

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程2003班**

**学 号： U202010851**

**姓 名： 侯皓斐**

**指导教师： 唐赫**

**报告日期： 2020.12.19**

**软件工程**

**目 录**

[**7 结构与联合实验 1**](#_Toc404837920)

[7.1 实验目的 1](#_Toc404837921)

[7.2 实验内容 1](#_Toc404837922)

7[.3 实验小结 20](#_Toc404837923)

# 7 指针实验

## 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)

**解答：**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

1) 先计算++p，且表达式的值为a[1]，取其x值，便为100。

2) 逗号表达式，先计算前面的表达式，p指向a[1]，但逗号表达式的值为最后一个表达式的值，表达式2的值因此为B。

3) 逗号表达式，先计算前面的表达式，p指向a[1]，由于成员选择运算符（指针）->的优先级最高，然后取出字符数组v的首元素。表达式3的值因此为x。

4) 先计算++p，且表达式的值为a[1]，由于成员选择运算符（指针）->的优先级最高，取其t指针指向的首元素，表达式4的值因此为x。

5) 由于成员选择运算符（指针）->的优先级最高，取其t指针（指向数组u）。自增运算符++和取值运算符\*优先级相同，但从右向左结合，先是指针指向u[1]。表达式5的值因此为V。

6) 由于成员选择运算符（指针）->的优先级最高，取其t指针（指向数组u）。自增运算符++和取值运算符\*优先级相同，但从右向左结合，先是取值u[0]，然后自加。表达式6的值因此为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。

#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 \*/

}

（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。

**解答：**

1. 错误修改：

通过运行程序发现，程序并未有输出的值。通过逐步调试发现，经过函数create\_list后，main函数内指针并未发生改变，指向位置仍然是空。具体原因是因为，void create\_list(struct s\_list \*headp,int \*p)函数是值传递，整个函数内的headp指针为局部变量，并未影响到main函数内的指针。解决方案是再取指针指向这个指向链表的指针，通过传递指针达到修改值的目的。

1) 第8行为保证编译正确，正确形式为：

int main()//void main(void)

2) 第7行为了能成功修改main函数内的需要再取指针，正确形式为：

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

3) 第20行同2，正确形式为：

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

4) 第36行为使头指针指向新创建的链表，正确形式为：

\*headp=loc\_head;

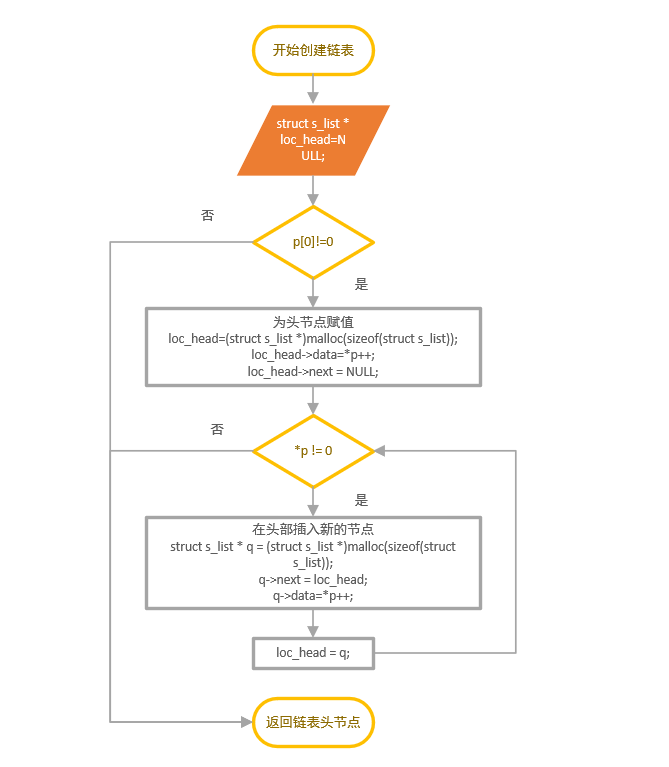
（2）错误修改后运行结果：



（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：**

整体代码框架基本保持不变，而修改为后进先出的链表仅需要修改创建链表的过程，其算法流程图如下：



代码实现如下：

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

{

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

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 \*/

loc\_head->next = NULL;

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

struct s\_list \* q = (struct s\_list \*)malloc(sizeof(struct s\_list));

q->next = loc\_head; /\* tail指向新创建的结点 \*/

q->data=\*p++; /\* 向新创建的结点的数据域赋值 \*/

loc\_head = q;

}

}

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

}

**7.2.3 程序设计**

（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。用菜单实现下列功能：

① 输入每个学生的各项信息。

② 输出每个学生的各项信息。

③ 修改指定学生的指定数据项的内容。

④ 统计每个同学的平均成绩（保留2位小数）。

⑤ 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：

本题分析和建模后应建立如下的链表节点域：

typedef struct Node {

int id; char Name[1000]; int a, b, c, d, sum;double ave;

struct Node \*next;

}Node;本题分析与建模后应完成如下函数：

1. Input函数：输入学生数量n，完成每个学生的学号、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的输入和保存至链表。
2. Output函数：遍历这个单向链表输出所有学生的各项信息。
3. Change函数：根据输入的学生的学号，和一个数字，0对应修改其名字，1到4对应修改其四门课的成绩。遍历单向链表，找到可以修改的地方然后修改。
4. CalcAve函数：遍历单向链表，统计每个同学的平均成绩保存并输出。
5. OutAve函数：遍历这个单向链表输出所有学生的总成绩和平均成绩。

为了保证能够修改头指针指向的链表，我们应该设立头节点，即头指针指向的第一个节点不记录任何信息，同时也便于插入节点程序的编写。

1. 源程序清单

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Node {

int id; char Name[1000]; int a, b, c, d, sum;double ave;

struct Node \*next;

}Node;

void input(Node \*S) {

int N;

scanf("%d", &N);

struct Node \*p = S;

for(int i = 1; i <= N; i++) {

struct Node\* newNode = (struct Node\*) malloc(sizeof(struct Node));

scanf("%d%s%d%d%d%d", &newNode->id, newNode->Name, &newNode->a, &newNode->b, &newNode->c, &newNode->d);

p->next = newNode;

newNode->next = NULL;

p = newNode;

}

printf("完成了%d位同学的成绩输入。\n", N);

}

void output(Node \*S) {

struct Node \*p = S->next;

while(p != NULL) {

printf("%d %s %d %d %d %d\n", p->id, p->Name, p->a, p->b, p->c, p->d);

p = p->next;

}

}

void Change(Node \*S) {

int id, waitChange;

scanf("%d%d",&id, &waitChange);

struct Node \*p = S->next;

while(p != NULL) {

if(p->id == id) {

if(waitChange == 0) {

char newName[100];

scanf("%s",newName);

memcpy(p->Name, newName, sizeof(newName));

}

else if(waitChange == 1) {

int A;scanf("%d", &A);p->a = A;

}

else if(waitChange == 2) {

int B;scanf("%d", &B);p->b = B;

}

else if(waitChange == 3) {

int C;scanf("%d", &C);p->c = C;

}

else if(waitChange == 4) {

int D;scanf("%d", &D);p->d = D;

}

p->sum = p->a + p->b + p->c + p->d;

p->ave = p->sum/4.0;

printf("%d %s %d %d %d %d\n", p->id, p->Name, p->a, p->b, p->c, p->d);

break;

}

p = p->next;

}

}

void CalcAve(Node \*S) {

struct Node \*p = S->next;

while(p != NULL) {

int sum = 0;

sum = p->a + p->b + p->c + p->d;

p->sum = sum;

p->ave = sum/4.0;

printf("%d %s %0.2lf\n", p->id, p->Name, sum/4.0);

p = p->next;

}

}

void OutAll (Node \*S) {

struct Node \*p = S->next;

while(p != NULL) {

printf("%d %s %d %0.2lf\n", p->id, p->Name, p->sum, p->ave);

p = p->next;

}

}

int main () {

struct Node\* S = (struct Node\*) malloc(sizeof(struct Node));

S->id = S->a = S->b = S->c = S->d = 0; (S->Name)[0] = 0;

S->next = NULL;

int opt;

while(scanf("%d", &opt) != EOF) {

if(opt == 0) {

//printf("（退出）\n");

break;

}

else if(opt == 1)

input(S);

else if(opt == 2)

output(S);

else if(opt == 3)

Change(S);

else if(opt == 4)

CalcAve(S);

else if(opt == 5)

OutAll(S);

}

}

3）测试

（a） 测试数据：

根据题目要求以及各种可能出现的错误情况构造数据如表6-1所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 输 入 | 理 论 结 果 | 运 行 结 果 |
| 操作与信息 |
| 例1 | 1  5  2021001 Jack 90 92 87 95  2021002 Mike 85 70 75 90  2021003 Joe 77 86 90 75  2021004 Andy 95 97 92 95  2021005 Rose 90 87 88 89  0 | 完成了5位同学的成绩输入。 | 完成了5位同学的成绩输入。 |
| 例2 | 1  5  2021001 Jack 90 92 87 95  2021002 Mike 85 70 75 90  2021003 Joe 77 86 90 75  2021004 Andy 95 97 92 95  2021005 Rose 90 87 88 89  3  2021003 0 Joan  3  2021003 2 96  4  5  0 | 完成了5位同学的成绩输入。  2021003 Joan 77 86 90 75  2021003 Joan 77 96 90 75  2021001 Jack 91.00  2021002 Mike 80.00  2021003 Joan 84.50  2021004 Andy 94.75  2021005 Rose 88.50  2021001 Jack 364 91.00  2021002 Mike 320 80.00  2021003 Joan 338 84.50  2021004 Andy 379 94.75  2021005 Rose 354 88.50 | 完成了5位同学的成绩输入。  2021003 Joan 77 86 90 75  2021003 Joan 77 96 90 75  2021001 Jack 91.00  2021002 Mike 80.00  2021003 Joan 84.50  2021004 Andy 94.75  2021005 Rose 88.50  2021001 Jack 364 91.00  2021002 Mike 320 80.00  2021003 Joan 338 84.50  2021004 Andy 379 94.75  2021005 Rose 354 88.50 |

（b） 对应测试测试用例1的运行结果如图7-1所示。

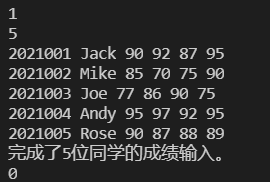


图7-2 编程题1的测试用例一的运行结果

对应测试测试用例2的运行结果如图7-3所示。

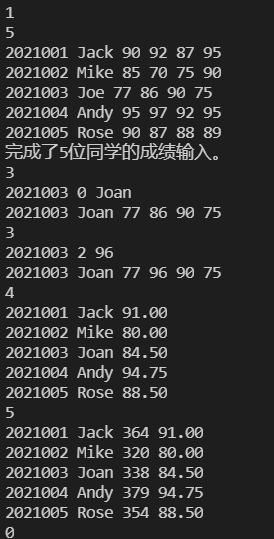


图7-3 编程题1的测试用例二的运行结果

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。

（2）对程序设计题第（1）题的程序，

⑥增加按照平均成绩进行升序排序的函数，写出用交换结点数据域的方法升序排序的函数，排序可用选择法或冒泡法。

**解答：**

1. 解题思路：

冒泡排序的中心思想：通过对无序序列的相邻元素两两比较交换，每次把最大（小）的数据放在最后，这个过程就像是冒泡一样，所以叫他冒泡排序。交换结点数据域的方法升序排序的函数，即与冒泡排序写法没有大的区别，遍历方式发生了改变，判断是否排序完的判断也发生了改变。而由于链表没有记录存储的学生信息数量如下：

2）部分函数清单：

void swap1 (int \*a,int \*b) {

int temp = \*a; \*a = \*b; \*b = temp;

}

void swap2 (double \*a,double \*b) {

double temp = \*a; \*a = \*b; \*b = temp;

}

void swap3 (char \*a,char \*b) {

char c[1000];

memcpy(c, a, sizeof(a));

memcpy(a, b, sizeof(b));

memcpy(b, c, sizeof(c));

}

void Sort1 (Node \*S) {

for(Node \*q = S->next; q != NULL; q = q->next) {

for(Node \*p = S->next; p->next != NULL; p = p->next) {

if((p->sum) > (p->next->sum)) {

swap1(&p->id, &p->next->id);

swap3(p->Name, p->next->Name);

swap1(&p->a, &p->next->a);

swap1(&p->b, &p->next->b);

swap1(&p->c, &p->next->c);

swap1(&p->d, &p->next->d);

swap1(&p->sum, &p->next->sum);

swap2(&p->ave, &p->next->ave);

}

}

}

Node \*p = S->next;

while(p != NULL) {

printf("%d %s %0.2lf\n", p->id, p->Name, p->ave);

p = p->next;

}

}

3）测试

（a） 测试数据：

1

5

2021001 Jack 90 92 87 95

2021002 Mike 85 70 75 90

2021003 Joe 77 86 90 75

2021004 Andy 95 97 92 95

2021005 Rose 90 87 88 89

3

2021003 0 Joan

3

2021003 2 96

4

5

6

0

（b） 对应测试数据的运行结果截图

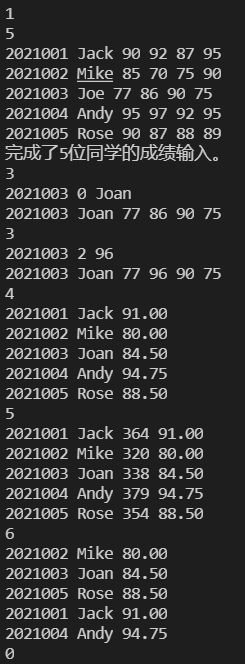


图7-4编程题2的测试运行结果

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。

（3）对程序设计题第（2）题，进一步写出用交换结点指针域的方法升序排序的函数。

**解答：**

1. 解题思路：

分析排序算法后，可以建立流程图如下：

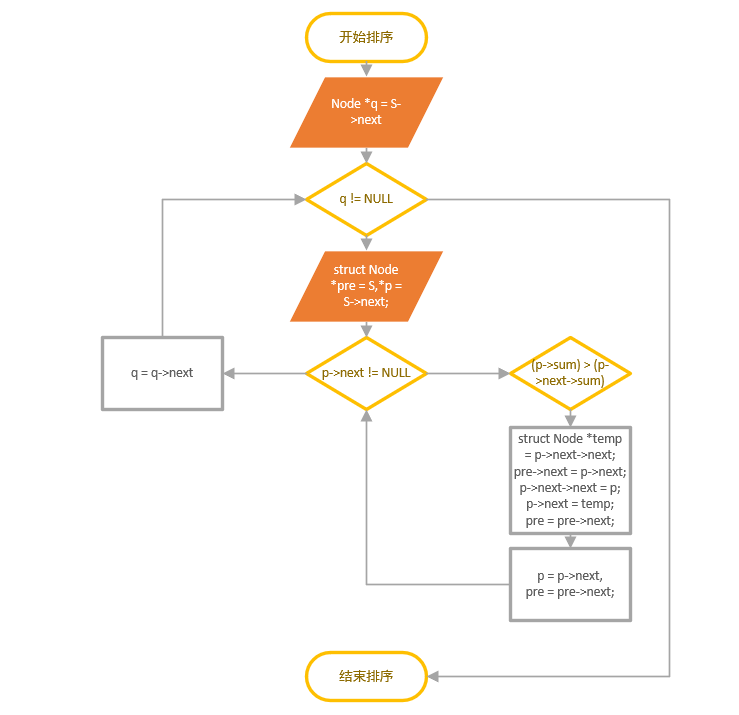


图7-5编程题3排序部分的算法流程图

2）程序清单

void Sort2(Node \*S) {

for(Node \*q = S->next; q != NULL; q = q->next) {

struct Node \*pre = S,\*p = S->next;

while(p->next != NULL) {

if((p->sum) > (p->next->sum)) {

struct Node \*temp = p->next->next;

pre->next = p->next;

p->next->next = p;

p->next = temp;

pre = pre->next;

continue;

}

p = p->next, pre = pre->next;

}

}

struct Node \*p = S->next;

while(p != NULL) {

printf("%d %s %0.2lf\n", p->id, p->Name, p->ave);

p = p->next;

}

}

3）测试

（a） 测试数据：

1

5

2021001 Jack 90 92 87 95

2021002 Mike 85 70 75 90

2021003 Joe 77 86 90 75

2021004 Andy 95 97 92 95

2021005 Rose 90 87 88 89

3

2021003 0 Joan

3

2021003 2 96

4

5

6

0

（b） 对应测试测试用例1的运行结果如图7-6所示。

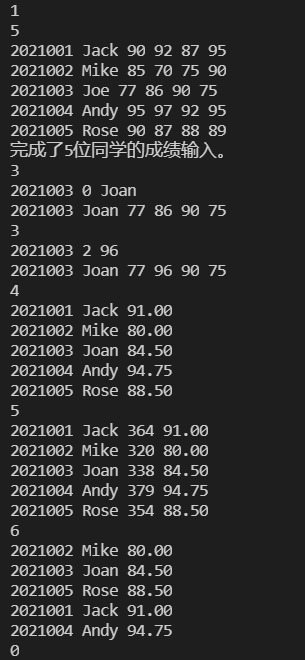


图7-6 编程题3的测试用例的运行结果

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。

## 7.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

通过本次实验，我较深入地理解了结构体和联合的实际写法和在具体问题中的应用。编写链表程序加深了对数据结构相关知识的理解。在表达式求值的程序验证题中，我将原先指针等知识整合，分析了他们的运算优先级，为编写正确的程序做好了准备。我也学会了链表交换节点的两种方式，节点值交换，节点来回倒。