C/C++ 程序设计综合训练设计说明书

（项目一）

目 录

一、项目总体设计方案

1. 系统功能需求分析

学生成绩管理系统面向单门课程、单个班级的成绩管理。系统启动后，从指定文本文件中读取固定格式的姓名、学号和原始成绩，从键盘获取参与统计的学生人数与课程满分，用于后续比例换算和合法性检查。数据读入并通过范围校验后统一保存在内存中，后续操作都在这一份数据上完成，无需重复读写文件。

系统提供循环菜单界面，用户可多次选择不同功能：一是按照成绩从高到低排序并生成名次，在终端输出“名次-学号-姓名-成绩”列表；二是按成绩区间统计人数，将原始成绩按满分比例划分为 0-30、30-60、60-80、80-100 四个分段，用于观察成绩分布；三是计算平均分，找出最高分和最低分，并给出对应学号和姓名；四是在退出前将排序后的完整成绩表导出为 CSV 文件，便于用表格软件查看和打印。

系统假定输入文件格式正确，输入如图 1，阶段主要进行人数上限、成绩范围和文件打开情况等基本检查。一旦检测到人数超限、成绩越界或文件无法打开，会在入口处给出提示并直接终止，避免基于错误数据继续处理。

2. 系统整体结构设计

整体上，如图 1，系统围绕主控流程展开，由数据获取与存储、成绩处理以及用户交互与结果展示三部分构成。主函数在程序入口完成提示和数据读入后，进入基于菜单的循环结构，根据用户选择调用相应功能模块。

数据获取与存储部分负责文件输入和基础校验：从磁盘读取学生姓名、学号和成绩，将每条记录封装为结构体元素顺序存入数组，同时检查成绩是否落在 [0, 满分] 区间、学生数量是否在预设上限以内，校验失败时向主控流程返回错误标志。

成绩处理部分包括排序与排名、分段统计和整体分析三个子模块。排序与排名模块在内存中对学生数组按成绩降序重排，并据此写入名次字段；分段统计模块根据课程满分，将成绩映射到相应分数区间，累计各分段人数；整体分析模块遍历数组，求出总分、平均分以及最大值和最小值所在位置，为结果输出提供基础数据。

用户交互与结果展示部分由控制台菜单和 CSV 导出组成。控制台负责显示操作选项、读取用户输入，并以文本形式展示排序结果、分段统计和分析结论；导出部分在用户选择退出时，将当前数组按名次顺序写入外部 CSV 文件，便于直接打开。

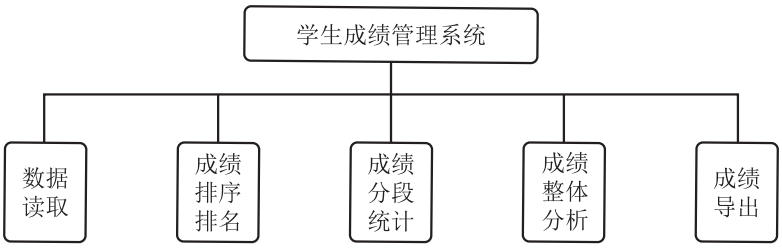

图 1 学生成绩管理系统功能模块图

二、项目详细设计方案

1. 数据输入与合法性检查模块——函数 inputstu 的设计

函数通过指针参数返回学生人数和课程满分，并将读取到的学生记录写入结构体数组 student，布尔返回值用于标记输入是否成功。

执行顺序如下：首先检查全局输入输出文件流是否成功打开，若有文件打开失败，则在终端输出错误信息并返回 false。文件正常时，提示用户输入需要统计的学生数量和课程满分，将键盘输入写入 *n 和 *full。随后进入循环，根据 *n 依次从文件读取每位学生的姓名、学号和成绩，每读入一条立即判断成绩是否在 [0, *full] 区间内；若发现越界，输出提示信息并返回 false，中止后续读取。

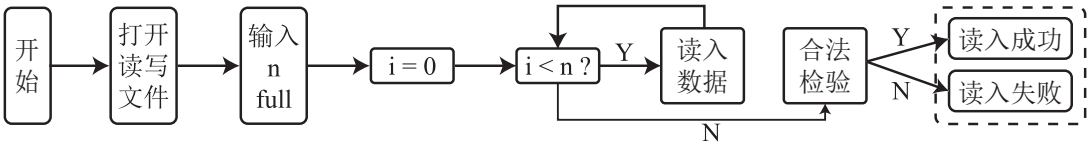

图 2 学生成绩输入模块图

再对整体数据做一次检查：若学生数量为 0、满分为 0 或人数超过 30，则给出读取失败提示并返回 false。只有文件状态正常且人数与成绩均在允许范围内时，函数才返回 true。

2. 成绩排序与排名模块

（1）比较函数 cmp 的设计

成绩排序基于 std::sort 实现，排序准则由比较函数 bool cmp(stu x, stu y) 提供。该函数仅比较两个学生结构体的 score 字段，当 x.score > y.score 时返回 true，表示在排序结果中 x 应排在 y 之前，从而实现按成绩从高到低的降序排序。

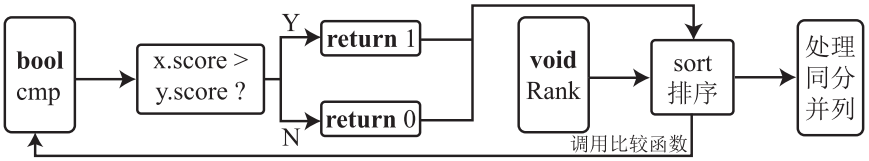

图 3 学生成绩成绩排序与名次生成函数

（2）排名函数 Rank 的设计

函数 Rank(stu student[], int n) 用于在排序基础上生成名次。函数首先调用 sort 函数，按成绩降序重排数组。排序完成后，从下标 0 开始依次扫描每个元素，维护一个临时变量 temp 记录当前连续并列的个数。

遍历过程中，若当前学生不是第一个且成绩与前一位相同，则说明出现并列，当前名次应与并列起始位置一致，因此通过 i - (temp++) + 1 计算名次并写入 ScoreRank，同时递增 temp；若成绩与前一位不同，则将当前学生名次设为 i+1，并将 temp 重置为 1，处理了同分并列的情况。

3. 成绩统计模块——函数 statistics 的设计

成绩分段统计由函数

```
statistics(stu student[], int n, int *score30, int *score60,
                int *score80, int *score100, int full)
```

完成。四个整型指针参数分别用于返回各分数段的人数计数，调用前先在函数内部将其清零。函数遍历学生数组，根据每个学生成绩占满分的比例进行区间判断：若 score < 0.3*full，则累计 score30；否则若小于 0.6*full，累计 score60；否则若小于 0.8*full，累计 score80；其余则视为 80-100 分段，累计 score100。

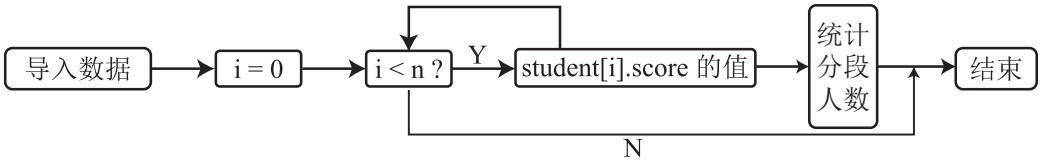

图 4 学生成绩分段统计函数

4. 成绩分析模块——函数 analysis 的设计

函数 analysis(stu student[], int *ScoreMax, int *ScoreMin, double *ScoreAve, int n, double full) 负责计算最高分、最低分和平均分。入口时 ScoreMax 与 ScoreMin 作为下标指针，一般初始化为 0，用于记录当前已知最高分和最低分所在位置；ScoreAve 用于返回平均分结果。

函数内部先将总分 sum 初始化为 0，然后从 0 开始遍历学生数组。在扫描过程中，若当前学生成绩大于 student[*ScoreMax].score，则更新 *ScoreMax 为当前下标；若成绩小于 student[*ScoreMin].score，则更新 *ScoreMin。同时，每次将当前成绩累加到 sum。循环结束后，用 sum/n 得到平均分，写入 *ScoreAve。

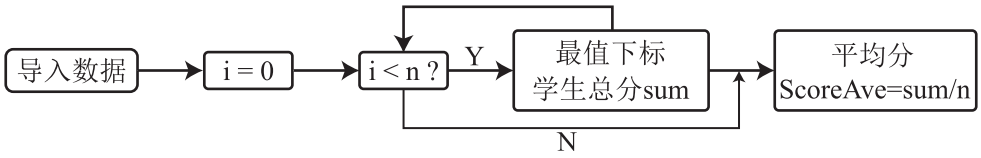

图 5 学生成绩统计模块

5. 数据导出模块——函数 stuoutput 的设计

导出结果由 stuoutput(stu student[], int n) 完成。函数开头再次调用 Rank，保证输出前数据已经按成绩排序并附带正确名次。随后通过全局输出文件流 fout 写入表头行“排名, 学号, 姓名, 成绩”，接着在循环中按数组顺序逐行输出每位学生的名次、学号、姓名和成绩，各字段之间以逗号分隔，每条记录一行。

6. 菜单与主控流程——主函数 main 的设计

主函数 main 负责整体控制流程和与用户的交互。程序启动时先输出提示信息，引导用户了解输入格式和样例。随后声明学生人数、满分、各分数段计数、最高分和最低分下标、平均分等变量，并定义容量为 30 的学生数组。接着提示“上传数据”，调用 inputstu 完成数据读入和合法性检查；若返回值为 false，说明输入阶段发生错误，主函数直接返回，程序结束；若读入成功，则输出“数据读取已完成，正在分析中”的提示，并进入菜单循环。

菜单部分通过 while(1) 实现，循环中先输出操作选项，包括成绩排序、成绩统计、成绩分析和导出退出，再读取用户输入的操作编号。根据编号进入 switch 结构：当用户选择排序时，调用 Rank 并在终端逐行打印名次、学号、姓名和成绩；选择统计时，调用 statistics，然后输出各分数段人数；选择分析时，调用 analysis，输出平均分以及最高分、最低分对应学生的信息；选择导出时，调用 stuoutput 将当前数据写入 CSV 文件，给出“数据已导出，正在退出”的提示，并通过 return 0 结束程序。若用户输入的选项不在 1–4 范围内，则输出错误提示并返回菜单顶部重新选择。

三、程序清单及运行结果

1. 程序清单

Listing 1: student grades.cpp

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <algorithm>
using namespace std;
ifstream fin("input.txt");
ofstream fout("output.csv");

struct stu{
   string name;
   int id;
   double score;
   int ScoreRank;
};

bool inputstu(int *n,int* full,stu student[]){
   if (!fin.is_open() || !fout.is_open()) {
      cerr << "无法打开文件\n";
      return 0;
   }

    cout<<"请输入需要统计的学生数量：";
    cin>>*n;
    cout<<"请输入该科目的满分：",
   cin>>*full;
```

```
    for(int i=0;i<*n;i++){
        fin>>student[i].name>>student[i].id>>student[i].score;
        if(student[i].score>*full || student[i].score<0){
            cerr<<"\n学生成绩范围错误，自动退出中。";
            return 0;
        }
    }
    if((*n)==0||(*full)==0||(*n)>30){
        cerr<<"\n数据读取失败，自动退出中。";
        return 0;
    }
    return 1;
}

bool cmp(stu x,stu y){
    if(x.score>y.score)
        return 1;
    return 0;
}

void Rank(stu student[],int n){
    sort(student,student+n,cmp);
    int temp=1;
    for(int i=0;i<n;i++){
        if(i>0 && student[i].score==student[i-1].score)
            student[i].ScoreRank=i-(temp++)+1;
        else{
            student[i].ScoreRank=i+1;
            temp=1;
        }
    }
}

void statistics(stu student[],int n,int *score30,int *score60,int *score80,int
     *score100,int full){
    *score30=0;
    *score60=0;
    *score80=0;
    *score100=0;
    for(int i=0;i<n;i++){
        if(student[i].score<0.3*full)
            (*score30)++;
        else if(student[i].score<0.6*full)
            (*score60)++;
        else if(student[i].score<0.8*full)
            (*score80)++;
        else
            (*score100)++;
    }
```

```
}
void analysis(stu student[],int *ScoreMax,int *ScoreMin,double *ScoreAve,int n,double
    full){
   double sum=0.0;
   for(int i=0;i<n;i++){
      if(student[i].score>student[*ScoreMax].score)
         *ScoreMax=i;
      if(student[i].score<student[*ScoreMin].score)
         *ScoreMin=i;
      sum+=student[i].score;
   }
   *ScoreAve=sum/n;
}

void stuoutput(stu student[], int n){
   Rank(student,n);
   fout<<"排名,学号,姓名,成绩\n";
   for(int i=0;i<n;i++){
      fout<<student[i].ScoreRank<<","<<student[i].id<<","
          <<student[i].name<<","<<student[i].score<<endl;
   }
}


int main(){
   cout<<"正在进入系统，请确保输入正确的数值。"<<endl
       <<"按照参考样例，共30名学生，试卷满分100分。\n\n";
   int n=0,full=0,score30=0,score60=0,score80=0,score100=0,ScoreMax=0,ScoreMin=0;
   double ScoreAve=0;
   stu student[30];
   cout<<"\n上传数据中，请确保学生信息按指定格式存储在 input.csv。\n";
   if(inputstu(&n,&full,student))
      cout<<"\n数据读取已完成,正在分析中。\n";
   else{
      system("pause");
      return 0;
   }

   while(1) {
      int operation=0;
      cout<<"\n\n数据分析已完成，可执行下列操作；\n请选择需要执行的操作\n";
      cout<<"1. 成绩排序\n";
      cout<<"2. 成绩统计\n";
      cout<<"3. 成绩分析\n";
      cout<<"4. 导出数据并退出\n\n";
      cin>>operation;

      switch(operation){
         case 1:
```

```
            Rank(student,n);
            for(int i=0;i<n;i++)
               cout<<student[i].ScoreRank<<" "<<student[i].id<<"号 "<<student[i].name<<"
                   "<<student[i].score<<" 分\n";
            cout<<"成绩排序已完成\n";
            break;
         case 2:
            cout<<"按原始分百分比转换至100分后\n";
            statistics(student,n,&score30,&score60,&score80,&score100,full);
            cout<<"0-30分:"<<score30<<"人; \n";
            cout<<"30-60分:"<<score60<<"人; \n";
            cout<<"60-80分:"<<score80<<"人; \n";
            cout<<"80-100分:"<<score100<<"人; \n";
            cout<<"已完成\n";
            break;
         case 3:
            analysis(student,&ScoreMax,&ScoreMin,&ScoreAve,n,full);
            cout<<"平均分为:  "<<ScoreAve<<endl;
            cout<<"最高分为: "<<student[ScoreMax].id<<"号 "<<student[ScoreMax].name<<"
                "<<student[ScoreMax].score<<"分。\n";
            cout<<"最低分为: "<<student[ScoreMin].id<<"号 "<<student[ScoreMin].name<<"
                "<<student[ScoreMin].score<<"分。\n";
            cout<<"成绩分析完成\n";


            break;
         case 4:
            stuoutput(student,n);
            cout<<"数据已导出，正在退出\n";
            system("pause");
            return 0;
            break;
         default:
            cerr<<"输入错误，请重新输入\n";
            break;
      }
   }
   system("pause");
   return 0;
}
```

2. 运行结果

```
正在进入系统，请确保输入正确的数值。
按照参考样例，共30名学生，试卷满分100分。


上传数据中，请确保学生信息按指定格式存储在 input.csv。
```

```
请输入需要统计的学生数量：30
请输入该科目的满分：100

数据读取已完成,正在分析中。



数据分析已完成，可执行下列操作;
请选择需要执行的操作
1．成绩排序
2．成绩统计
3．成绩分析
4．导出数据并退出

1
1 23号 Stu23 69.7 分
2 1号 Stu01 67.7 分
3 15号 Stu15 67.5 分
4 29号 Stu29 63.4 分
5 8号 Stu08 63.1 分
6 12号 Stu12 61.7 分
7 30号 Stu30 61.5 分
8 13号 Stu13 58.3 分
9 4号 Stu04 58 分
10 9号 Stu09 56 分
11 20号 Stu20 55.1 分
12 18号 Stu18 54.4 分
13 3号 Stu03 53 分
14 26号 Stu26 52.8 分
15 22号 Stu22 52.6 分
16 16号 Stu16 51.5 分
17 19号 Stu19 50.5 分
18 14号 Stu14 49.7 分
19 2号 Stu02 48.8 分
20 5号 Stu05 48.4 分
21 17号 Stu17 47.2 分
22 28号 Stu28 46.6 分
23 6号 Stu06 45 分
24 10号 Stu10 43.5 分
25 11号 Stu11 40.3 分
26 21号 Stu21 38.2 分
27 27号 Stu27 35 分
28 24号 Stu24 30.1 分
29 25号 Stu25 29.4 分
30 7号 Stu07 27.5 分
成绩排序已完成



数据分析已完成，可执行下列操作;
请选择需要执行的操作
```

```
1．成绩排序
2．成绩统计
3．成绩分析
4．导出数据并退出

2
按原始分百分比转换至100分后
0-30分:2人；
30-60分:21人；
60-80分:7人；
80-100分:0人；
已完成


数据分析已完成，可执行下列操作；
请选择需要执行的操作
1．成绩排序
2．成绩统计
3．成绩分析
4．导出数据并退出

3
平均分为： 50.8833
最高分为：23号 Stu23 69.7分。
最低分为：7号 Stu07 27.5分。
成绩分析完成


数据分析已完成，可执行下列操作；
请选择需要执行的操作
1．成绩排序
2．成绩统计
3．成绩分析
4．导出数据并退出

4
数据已导出，正在退出
请按任意键继续．．．

--------------------------------
Process exited after 7.047 seconds with return value 0
```