**《数据结构实验》报告**

|  |  |  |
| --- | --- | --- |
| **第 二 次实验** | **日期：2021.10.18** | **得分：** |
| **学号：** | **姓名：*Steven*** | **专业：** |

**一、实验目的**

1. 练习C语言函数的使用方法。

2. 练习使用malloc、realloc、free等函数动态分配内存的方法。

3. 练习指针变量的使用，了解其用法。

4. 理解并熟练结构体的使用方法。

**二、实验内容**

1、每个学生的成绩信息包括：学号、语文、数学、英语、总分、加权平均分；采用动态方法创建数组用于存储若干学生的成绩信息； 输入学生的学号、语文、数学、英语成绩；计算学生的总分和加权平均分（语文占30％，数学占50％，英语占20％）；输出学生的成绩信息。

2、可以在成绩数组末尾追加新学生的成绩信息；可以根据学号，从成绩数组中删除该学生的成绩信息。

3、可以根据学号或总分，升序排序成绩数组中学生的成绩信息。

4、设一元n次多项式P= a0+a1x+a2x2+……+anxn；用动态方法创建数组用于存储多项式P的系数a0、a1、a2、……、an；输入多项式P的各个系数及未知数x的值；计算并输出多项式P的值。（注：不允许使用求幂函数pow( )）

**三、数据结构及算法设计**

1、

对于每个学生对象，都拥有相同的属性——学号和各种成绩，所以可以建立学生结构体，用以存放每个学生的各种信息；同时由于总分和加权平均分都可以计算，故将计算总分和加权平均分的函数写在结构体中。

此外再分析功能：需要动态创建学生数组、能够逐对象输入信息、能够输出学生成绩。因此还需要3个函数，分别负责：(1)为学生对象申请内存空间，也就是动态创建学生数组；(2)为这些学生中的各个属性赋值；(3)能够输出单个或全部学生。

具体实现方面，(1)根据询问到的学生人数，使用malloc函数为学生指针申请内存。(2)通过指针遍历学生数组，并且顺次输入属性信息。要注意学号长度事先并不清楚，所以需要在输入前询问并动态分配字符串长度。(3)首先询问是查看单个学生信息还是查看所有学生信息，然后输出单个或多个对象的信息。由于每个对象的信息项目相同，所以可以把输出信息的函数写在结构体中。

2、

在第一题的基础上增加了增添和删除学生的功能，也就是对原本的数组进行realloc函数操作。本身不难，只是需要考虑内存分配的问题。

增添学生时先询问增添量，然后使用realloc函数扩大数组，并且根据之前的学生数量，利用数组的下表访问方式遍历未初始化的几个对象，进行初始化操作。

删除学生时要先获得被删除学生的数组序号，然后将其后的学生信息都向前一位进行拷贝，这样就将被删除的学生信息覆盖掉了，且数组最后两位重复。随后再使用realloc函数将数组空间占用减少1，这样就完成了删除。

3、

本质上不难理解，不过由于这是结构体数组的排序，所以要综合运用排序算法和指针赋值进行。

先请求排序方式，然后对数组元素进行希尔排序。在排序的时候根据用户的选择，选择进行比较的值。希尔排序需要1个辅助空间，在本题中使用1个student对象充当，因为对于student对象来说，赋值号就是值而非地址的复制。

4、

分析题目，功能就只是完成一个简单的多项式求和问题。难点有两个，一个是阶数不确定，需要动态分配系数数组的长度，二是不允许使用求幂函数pow( )，直接用循环进行高阶计算会略显繁琐。所以在用代码实现的时候，另外附加了一个函数用来求幂。

**四、核心程序代码（给出必要注释）**

1.

|  |
| --- |
| typedef struct student\_message student; //结构体设计 |
| typedef struct student\_message {  char\* ID;//学号  double chineseScore;//语文  double mathScore;//数学  double englishScore;//英语  double totalScore;//总分  double aveScore = 0;//加权平均分  //计算加权平均分  void calcAveScore() {  aveScore = chineseScore \* 0.3 + mathScore \* 0.5 + englishScore \* 0.2;  }  //计算总分  void calcTotalScore() {  totalScore = chineseScore + mathScore + englishScore;  }  //输出学生信息  void printStuMes() {  printf(" 学号：%s\n", ID);  printf(" 语文成绩：%5.2lf\n", chineseScore);  printf(" 数学成绩：%5.2lf\n", mathScore);  printf(" 英语成绩：%5.2lf\n", englishScore);  printf(" 总成绩：%5.2lf\n", totalScore);  printf(" 加权平均分：%5.2lf\n", aveScore);  }  }student; |

|  |
| --- |
| 全局变量 |
| student\* stu = NULL;//学生数组，用来存放学生信息  int stu\_num;//学生人数  int stu\_ID\_length;//学号的长度 |
| int createStuList()//申请内存，并返回申请结果 |
| int createStuList() {  system("cls");  printf("-> 初始化学生人数\n");  printf("请输入学生人数：");  scanf("%d", &stu\_num);  stu = (student\*)malloc(stu\_num \* sizeof(student));  //检测是否分配成功  return stu ? 0 : 1;  } |
| void setStuMes(student\* stu\_pointer, int stu\_ID\_length, int index) //设置学生信息 |
| void setStuMes(student\* stu\_pointer, int stu\_ID\_length, int index) {  stu\_pointer->ID = (char\*)malloc((stu\_ID\_length + 1) \* sizeof(char));  printf("请输入第%d个学生的学号：", index + 1);  scanf("%s", stu\_pointer->ID);  printf("请输入第%d个学生的语文成绩：", index + 1);  scanf("%lf", &stu\_pointer->chineseScore);  printf("请输入第%d个学生的数学成绩：", index + 1);  scanf("%lf", &stu\_pointer->mathScore);  printf("请输入第%d个学生的英语成绩：", index + 1);  scanf("%lf", &stu\_pointer->englishScore);  fflush(stdin);  stu\_pointer->calcTotalScore();  stu\_pointer->calcAveScore();  } |
| void iniStuMes() //初始化学生信息 |
| void iniStuMes() {  system("cls");  printf("-> 初始化学生信息\n");  student\* stu\_pointer = stu;  printf("请输入学号长度：");  scanf("%d", &stu\_ID\_length);  fflush(stdin);  for (int index = 0; index < stu\_num; index++, stu\_pointer++) {  setStuMes(stu\_pointer, stu\_ID\_length, index);  }  printf("您已初始化完成%d个同学的信息\n", stu\_num);  system("pause");  } |
| int getStuIndex(char\* targetID) //依据学号，获取学生在数组中的序号（输出时需要） |
| int getStuIndex(char\* targetID) {  int index;  for (index = 0; index < stu\_num; index++) {  if (!strcmp(targetID, stu[index].ID)) {  return index;  }  }  return -1;  } |
| void printStuMes()//输出学生信息 |
| void printStuMes() {  system("cls");  printf("-> 查看单个学生信息\n");  char\* targetID = (char\*)malloc((stu\_ID\_length + 1) \* sizeof(char));  printf("请输入想要查看信息的学生的学号：");  scanf("%s", targetID);  int targetIndex = getStuIndex(targetID);//获取被删除的index  if (targetIndex == -1) {  printf("未查找到该学生，即将退出\n");  system("pause");  return;  }  printf("\n--------------学生信息--------------\n");  stu[targetIndex].printStuMes();  printf("输出完成\n");  free(targetID);  system("pause");  } |
| void printAllStuMes()//输出所有学生信息 |
| void printAllStuMes() {  system("cls");  printf("-> 查看所有学生信息\n");  for (int index = 0; index < stu\_num; index++) {  printf("\n-------------学生信息%-2d-------------\n", index + 1);  stu[index].printStuMes();  }  printf("输出完成\n");  system("pause");  } |

2.

增添append和del函数，用于修改学生数量

|  |
| --- |
| void appendStuMes()//追加学生信息 |
| void appendStuMes() {  system("cls");  printf("-> 追加学生信息\n");  int append\_num;  printf("当前已有%d位同学的信息，仍需追加多少同学的信息：", stu\_num);  scanf("%d", &append\_num);  stu = (student\*)realloc(stu, (append\_num + stu\_num) \* sizeof(student));  if (!stu) {  printf("追加内存失败，程序即将退出\n");  exit(-1);  }  student\* stu\_pointer = &stu[stu\_num];//现在指向了第一个未初始化的地址  for (int append = 0; append < append\_num; append++, stu\_pointer++) {  setStuMes(stu\_pointer, stu\_ID\_length, stu\_num + append);  }  printf("您已完成对%d位同学信息的追加\n", append\_num);  stu\_num += append\_num;//更新学生人数  system("pause");  } |
| int delStuMes()//依据学号删除学生信息 |
| int delStuMes() {  system("cls");  printf("-> 删除学生信息\n");  char\* targetID = (char\*)malloc((stu\_ID\_length + 1) \* sizeof(char));  printf("请输入被删除的学生的学号：");  scanf("%s", targetID);  int targetIndex = getStuIndex(targetID);//获取被删除的index  if (targetIndex == -1) {  printf("未查找到该学生，删除无法执行，即将退出删除\n");  system("pause");  return -1;  }  //将其之后的学生信息都向前平移，覆盖掉该学生  for (int i = targetIndex; i < stu\_num - 1; i++) {  strcpy(stu[i].ID, stu[i + 1].ID);  stu[i].chineseScore = stu[i + 1].chineseScore;  stu[i].mathScore = stu[i + 1].mathScore;  stu[i].englishScore = stu[i + 1].englishScore;  stu[i].totalScore = stu[i + 1].totalScore;  stu[i].aveScore = stu[i + 1].aveScore;  }  //重分配空间  stu = (student\*)realloc(stu, (stu\_num - 1) \* sizeof(student));  stu\_num--;  printf("删除成功\n");  system("pause");  free(targetID);  return 0;  } |

3.

增添两个用于排序的函数

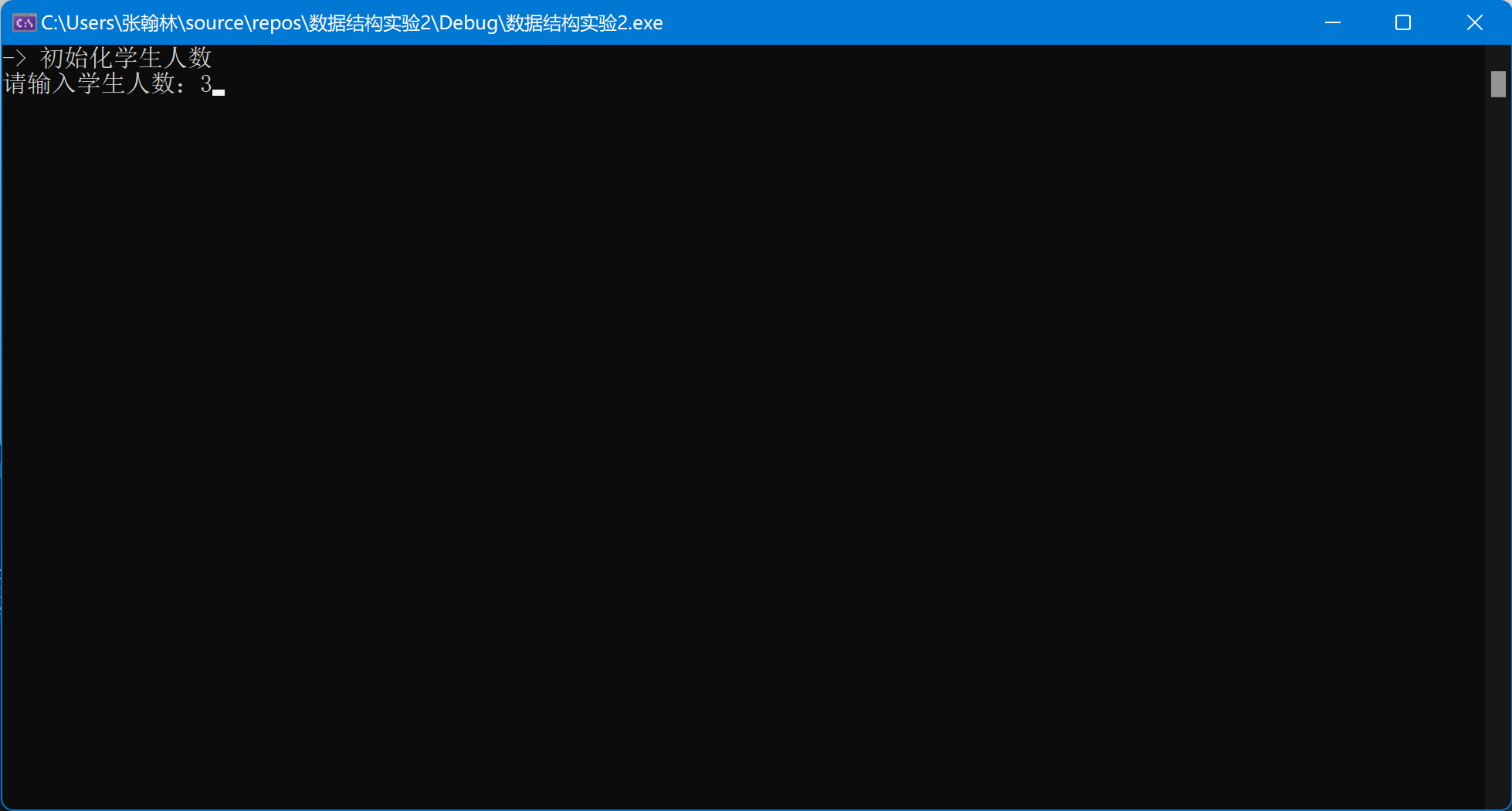
|  |
| --- |
| void sortStuMes()//学生信息排序 |
| void sortStuMes() {  system("cls");  printf("-> 学生信息排序\n");  void sort\_ID(student \* stu, int stu\_num);  void sort\_totalScore(student \* stu, int stu\_num);  int rank\_basis;  printf("请选择排序依据：\n");  printf("1. 学号\n2. 总分\n");  scanf("%d", &rank\_basis);  switch (rank\_basis) {  case 1:  sort\_ID(stu, stu\_num);  printf("按学号升序排列完成，主菜单中输出所有学生信息即可查看\n");  break;  case 2:  sort\_totalScore(stu, stu\_num);  printf("按总分升序排列完成，主菜单中输出所有学生信息即可查看\n");  break;  default:  printf("您的输入有误，排序已终止\n");  break;  }  system("pause");  } |
| void sort\_ID(student\* stu, int stu\_num)//根据学号升序排列 |
| void sort\_ID(student\* stu, int stu\_num) {  int i, j, k, gap;// gap 为步长  student tmp;  for (gap = stu\_num / 2; gap > 0; gap /= 2) { // 步长初始化为数组长度的一半，每次遍历后步长减半,  for (i = 0; i < gap; ++i) { // 变量 i 为每次分组的第一个元素下标  for (j = i + gap; j < stu\_num; j += gap) {  tmp = stu[j];  k = j - gap;  while (k >= 0 && atoi(stu[k].ID) > atoi(tmp.ID)) {  stu[k + gap] = stu[k]; // 将在stu[i]前且比tmp的值大的元素向后移动一位  k -= gap;  }  stu[k + gap] = tmp;  }  }  }  } |
| void sort\_totalScore(student\* stu, int stu\_num) //根据总分升序排列 |
| void sort\_totalScore(student\* stu, int stu\_num) {  int i, j, k, gap;// gap 为步长  student tmp;  for (gap = stu\_num / 2; gap > 0; gap /= 2) { // 步长初始化为数组长度的一半，每次遍历后步长减半,  for (i = 0; i < gap; ++i) { // 变量 i 为每次分组的第一个元素下标  for (j = i + gap; j < stu\_num; j += gap) { //对步长为gap的元素进行直插排序，当gap为1时，就是直插排序  tmp = stu[j];  k = j - gap;  while (k >= 0 && stu[k].totalScore > tmp.totalScore) {  stu[k + gap] = stu[k]; // 将在stu[i]前且比tmp的值大的元素向后移动一位  k -= gap;  }  stu[k + gap] = tmp;  }  }  }  } |

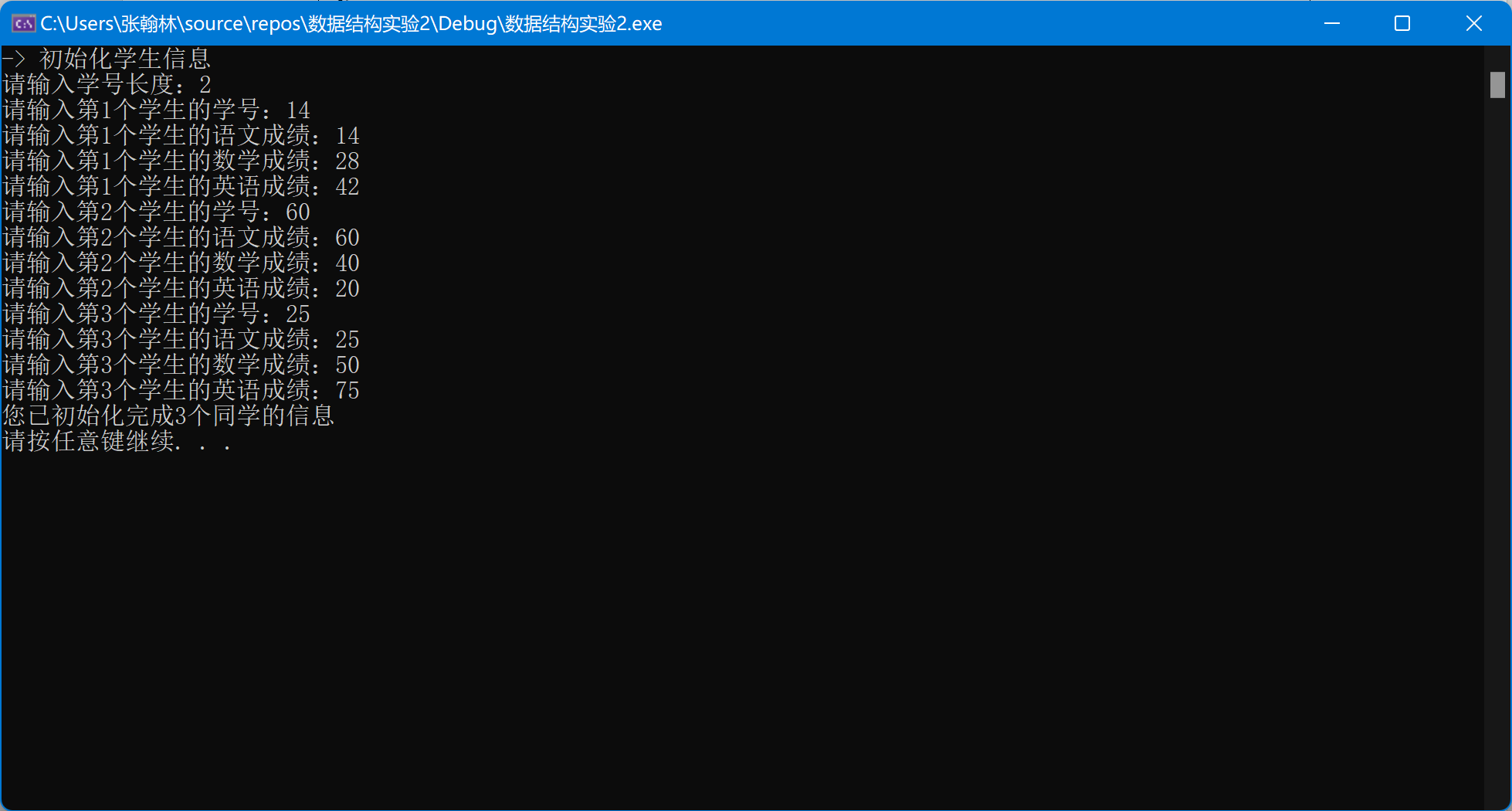
4.

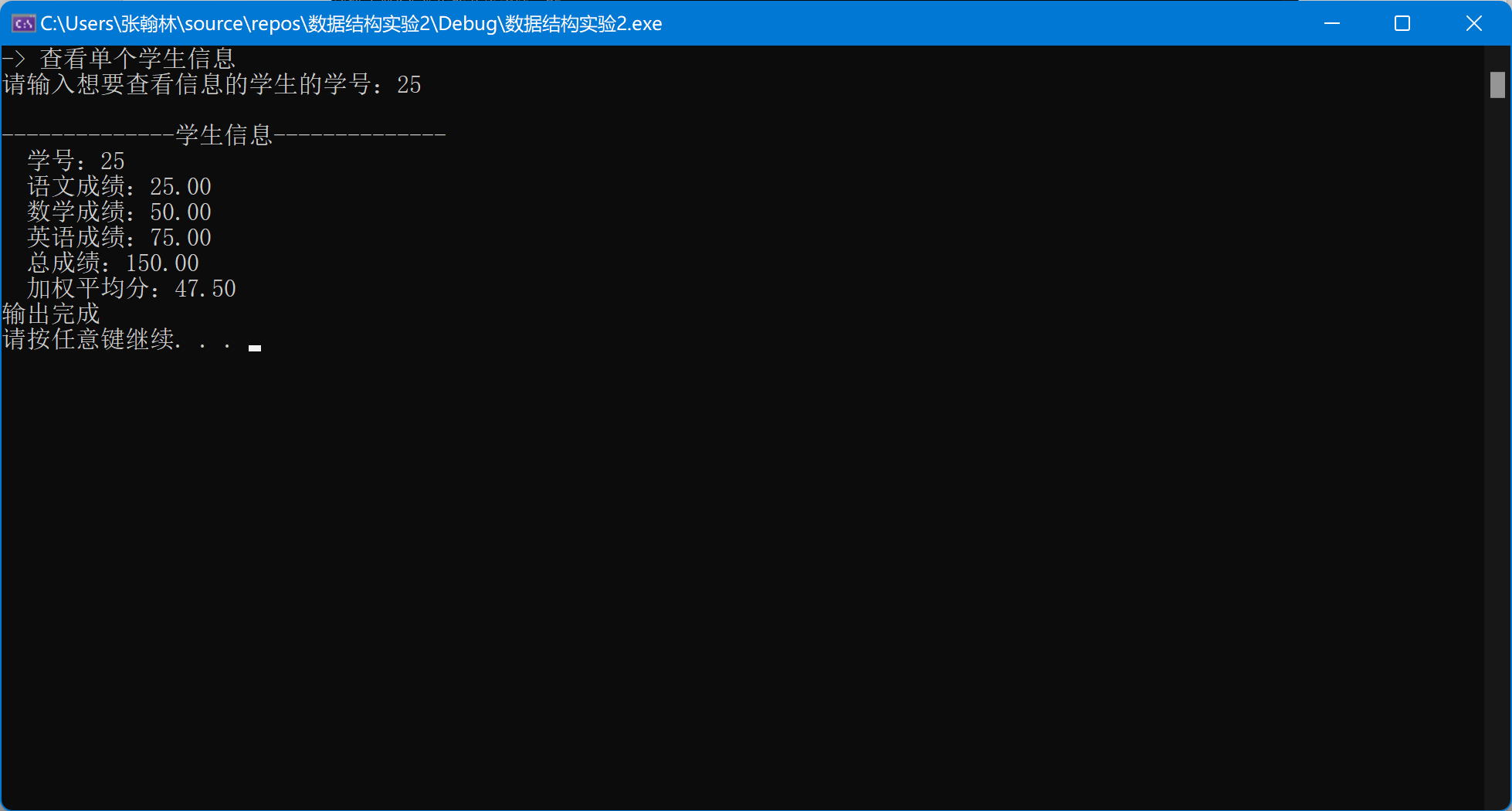
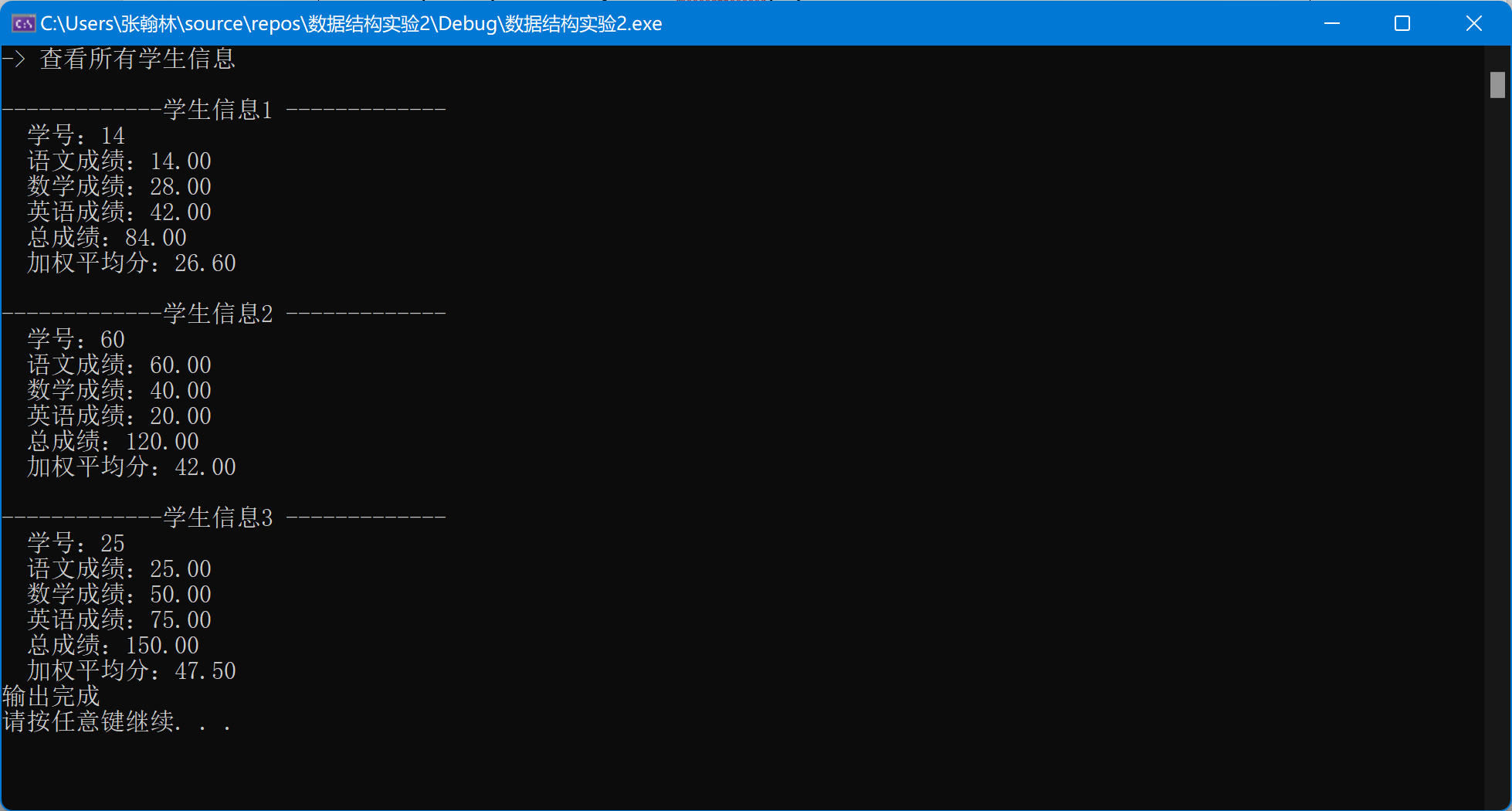
|  |
| --- |
| int main() |
| int main() {  int power\_num;//阶数  double x, result = 0, \* para; //分别表示未知数、结果、参数数组  double power(double x, int power);//函数头声明  printf("请输入多项式的次数：");  scanf("%d", &power\_num);  para = (double\*)malloc((power\_num + 1) \* sizeof(double));  for (int i = 0; i < power\_num + 1; i++) {  printf("请输入%d次项的系数，如无则填0：", i);  scanf("%lf", &para[i]);  }  printf("请输入未知量的赋值:");  scanf("%lf", &x);  for (int i = 0; i < power\_num + 1; i++) {  result += para[i] \* power(x, i);  }  printf("%lf", result);  } |
| double power(double x, int pow\_num) |
| double power(double x, int pow\_num) {  if (pow\_num == 0) return 1;  else return x \* power(x, pow\_num - 1);  } |

**五、测试及结果（给出测试用例及测试结果）**

1.测试用例：学生数为3，学号长度为2；成绩分配如下：

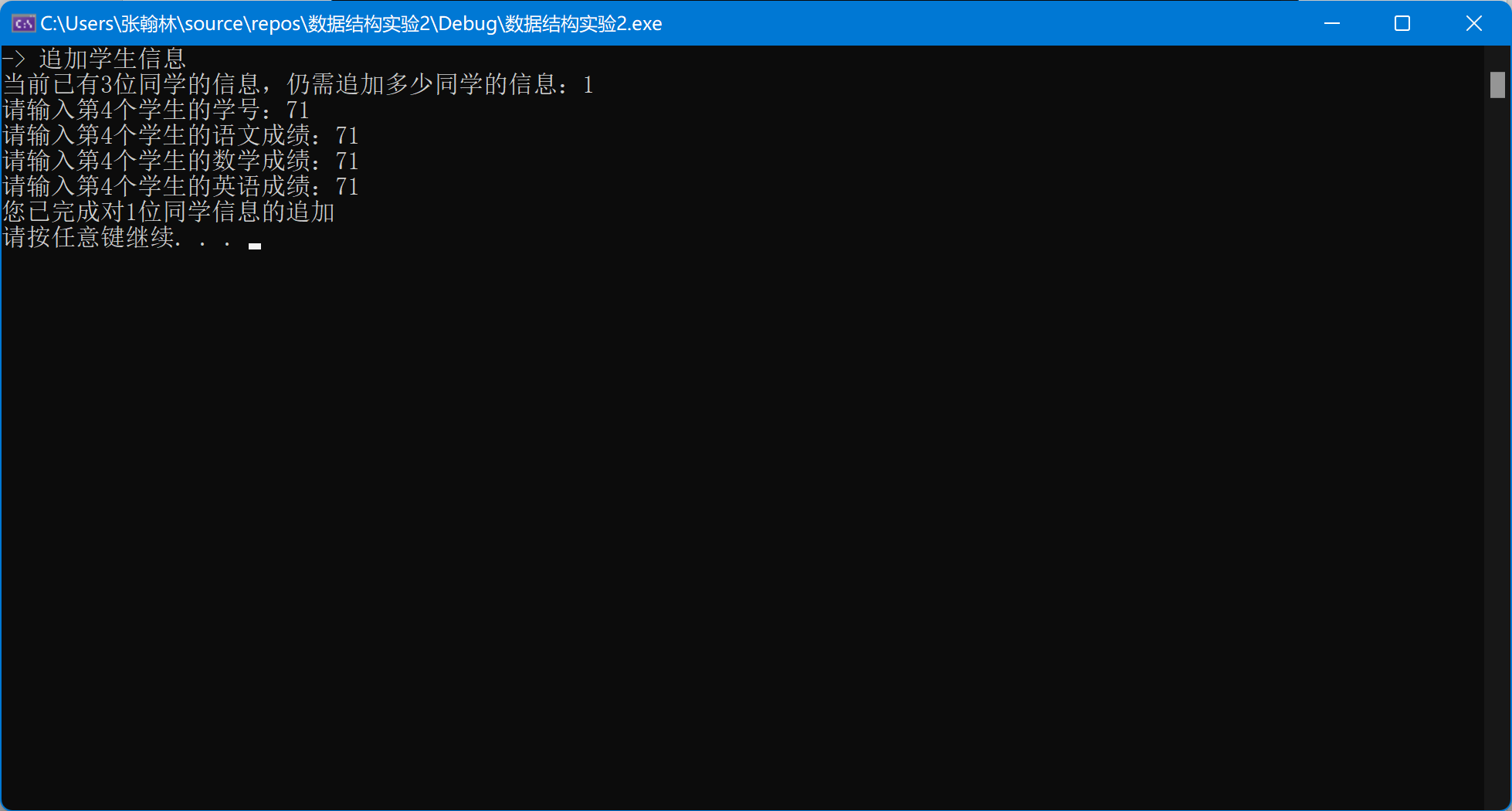


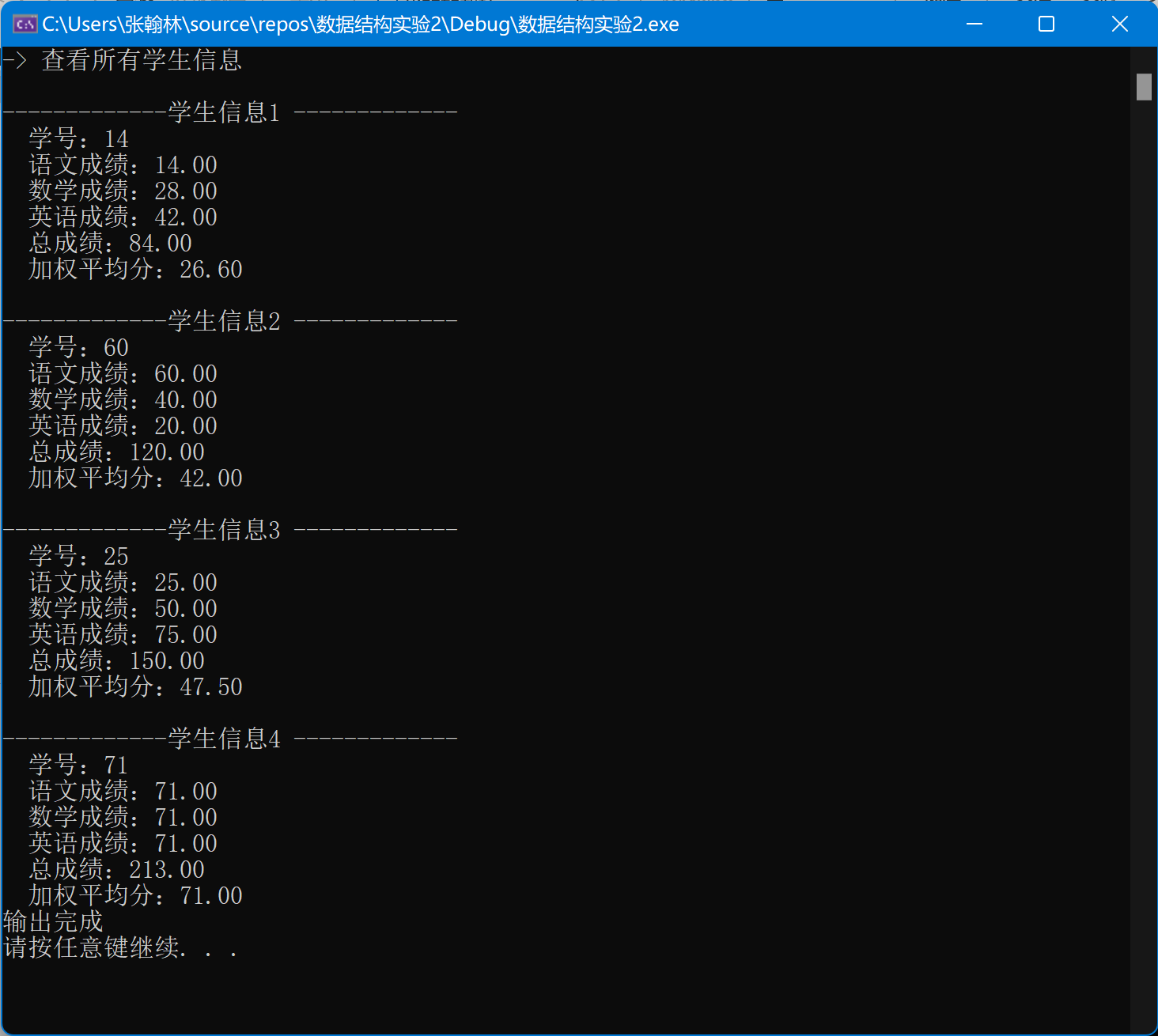




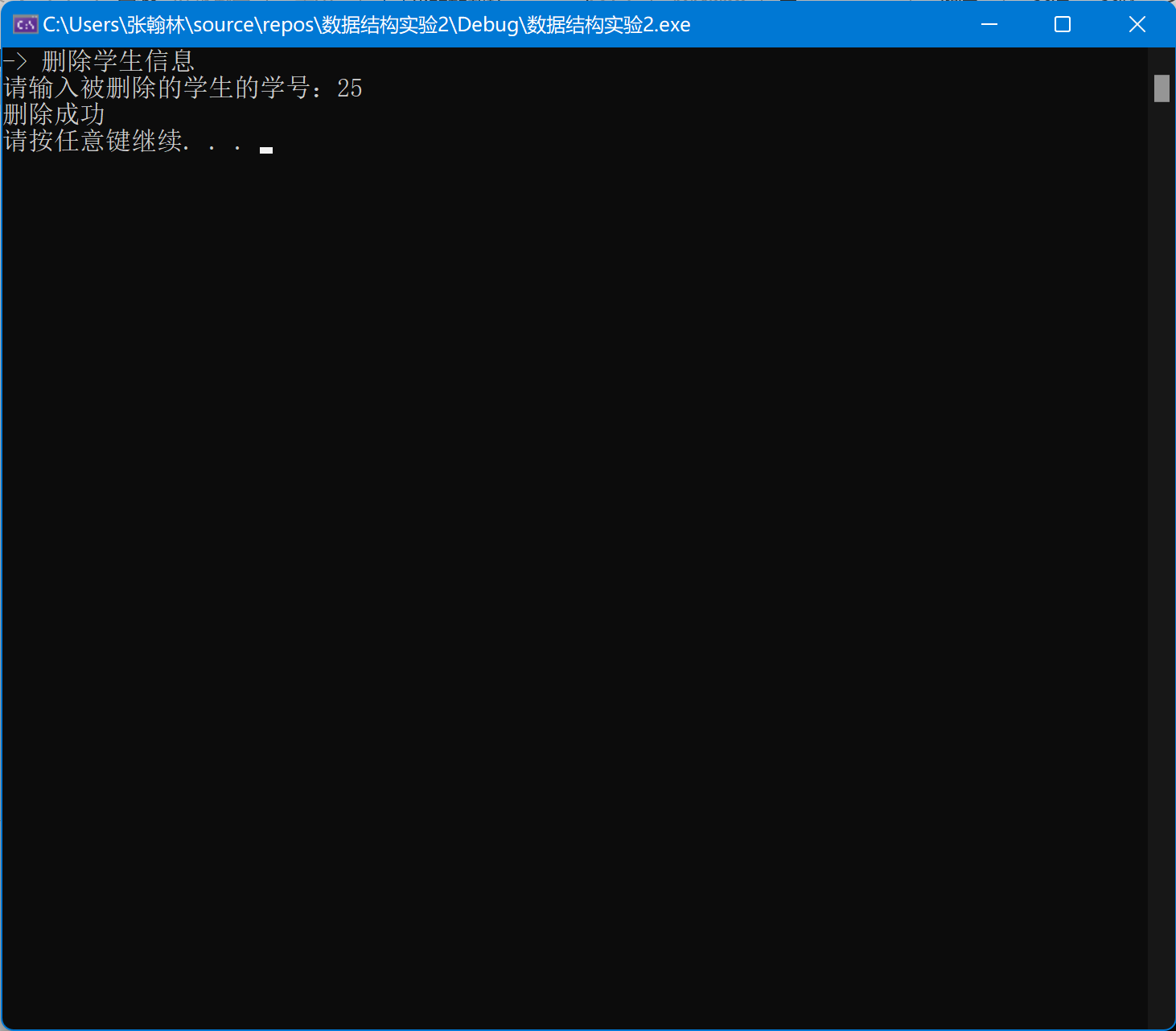
2.

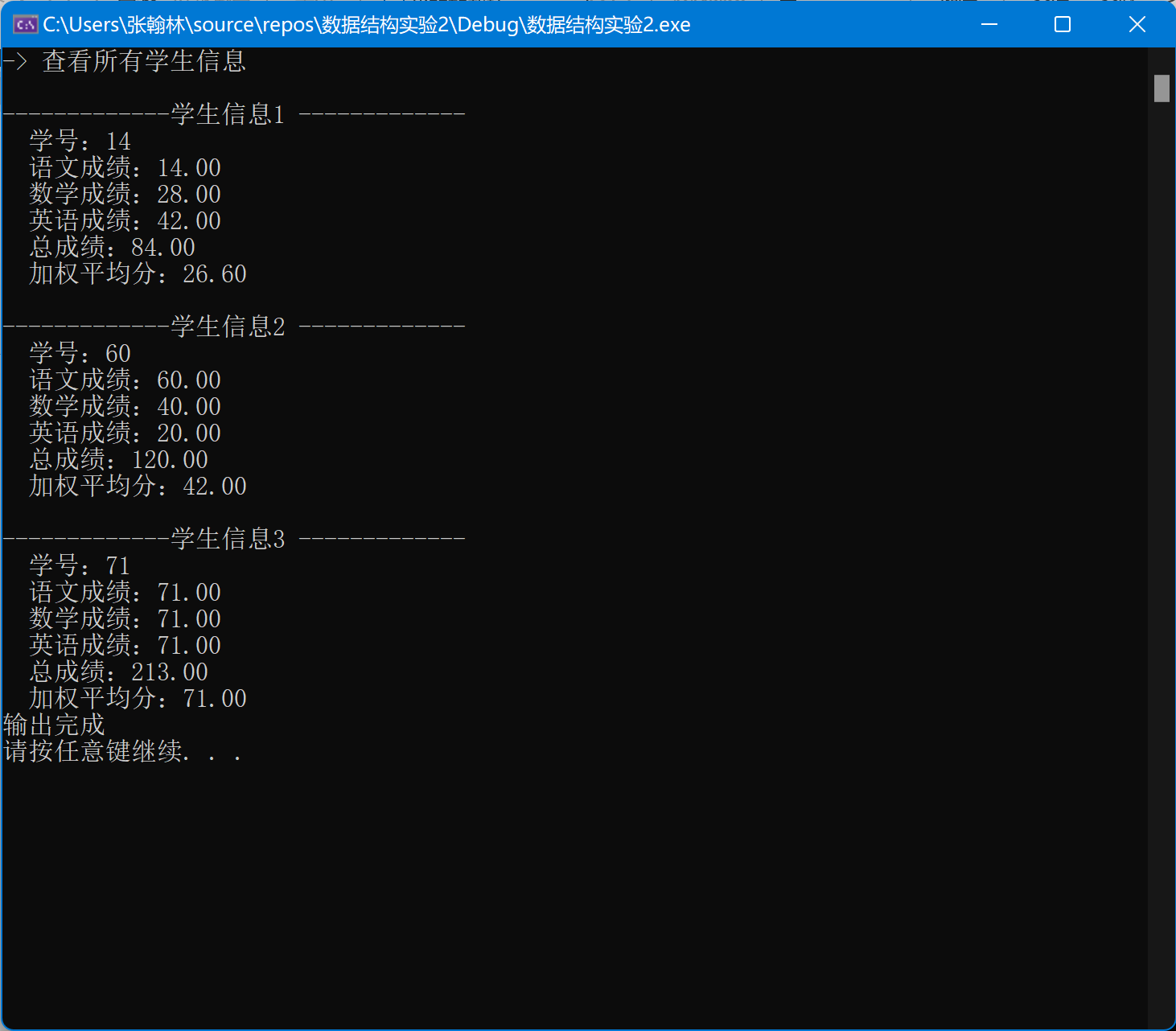
追加1个：





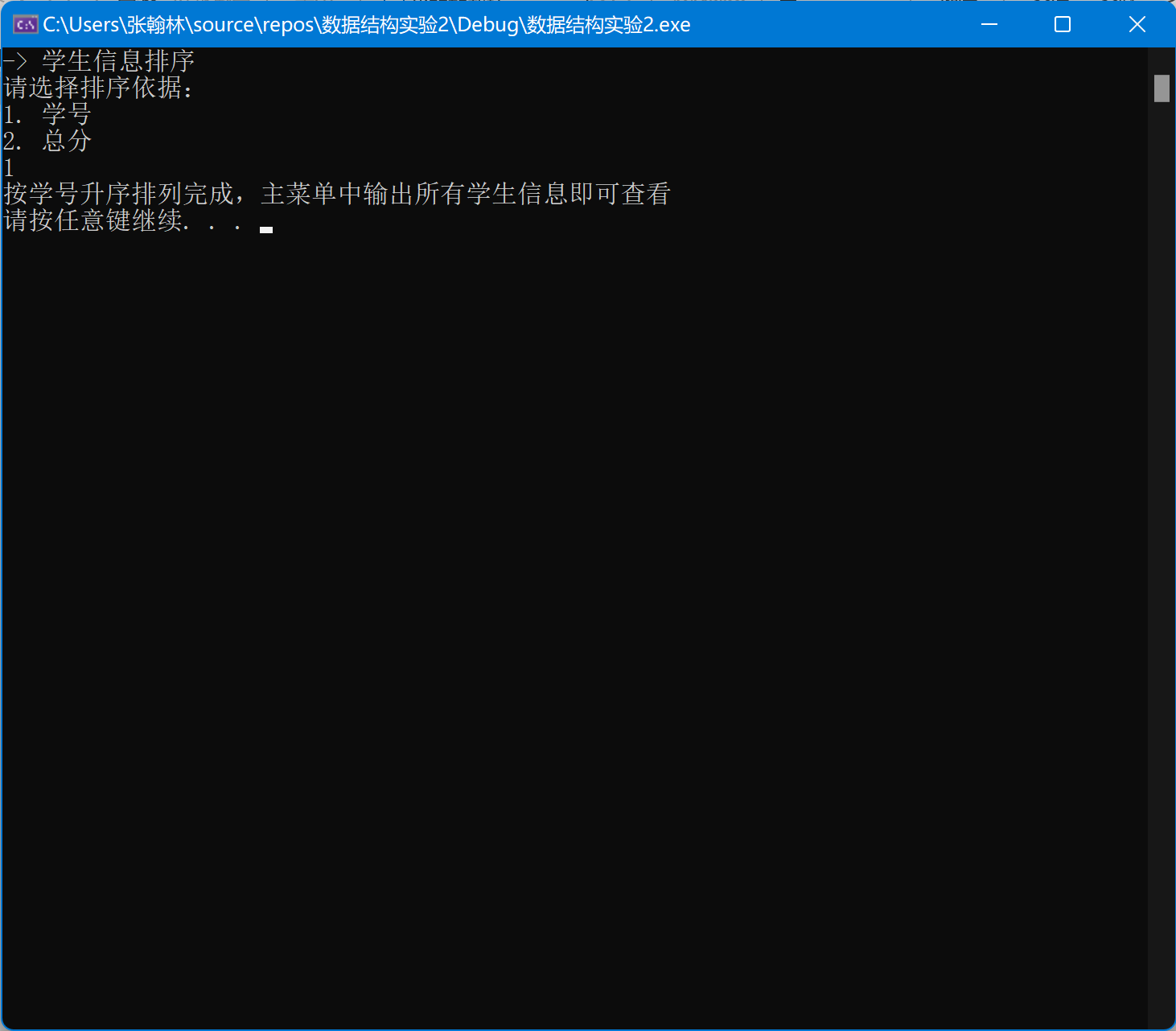
删除1个：

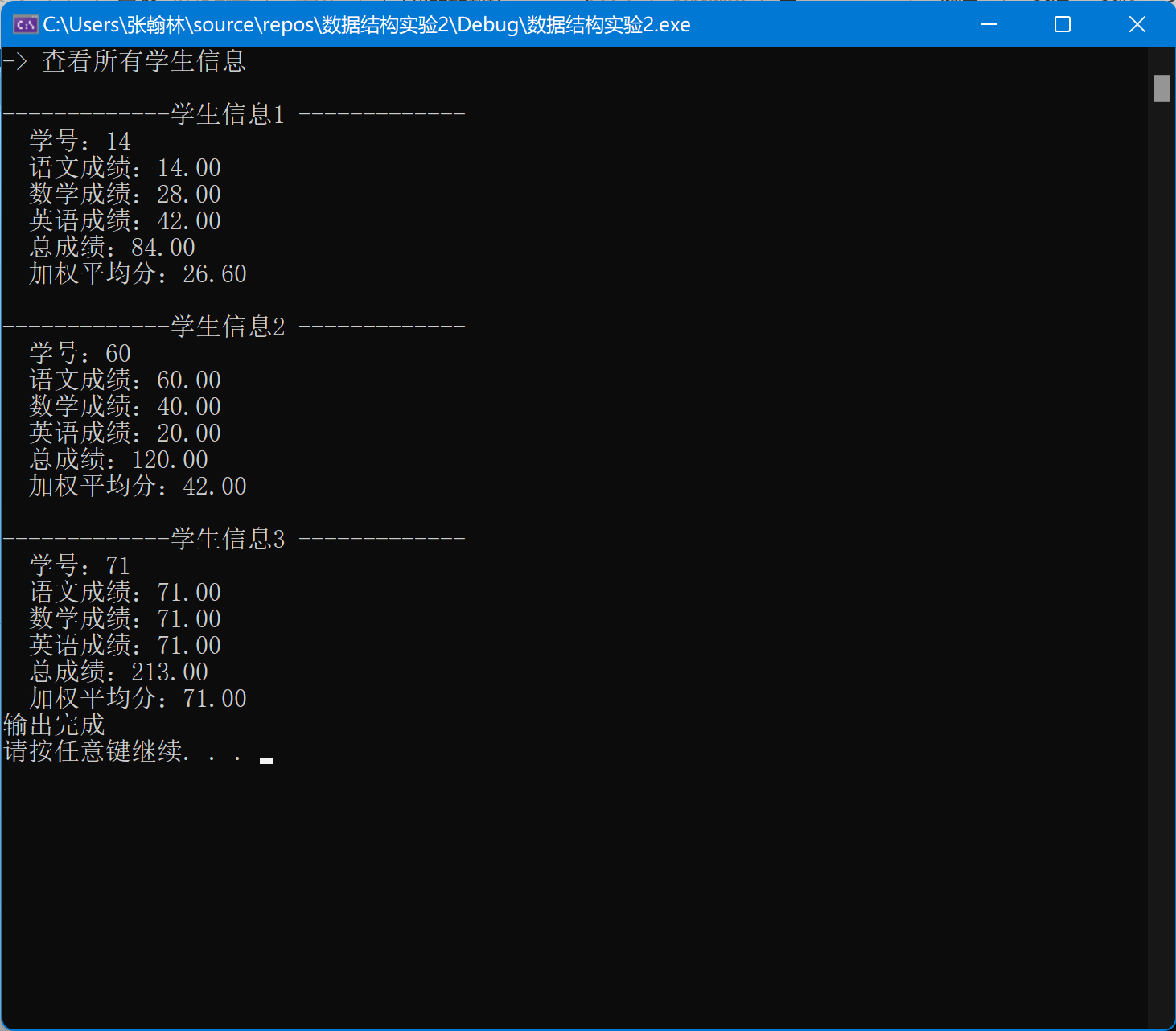




3.

按照学号排序：

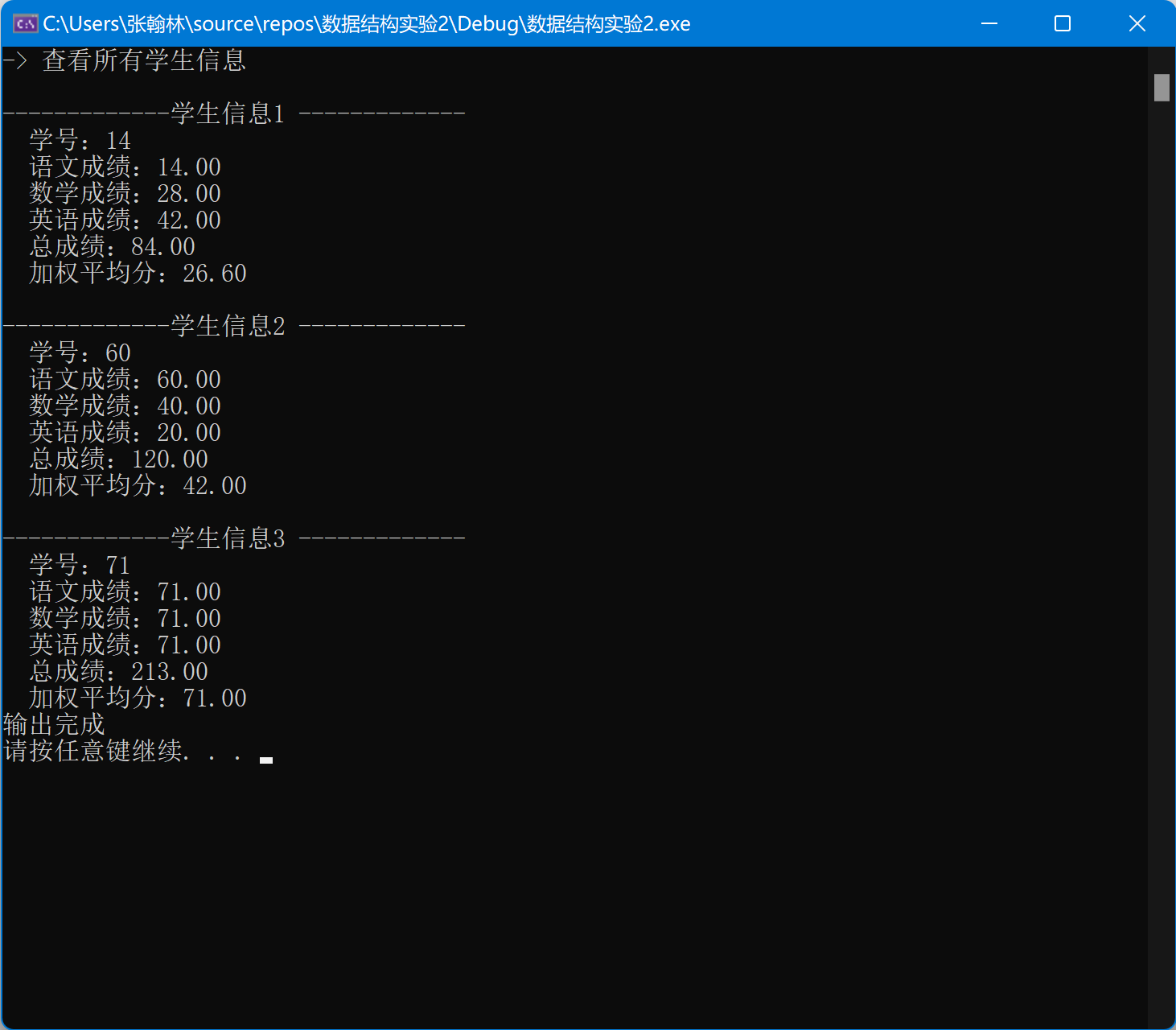




按照总分排序：

文本

描述已自动生成



4.测试用例：1+2\*6+3\*6^2+4\*6^3+5\*6^4

