**《数据结构实验》报告**

|  |  |  |
| --- | --- | --- |
| **第 三 次实验** | **日期：2021.11.1** | **得分：** |
| **学号：** | **姓名：*Steven*** | **专业：智能科学与技术** |

**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。

5. 理解并学会链表的构建和操作

**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；采用单链表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、可以在成绩链表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，升序排序成绩链表中学生的成绩信息。

4、给定存储若干整数的单链表，实现单链表逆转。

**三、数据结构及算法设计**

1、

此题和上次用动态数组功能相同，结构体设计和之前相同，故不作多言。

此外再分析功能：需要创建链表、能够逐对象输入信息、能够输出学生成绩。因此还需要若干个函数，分别负责：(1)创建每个结构体节点并连成链表；(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信息。

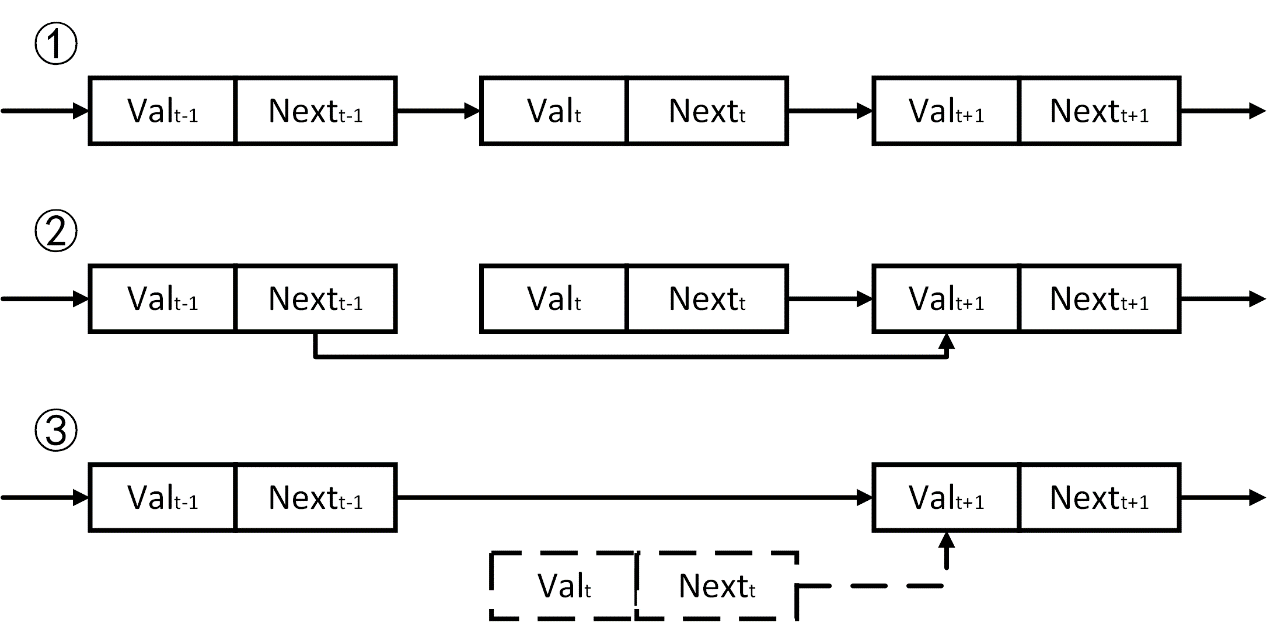
具体实现方面，(1)根据询问到的学生人数，使用for循环连续使用malloc创建节点，并进行链接。需要注意的是，函数内定义局部变量在函数结束后即销毁，所以创建的节点不能是变量类型，只有创建指针并为之malloc分配过空间才不会被自动销毁。(2)通过指针遍历学生链表，并依次输入属性信息即可。要注意学号长度事先并不清楚，所以需要在输入前询问并动态分配字符串长度。(3)首先询问是查看单个学生信息还是查看所有学生信息，然后输出单个或多个对象的信息。由于希望本题中返回查找用户地址的方法能尽量多地被调用，所以本题中的查找方法返回的是目标对象的前一个，以便于能在删除对象时使用。

2、

在第一题的基础上增加了增添和删除学生的功能，也就是链表的尾插法增添和链表的节点删除。

增添学生时先询问增添量，然后用循环将指针遍历到表尾，记录下当前表尾的地址；再使用for循环依次进行尾插法增添节点，最后从之前表尾的地址开始遍历新创建的部分，为其写入属性。注意，新创建的节点里，ID是没有被分配空间的，所以在写入属性之前先为新创建的节点分配空间。

删除学生时要先获得被删除学生的前一个节点和该节点，然后前一个节点的指针域跳过被删除的节点，直接指向其下一个，最后把被删除的节点free掉即可，示意图如下。

图示

描述已自动生成图示

描述已自动生成

3、

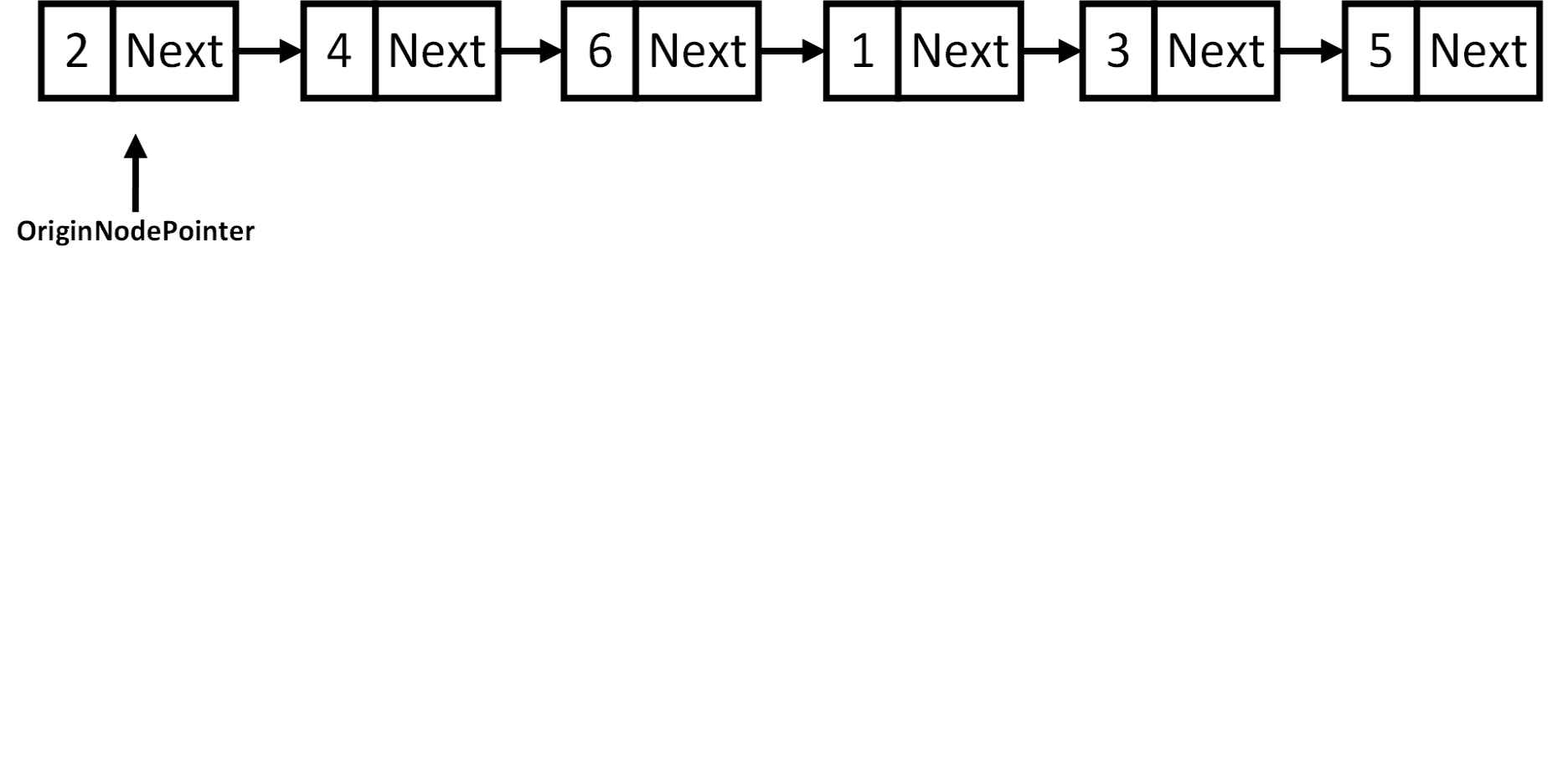
本质和上次的相同，只不过本题中由于链表元素的操作太过麻烦且不能随机取值，所以本题中先把所有元素拷贝进一个数组，在数组中进行排序操作，最后再将排序好的内容依次写入链表，并把数组清除掉，以完成排序。

4、

由于本题并没有说需要在原链表的基础上操作，所以本题最简单的方式就是重建一个新的倒序链表，结合头插法可以轻松实现。

首先需要originNodePointer正向遍历原链表，lastNode保存上次创建的节点。然后创建一个新节点newNode，val为当前值，指针域指向lastNode，再更新lastNode为newNode以便下次循环使用。

示意图如下：



**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=NULL;//学号  double chineseScore = 0;//语文  double mathScore = 0;//数学  double englishScore = 0;//英语  double totalScore = 0;//总分  double aveScore = 0;//加权平均分  student\_message\* next = NULL;  //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;  }  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);  }  }stuNode; |

|  |
| --- |
| 全局变量 |
| stuNode\* stu\_top = (stuNode\*)malloc(sizeof(stuNode));  stuNode\* stu\_end = NULL;//指向尾节点，在增添学生时有用  stuNode\* stu\_pointer = stu\_top;//共用的pointer  int stu\_num;//学生人数  int stu\_ID\_length;//学号的长度 |

|  |
| --- |
| int createStuList()//构建链表 |
| 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\_pointer = stu\_top;//指向头节点  for (int i = 0; i < stu\_num; i++) {  stuNode\* stu\_next = (stuNode\*)malloc(sizeof(stuNode));//创建下个  stu\_next->next = NULL;  stu\_pointer->next = stu\_next;  stu\_pointer = stu\_pointer->next;  }  stu\_pointer->next = NULL;  stu\_end = stu\_pointer;  return stu\_pointer ? 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();//计算总分  stu\_pointer->calcAveScore();//执行计算加权平均分  } |

|  |
| --- |
| void iniStuMes() //初始化学生信息 |
| void iniStuMes() {  system("cls");  printf("-> 初始化学生信息\n");  int index = 0;  stuNode\* stu\_pointer = stu\_top->next;//指向实际的第一个学生  printf("请输入学号长度：");  scanf("%d", &stu\_ID\_length);  while (stu\_pointer) {  setStuMes(stu\_pointer, stu\_ID\_length, index);  index++;  stu\_pointer = stu\_pointer->next;  }  printf("您已初始化完成%d个同学的信息\n", stu\_num);  system("pause");  } |

|  |
| --- |
| stuNode\* getPreTargetStu(char\* targetID) //依据学号，获取目标的前一个学生（为了删除时也能使用） |
| stuNode\* getPreTargetStu(char\* targetID) {  stu\_pointer = stu\_top;//指向头节点  for (int index = 0; index < stu\_num; index++) {  if (index == 0 && !strcmp(targetID, stu\_pointer->next->ID)) {//表示查找的数据恰好为链表中的头节点  return stu\_top;  }  else if (!strcmp(targetID, stu\_pointer->next->ID)) {//查找的数据不是第一个  return stu\_pointer;  }  stu\_pointer = stu\_pointer->next;  }  return NULL;  } |

|  |
| --- |
| void printStuMes()//输出学生信息 |
| void printStuMes() {  system("cls");  printf("-> 查看单个学生信息\n");  stuNode\* stu\_target;  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);  stu\_target = getPreTargetStu(targetID)->next;//获取到真正目标学生的地址  if (!stu\_target) {  printf("未查找到该学生，即将退出\n");  system("pause");  return;  }  printf("\n--------------学生信息--------------\n");  stu\_target->printStuMes();  printf("输出完成\n");  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");  int index = 0;  stuNode\* stu\_pointer = stu\_top->next;  while (stu\_pointer) {  printf("\n-------------学生信息%-2d-------------\n", index + 1);  stu\_pointer->printStuMes();  stu\_pointer = stu\_pointer->next;  index++;  }  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;//要增添的学生数量  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\_pointer = stu\_end;//现在pointer指向了原链表的最后一项。  for (int i = 0; i < append\_num; i++) {  stuNode\* stu\_next = (stuNode\*)malloc(sizeof(stuNode));//创建节点  stu\_next->next = NULL;  stu\_pointer->next = stu\_next;  stu\_pointer = stu\_pointer->next;  setStuMes(stu\_pointer, stu\_ID\_length, stu\_num + i);  }  stu\_end = stu\_pointer;  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));  stuNode\* stu\_preTarget = NULL, \* stu\_target = NULL;  printf("请输入被删除的学生的学号：");  scanf("%s", targetID);  stu\_preTarget = getPreTargetStu(targetID);//获取被删除的前一项  if (!stu\_preTarget) {  printf("未查找到该学生，删除无法执行，即将退出删除\n");  system("pause");  return FALSE;  }  //执行删除程序  stu\_target = stu\_preTarget->next;//被删除的那一项  stu\_preTarget->next = stu\_target->next;//被删除的前一项的指针域指向被删除的后一项  free(stu\_target);  stu\_num--;  printf("删除成功\n");  system("pause");  return TRUE;  } |

3.

排序的过程如下：

|  |
| --- |
| void sortStuMes()//学生信息排序 |
| void sortStuMes() {  system("cls");  printf("-> 学生信息排序\n");  int rank\_basis;  int sort(int choice);  printf("请选择排序依据：\n");  printf("1. 学号\n2. 总分\n");  scanf("%d", &rank\_basis);  printf(sort(rank\_basis) ? "排序完成" : "排序过程中出现问题，未完成排序");  system("pause");  } |

|  |
| --- |
| int sort(int choice) //具体的排序方法 |
| int sort(int choice) {  int RANK\_ID = 1, RANK\_SCORE = 2;  stuNode\* stu = (stuNode\*)malloc(sizeof(stuNode) \* stu\_num);  stu\_pointer = stu\_top->next;  for (int i = 0; i < stu\_num; i++) {  stu[i].ID = (char\*)malloc(sizeof(char)\*stu\_ID\_length);//分配空间  //信息拷贝  strcpy(stu[i].ID, stu\_pointer->ID);  stu[i].chineseScore = stu\_pointer->chineseScore;  stu[i].mathScore = stu\_pointer->mathScore;  stu[i].englishScore = stu\_pointer->englishScore;  stu[i].totalScore = stu\_pointer->totalScore;  stu[i].aveScore = stu\_pointer->aveScore;  stu[i].next = NULL;  stu\_pointer = stu\_pointer->next;  }  //接下来是排序  int i, j, k, gap;// gap 为步长  stuNode tmp;  for (gap = stu\_num / 2; gap > 0; gap /= 2) { // 步长初始化及减半  for (i = 0; i < gap; ++i) {  for (j = i + gap; j < stu\_num; j += gap) { //对步长为gap的元素进行排序，当gap为1时，就是直插排序  tmp = stu[j];// 备份stu[j]  k = j - gap;// j初始化为i的前一个元素（与i相差gap）  while (k >= 0 && (choice == RANK\_ID ? (atoi(stu[k].ID) > atoi(tmp.ID)) : (stu[k].totalScore > tmp.totalScore))) {  stu[k + gap] = stu[k]; //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;  }  }  }  stu\_pointer = stu\_top;  for (int i = 0; i < stu\_num; i++) {  //拷贝信息  strcpy(stu\_pointer->next->ID, stu[i].ID);  stu\_pointer->next->chineseScore = stu[i].chineseScore;  stu\_pointer->next->mathScore = stu[i].mathScore;  stu\_pointer->next->englishScore = stu[i].englishScore;  stu\_pointer->next->totalScore = stu[i].totalScore;  stu\_pointer->next->aveScore = stu[i].aveScore;  stu\_pointer = stu\_pointer->next;  }  free(stu);//删去整个暂存的数组  return TRUE;  } |

4.

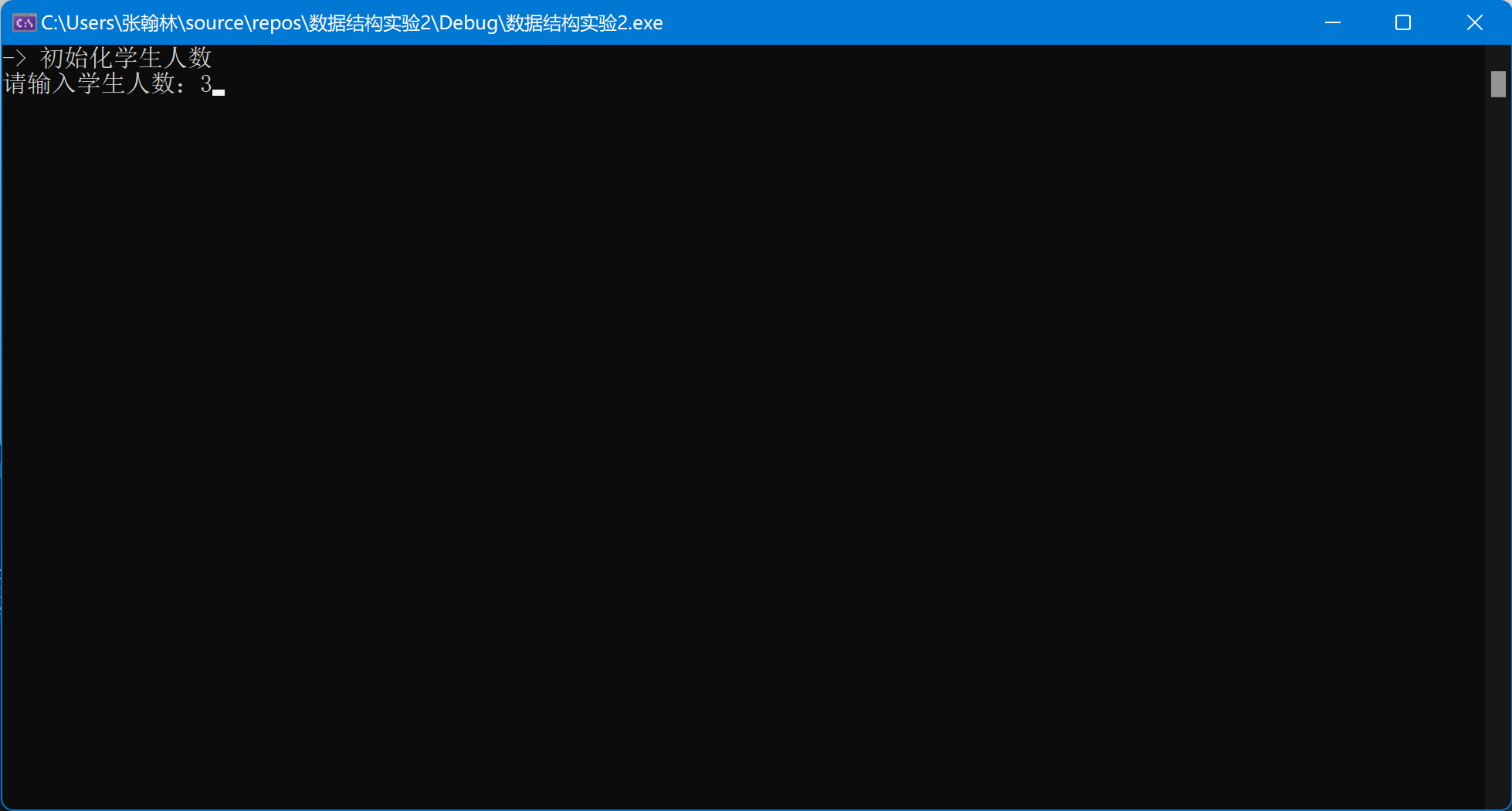
|  |
| --- |
| struct listNode {  int val;  listNode\* nextNode=NULL;  }; |

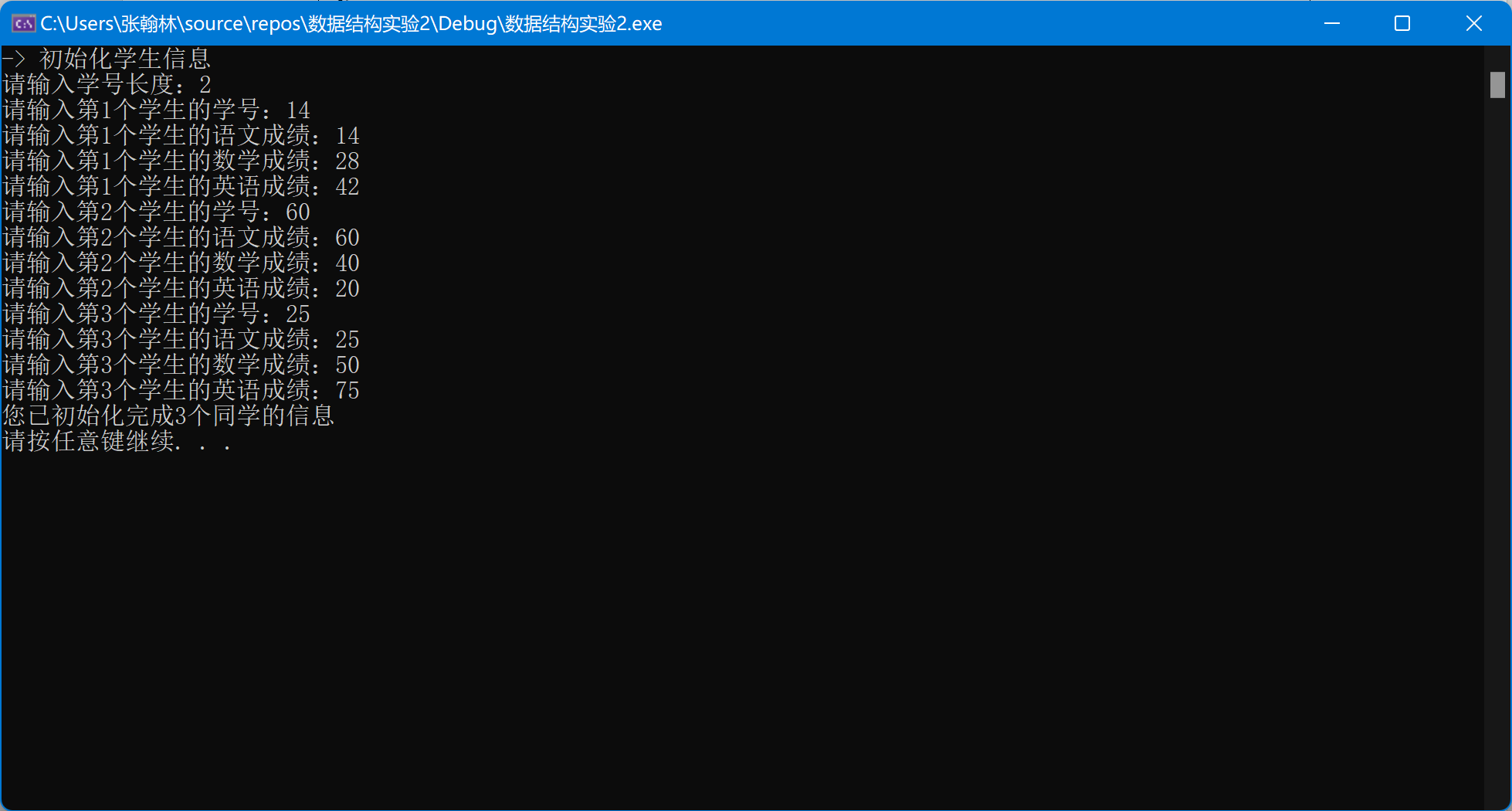
|  |
| --- |
| listNode\* reverseList(listNode\* headNode) |
| listNode\* reverseList(listNode\* headNode) {  listNode\* originNodePointer = headNode;//该指针遍历原链表  listNode\* lastNode = NULL;  while (originNodePointer->nextNode) {  listNode\* newNode = (listNode\*)malloc(sizeof(listNode));//创建一个节点  newNode->val = originNodePointer->val;//新建的节点拷贝原节点值  newNode->nextNode = lastNode;//其nextNode指针域指向上次创建的节点，即头插法  lastNode = newNode;//为下次循环做准备，把这次新建的newNode作为下次的behindNode使用  originNodePointer = originNodePointer->nextNode;//原链表接着遍历  }  listNode\* newNode = (listNode\*)malloc(sizeof(listNode));//创建一个节点  newNode->val = originNodePointer->val;//新建的节点拷贝原节点值  newNode->nextNode = lastNode;//其nextNode指针域指向上次创建的节点，即头插法  return newNode;  } |

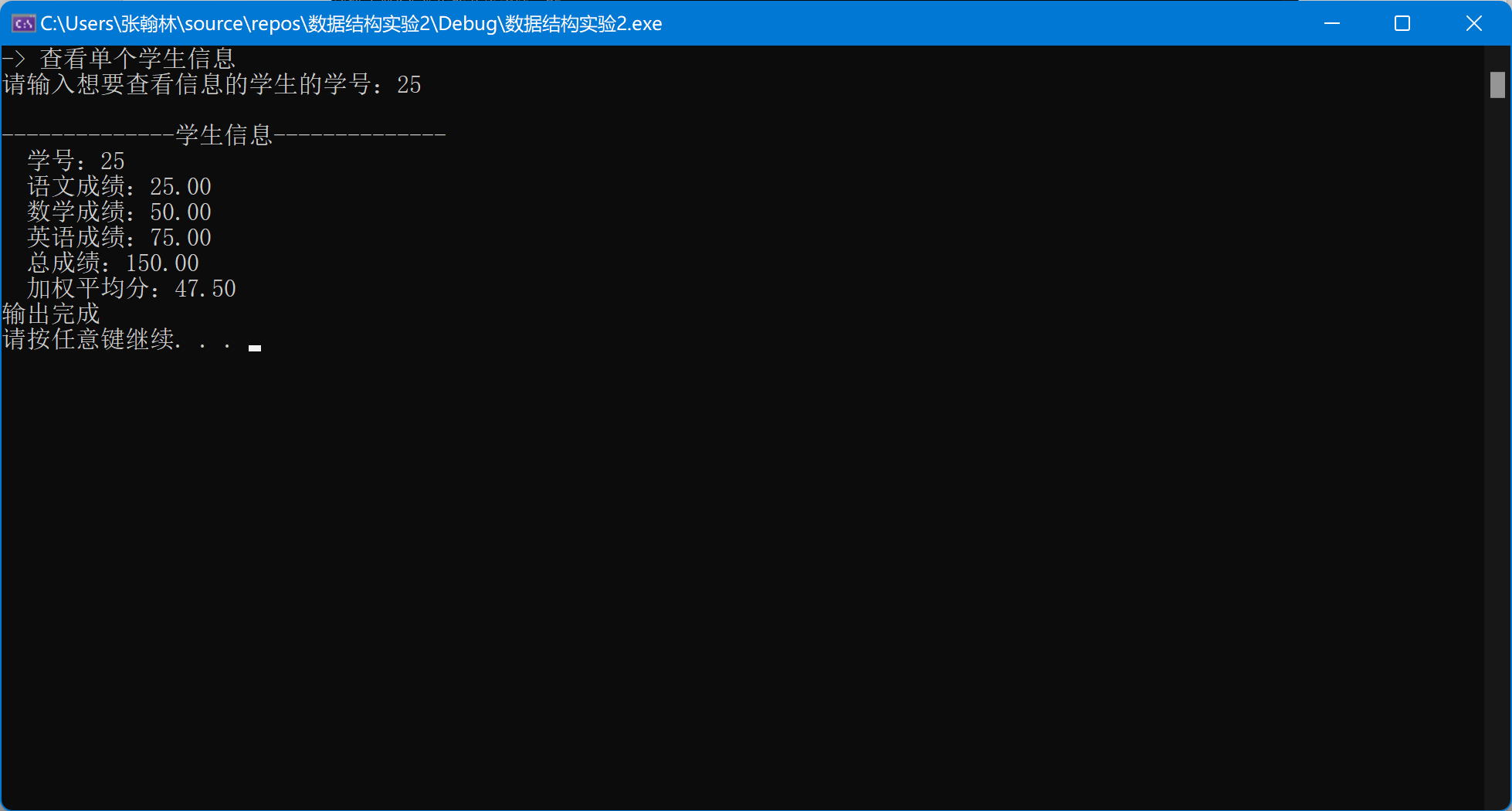
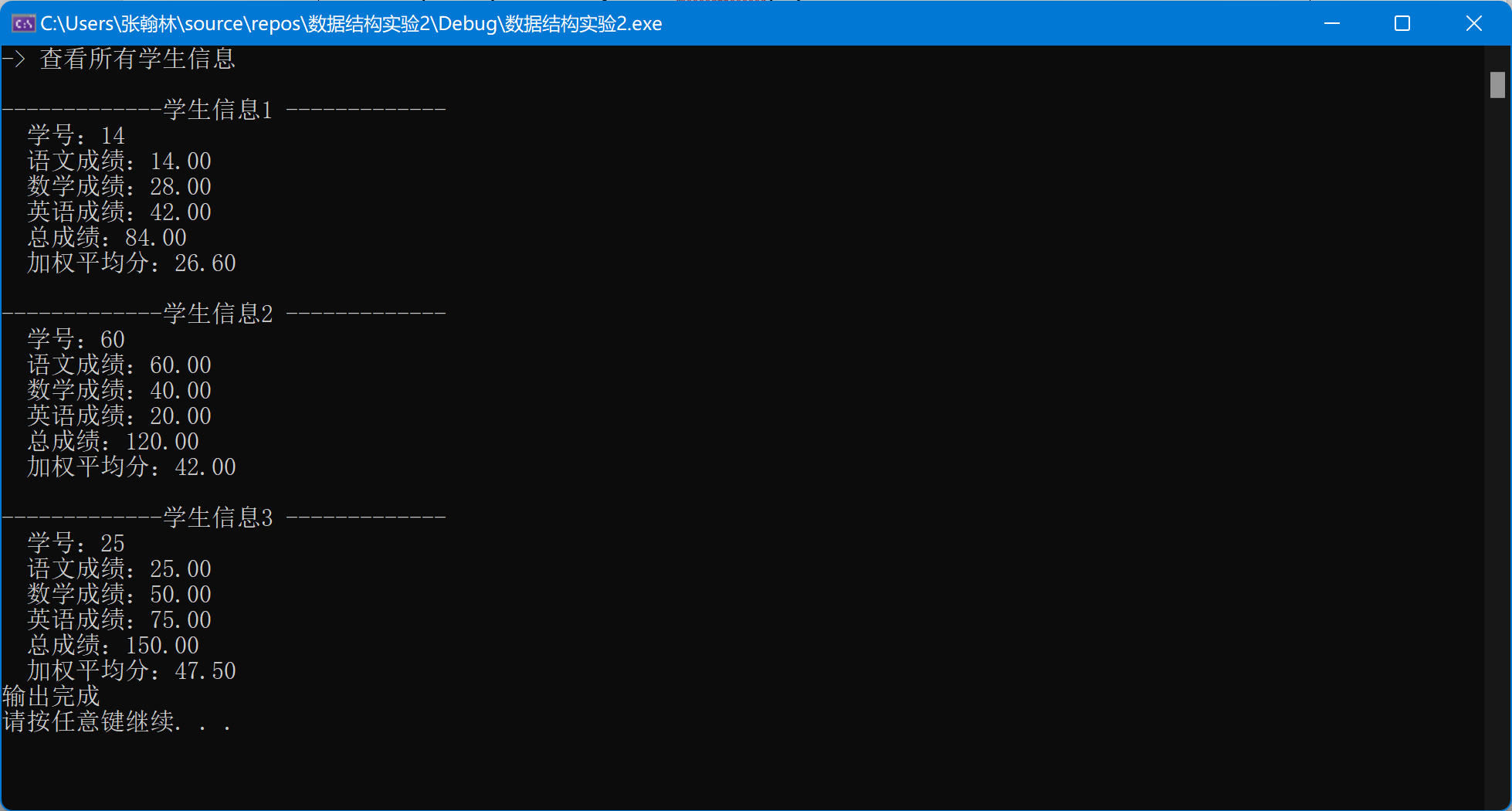
|  |
| --- |
| void question4() |
| void question4() {  listNode\* reverseList(listNode \* headNode);  listNode testList[6];  testList[0].val = 2;  testList[0].nextNode = &testList[1];  testList[1].val = 4;  testList[1].nextNode = &testList[2];  testList[2].val = 6;  testList[2].nextNode = &testList[3];  testList[3].val = 1;  testList[3].nextNode = &testList[4];  testList[4].val = 3;  testList[4].nextNode = &testList[5];  testList[5].val = 5;  testList[5].nextNode = NULL;  printf("测试用例为：2 -> 4 -> 6 -> 1 -> 3 -> 5\n");  listNode\* pointer = reverseList(testList);  printf("反转结果为：");  while (pointer)  {  printf("%d ", pointer->val);  pointer = pointer->nextNode;  }  free(pointer);  } |

**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：

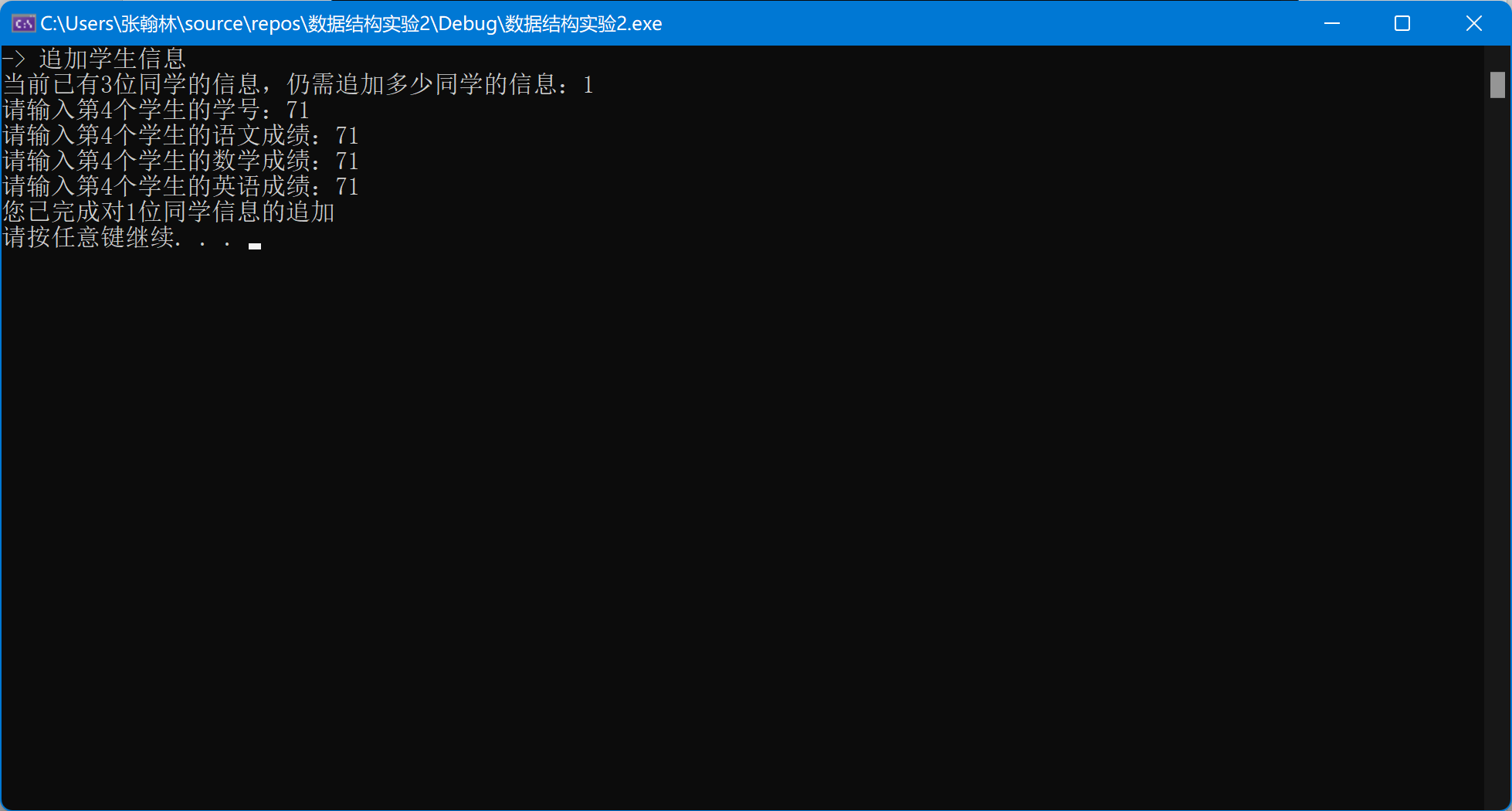


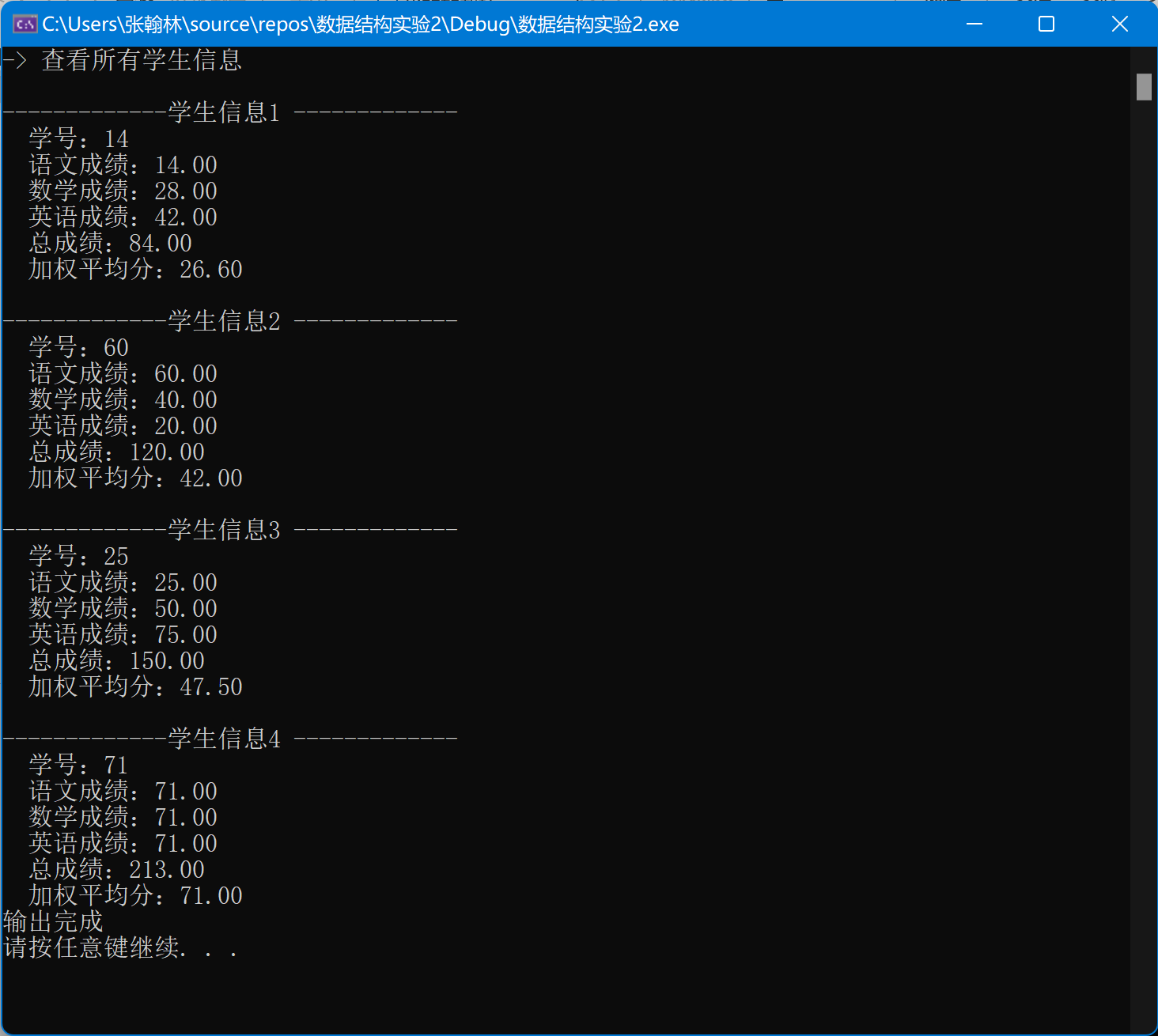




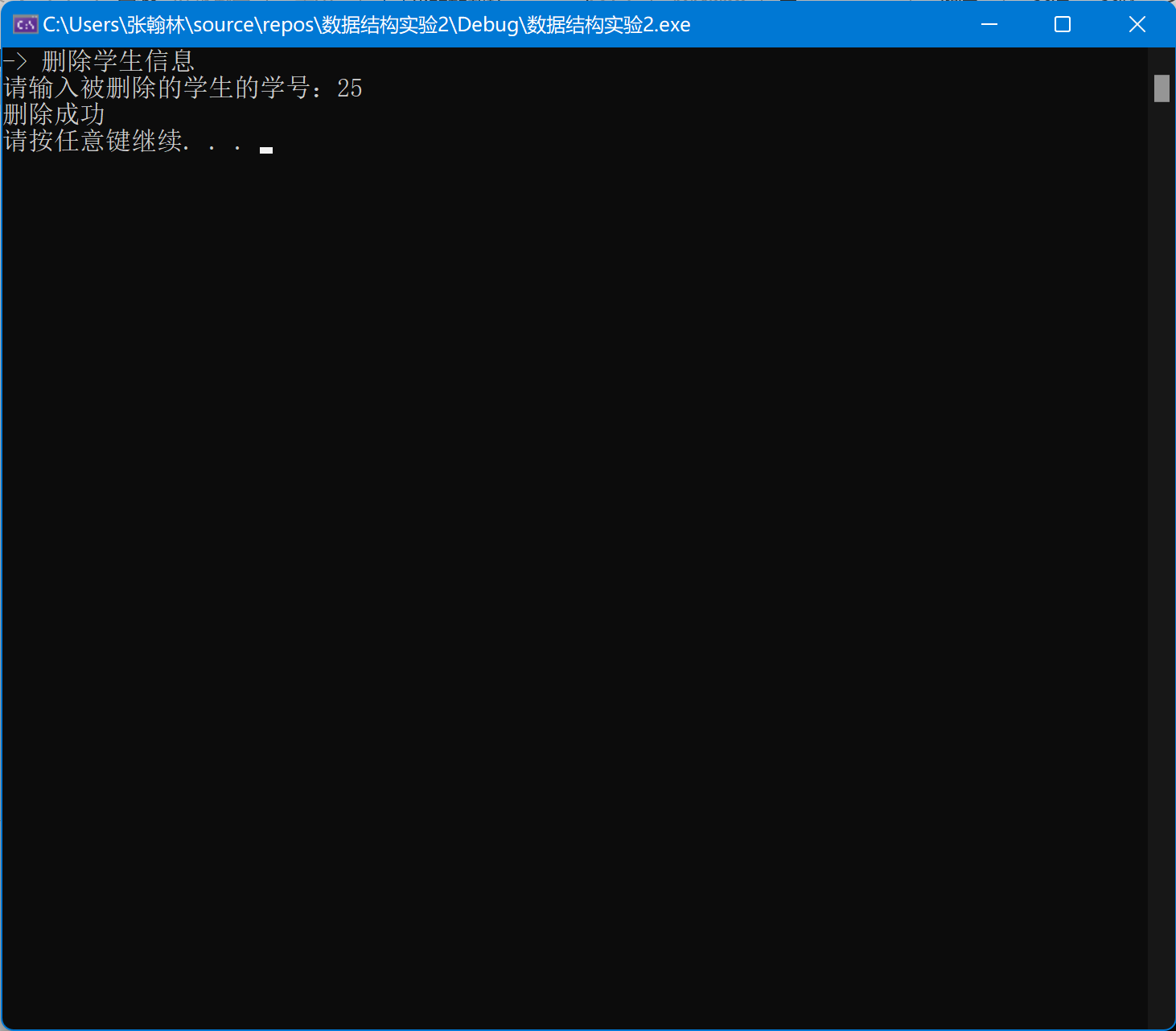
2.

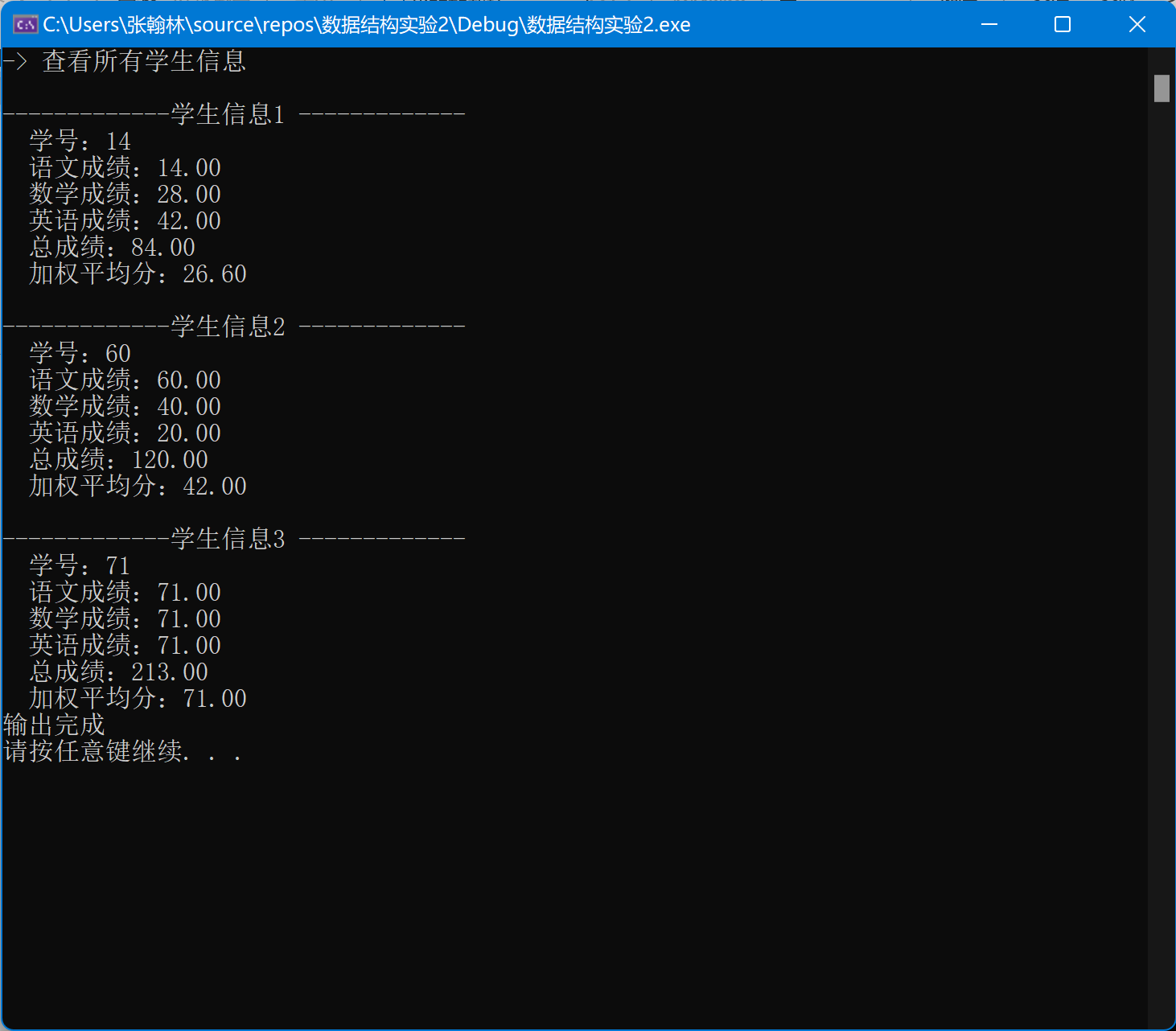
追加1个：





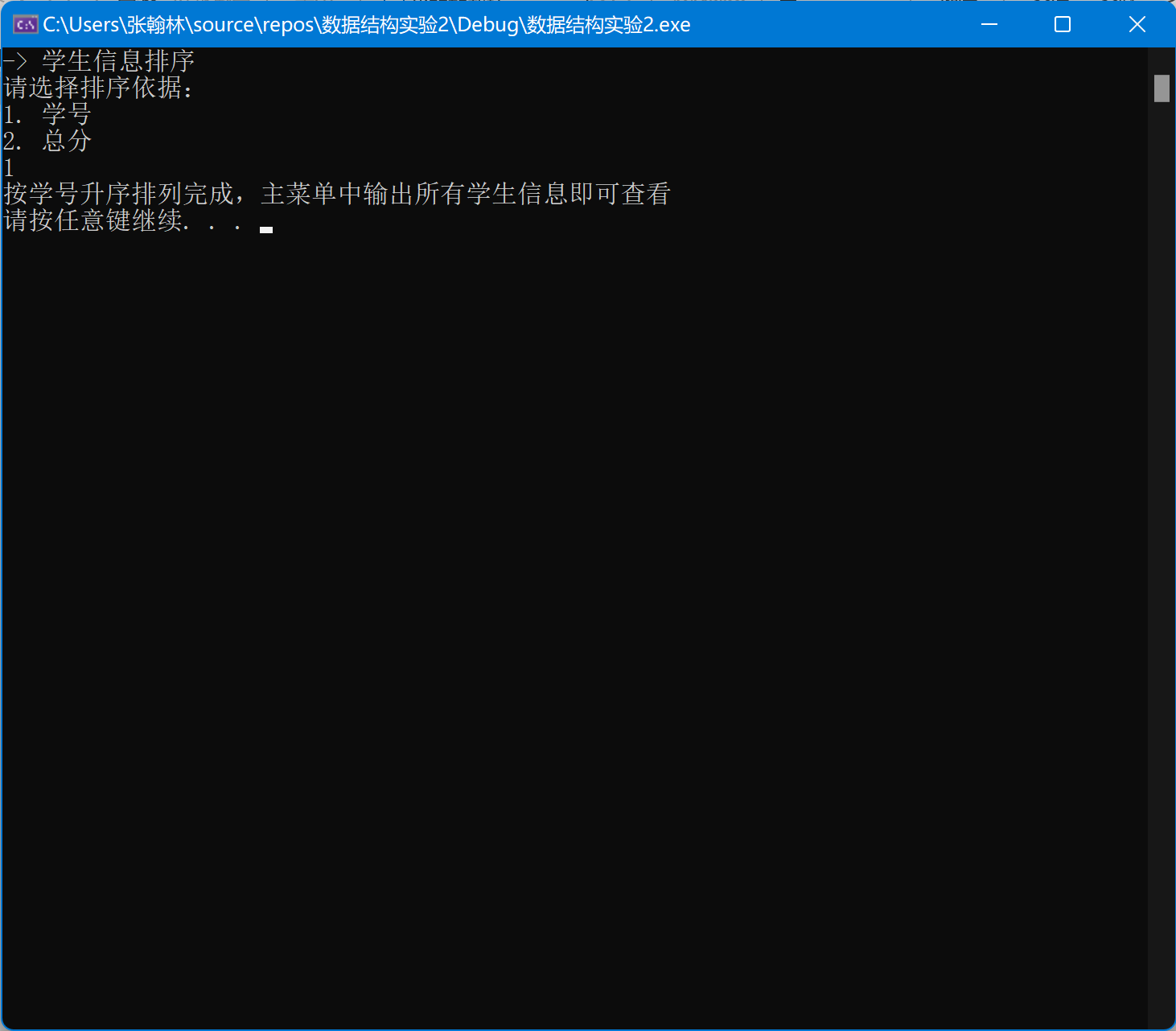
删除1个：

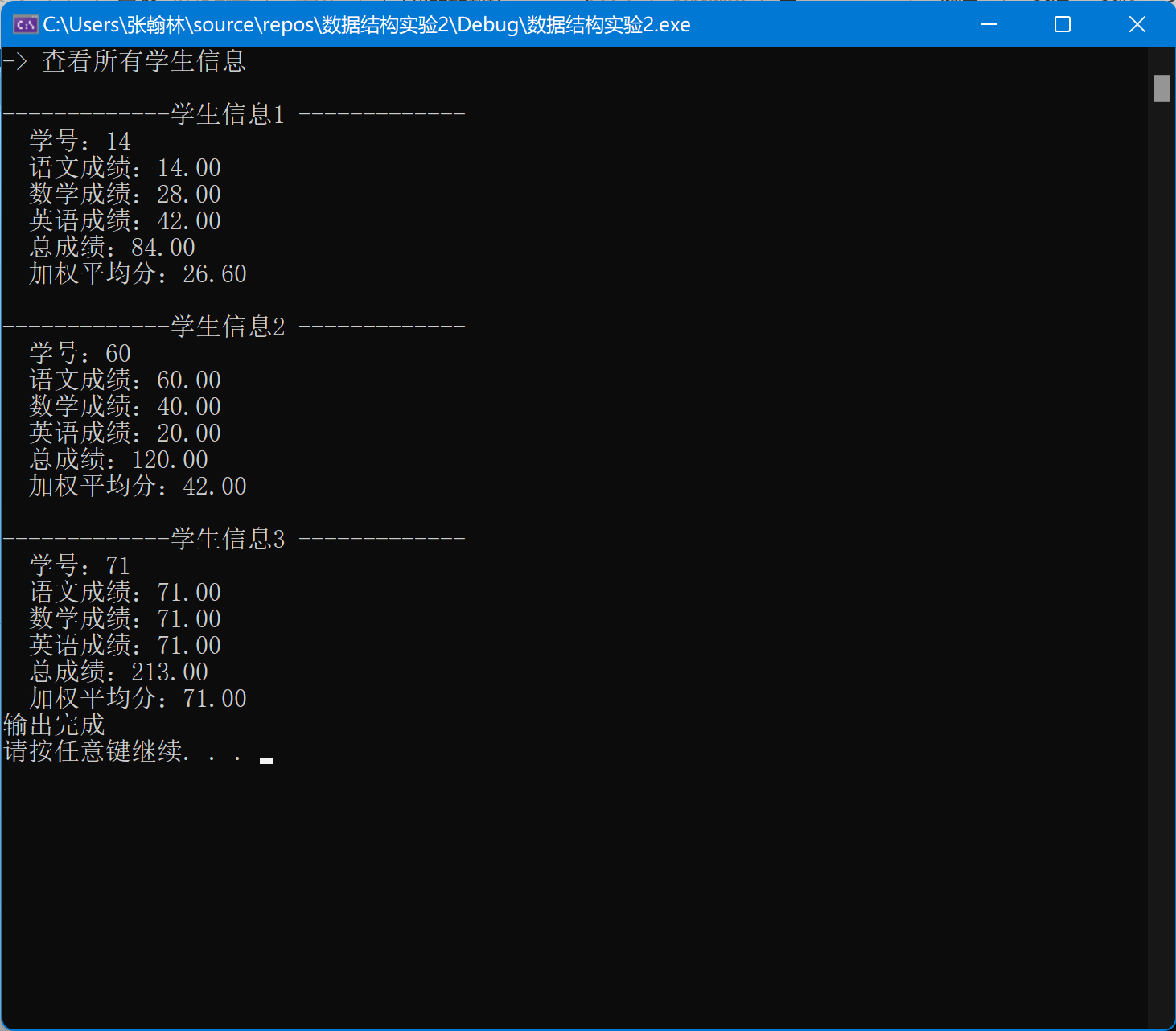




3.

按照学号排序：

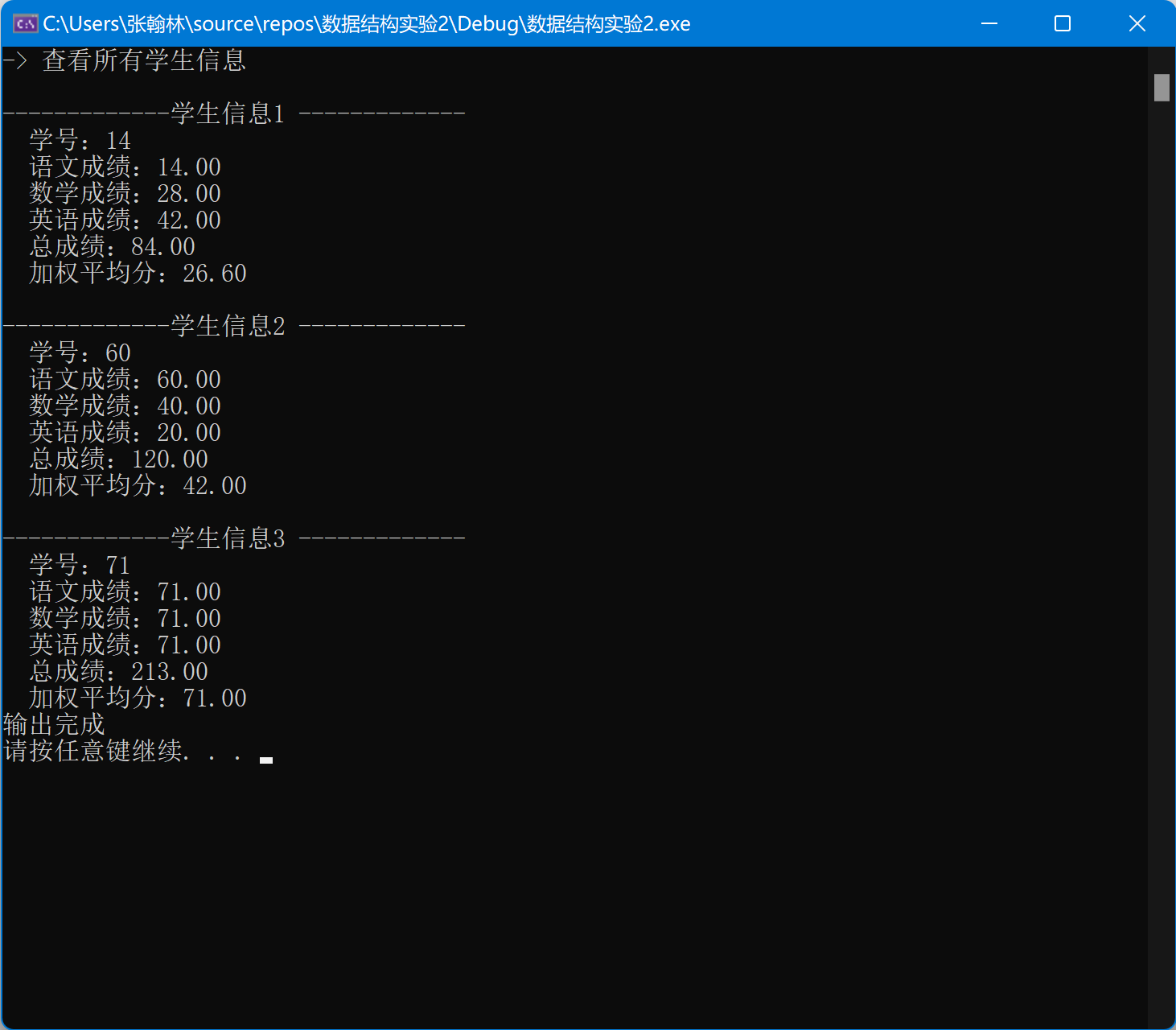




按照总分排序：

文本

描述已自动生成



4.测试用例：2 -> 4 -> 6 -> 1 -> 3 -> 5

手机屏幕的截图

中度可信度描述已自动生成