int** main(){
363. **int** opt;
364. Student \*HEAD = NULL;
365. printf("Welcome to Students' score system!\n");
366. printf("1.Input students.\n");
367. printf("2.Input courses.\n");
368. printf("3.Input grades.\n");
369. printf("4.Output students.\n");
370. printf("5.Output courses.\n");
371. printf("6.Output grades.\n");
372. printf("7.Output all.\n");
373. printf("8.Output all of a course.\n");
374. printf("9.Output all below 60.\n");
375. printf("10.Reverse students.\n");
376. printf("11.Output all.\n");
378. **while**(1){
379. printf("\nPlease input an option:");
380. scanf("%d", &opt);
381. **if**(opt == -1) **break**;
382. **switch**(opt){
383. **case** 1:{
384. Write\_Student();
385. printf("done.\n");
386. **break**;
387. }
389. **case** 2:{
390. Write\_Course();
391. printf("done.\n");
392. **break**;
393. }
395. **case** 3:{
396. Write\_Grade();
397. printf("done.\n");
398. **break**;
399. }
401. **case** 4:{
402. StudentList = Read\_Student();
403. BubbleSort\_Student();
404. Print\_Student();
405. **break**;
406. }
408. **case** 5:{
409. CourseList = Read\_Course();
410. BubbleSort\_Course();
411. Print\_Course();
412. **break**;
413. }
415. **case** 6:{
416. GradeList = Read\_Grade();
417. BubbleSort\_Grade();
418. Print\_Grade();
419. **break**;
420. }
422. **case** 7:{
423. StudentList = Read\_Student();
424. CourseList = Read\_Course();
425. GradeList = Read\_Grade();
426. BubbleSort\_StudentsGrade();//按照成绩排序
427. Print\_Seven();
429. **break**;
430. }
432. **case** 8:{
433. StudentList = Read\_Student();
434. CourseList = Read\_Course();
435. GradeList = Read\_Grade();
436. BubbleSort\_StudentsGrade();//按照成绩排序
437. **char** str[10];
438. printf("input course name:");
439. scanf("%s", str);
440. Print\_Eight(str);
442. **break**;
443. }
445. **case** 9:{
446. StudentList = Read\_Student();
447. CourseList = Read\_Course();
448. GradeList = Read\_Grade();
449. BubbleSort\_StudentsGrade();//按照成绩排序
450. Print\_Nine();
452. **break**;
453. }
455. **case** 10:{
456. StudentList = Read\_Student();
457. ReverseStudent();//逆序生成新的链表
458. Print\_Student();//输出该链表
459. **break**;
460. }
462. **case** 11:{
463. StudentList = Read\_Student();
464. CourseList = Read\_Course();
465. GradeList = Read\_Grade();
466. BubbleSort\_StudentsGrade();//按照成绩排序
467. Print\_Seven();

470. **break**;
471. }
472. }
474. }
476. }