# 7 结构与联合实验

## 7.1 实验目的

1. 通过实验，熟悉和掌握结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。
2. 通过实验，掌握动态储存分配函数的用法，掌握自引用结构，单向链表的创建、遍历、结点的增删、查找等操作。
3. 了解字段结构和联合的用法。

## 7.2 实验内容

**7.2.1 表达式求值的程序验证题**

设有说明：

char u[]="UVWXYZ";

char v[]="xyz";

struct T{

int x;

char c;

char \*t;

}a[]={{11,ˊAˊ,u},{100, ˊBˊ,v}},\*p=a;

请先自己计算下面表达式的值，然后通过编程计算来加以验证。(各表达式相互无关)

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **表达式** | **计算值** | **验证值** |
| 1 | (++p)->x |  |  |
| 2 | p++,p->c |  |  |
| 3 | \*p++->t,\*p->t |  |  |
| 4 | \*(++p)->t |  |  |
| 5 | \*++p->t |  |  |
| 6 | ++\*p->t |  |  |

**解答：**

表达式求值的程序验证题的结果如表7-1所示：

表7-1 表达式求值的程序验证题的结果

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **表达式** | **计算值** | **验证值** |
| 1 | (++p)->x | 100 | 100 |
| 2 | p++,p->c | ‘B’ | ‘B’ |
| 3 | \*p++->t,\*p->t | ‘x’ | ‘x’ |
| 4 | \*(++p)->t | ‘x’ | ‘x’ |
| 5 | \*++p->t | ‘V’ | ‘V’ |
| 6 | ++\*p->t | ‘V’ | ‘V’ |

**7.2.2 源程序修改替换题**

给定一批整数，以0作为结束标志且不作为结点，将其建成一个先进先出的链表，先进先出链表的指头指针始终指向最先创建的结点（链头），先建结点指向后建结点，后建结点始终是尾结点。

1. 源程序中存在什么样的错误（先观察执行结果）？对程序进行修改、调试，使之能够正确完成指定任务。

源程序如下：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

void create\_list (struct s\_list \*headp,int \*p);

void main(void)

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0}; /\* 0为结束标记 \*/

create\_list(head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

printf("%d\t",p->data); /\* 输出数据域的值 \*/

p=p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

}

void create\_list(struct s\_list \*headp,int \*p)

{

struct s\_list \* loc\_head=NULL,\*tail;

if(p[0]==0) /\* 相当于\*p==0 \*/

;

else { /\* loc\_head指向动态分配的第一个结点 \*/

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++; /\* 对数据域赋值 \*/

tail=loc\_head; /\* tail指向第一个结点 \*/

while(\*p){ /\* tail所指结点的指针域指向动态创建的结点 \*/

tail->next=(struct s\_list \*)malloc(sizeof(struct s\_list));

tail=tail->next; /\* tail指向新创建的结点 \*/

tail->data=\*p++; /\* 向新创建的结点的数据域赋值 \*/

}

tail->next=NULL; /\* 对指针域赋NULL值 \*/

}

headp=loc\_head; /\* 使头指针headp指向新创建的链表 \*/

}

（2）修改替换create\_list函数，将其建成一个后进先出的链表，后进先出链表的头指针始终指向最后创建的结点（链头），后建结点指向先建结点，先建结点始终是尾结点。

**解答：**

1. 存在的错误：在create\_list函数中传入结构指针无法改变其该指针作为头指针所指的位置；
2. 程序改错：
3. 第8行函数声明void create\_list(struct s\_list \*headp, int \*p);，改为使用指向结构体的指针的指针，正确形式为：

void create\_list(struct s\_list \*\*headp, int \*p);

1. 第13行create\_list(head, s);，改为向create\_list函数传入head的地址，正确形式为：

create\_list(&head, s);

1. 第23行函数定义void create\_list(struct s\_list \*headp, int \*p)改为使用指向结构体指针的指针，正确形式为：

void create\_list(struct s\_list \*\*headp, int \*p)

1. 第43行headp=loc\_head;，无法使headp指向新创建的链表的头指针，改为headp所指的结构指针指向loc\_head，正确形式为：

\*headp=loc\_head;

1. 调试与修改后的源代码清单：
2. #include "stdio.h"
3. #include "stdlib.h"
4. struct s\_list
5. {
6. int data; /\* 数据域 \*/
7. struct s\_list \*next; /\* 指针域 \*/
8. };
9. void create\_list(struct s\_list \*\*headp, int \*p); // 改为指向结构指针的指针
10. int main(void)
11. {
12. struct s\_list \*head = NULL, \*p;
13. int s[] = {1, 2, 3, 4, 5, 6, 7, 8, 0}; /\* 0为结束标记 \*/
14. create\_list(&head, s); /\* 创建新链表 \*/ // 输入head的地址
15. p = head; /\*遍历指针p指向链头 \*/
16. while (p)
17. {
18. printf("%d\t", p->data); /\* 输出数据域的值 \*/
19. p = p->next; /\*遍历指针p指向下一结点 \*/
20. }
21. printf("\n");
22. return 0;
23. }
24. void create\_list(struct s\_list \*\*headp, int \*p) // 改为指向结构指针的指针
25. {
26. struct s\_list \*loc\_head = NULL, \*tail;
27. if (p[0] == 0) /\* 相当于\*p==0 \*/
28. ;
29. else
30. {
31. /\* loc\_head指向动态分配的第一个结点 \*/
32. loc\_head = (struct s\_list \*)malloc(sizeof(struct s\_list));
33. loc\_head->data = \*p++; /\* 对数据域赋值 \*/
34. tail = loc\_head; /\* tail指向第一个结点 \*/
35. while (\*p)
36. {
37. /\* tail所指结点的指针域指向动态创建的结点 \*/
38. tail->next = (struct s\_list \*)malloc(sizeof(struct s\_list));
39. tail = tail->next; /\* tail指向新创建的结点 \*/
40. tail->data = \*p++; /\* 向新创建的结点的数据域赋值 \*/
41. }
42. tail->next = NULL; /\* 对指针域赋NULL值 \*/
43. }
44. \*headp = loc\_head; /\* 使头指针headp指向新创建的链表 \*/ // 使headp所指的结构指针指向loc\_head
45. }
46. 修改替换题优化和替换后运行结果如图7-1所示。

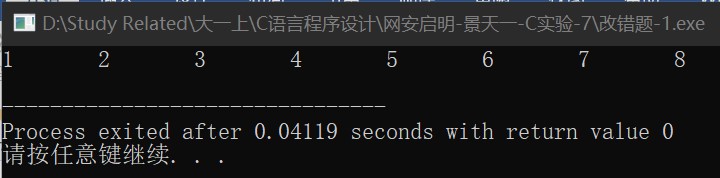


图7-1 源程序修改替换题优化替换后运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 修改替换create\_list函数后的源代码清单：
2. #include "stdio.h"
3. #include "stdlib.h"
4. struct s\_list
5. {
6. int data; /\* 数据域 \*/
7. struct s\_list \*next; /\* 指针域 \*/
8. };
9. void create\_list(struct s\_list \*\*headp, int \*p); // 改为指向结构指针的指针
10. int main(void)
11. {
12. struct s\_list \*head = NULL, \*p;
13. int s[] = {1, 2, 3, 4, 5, 6, 7, 8, 0}; /\* 0为结束标记 \*/
14. create\_list(&head, s); /\* 创建新链表 \*/ // 输入head的地址
15. p = head; /\*遍历指针p指向链头 \*/
16. while (p)
17. {
18. printf("%d\t", p->data); /\* 输出数据域的值 \*/
19. p = p->next; /\*遍历指针p指向下一结点 \*/
20. }
21. printf("\n");
22. return 0;
23. }
24. void create\_list(struct s\_list \*\*headp, int \*p) // 改为指向结构指针的指针
25. {
26. struct s\_list \*tail = NULL, \*loc\_head;
27. if (p[0] == 0)
28. ;
29. else
30. {
31. tail = (struct s\_list \*)malloc(sizeof(struct s\_list));
32. tail->data = \*p++;
33. tail->next = NULL;
34. while (\*p)
35. {
36. loc\_head = (struct s\_list \*)malloc(sizeof(struct s\_list));
37. loc\_head->data = \*p++;
38. loc\_head->next = tail;
39. tail = loc\_head;
40. }
41. }
42. \*headp = loc\_head;
43. }
44. 修改create\_list函数后的运行结果如图7-2所示：

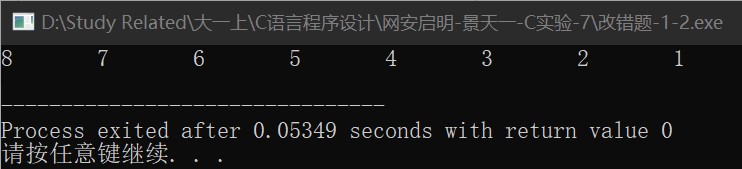


图7-2 create\_list函数修改后后运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

**7.2.3 程序设计**

1. 设计一个字段结构struct bits，它将一个8位无符号字节从最低位向最高位声明为8个字段，各字段依次为bit0, bit1, …, bit7，且bit0的优先级最高。同时设计8个函数，将8个函数的名字存入一个函数指针数组p\_fun。如果bit0为1，调用p\_fun[0]指向的函数。如果struct bits中有多位为1，则根据优先级从高到低依次调用函数指针数组p\_fun中相应元素指向的函数。8个函数中的第0个函数可以设计为：

void f0( int n)

{

printf(“the function %d is called!\n”,n);

}

**解答：**

1. 程序设计题1的设计思路如下所示：
2. 声明一个函数指针数组p\_fun，分别指向f0到f7 八个函数；
3. 声明一个长度为一个字节的字段类型bit，声明一个bit类型变量b；
4. 输入指令，存储在字符串order中；
5. 依次将order中的数据存储在b的每一位中；
6. 遍历b的每一位，若第i位不为0，则执行p\_fun[i](i)；
7. 源程序清单：
8. #include <stdio.h>
9. struct bits {
10. int bit0 : 1;
11. int bit1 : 1;
12. int bit2 : 1;
13. int bit3 : 1;
14. int bit4 : 1;
15. int bit5 : 1;
16. int bit6 : 1;
17. int bit7 : 1;
18. } b;
19. void f0(int n) {
20. printf("The function %d is called!\n", n);
21. };
22. void f1(int n) {
23. printf("The function %d is called!\n", n);
24. };
25. void f2(int n) {
26. printf("The function %d is called!\n", n);
27. };
28. void f3(int n) {
29. printf("The function %d is called!\n", n);
30. };
31. void f4(int n) {
32. printf("The function %d is called!\n", n);
33. };
34. void f5(int n) {
35. printf("The function %d is called!\n", n);
36. };
37. void f6(int n) {
38. printf("The function %d is called!\n", n);
39. };
40. void f7(int n) {
41. printf("The function %d is called!\n", n);
42. };
43. int main(void) {
44. void (\*p\_fun[8])(int) = {
45. f0, f1, f2, f3, f4, f5 ,f6, f7,
46. };
47. char order[9];
48. printf("Input your order:");
49. scanf("%s", order);
50. b.bit0 = order[0]-'0';
51. b.bit1 = order[1]-'0';
52. b.bit2 = order[2]-'0';
53. b.bit3 = order[3]-'0';
54. b.bit4 = order[4]-'0';
55. b.bit5 = order[5]-'0';
56. b.bit6 = order[6]-'0';
57. b.bit7 = order[7]-'0';
58. if (b.bit0) p\_fun[0](0);
59. if (b.bit1) p\_fun[1](1);
60. if (b.bit2) p\_fun[2](2);
61. if (b.bit3) p\_fun[3](3);
62. if (b.bit4) p\_fun[4](4);
63. if (b.bit5) p\_fun[5](5);
64. if (b.bit6) p\_fun[6](6);
65. if (b.bit7) p\_fun[7](7);
66. return 0;
67. }
68. 测试：
69. 测试数据：

程序设计题1的测试数据如表7-2所示：

表7-2 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 输入 | 预计输出 |
| 用例1 | 10010111 | The function 0 is called!  The function 3 is called!  The function 5 is called!  The function 6 is called!  The function 7 is called! |
| 用例2 | 00111001 | The function 2 is called!  The function 3 is called!  The function 4 is called!  The function 7 is called! |

1. 对应测试数据的运行结果截图：

测试用例一对应的运行结果如图7-3所示：

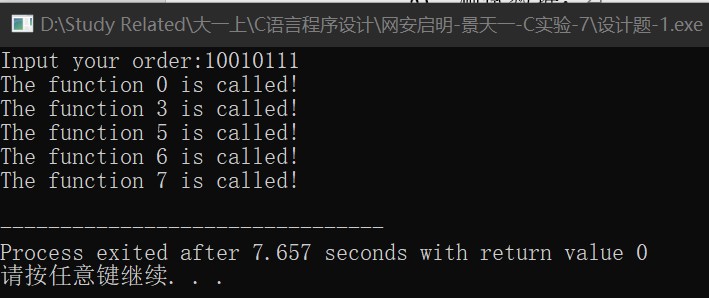


图7-3 程序设计题1的测试用例一的运行结果

测试用例二对应的运行结果如图7-4所示：

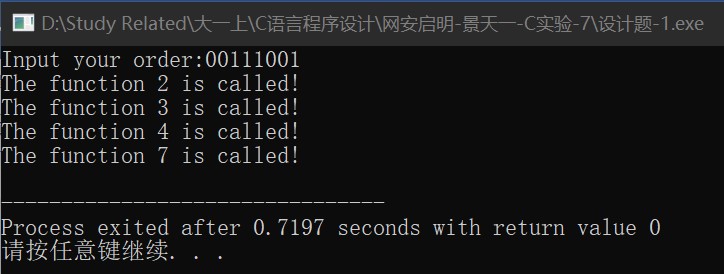


图7-4 程序设计题1的测试用例二的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

1. 用单向链表建立一张班级成绩单，包括每个学生的学号、姓名、英语、高等数学、普通物理、C语言程序设计四门课程的成绩。用函数编程实现下列功能：
2. 输入每个学生的各项信息。
3. 输出每个学生的各项信息。
4. 修改指定学生的指定数据项的内容。
5. 统计每个同学的平均成绩（保留2位小数）。
6. 输出各位同学的学号、姓名、四门课程的总成绩和平均成绩。

**解答：**

1. 解题思路：

程序设计题2的解题思路如图7-5所示：

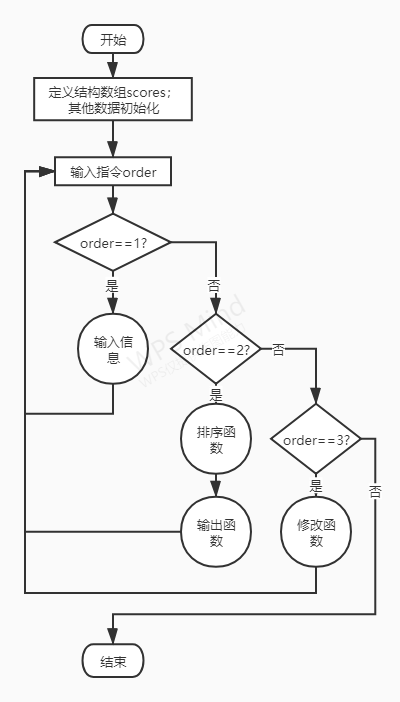


图7-5 程序设计题2的程序流程图

1. 源程序清单：
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <string.h>
5. #define SWAP(a,b,c) \
6. temp->c = a->c; \
7. a->c = b->c; \
8. b->c = temp->c;
9. // 外部变量声明
10. struct scores
11. {
12. struct scores \*prior;
13. char std\_num[20];
14. char name[10];
15. int math;
16. int english;
17. int physics;
18. int cpl;
19. float average;
20. struct scores \*next;
21. };
22. // 函数原型声明
23. int main(void); // 主函数
24. void input(struct scores \*\*); // 输入函数
25. void output(struct scores \*); // 输出函数
26. void change\_info(struct scores \*); // 改变内容
27. struct scores \*search(struct scores \*, char \*); // 查找学生
28. // 函数定义
29. int main(void)
30. { // 主函数：目录和指令输入
31. int order;
32. int ifadd;
33. struct scores \*head = NULL;
34. struct scores \*\*tail = &head;
35. do
36. {
37. system("cls");
38. printf("\t==学生成绩信息管理系统==\n");
39. printf("\t0.输入 0 退出应用\n");
40. printf("\t1.输入 1 录入新同学成绩\n");
41. printf("\t5.输入 2 输出学生成绩和平均分\n");
42. printf("\t4.输入 3 修改指定学生的某项信息\n");
43. printf("\tUser:Admin />");
44. scanf("%d", &order);
45. ifadd = 0;
46. switch (order)
47. {
48. case 1:
49. input(tail);
50. ifadd = 1;
51. break;
52. case 2:
53. output(head);
54. break;
55. case 3:
56. change\_info(head);
57. break;
58. case 0:
59. return 0;
60. default:
61. continue;
62. }
63. if (head == NULL)
64. head = \*tail;
65. if (ifadd)
66. tail = &(\*tail)->next;
67. } while (true);
68. while ((head->next) != NULL)
69. { // 清理内存
70. head = head->next;
71. free(head->prior);
72. }
73. return 0;
74. }
75. // 输入函数：输入新学生的信息
76. void input(struct scores \*\*head)
77. {
78. struct scores \*p = (struct scores \*)malloc(sizeof(struct scores));
79. printf("\n输入学号：");
80. scanf("%s", p->std\_num);
81. printf("\n输入学生姓名：");
82. scanf("%s", p->name);
83. printf("\n数学成绩：");
84. scanf("%d", &p->math);
85. printf("\n英语成绩：");
86. scanf("%d", &p->english);
87. printf("\n物理成绩：");
88. scanf("%d", &p->physics);
89. printf("\nC语言程序设计成绩：");
90. scanf("%d", &p->cpl);
91. int score\_full = p->english + p->math + p->physics + p->cpl;
92. p->average = score\_full / 4.0;
93. p->prior = \*head;
94. p->next = NULL;
95. \*head = p;
96. }
97. // 修改函数：修改指定学生的指定信息
98. void change\_info(struct scores \*head)
99. {
100. char target[10];
101. int order;
102. int data;
103. printf("请输入需要修改的学生名：");
104. scanf("%s", target);
105. struct scores \*temp = search(head, target);
106. if (temp)
107. {
108. printf("\t请输入需要修改的信息项：\n");
109. printf("\t输入 1 修改数学成绩\n");
110. printf("\t输入 2 修改物理成绩\n");
111. printf("\t输入 3 修改英语成绩\n");
112. printf("\t输入 4 修改C语言程序设计成绩\n\n");
113. printf(">");
114. scanf("%d", &order);
115. switch (order)
116. {
117. case 1:
118. printf("数学成绩:\n");
119. scanf("%d", &data);
120. temp->math = data;
121. break;
122. case 2:
123. printf("物理成绩:\n");
124. scanf("%d", &data);
125. temp->physics = data;
126. break;
127. case 3:
128. printf("英语成绩:\n");
129. scanf("%d", &data);
130. temp->english = data;
131. break;
132. case 4:
133. printf("C 语言程序设计成绩:\n\n");
134. scanf("%d", &data);
135. temp->cpl = data;
136. break;
137. }
138. int score\_full = head->english + head->math + head->physics + head->cpl;
139. temp->average = score\_full / 4.0;
140. }
141. else
142. printf("Student not found\n");
143. system("pause");
144. }
145. // 搜索函数：搜索是否输入了指定学生的信息
146. struct scores \*search(struct scores \*head, char \*target)
147. {
148. while (head && strcmp(head->name, target))
149. head = head->next;
150. return head;
151. }
152. // 输出函数：输出平均分 总分 单科成绩
153. void output(struct scores \*head)
154. {
155. int score\_full = 0;
156. while (head)
157. {
158. score\_full = head->english + head->math + head->physics + head->cpl;
159. printf("学号: %s\n", head->std\_num);
160. printf("姓名: %s\n", head->name);
161. printf("数学: %d\n", head->math);
162. printf("英语: %d\n", head->english);
163. printf("物理: %d\n", head->physics);
164. printf("C语言程序设计: %d\n", head->cpl);
165. printf("总分: %d\n", score\_full);
166. printf("%s的平均分为%.2f\n\n", head->name, head->average);
167. head = head->next;
168. }
169. system("pause");
170. }
171. 测试
172. 测试数据：

设计题-2的测试数据如表7-3所示：

表7-3 编程题2的测试数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 第一次指令 | 第二次指令 | 第三次指令 | 第四次指令 | 第五次指令 |
| 用例1 | 1  01 aaa 89 89 90 90  02 sss 78 98 90 80  03 ddd 86 96 85 97 | 2 | 3  sss  1  100 | 1 | 0 |
| 预计输出 | 无 | 输出三名学生的各科成绩，总成绩和平均成绩 | 无 | 输出三名学生的各科成绩，总成绩和平均成绩；学生sss的数学成绩改为100 | 退出 |

1. 对应测试测试用例1的运行结果如图7-6到图7-12所示所示。

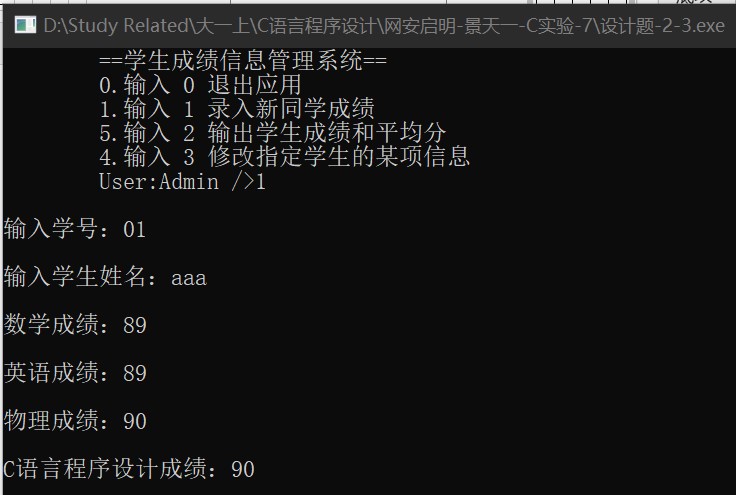


图7-6 程序设计题2的测试用例一的运行结果

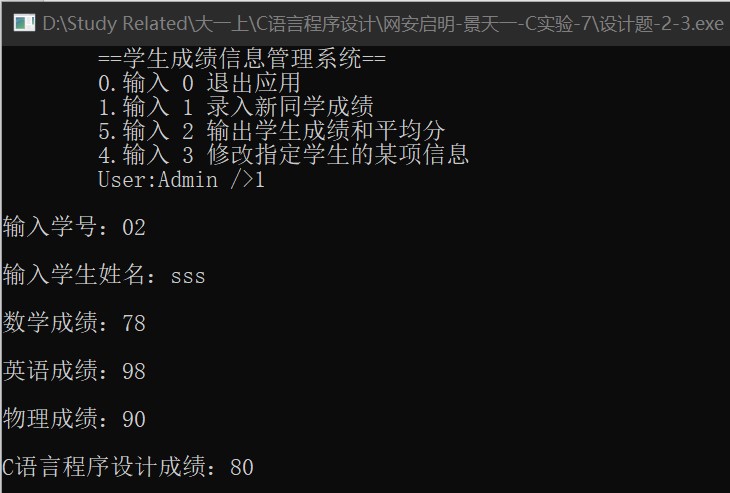


图7-7 程序设计题2的测试用例一的运行结果

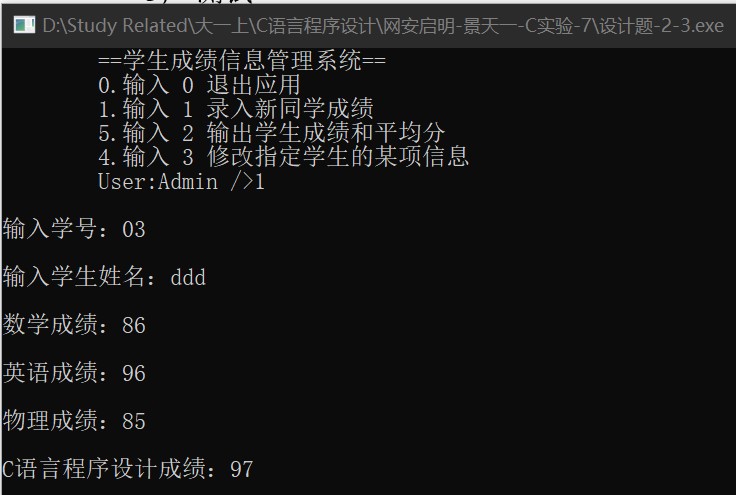


图7-8 程序设计题2的测试用例一的运行结果

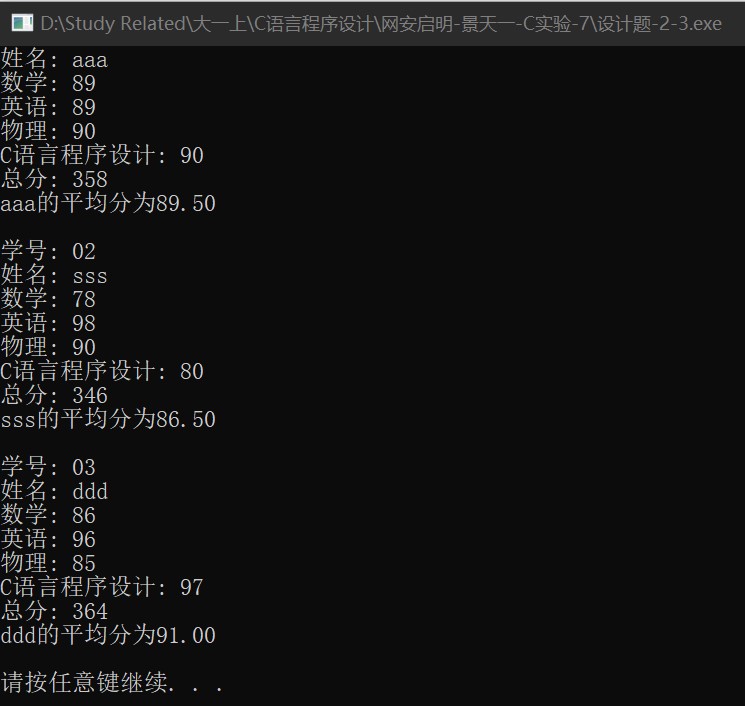


图7-9 程序设计题2的测试用例一的运行结果

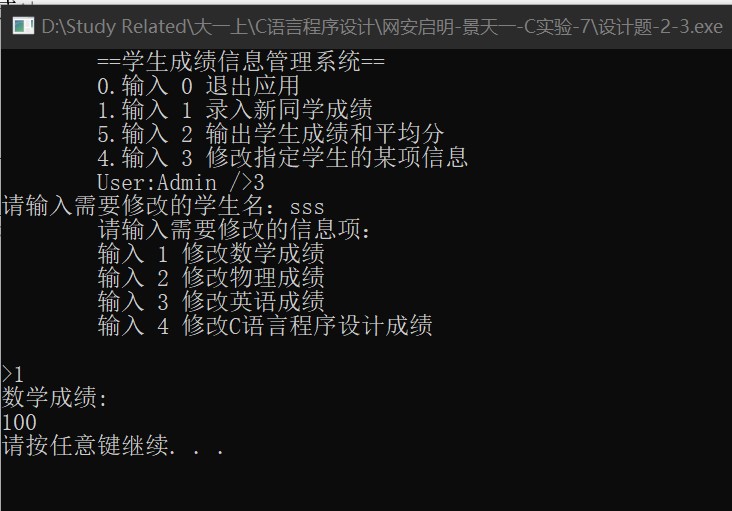


图7-10 程序设计题2的测试用例一的运行结果

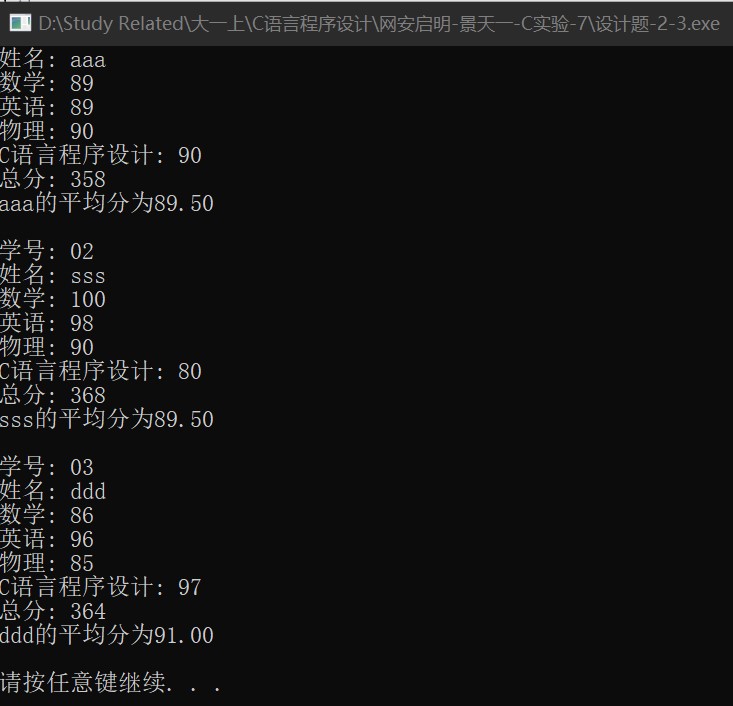


图7-11 程序设计题2的测试用例一的运行结果

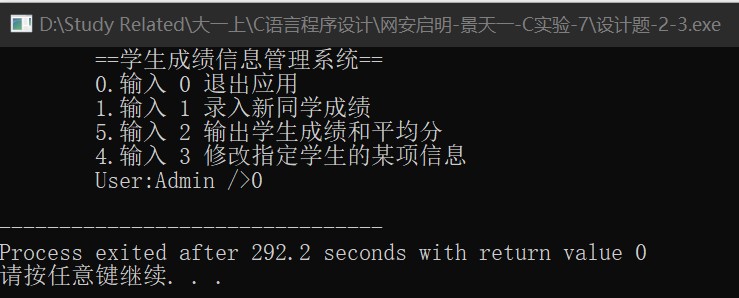


图7-12 程序设计题2的测试用例一的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

## 1.3 实验小结

本次实验主要学习了结构体、结构体指针的使用，字段结构的声明和使用，联合的声明和使用，链表的声明和使用。结构体是C语言中一种重要的构造类型，借助结构体，可以实现链表等多种复杂的数据类型，突破数组仅能存储相同类型数据的限制。结构体和结构指针的应用，也可以达到接近面对对象语言中“对象”的效果。

在实际运用中，我发现字段结构类型变量的使用具有一定的局限性，如：当比特数小于一字节时，无法通过键访，指针等方式进行引用，进而进行遍历操作。另外，链表在进行遍历和删除操作时，虽然效率比压缩数组高很多，但是将对记数造成很大影响，（例如排序时）必须对引入的计数变量进行更复杂的操作。