

**哈尔滨工业大学**

**高级语言程序设计实验报告**

**实验项目名称 学生成绩管理系统 成绩评定**

**实验项目类型： 综合设计型 实验时间： 2016/12/20**

**学生姓名： 王世川 班级： 1637102 学号： 1163710227**

**学院： 计算机学院 专业： 软件工程 任课教师： 苏小红教授**

1. **实验目的**

熟悉一维、二维数组做函数参数，排序、查找、统计分析等常用算法，自顶向下、逐步求精的模块化设计以及增量测试方法

1. **实验内容和要求**

**实验内容：**

对学生成绩管理系统V6.0的升级

某班有最多不超过30人（具体人数由键盘输入）参加哈工大本专业大一秋季学期所有课程的考试，用结构体作函数参数编程实现对学生成绩的管理。

新增了命令行终端风格的操作模式，更加友好的交互界面设计，提升了程序实际的用户体验。

**实验要求：**

采用自顶向下、逐步求精的模块化设计思想设计的一个小型数据库管理系统，也可以是一个游戏，或者其他作品。要求按系统的需求分析、系统设计（总体设计、各个模块的设计）、系统实现（各个模块的实现）、系统测试等过程组织报告内容，说明采用了什么数据结构和算法，遇到哪些问题，这些问题是如何解决的，本设计的亮点在哪里。

本系统主要实现下面的功能：

1：录入学生信息。

2：修改学生信息。

3：删除学生信息。

4：按 学号/姓名 查找学生信息。

5：自动计算学生的平均成绩。

6：自动计算学生的总成绩。

7：排出成绩表（按字典升/降顺序）。

8：排出成绩表（按学号升/降顺序）。

9：排出成绩表（按总成绩升/降顺序）。

10：将所有学生的记录信息写入文件。

11：从文件中读取每个学生的记录信息。

12：打开程序自动读档。

13：退出程序自动保存

14：退出程序。

1. **实验环境**

**操作系统：Win10**

**编译器：[Visual Studio Community 2015](https://go.microsoft.com/fwlink/?LinkId=691978&clcid=0x409" \t "_self)**

1. **系统设计**

**系统功能模块结构图：**

**学生成绩管理系统**

保存并退出

打开帮助菜单

导出学生信息

打印成绩单

导入学生信息

查找学生信息

修改学生信息

添加学生信息

删除学生信息

按学号/姓名 升/降序排序并输出

按学号/姓名检索并修改/输出

**算法设计：**（请写出程序中用到的算法，并说明所用的算法的思想）

在查找学生信息部分运用了遍历算法，其核心思想是遍历每一个元素直到找到合乎条件的那一个。在数据没有进行排序以及经常变动的情况下，这种排序比二分排序要更适用。

在排序部分运用了选择法排序，其核心思想是：（按从小到大排列一组数字为例）第1趟，在待排序记录r[1]~r[n]中选出最小的记录，将它与r[1]交换；第2趟，在待排序记录r[2]~r[n]中选出最小的记录，将它与r[2]交换；以此类推，第i趟在待排序记录r[i]~r[n]中选出最小的记录，将它与r[i]交换，使有序序列不断增长直到全部排序完毕。

**数据结构设计：**

（请说明采用数组、结构体还是链表存储学生个人信息，并列出所定义的数据各种信息，如：）

结构体1-STU：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 字段含义 | 类型 | 长度 | 默认值 | 说明 |
| 1 | Number | 学生学号 | Long | 64位 | 0 |  |
| 2 | Name | 学生姓名 | Char [15] | 15位 | Null |  |
| 3 | Score | 学生各科成绩 | Int [6] | 6\*32位 | 0 |  |
| 4 | Sum | 学生总成绩 | Double | 64位 | 0.0 |  |
| 5 | Avera | 学生平均成绩 | Double | 64位 | 0.0 |  |

结构体2-FLAG：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 字段含义 | 类型 | 长度 | 默认值 | 说明 |
| 1 | Course | 课程数 | Int | 32位 | 0 | 本次考试的课程数 |
| 2 | Iniflag | 初始化状态 | Int | 32位 | Null | 当前是否数据正常 |
| 3 | Sortflag | 排序状态 | Int | 32位 | 0 | 当前是否有序 |
| 4 | SuccessfulNumber | 录入人数 | Int | 32位 | 0 | 当前成功记录的人数 |

**各模块（函数）的功能和接口设计：**

**主函数：** 完成指令读取并按要求调用各函数。

（1）Initialization**函数原型**：int Initialization(FLAG \*flag);

功能：初始化各项参数，清空缓冲区，当有未知错误时，返回FLASE以退出程序。

参数：

FLAG \*flag ：程序各项状态的集合

返回值：0 FALSE/1 TURE

（2）AnalyzeOrder**函数原型**：int AnalyzeOrder(char ml[], int x);

功能：分析用户输入的命令

参数：

char ml[]：存储用户输入的命令的数组

int x：指令的集合序号

返回值：对应命令的数字记号

（3）SearchStudent**函数原型**：int SearchStudent(FLAG\*flag, STU student[], int x);

功能：按条件查询学生信息

参数：

FLAG \*flag ：程序各项状态的集合

STU student[]：所有学生信息

int x：查询的类型

返回值：对应学生在所有学生信息数组中的下标

（4）NewStudent**函数原型**：void NewStudent(FLAG \*flag, STU student[]);

功能：批量新建学生信息

参数：

FLAG \*flag ：程序各项状态的集合

STU student[]：所有学生信息

返回值：无

（5）ModifyStudent**函数原型**：void ModifyStudent(FLAG \*flag, STU student[]);

功能：修改指定学生信息

参数：

FLAG \*flag ：程序各项状态的集合

STU student[]：所有学生信息

返回值：无

（6）ReadStudent**函数原型**：void ReadStudent(FLAG \*flag, STU stu[]);

功能：读取学生信息的存档文件

参数：

FLAG \*flag ：程序各项状态的集合

STU student[]：所有学生信息

返回值：无

（7）ListStudent**函数原型**：void ListStudent(FLAG\*flag, STU student[]);

功能：按要求分析用户的排序指令,并最终按要求打印学生成绩单

参数：

FLAG \*flag ：程序各项状态的集合

STU student[]：所有学生信息

返回值：无

（8）DeleteStudent**函数原型**：void DeleteStudent(FLAG \*flag, STU student[]);

功能：引导用户删除指定学生信息

参数：

FLAG \*flag ：程序各项状态的集合

STU student[]：所有学生信息

返回值：无

（9）WriteStudent**函数原型**：void WriteStudent(FLAG \*flag, STU student[],int x);

功能：引导用户删除指定学生信息

参数：

FLAG \*flag ：程序各项状态的集合

STU student[]：所有学生信息

int x：用于确定存档类型（自动/手动存档）

返回值：无

（10）SortName**函数原型**：void SortName(FLAG \*flag, STU student[], int mode);

功能：对学生信息按照姓名进行排序

参数：

FLAG \*flag ：程序各项状态的集合

STU student[]：所有学生信息

int mode：用于确定排序方式

返回值：无

（11）SortScore**函数原型**：void SortScore(FLAG \*flag, STU student[], int mode);

功能：对学生信息按照总分进行排序

参数：

FLAG \*flag ：程序各项状态的集合

STU student[]：所有学生信息

int mode：用于确定排序方式

返回值：无

（12）SortNumber**函数原型**：void SortNumber(FLAG \*flag, STU student[], int mode);

功能：对学生信息按照学号进行排序

参数：

FLAG \*flag ：程序各项状态的集合

STU student[]：所有学生信息

int mode：用于确定排序方式

返回值：无

（13）PrintStudent**函数原型**：void PrintStudent(FLAG\*flag, STU student[], int x);

功能：输出学生的成绩单

参数：

FLAG \*flag ：程序各项状态的集合

STU student[]：所有学生信息

int x：用于确定输出方式

返回值：无

（14）Swap**函数原型**：void Swap(STU \*x,STU \*y);

功能：交换学生信息存储的位置

参数：

STU \*x：需要交换位置的学生信息

STU \*y：需要交换位置的学生信息

返回值：无

（15）ViewHelp**函数原型**：void ViewHelp(void);

功能：各项帮助信息导航

参数：无

返回值：无

（16）PutMenu**函数原型**：void PutMenu(void);

功能：输出菜单

参数：无

返回值：无

1. **系统实现**

**主要模块的程序流程图：**

自动读档

ReadStudent(…);

AnalyzeOrder(…)

清空缓冲区

读取指令

exitnewrddelwr

WriteStudent(…);

Break;

DeleteStudent(…);

Break;

exit(1);

ReadStudent(…);

Break;

NewStudent(…);

Break;

re ls

ListStudent(…);

Break;

ViewHelp(…);

Break;

Menu(…);

Break;

SearchStudent(…);

Break;

ModifyStudent(…);

Break;

输出帮助信息

AnalyzeOrder

输出成绩单

SortName(…);

SortScore(…);

SortNumber(…);

**源程序的全部代码：**

文件：源.c

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399  400  401  402  403  404  405  406  407  408  409  410  411  412  413  414  415  416  417  418  419  420  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435  436  437  438  439  440  441  442  443  444  445  446  447  448  449  450  451  452  453  454  455  456  457  458  459  460  461  462  463  464  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480  481  482  483  484  485  486  487  488  489  490  491  492  493  494  495  496  497  498  499  500  501  502  503  504  505  506  507  508  509  510  511  512  513  514  515  516  517  518  519  520  521  522  523  524  525  526  527  528  529  530  531  532  533  534  535  536  537  538  539  540  541  542  543  544  545  546  547  548  549  550  551  552  553  554  555  556  557  558  559  560  561  562  563  564  565  566  567  568  569  570  571  572  573  574  575  576  577  578  579 | **#include<stdio.h>**  **#include<windows.h>**  **#include<string.h>**  **#include<stdlib.h>**  **#include<conio.h>**    **#define STUDENTS 30 //学生人数**  **#define COURSE 6 //课程数上限**  **#define NAME 15 //姓名长度上限**  **#define MAXLEN 7 //单条命令长度上限**  **typedef struct MyStruct**  **{**  **long number; //学号**  **int totalscore; //总分**  **double averscore; //平均分**  **int score[COURSE]; //各科成绩**  **char name[NAME]; //姓名**  **}STU; //用于存储学生信息的结构体**  **typedef struct MyStruct2**  **{**  **int courseN; //课程数**  **int Iniflag; //初始化状态**  **int Successfulnumber; //已登记人数**  **int sortflag; //当前排序状态**  **}FLAG; //用于存储程序运行状态的结构体**  **int Initialization(FLAG \*flag);**  **int AnalyzeOrder(char ml[], int x);**  **/\*功能:分析用户输入的命令**  **参数:存储用户输入的命令的数组,以及命令类型**  **返回值:对应命令的数字记号**  **\*/**  **int SearchStudent(FLAG\*flag, STU student[], int x);**  **/\*功能:按条件查询学生信息**  **参数:flag是本程序各项状态的集合,student[]是所有学生信息,x为查询的类型**  **返回值:对应学生在所有学生信息数组中的下标**  **\*/**  **void NewStudent(FLAG \*flag, STU student[]);**  **/\*功能:批量新建学生信息**  **参数:flag是本程序各项状态的集合,student[]用于存储所有学生信息**  **返回值:无**  **\*/**  **void ModifyStudent(FLAG \*flag, STU student[]);**  **/\*功能:修改指定学生信息**  **参数:flag是本程序各项状态的集合,student[]用于存储所有学生信息**  **返回值:无**  **\*/**  **void ReadStudent(FLAG \*flag, STU stu[]);**  **/\*功能:读取文件中的学生信息**  **参数:flag是本程序各项状态的集合,student[]用于存储所有学生信息**  **返回值:无**  **\*/**  **void ListStudent(FLAG\*flag, STU student[]);**  **/\*功能:按要求分析用户的排序指令,并最终按要求打印学生成绩单**  **参数:flag是本程序各项状态的集合,student[]用于存储所有学生信息**  **返回值:无**  **\*/**  **void DeleteStudent(FLAG \*flag, STU student[]);**  **/\*功能:引导用户删除指定学生信息**  **参数:flag是本程序各项状态的集合,student[]用于存储所有学生信息**  **返回值:无**  **\*/**  **void WriteStudent(FLAG \*flag, STU student[],int x);**  **/\*功能:将学生信息输出到对应文件中**  **参数:flag是本程序各项状态的集合,student[]用于存储所有学生信息**  **返回值:无**  **\*/**  **void SortName(FLAG \*flag, STU student[], int mode);**  **/\*功能:对学生信息按照姓名进行排序**  **参数:flag是本程序各项状态的集合,student[]用于存储所有学生信息**  **返回值:无**  **\*/**  **void SortScore(FLAG \*flag, STU student[], int mode);**  **/\*功能:对学生信息按照总分进行排序**  **参数:flag是本程序各项状态的集合,student[]用于存储所有学生信息**  **返回值:无**  **\*/**  **void SortNumber(FLAG \*flag, STU student[], int mode);**  **/\*功能:对学生信息按照学号进行排序**  **参数:flag是本程序各项状态的集合,student[]用于存储所有学生信息**  **返回值:无**  **\*/**  **void PrintStudent(FLAG\*flag, STU student[], int x);**  **/\*功能:输出学生的成绩单**  **参数:flag是本程序各项状态的集合,student[]用于存储所有学生信息,x是指定输出的学生下标**  **返回值:无**  **\*/**  **void Swap(STU \*x,STU \*y);**  **/\*功能:交换学生信息存储的位置**  **参数:需要交换位置的两项学生信息**  **返回值:无**  **\*/**  **void ViewHelp(void);**  **/\*功能:各项帮助信息导航**  **参数:无**  **返回值:无**  **\*/**  **void PutMenu(void);**  **/\*功能:打印功能菜单**  **参数:无**  **返回值:无**  **\*/**  **int main(int argc, char\* arvg[])**  **{**  **STU student[STUDENTS]; //储存所有学生信息的数组**  **FLAG Flag = { -1,1,0 ,0 }; //定义用于存储程序各项运行状态的结构体**  **int i;**  **char ml[MAXLEN] = { 0 };**  **//欢迎界面**  **puts("欢迎使用学生成绩管理终端! --作者HIT1163710227黑川");**  **PutMenu();**  **ReadStudent(&Flag, student);**  **//开始程序**  **while (Initialization(&Flag)) {**  **printf("\n请输入命令进行操作:>>> ");**  **while (!scanf("%4s", ml))//错误检查**  **{**  **puts("无效命令，请重新输入\n");**  **while (getchar() != '\n');**  **}**  **switch (AnalyzeOrder(ml, 1))//分析指令并引导到各个功能函数**  **{**  **case 0:**  **NewStudent(&Flag, student); break;**  **case 1:**  **DeleteStudent(&Flag, student); break;**  **case 2:**  **ModifyStudent(&Flag, student); break;**  **case 3:**  **i = -1;**  **i = SearchStudent(&Flag, student, -1);**  **if (i > -1)PrintStudent(&Flag, student, i); break;**  **case 4:**  **ListStudent(&Flag, student); break;**  **case 5:**  **ReadStudent(&Flag, student); break;**  **case 6:**  **WriteStudent(&Flag, student,0); break;**  **case 7:**  **ViewHelp(); break;**  **case 8:**  **PutMenu(); break;**  **case 9:**  **WriteStudent(&Flag, student,1); exit(1); break;**  **default:**  **printf("'%s'不是有效命令,请重新输入:", ml); break;**  **}**  **}**  **//特殊原因退出**  **puts("异常退出!!!");**  **return 0;**  **}**  **void PutMenu(void) {**  **system("cls");//清屏**  **printf("\t\t\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\n\**  **\t\t\* 命令简介如下: \*\n\**  **\t\t\* new n(n<=30) 新建学生信息 \*\n\**  **\t\t\* del name/number 按姓名/学号查找并删除学生信息 \*\n\**  **\t\t\* re name/number 按姓名/学号查找并修改学生信息 \*\n\**  **\t\t\* sch name/number 按姓名/学号查找学生信息 \*\n\**  **\t\t\* ls score/name/number asc/desc 按分数/学号/姓名 升序/降序 打印成绩单 \*\n\**  **\t\t\* rd xxx 读取文件xxx \*\n\**  **\t\t\* wr xxx 导出文件于xxx \*\n\**  **\t\t\* help xx 查看xx命令的详细帮助信息 \*\n\**  **\t\t\* menu 查看本菜单 \*\n\**  **\t\t\* exit 自动保存数据并退出程序 \*\n\**  **\t\t\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\n\n");**  **}**  **int Initialization(FLAG \*flag) {**  **if (flag->Iniflag) {//自动读档失败,要求用户手动输入各项参数**  **printf("\n请输入本次考试有效科目数n,(n<=6):\n");**  **while (!scanf("%2d", &flag->courseN) ||flag->courseN >6 || flag->courseN<0)**  **{**  **puts("错误数据,请重新输入\n");**  **while (getchar() != '\n');**  **}**  **flag->Iniflag = 0;**  **}**  **while(getchar() != '\n');//清空缓冲区**  **return 1;**  **}**  **void NewStudent(FLAG \*flag,STU student[]) {**  **int n = flag->courseN,m=flag->Successfulnumber;**  **int i=0, j=0, number = 0;**  **//输入及输入检查**  **if (!scanf("%2d", &number)||number<1||number+m >STUDENTS) {**  **puts("输入的学生人数无效或越界,新建学生信息失败!");**  **}else {//输入无误,读取学生信息,并计算总分平均分**  **printf("请按如下格式输入%d个学生的信息:\n",number);**  **printf("姓名\t学号");**  **for (i = 0; i < n; i++)printf("\t科目%d", i + 1);**  **printf("\n");**  **for (i = 0; i < number; i++) {**  **scanf("%14s%ld",student[m+i].name,&student[m+i].number);**  **student[i].totalscore = 0;**  **for (j = 0; j < n; j++) {**  **scanf("%3d", &student[m + i].score[j]);**  **student[i].totalscore += student[i].score[j];**  **}**  **student[i].averscore = student[i].totalscore / flag->courseN;**  **flag->Successfulnumber++;//每读取一名学生信息,成功读取信息人数加一**  **}**  **}**  **flag->sortflag = -1;//有新信息写入,需要重新排序**  **}**  **void ModifyStudent(FLAG \*flag, STU student[]) {**  **char ml[7] = { 0 };**  **int n = -1, x = -1, i =0;**  **if (!scanf("%6s", ml))//读取命令并检查**  **{**  **puts("无效命令,请重新输入\n");**  **return;**  **}**  **//分析命令并检索学生信息**  **n = AnalyzeOrder(ml, 2);**  **if (n == 1 || n == 2)x = SearchStudent(flag, student, n);**  **else printf("'re %s'不是有效指令,请重新输入", ml);**  **if (x != -1) {//引导用户修改学生信息**  **printf("请按如下格式输入正确的学生信息:\n");**  **printf("姓名\t学号");**  **for (i = 0; i < n; i++)printf("\t科目%d", i + 1);**  **printf("\n");**  **scanf("%14s%ld", student[x].name, &student[x].number);**  **student[x].totalscore = 0;**  **for (i = 0; i < flag->courseN; i++) {**  **scanf("%3d", &student[x].score[i]);**  **student[x].totalscore += student[x].score[i];**  **}**  **student[x].averscore = student[x].totalscore / flag->courseN;**  **flag->sortflag = -1;//有信息改动,需要重新排序**  **}**  **}**  **void ReadStudent(FLAG \*flag,STU student[]) {**  **FILE \*fp;**  **char filename[20] = {0};**  **char \*file = "student.txt";**  **int i, j;**    **//打开文件**  **if (flag->Iniflag ) {**  **fp = fopen(file, "r");**  **}else {**  **while (!scanf("%19s", filename)) {**  **puts("请输入正确文件地址及文件名");**  **while (getchar() != '\n');**  **}**  **fp = fopen(filename, "r");**  **}**  **if (fp != NULL) {//打开成功则读取文件**  **fscanf(fp,"%1d%2d",&flag->courseN,&flag->Successfulnumber);**  **for (i = 0; i < flag->Successfulnumber; i++) {**  **student[i].totalscore = 0;**  **fscanf(fp, "%14s%ld", &student[i].name, &student[i].number);**  **for (j = 0; j < flag->courseN; j++){**  **fscanf(fp,"%3d",&student[i].score[j]);**  **student[i].totalscore += student[i].score[j];**  **}**  **fscanf(fp, "%\*d%\*lf");**  **student[i].averscore = student[i].totalscore / flag->courseN;**  **}**  **fclose(fp);**  **flag->sortflag = -1;**  **flag->Iniflag = 0;**  **puts("读取存档成功,按回车键继续") ;**  **}else if (flag->Iniflag)puts("自动读档失败!");**  **else puts("读取失败!!!");//打开失败则根据原因报出错误**  **}**  **void WriteStudent(FLAG \*flag, STU student[],int x) {**  **FILE \*fp;**  **char filename[20] = { 0 };**  **char \*file = "student.txt";**  **int i=0, j=0;**  **//打开文件**  **if (x) {**  **fp = fopen(file, "w");**  **}else {**  **while (!scanf("%19s", filename)) {**  **puts("请输入正确文件地址及文件名");**  **while (getchar() != '\n');**  **}**  **fp = fopen(filename, "w");**  **}**  **if (fp != NULL) {//打开成功则输出学生信息**  **fprintf(fp,"%d\t%d",flag->courseN,flag->Successfulnumber);**  **for (i = 0; i < flag->Successfulnumber; i++) {**  **fprintf(fp, "\n%s\t%ld", student[i].name,student[i].number);**  **for (j = 0; j < flag->courseN; j++) {**  **fprintf(fp, "\t%d", student[i].score[j]);**  **}**  **fprintf(fp, "\t%d\t%.1f",student[i].totalscore,student[i].averscore);**  **}**  **fclose(fp);**  **}else puts("保存失败!");//打开失败则报错**  **}**  **void Swap(STU \*x, STU \*y) {//交换两项学生信息**  **STU temp=\*x;**  **\*x = \*y;**  **\*y = temp;**  **}**  **void SortName(FLAG \*flag,STU student[],int mode) {**  **int i = 0, j = 0, k = 0;**  **int temp;**  **//如果已排序则无需重复排序**  **if (flag->sortflag == 10+mode)return;**  **for (i = 0; i < flag->Successfulnumber - 1; i++) {**  **k = i;**  **for (j = i + 1; j < flag->Successfulnumber; j++){**  **temp = strcmp(student[j].name, student[k].name);**  **if (mode == 4)temp = -temp;**  **if ( temp< 0)k = j;**  **}**  **if (k != i)Swap(&student[k],&student[i]);**  **}**  **flag->sortflag = 10+mode;//排序完成后标记当前排序状态**  **}**  **void SortScore(FLAG \*flag, STU student[], int mode) {**  **int i = 0, j = 0, k = 0;**  **int temp;**  **//如果已排序则不必重复排序**  **if (flag->sortflag == mode)return;**  **for (i = 0; i < flag->Successfulnumber - 1; i++) {**  **k = i;**  **for (j = i + 1; j < flag->Successfulnumber; j++) {**  **temp = (student[j].totalscore<student[k].totalscore);**  **if (mode == 4)temp = !temp;**  **if (temp)k = j;**  **}**  **if (k != i)Swap(&student[k], &student[i]);**  **}**  **flag->sortflag = mode;//排序完成则标记当前排序状态**  **}**  **void SortNumber(FLAG \*flag, STU student[], int mode) {**  **int i = 0, j = 0, k = 0;**  **int temp;**  **//如果已排序则无需重复排序**  **if (flag->sortflag == 20+mode)return;**  **for (i = 0; i < flag->Successfulnumber - 1; i++) {**  **k = i;**  **for (j = i + 1; j < flag->Successfulnumber; j++) {**  **temp = (student[j].number<student[k].number);**  **if (mode == 4)temp = !temp;**  **if (temp)k = j;**  **}**  **if (k != i)Swap(&student[k], &student[i]);**  **}**  **flag->sortflag = 20+mode;//排序完成后标记排序状态**  **}**  **int AnalyzeOrder(char ml[], int x) {**  **const char ML1[10][5] = { "new","del","re","sch","ls","rd","wr","help","menu","exit" };**  **const char ML2[5][7] = { "score","name","number","asc","desc" };**  **int i;//定义用于存储有效命令的数组**  **if (x == 1) {**  **for (i = 0; i < 10;i++)if (strcmp(ml, ML1[i]) == 0)return i;**  **}**  **else if (x == 2) {**  **for (i = 0; i < 5;i++)if (strcmp(ml, ML2[i]) == 0)return i;**  **}**  **return -1;//匹配不成功则返回-1**  **}**  **void ListStudent(FLAG\*flag, STU student[]) {**  **char ml1[7] = { 0 }, ml2[5] = { 0 };**  **int n = -1, m = -1;**  **//读取命令并检查**  **if (scanf("%s %s", ml1, ml2) != 2) {**  **printf("'ls %s %s'不是完整命令或不是有效命令!\n", ml1, ml2);**  **return;**  **}**  **n = AnalyzeOrder(ml1, 2);**  **m = AnalyzeOrder(ml2, 2);**  **if ( m == 3 || m == 4)**  **{//根据命令排序,出错则输出错误信息**  **switch (n)**  **{**  **case 0:SortScore(flag, student, m); break;**  **case 1:SortName(flag, student, m); break;**  **case 2:SortNumber(flag, student, m); break;**  **default:printf("'%s %s'不是有效的命令参数!\n", ml1, ml2);**  **break;**  **}**  **PrintStudent(flag, student,-1);**  **}**  **else { //出错则输出错误信息**  **printf("'%s %s'不是有效的命令参数!\n",ml1,ml2);**  **return;**  **}**  **}**  **int SearchStudent(FLAG\*flag, STU student[], int x) {**  **char ml[7] = { 0 }, name[NAME] = {0};**  **long number = 0;**  **int i=0, n = flag->Successfulnumber;**  **//未知检索条件则先读取命令分析检索条件**  **if (x == -1) {**  **scanf("%6s", ml);**  **x = AnalyzeOrder(ml, 2);**  **}**  **if (x == -1) printf("'sch %s'不是有效的命令,请重新输入!\n", ml);**  **if (x == 1) { //检索姓名并返回下标**  **printf("请输入指定学生的姓名:");**  **scanf("%14s", name);**  **for (i = 0; i < n; i++) {**  **if (strcmp(name, student[i].name) == 0)**  **return i;**  **}**  **}else if (x == 2) { //检索学号并返回下标**  **printf("请输入指定学生的学号:");**  **scanf("%ld", &number);**  **for (i = 0; i <n; i++) {**  **if (number == student[i].number)**  **return i;**  **}**  **}**  **puts("未检索到该学生信息!");**  **return -1;//未检索到返回-1**  **}**  **void DeleteStudent(FLAG \*flag, STU student[]) {**  **char ml[7] = { 0 };**  **int n=-1,x=-1,i=flag->Successfulnumber-1;**  **//从缓冲区读取指令并错误则报错**  **if (!scanf("%6s", ml))**  **{**  **puts("无效命令,请重新输入\n");**  **return;**  **}**  **n = AnalyzeOrder(ml, 2);**  **if(n==1||n==2)x = SearchStudent(flag,student,n);**  **else printf("'del %s'不是有效指令,请重新输入",ml);**  **if (x != -1) {//检索到学生信息则用最后一条信息覆盖该信息**  **if (x != i) {**  **student[x] = student[i];**  **flag->sortflag = -1;//信息变动需要重新排序**  **}**  **flag->Successfulnumber--;//已登记人数减一**  **}**  **}**  **void ViewHelp(void) {**  **char ml2[5];**  **scanf("%4s",ml2);**  **switch (AnalyzeOrder(ml2,1))//分析指令分别输出对应详细帮助**  **{**  **case 0:puts("new n命令用于新建n(<30人)个学生信息,输入该指令将引导您新建学生信息"); break;**  **case 1:puts("del name/numble命令用于按姓名/学号索引学生信息并删除"); break;**  **case 2:puts("re name/numble命令用于按姓名/学号检索学生信息并修改"); break;**  **case 3:puts("sch name/numble命令用于按姓名/学号检索学生信息"); break;**  **case 4:puts("ls name/numble/score asc/desc命令由三部分组成,\n表示按照姓名/学号/总分进行升/降序排序后打印成绩单"); break;**  **case 5:puts("rd xxx用于读取成绩存档,xxx为文件地址和文件名"); break;**  **case 6:puts("wr xxx用于导出成绩存档文件,xxx为文件地址和文件名"); break;**  **case 7:puts("help xx命令用于查看xx命令的详细使用帮助"); break;**  **case 8:puts("menu 命令用于查看命令菜单"); break;**  **case 9:puts("exit 命令用于保存并退出程序,默认保存的在程序同目录下student.txt文件中"); break;**  **default:**  **printf("' %s'不是有效命令，请输入正确命令",ml2);**  **break;**  **}**  **}**  **void PrintStudent(FLAG \*flag, STU student[],int x) {**  **int i = 0,j=0;**  **//打印表头**  **printf("\n姓名\t学号");**  **for (i = 0; i < flag->courseN; i++)printf("\t科目%d", i + 1);**  **printf("\t总分\t平均分");**  **if (x != -1) {//输出指定学生信息**  **printf("\n%s\t%ld", student[x].name, student[x].number);**  **for (i = 0; i <flag->courseN; i++)**  **printf("\t%d", student[x].score[i]);**  **printf("\t%d\t%.1f",student[x].totalscore,student[x].averscore);**  **}**  **else for (i = 0; i < flag->Successfulnumber; i++) {**  **//输出全部 学生信息**  **printf("\n%s\t%ld", student[i].name, student[i].number);**  **for (j = 0; j < flag->courseN; j++)**  **printf("\t%d", student[i].score[j]);**  **printf("\t%d\t%.1f", student[i].totalscore, student[i].averscore);**  **}**  **}** |

1. **实验步骤及调试过程**

**遇到的问题有：**

1、输入非法数据时程序崩溃退出；

解决方法：对输入数据进行合法性检查和防溢出检查，当输入错误数据时，提示输入有误，并要求用户重新输入；

2、出现少数逻辑错误使得排序结果不符合要求；

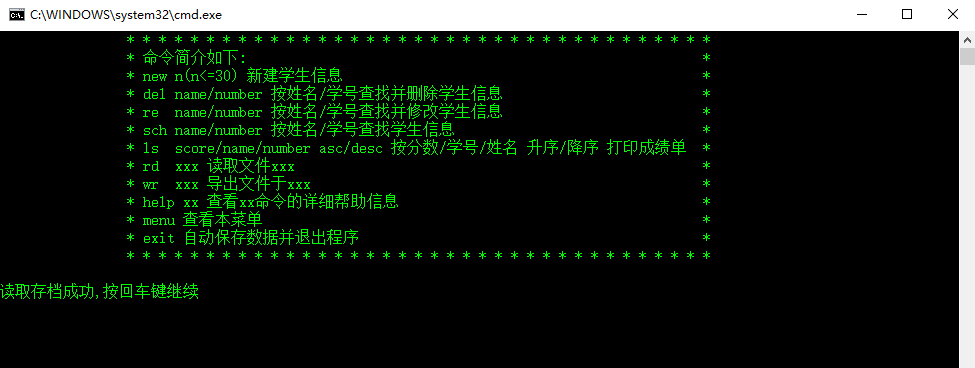
解决方法：重新检查代码，谨慎选择排序方法，逐条运行代码检查错误步骤并修改；

3、输出学生信息时，输出结果不整齐，不美观；

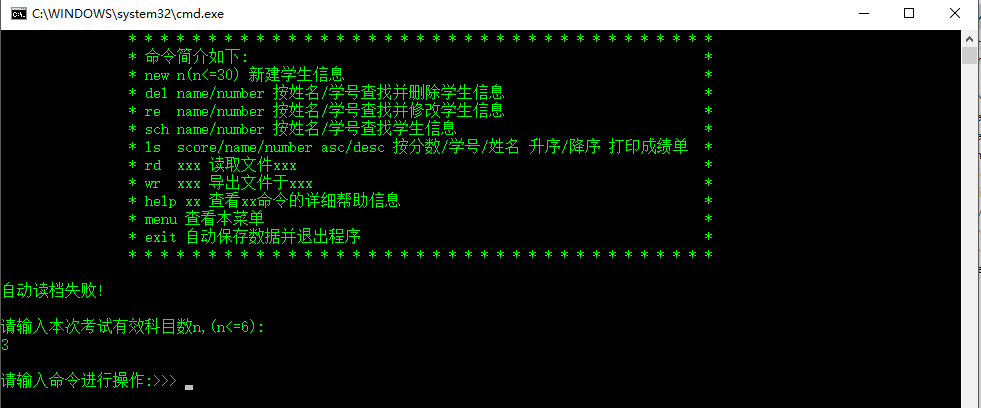
解决方法：调整输出格式，使用’\t’使输出网格对齐，再次运行程序观察是否合适美观。

1. **系统测试结果**

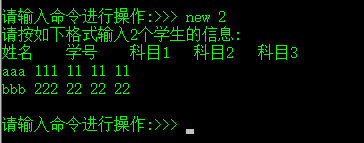
**正常打开并自动读档成功：**



**自动读档失败：**

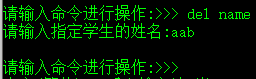


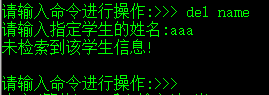
**新建学生信息：**



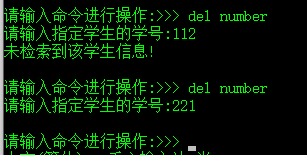
**删除学生信息：**

**-1、按姓名查找并删除（成功/失败）**



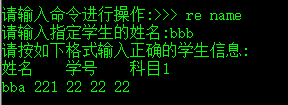


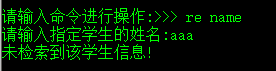
**-2、按学号查找并删除**



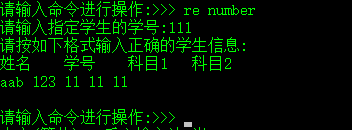
**修改学生信息：**

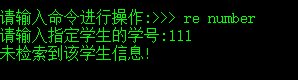
**-1、按姓名检索并修改（成功/失败）**



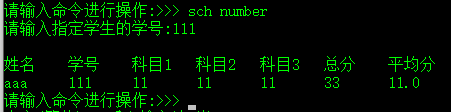


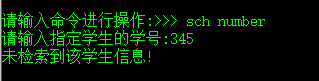
**-2、按学号检索并修改（成功/失败）**



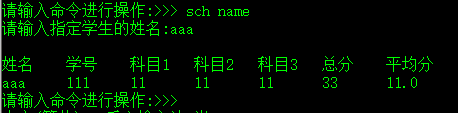


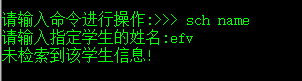
**按学号查找学生信息：（成功/失败）**



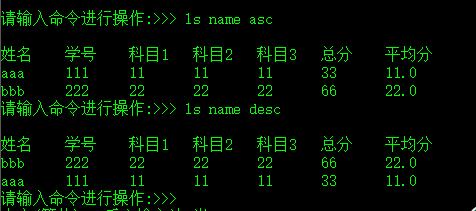


**按姓名查找学生信息：（成功/失败）**





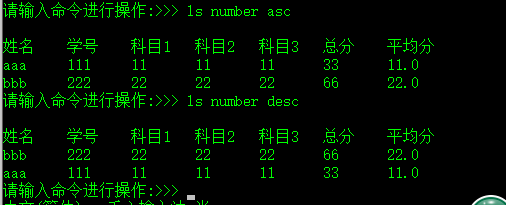
**按姓名字典序 升序/降序 排序**



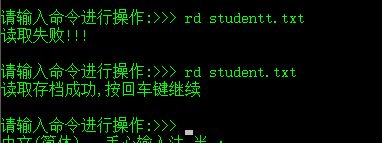
**按总分 升/降序 排序：**



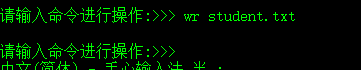
**按学号 升/降序 排序：**



**读取文件：（成功/失败）**



**输出文件：**



**自动保存并退出：**



1. **分析总结**

**我对题目的理解是**：管理系统的开发，是对熟练选择使用合适的数据结构，和程序性思维，对程序整体框架、流程的掌控能力，以及语法正确性、函数封装、程序调试修改等基础综合能力的考察。

同时我认为，一个好的学生成绩管理系统，好的程序，应当要再实现基本功能的基础上，更进一步考虑：程序界面是否友好？系统升级和修改是否尽量做到便易？甚至长远来看，维护人员的交接是否能够较为顺利？

这要求我们不仅要追求程序运行的正确性，更要考虑用户界面的友好，代码风格便于阅读理解，以助于团队合作中他人进行更改和升级。

**我的思路和我的代码的特点是：**我阅读了用户需求，充分理解用户的需要并在此基础上给出更专业化的、更明确的功能实现目标，并在草稿纸上模拟程序运行的流程，然后设想好了本程序的框架，并依据框架分块完成各项函数功能，并在主函数中统筹地转接调用。

得益于充分的思考构想，我的代码有简洁，高利用率的特点，充分合并同类功能，简短而有力，同时拥有美观易读的代码风格，合理缩进、注释，简单易懂的变量名，符合规范的函数名，注释适当，充分而不繁琐。整体上结构紧凑，封装有力，使得Debug时能够迅速分析错因，找到错语句并且易于修改。

**亮点与缺点（可改进的地方）：**亮点是充分考虑了用户界面的友好，作为管理系统，易于使用，容易上手。代码风格严谨有序，简洁易懂，便于后期的维护和升级。

可改进的地方是，还没有进行充分的实践测试，提供的帮助信息、失败报错的功能还不够完善，有待补充和改进。

**参考文献**

[1]  苏小红，王宇颖，孙志岗.C语言程序设计（第3版）[M].北京：高等教育出版社.2015

[2] 苏小红，王宇颖，孙志岗.C语言程序设计学习指导（第3版）[M].北京：高等教育出版社.2015