

# 华中科技大学

## 课程实验报告

课程名称： C 语言程序设计实验

专业班级： CS2208

学 号： U202215595

姓 名： 郭凯锐

指导教师： 毛伏兵

报告日期： 2022 年 11 月 30 日

计算机科学与技术学院

# 目 录

目 录.....	II
<b>1 实验 6 指针程序设计实验 .....</b>	<b>1</b>
1.1 程序改错与跟踪调试 .....	1
1.2 程序完善与修改替换 .....	3
1.3 程序设计 .....	6
1.4 小结.....	12
<b>2 实验 7 结构与联合 .....</b>	<b>13</b>
2.1 表达式求值的程序验证 .....	13
2.2 源程序修改替换 .....	14
2.3 程序设计 .....	16
2.4 小结.....	26
参考文献.....	27

# 1 实验 6 指针程序设计实验

## 1.1 程序改错与跟踪调试

在下面所给的源程序中，函数 strcpy(t, s) 的功能是将字符串 s 复制给字符串 t，并且返回串 t 的首地址。请单步跟踪程序，根据程序运行时出现的现象或观察到的字符串的值，分析并排除源程序的逻辑错误，使之能按照要求输出如下结果：

Input a string:  
programming✓（键盘输入）  
programming  
Input a string again:  
language✓（键盘输入）  
language

源程序：

```
1  #include<stdio.h>
2  char *strcpy(char *, const char *);
3  int main(void)
4  {
5      char *s1, *s2, *s3;
6      printf("Input a string:\n", s2);
7      scanf("%s", s2);
8      strcpy(s1, s2);
9      printf("%s\n", s1);
10     printf("Input a string again:\n", s2);
11     scanf("%s", s2);
12     s3 = strcpy(s1, s2);
13     printf("%s\n", s3);
14     return 0;
15 }
16 /*将字符串s复制给字符串t，并且返回串t的首地址*/
17 char * strcpy(char *t, const char *s)
18 {
19     while(*t++ = *s++);
20     return (t);
21 }
22
```

图 1-1-1-1 源程序

解答：

第 5 行的字符数组定义错误，正确形式为：

```
#define maxn 10005
```

```
char s1[maxn] , s2[maxn] , s3[maxn];
```

第 12 行的语句错误，正确形式为：

```
strcpy(s3, s2);
```

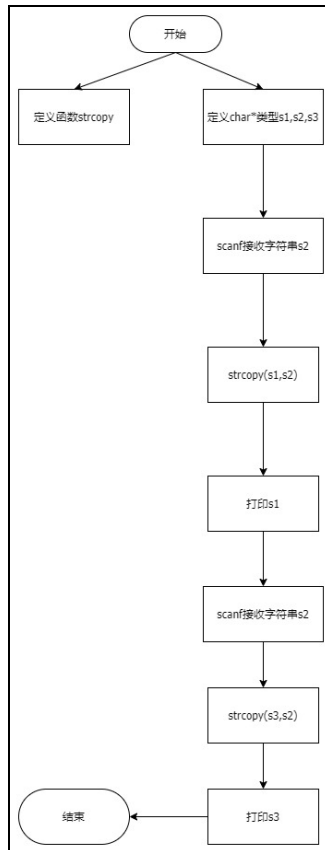


图 1-1-1-2 流程图

```

c:\Users\flans\Desktop\C | × + ∨
Input a string:
programming
programming
Input a string again:
language
language

-----
-
进程已结束。按任意键关闭窗口...

```

图 1-1-2 源程序改错题的程序运行结果示意图

## 1.2 程序完善与修改替换

(1) 下面程序中函数 `strsort` 用于对字符串进行升序排序，在主函数中输入 `N` 个字符串（字符串长度不超过 49）存入通过 `malloc` 动态分配的存储空间，然后调用 `strsort` 对这 `N` 个串按字典序升序排序。

①请在源程序中的下划线处填写合适的代码来完善该程序。

源程序：

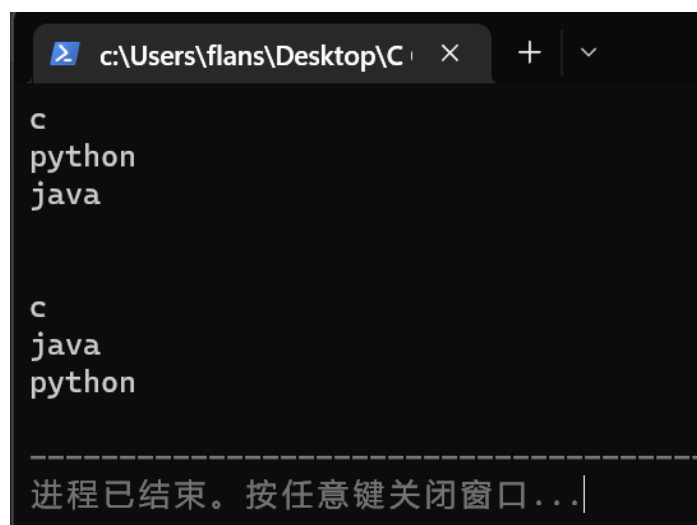
```
1  #include<stdio.h>
2  #include<_____>
3  #include<string.h>
4  #define N 4
5  /*对指针数组s指向的size个字符串进行升序排序*/
6  void strsort(char *s[], int size)
7  {
8      _____temp;
9      int i, j;
10     for(i=0; i<size-1; i++)
11         for (j=0; j<size-i-1; j++)
12             if (_____)
13             {
14                 temp = s[j];
15                 _____;
16                 s[j+1] = temp;
17             }
18     }
19
20     int main()
21     {
22         int i;
23         char *s[N], t[50];
24         for (i=0; i<N; i++)
25         {
26             gets(t);
27             s[i] = (char *)malloc(strlen(t)+1);
28             strcpy(_____);
29         }
30         strsort(_____);
31         for (i=0; i<N; i++)    {puts(s[i]); free(s[i]);}
32         return 0;
33     }
34
35
```

图 1-2-1 程序完善源程序

替换后的程序如下图所示：

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4  #define N 4
5  /*对指针数组s指向的size个字符串进行升序排序*/
6  void strsort(char *s[], int size)
7  {
8      char *temp ;
9      int i, j;
10     for(i=0; i<size-1; i++)
11         for (j=0; j<size-i-1; j++)
12             if ( strcmp( s[j] , s[j+1]) > 0 )
13             {
14                 temp = s[j];
15                 s[j] = s[j+1] ;
16                 s[j+1] = temp;
17             }
18 }
19
20 int main()
21 {
22     int i;
23     char *s[N], t[50];
24     for ( i=0 ; i<N ; i++)
25     {
26         gets(t);
27         s[i] = (char *)malloc(strlen(t)+1);
28         strcpy( s[i] , t );
29     }
30     strsort( s , N );
31     for (i=0; i<N; i++)    {puts(s[i]); free(s[i]);}
32     return 0;
33 }
34
```

图 1-2-2 修改之后的程序



```
c
python
java

c
java
python

-----
进程已结束。按任意键关闭窗口...|
```

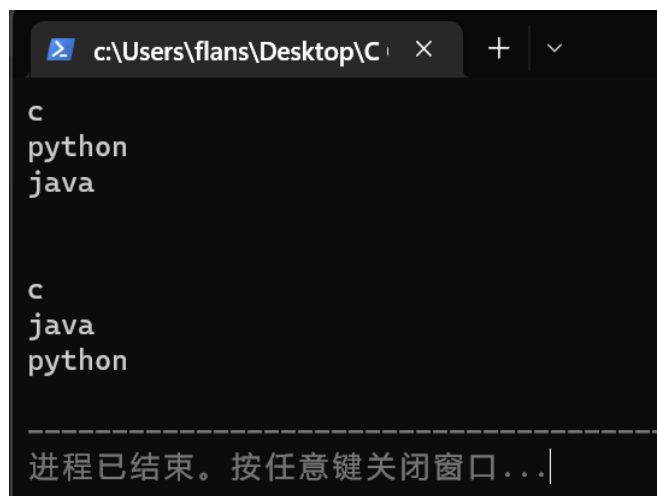
图 6-2-3 运行结果

②数组作为函数参数其本质类型是指针。例如，对于形参 `char *s []`，编译器将其解释为 `char **s`，两种写法完全等价。请用二级指针形参重写 `strsort` 函数，并且在该函数体的任何位置都不允许使用下标引用。

替换后的程序如下所示：

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4  #define N 4
5  /*对指针数组s指向的size个字符串进行升序排序*/
6  void strsort(char **s , int size)
7  {
8      char *temp ;
9      int i, j;
10     for(i=0; i<size-1; i++)
11         for (j=0; j<size-i-1; j++)
12             if ( strcmp( *( s + j ) , *( s + j + 1 ) ) > 0 )
13             {
14                 temp = *( s + j ) ;
15                 ( *( s + j ) ) = *( s + j + 1 );
16                 *( s + j + 1 ) = temp;
17             }
18 }
19
20 int main()
21 {
22     int i;
23     char *s[N], t[50];
24     for ( i=0 ; i<N ; i++)
25     {
26         gets(t);
27         s[i] = (char *)malloc(strlen(t)+1);
28         strcpy( s[i] , t );
29     }
30     strsort( s , N );
31     for (i=0; i<N; i++)    {puts(s[i]); free(s[i]);}
32     return 0;
33 }
34
```

图 1-2-4 替换之后的程序



```
c
python
java

c
java
python

-----
进程已结束。按任意键关闭窗口...|
```

图 1-2-5 运行结果

## 1.3 程序设计

### 1. 指针取出每个字节：

编写一个程序，从整型变量的高字节开始，依次取出每字节的高 4 位和低 4 位并以十六进制数字字符的形式进行显示，要求通过指针取出每字节。

解答：

```
1  #include<stdio.h>
2  int main(){
3      unsigned int n;
4      scanf("%d",&n);
5      unsigned char *p=(char *)&n;
6      char up,low;
7      char s[8];
8      int e=7;
9      for(int k=0;k<4;k++){
10         low=(*p)&0x0f;
11         if(low<10) low=low|'0';
12         else low=low-10+'a';
13         up=(*p>>4)&0x0f;
14         if(up<10) up|= '0';
15         else up=up-10+'a';
16         p++;
17         s[e]=low;
18         e--;
19         s[e]=up;
20         e--;
21     }
22     for(e=0;e<8;e++){
23         printf("%c ",s[e]);
24     }
25     return 0;
26 }
```

图 1-3-1 程序代码



图 1-3-2 运行结果



## 2. 删除重复字符:

定义函数 RemoveSame(a, n)，去掉有 n 个元素的有序整数序列 a 中的重复元素，返回去重后序列的长度。

解答:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int RemoveSame(int a[], int n);
4  int main(void)
5  {
6      int n, a[100], i;
7      scanf("%d", &n);
8      for (i = 0; i < n; ++i) scanf("%d", &a[i]);
9
10     n = RemoveSame(a, n);
11     for (i = 0; i < n - 1; ++i) printf("%d ", a[i]);
12     printf("%d\n", a[n - 1], n);
13
14 }
15
16 int RemoveSame(int a[], int n)
17 {
18     int key, cnt = 0, *ptr, *q;
19     for (ptr = a; ptr < a + n; ptr++)
20     {
21         key = 1;
22         for (q = a; q < a + cnt; q++)
23             if (*ptr == *q)
24             {
25                 key = 0;
26                 break;
27             } // 如果相等, key标识为0代表有重复的, 后面cnt不进行累加
28
29         if (key) a[cnt++] = *ptr; // 如果key=1说明ptr与之前cnt个都不重复, cnt可以加1
30     }
31     return cnt;
32 }
33 /*直接通过指针a重新赋值, 用剩下的元素与已知的新数组a比较, 重复则舍弃, 不同则cnt++,
34 最后输出cnt, 此种做法基于不需要输出比较后的结果。若需输出, 还需将重复元素删除。*/
```

图 1-3-3 程序代码



```
c:\Users\flans\Desktop\C 5
3 3 3 6 6
3 6
2

-----
进程已结束。按任意键关闭窗口...
```

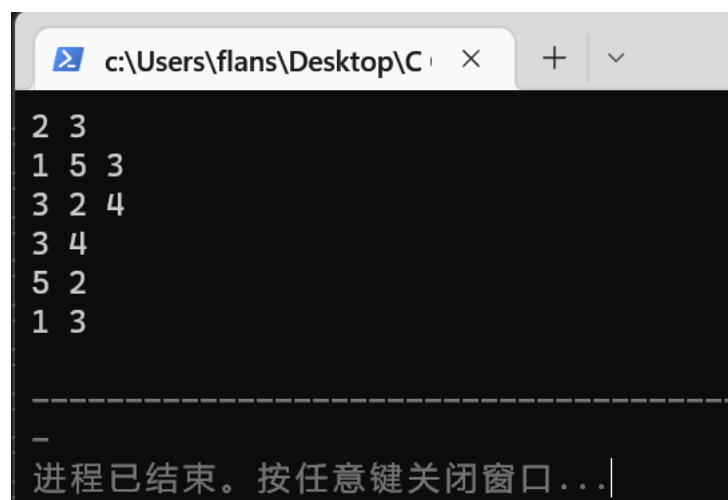
图 1-3-4 运行结果

### 3. 旋转图像：

输入图像矩阵的行数  $n$  和列数  $m$ ，接下来的  $n$  行每行输入  $m$  个整数，表示输入的图像，输出原始矩阵逆时针旋转  $90^\circ$  后的矩阵。

```
1  #include <stdio.h>
2
3  #define re register
4  #define maxn 10
5
6  int n , m ;
7  int a[maxn][maxn] ;
8
9  int main()
10 {
11     scanf("%d%d",&n,&m) ;
12     for( re int i = 1 ; i <= n ; i++)
13         for( re int j = 1 ; j <= m ; j++)
14             scanf("%d",&a[i][j]) ;
15     for ( re int i = m ; i >= 1 ; i-- )
16     {
17         for( re int j = 1 ; j <= n-1 ; j++ )
18             printf("%d ",a[j][i]) ;
19         printf("%d\n",a[n][i]) ;
20     }
21     return 0 ;
22 }
```

图 1-3-5 程序代码



```
c:\Users\flans\Desktop\C> 2 3
1 5 3
3 2 4
3 4
5 2
1 3
-----
进程已结束。按任意键关闭窗口...
```

图 1-3-6 运行结果

4. 命令行实现对 N 个整数排序：  
解答：

```
1  #include <stdio.h>
2  #include<stdlib.h>
3  void BubbleSort(int a[], int n)
4  {
5      int i, j;
6      for (i = 0; i < n; i++) {
7          for (j = i+1; j < n; j++) {
8              if (a[i] > a[j]) {
9                  int t = a[i]; a[i] = a[j]; a[j] = t;
10             }
11         }
12     }
13 }
14 void BubbleSort2(int a[], int n)
15 {
16     int i, j;
17     for (i = 0; i < n; i++) {
18         for (j = i+1; j < n; j++) {
19             if (a[i] < a[j]) {
20                 int t = a[i]; a[i] = a[j]; a[j] = t;
21             }
22         }
23     }
24 }
25
26 int main(int argc, char* argv[])
27 {
28     int length = atoi(argv[1]);
29     int arr[length];
30     int i;
31
32     for (i = 0; i < atoi(argv[1]); i++) {
33         scanf("%d",&arr[i]);
34     }
35     if(argv[2]==NULL) BubbleSort(arr,length);
36     else BubbleSort2(arr,length);
37
38     for (i = 0; i < length; i++)
39     {
40         printf("%d ", arr[i]);
41     }
42
43
44     return 0;
45 }
```

图 1-3-7 程序代码

```
PS C:\Users\flans\Desktop\C CODES> sort 5 -d
4 3 8 5 1
8 5 4 3 1
```

图 1-3-6 运行结果

## 5. 删除子串:

解答:

```
1  #include <stdio.h>
2  #include <string.h>
3  char* delete(char result[], char str1[], const char str2[]);
4  //创建函数,实现删除字符串中的子串,利用到递归。
5  int main(){
6      char str1[100]; //设置数组长度,要大于80
7      char str2[100];
8      char result[100]={}; //新建空字符串用来存储与子串不同的字符串
9      gets(str1); //获取字符串
10     gets(str2); //获取子串
11     delete(result,str1, str2);
12     return 0;
13 }
14
15
16 char* delete(char result[], char str1[], const char str2[])
17 {
18     int flag = 0;
19     int count = 0;
20     //统计字符串中与子串相同的字符长度,判断是否要递归
21     int i = 0; //将其放在循环外,为了更容易在串最后加上'\0'
22     int len = strlen(str2); //strlen函数获取子串长度
23     while (str1[0] != '\0')//进行循环,如果字符串第一个位置'\0'则停止循环
24     {
25         count = 0; //每次循环开始重置count
26         if (str1[0] == str2[0])//如果字符串的第一位和子串的第一位相同,那么开始遍历判断
27         {
28             for (int k = 0; k < len; k++)//字符串中的字符与子串长度是否(也就是两者相同)相同
29             {
30                 if (str1[k] == str2[k]) count++;
31                 else break;
32             }
33             if (count == len)//如果count和子串长度相同,那么就将字符串往后移动子串的长度
34             {
35                 str1 += len;
36                 flag = 1; //如果满足上面的条件,表示可以进行遍历(该遍历应该可以再优化)
37             }
38             else
39             {
40                 result[i] = str1[0]; //不满足则将字符串存入result中
41                 str1++; //将字符串往后移动一位
42                 i++; //每次存入一个字符串则将i加一
43             }
44         }
45         if (str1[0] != str2[0])//不满足的情况,与上面else同理
46         {
47             result[i] = str1[0];
48             str1++;
49             i++;
50         }
51     }
52     result[i] = '\0'; //串的结尾以'\0'结尾
53     if (flag == 1)
54     {
55         printf("%s\n",result);//不满足递归则直接return
56         printf("1");
57     }
58     else
59     {
60         printf("%s\n",result);//不满足递归则直接return
61         printf("0");
62     }
63 }
```

图 1-3-9 程序代码



图 1-3-10 运行结果

\*\*6. 求两个不超过 200 位的非负整数积:

解答:

```

1  #include<stdio.h>
2  #include<string.h>
3  int main()
4  {
5      char a[201], b[201]; //不能是200, 因为数组可能越界
6      int i;
7      int j, k, q;
8      scanf("%s", a);
9      scanf("%s", b);
10     j = strlen(a); //算出a, b中元素的个数
11     k = strlen(b);
12
13     int c[401] = { 0 };
14     int cnt = 0;
15     int w, e, r;
16     r = 0;
17     int cnt1; //用来统计c中个数
18

```

图 1-3-11 程序代码

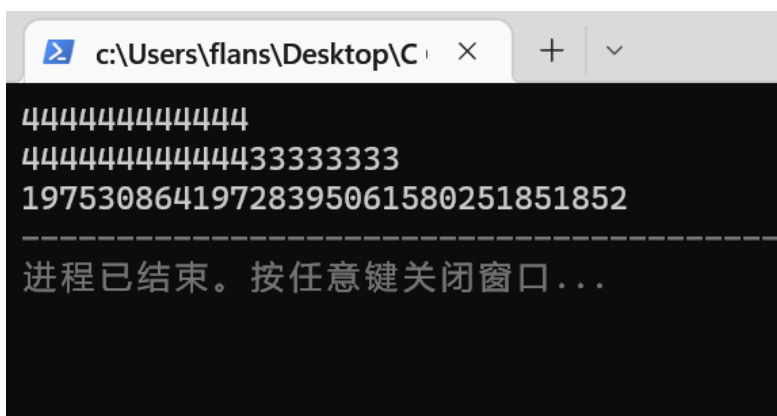


图 1-3-12 运行结果

## 1.4 小结

指针是 C 语言学习和使用的重难点。本次实验主要练习了指针的使用，包括指针变量的声明和引用，利用指针引用所指变量，作为函数参数和返回值的指针的使用，复杂类型指针的使用等。

指针常常被称为 C 语言的精华，我认识到了它的重要意义。借由指针，C 语言可以对计算机内存进行直接操作，更加贴近底层，大大提高了执行效率。但同时，这也要求我们设计程序时格外重视对指针的使用，防止出现野指针，内存泄漏等情况。

## 2 实验 7 结构与联合

### 2.1 表达式求值的程序验证

```
1 #include<stdio.h>
2 int main()
3 {
4     char u[]="UVWXYZ";
5     char v[]="xyz";
6     struct T{
7         int x;
8         char c;
9         char *t;
10    }a[]={11,'A',u},{100,'B',v}},*p=a;
11    int choice=0;
12    scanf("%d",&choice);
13    switch (choice)
14    {
15        case 1:
16        {
17            int a1=(++p)->x;
18            printf("%d\n",a1);
19            break;
20        }
21        case 2:
22        {
23            int a2=(++p)->c;
24            printf("%c\n",a2);
25            break;
26        }
27        case 3:
28        {
29            *p++->t;
30            int a3=*p->t;
31            printf("%c\n",a3);
32            break;
33        }
34        case 4:
35        {
36            int a4=(++p)->t;
37            printf("%c\n",a4);
38            break;
39        }
40        case 5:
```

图 2-1-1 程序代码

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

图 2-1-2 计算结果

## 2.2 源程序修改替换

(1) 给定一批整数，以 0 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。

修改后的程序：

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  struct s_list
4  {
5      int data; /* 数据域 */
6      struct s_list *next; /* 指针域 */
7  };
8  void create_list (struct s_list **headp,int *p);
9  int main(void)
10 {
11     struct s_list *head=NULL,*p;
12     int s[9]={0};/* 0为结束标记 */
13     for (int i=0;i<=8;i++)
14     {
15         scanf("%d",&s[i]);
16     }
17     create_list(&head,s); /* 创建新链表 */
18     p=head; /* 遍历指针p指向链头 */
19     while(p)
20     {
21         printf("%d\t",p->data); /* 输出数据域的值 */
22         p=p->next; /* 遍历指针p指向下一结点 */
23     }
24     printf("\n");
25     return 0 ;
26 }
27 void create_list(struct s_list **headp,int *p)
28 {
29     struct s_list * loc_head=NULL, * tail;
30     if(p[0]==0) /* 相当于*p==0 */;
31     else
32     { /* loc_head指向动态分配的第一个结点 */
33         loc_head=(struct s_list *)malloc(sizeof(struct s_list));
34         loc_head->data=*p++; /* 对数据域赋值 */
35         tail=loc_head; /* tail指向第一个结点 */
36         while(*p)
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             tail->data=*p++; /* 向新创建的结点的数据域赋值 */
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 */
45     }
46 }
```

图 2-2-1 程序代码

图 2-2-2 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。

修改后的程序：

```
1  #include "stdio.h"
2  #include "stdlib.h"
3  struct s_list
4  {
5      int data; /* 数据域 */
6      struct s_list *next; /* 指针域 */
7  };
8  void create_list (struct s_list **headp,int *p);
9  int main(void)
10 {
11     struct s_list *head=NULL,*p;
12     int s[9]={0};/* 0为结束标记 */
13     for (int i=0;i<=8;i++)
14     {
15         scanf("%d",&s[i]);
16     }
17     create_list(&head,s); /* 创建新链表 */
18     p=head; /*遍历指针p指向链头 */
19     while(p)
20     {
21         printf("%d\t",p->data); /* 输出数据域的值 */
22         p=p->next; /*遍历指针p指向下一结点 */
23     }
24     printf("\n");
25     return 0 ;
26 }
27 void create_list(struct s_list **headp,int *p)
28 {
29     struct s_list * loc_head=NULL, * tail , * tmp = NULL ;
30     if(p[0]==0) /* 相当于*p==0 */;
31     else
32     { /* loc_head指向动态分配的第一个结点 */
33         loc_head=(struct s_list *)malloc(sizeof(struct s_list));
34         loc_head->data=*p++; /* 对数据域赋值 */
35         tail=loc_head; /* tail指向第一个结点 */
36         while(*p)
37         { /* tail所指结点的指针域指向动态创建的结点 */
38             tmp = (struct s_list *)malloc(sizeof(struct s_list)) ;
39             tmp->data = *p++;
40             tmp->next = loc_head ;
41             loc_head = tmp ;
42             // loc_head->data = tmp->data ;
43         }
44         tail->next=NULL; /* 对指针域赋NULL值 */
45     }
46     *headp=loc_head; /* 使头指针headp指向新创建的链表 */
47 }
```

图 2-2-3 程序代码

图 2-2-4 运行结果

## 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到最高位声明为 8 个字段，依次为 `bit0`, `bit1`, ..., `bit7`，且 `bit0` 优先级最高。同时设计 8 个函数，第 `i` 个函数以 `biti` ( $i=0, 1, \dots, 7$ ) 为参数，并且在函数体内输出 `biti` 的值。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顺序依次调用函数指针数组 `p_fun` 中相应元素指向的函数。

```
2  #include <stdio.h>
3  struct bits {
4      int bit0 : 1;
5      int bit1 : 1;
6      int bit2 : 1;
7      int bit3 : 1;
8      int bit4 : 1;
9      int bit5 : 1;
10     int bit6 : 1;
11     int bit7 : 1;
12 } b;
13 void f0(int n) {
14     printf("the function %d is called!\n", n);
15 };
16 void f1(int n) {
17     printf("the function %d is called!\n", n);
18 };
19 void f2(int n) {
20     printf("the function %d is called!\n", n);
21 };
22 void f3(int n) {
23     printf("the function %d is called!\n", n);
24 };
25 void f4(int n) {
26     printf("the function %d is called!\n", n);
27 };
28 void f5(int n) {
29     printf("the function %d is called!\n", n);
30 };
31 void f6(int n) {
32     printf("the function %d is called!\n", n);
33 };
34 void f7(int n) {
35     printf("the function %d is called!\n", n);
36 };
37 int main(void)
38 {
39     void (*p_fun[8])(int) = {f0, f1, f2, f3, f4, f5, f6, f7};
40     int order[8]={0,0,0,0,0,0,0,0};
41
42     int c=0;
43     scanf("%d",&c);
44     int p=0,yushu;
45     while(1)
46     {
47         yushu=c%2;
48         c/=2;
49         order[p]=yushu;
50         p++;
51         if(c==0) break;
52     }
53     b.bit0 = order[0] ;
54     b.bit1 = order[1] ;
55     b.bit2 = order[2] ;
56     b.bit3 = order[3] ;
57     b.bit4 = order[4] ;
58     b.bit5 = order[5] ;
59     b.bit6 = order[6] ;
60     b.bit7 = order[7] ;
61     if (b.bit0) p_fun[0](0);
62     if (b.bit1) p_fun[1](1);
63     if (b.bit2) p_fun[2](2);
64     if (b.bit3) p_fun[3](3);
65     if (b.bit4) p_fun[4](4);
66     if (b.bit5) p_fun[5](5);
67     if (b.bit6) p_fun[6](6);
68     if (b.bit7) p_fun[7](7);
69     return 0;
70 }
```

图 2-2-5 程序代码

```
c:\Users\flans\Desktop\C \times + \n
111
the function 0 is called!
the function 1 is called!
the function 2 is called!
the function 3 is called!
the function 5 is called!
the function 6 is called!

-----
进程已结束。按任意键关闭窗口...|
```

图 2-2-6 运行结果

(2) 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 4 门课程的成绩。

```
2  #include<stdio.h>
3  #include<stdlib.h>
4  #include<string.h>
5  #define N 4
6  struct list {
7      char num[15], name[9];
8      int c, phy, e, math;
9      struct list* next;
10 };
11 struct list* creat()
12 {
13     int n;
14     scanf("%d",&n);
15     int i;
16     struct list* h = (struct list*)malloc(sizeof(struct list));
17     struct list* p2 = h;
18     for (i = 0; i < n; i++){
19         struct list* p = (struct list*)malloc(sizeof(struct list));
20         scanf("%s %s %d %d %d %d",&p->num,&p->name,&p->c,&p->phy,&p->e,&p->math);
21         p2->next = p;
22         p2 = p;
23         p->next = NULL;
24     }
25     return h;
26 }
27 void input(struct list* p3) {
28     int n;
29     scanf("%d",&n);
30     int i;
31     struct list* p2 = p3;
32     while(p2->next!=NULL)
33         p2=p2->next;
34     for (i = 0; i < n; i++){
35         struct list* p = (struct list*)malloc(sizeof(struct list));
36         scanf("%s %s %d %d %d %d",&p->num,&p->name,&p->c,&p->phy,&p->e,&p->math);
37         p2->next = p;
38         p2 = p;
39         p->next = NULL;
40     }
```

```

41 }
42 void out(struct list* p) {
43     struct list *q = p->next;
44     while (q != NULL) {
45         printf("%s %s ", q->num, q->name);
46         printf("%d %d %d %d\n", q->c, q->phy, q->e, q->math);
47         q = q->next;
48     }
49 }
50 void change(struct list* p, char* p1) {
51     while (p != NULL) {
52         if (strcmp(p->num, p1)==0){
53             int x;
54             scanf("%d",&x);
55             if(x==1) scanf("%d",&p->c);
56             else if(x==2) scanf("%d",&p->phy);
57             else if(x==3) scanf("%d",&p->e);
58             else if(x==4) scanf("%d",&p->math);
59         }
60         p = p->next;
61     }
62 }
63 void sum1(struct list* q) {
64     int i = 0, j = 0, k = 0, l = 0;
65     int sum = 0;
66     struct list *p = q->next;
67     while (p != NULL) {
68         sum += p->c;
69         sum += p->phy;
70         sum += p->e;
71         sum += p->math;
72         printf("%s %s %.2f\n",p->num,p->name, (float)sum / 4);
73         sum = 0;
74         p = p->next;
75     }
76 }

```

```

77 void sum2(struct list* q) {
78     int i = 0, j = 0, k = 0, l = 0;
79     int sum = 0;
80     struct list *p = q->next;
81     while (p != NULL) {
82         sum += p->c;
83         sum += p->phy;
84         sum += p->e;
85         sum += p->math;
86         printf("%s %s %d %.2f\n",p->num,p->name,sum, (float)sum / 4);
87         sum = 0;
88         p = p->next;
89     }
90 }
91 void main() {
92     struct list* p=NULL, * h;
93     char c, *p1,ex[15];
94     int u=0;
95     while((c=getchar())!=EOF){
96         if(c=='0') break;
97         else if(c=='1'&&u==0) {h=creat();u=1;}
98         else if(c=='1'&&u==1) input(h);
99         else if(c=='2') out(h);
100        else if(c=='3'){
101            scanf("%s", &ex);
102            p1=ex;
103            change(h, p1);
104        }
105        else if(c=='4') sum1(h);
106        else if(c=='5') sum2(h);
107    }
108 }

```

图 2-2-7 程序代码

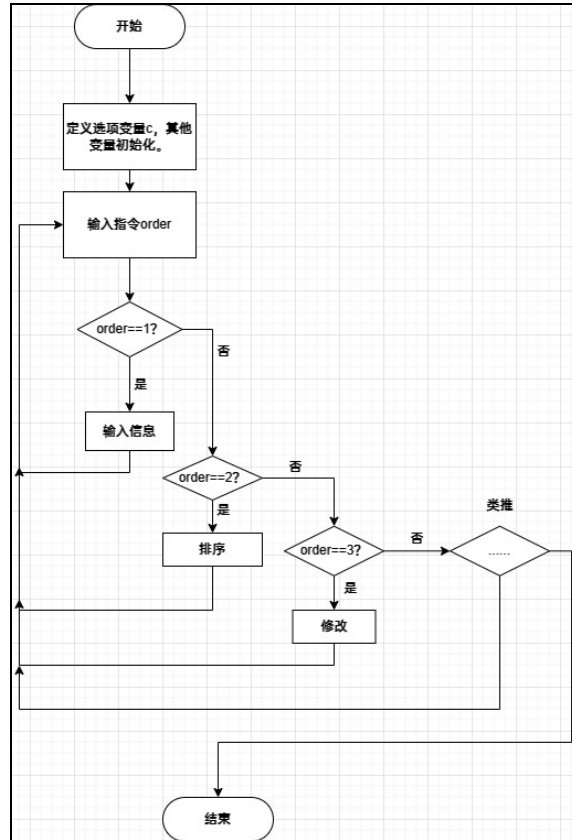


图 2-2-8-1 流程图

```

c:\Users\flans\Desktop\C  x + v
1
2
U202012345 Jack 99 100 80 96
U202054321 Rose 89 94 85 100
2
U202012345 Jack 99 100 80 96
U202054321 Rose 89 94 85 100
3
U202054321 1 66

4
U202012345 Jack 93.75
U202054321 Rose 86.25
5
U202012345 Jack 375 93.75
U202054321 Rose 345 86.25
0

-----
进程已结束。按任意键关闭窗口...|
  
```

图 2-2-8-2 运行结果

(3) 对程序设计的第二题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。

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4  int all=0;
5  struct list {
6      char num[15], name[9];
7      int c, phy, e, math;
8      float ave;
9      struct list* next;
10 };
11 struct list* creat()
12 {
13     int n;
14     scanf("%d",&n);
15     all+=n;
16     int i;
17     struct list* h = (struct list*)malloc(sizeof(struct list));
18     struct list* p2 = h;
19     for (i = 0; i < n; i++){
20         struct list* p = (struct list*)malloc(sizeof(struct list));
21         scanf("%s %s %d %d %d %d",&p->num,&p->name,&p->c,&p->phy,&p->e,&p->math);
22         int sum;
23         sum=p->c + p->phy + p->e + p->math;
24         p->ave=(float)sum/4;
25         p2->next = p;
26         p2 = p;
27         p->next = NULL;
28     }
29     return h;
30 }
31 void input(struct list* p3) {
32     int n;
33     scanf("%d",&n);
34     all+=n;
35     int i;
36     struct list* p2 =p3;
37     while(p2->next!=NULL)
38         p2=p2->next;
39     for (i = 0; i < n; i++){
40         struct list* p = (struct list*)malloc(sizeof(struct list));

```

```

41         scanf("%s %s %d %d %d %d",&p->num,&p->name,&p->c,&p->phy,&p->e,&p->math);
42         int sum=p->c + p->phy + p->e + p->math;
43         p->ave=(float)sum/4;
44         p2->next = p;
45         p2 = p;
46         p->next = NULL;
47     }
48 }
49 void out(struct list* p) {
50     struct list *q = p->next;
51     while (q != NULL) {
52         printf("%s %s ", q->num, q->name);
53         printf("%d %d %d %d\n", q->c, q->phy, q->e, q->math);
54         q = q->next;
55     }
56 }
57 void change(struct list* p, char* p1) {
58     while (p != NULL) {
59         if (strcmp(p->num, p1)==0){
60             int x;
61             scanf("%d",&x);
62             if(x==1) scanf("%d",&p->c);
63             else if(x==2) scanf("%d",&p->phy);
64             else if(x==3) scanf("%d",&p->e);
65             else if(x==4) scanf("%d",&p->math);
66             int sum;
67             sum=p->c + p->phy + p->e + p->math;
68             p->ave=(float)sum/4;

```

```

69     }
70     p = p->next;
71 }
72 }
73 void sum1(struct list* q) {
74     struct list *p = q->next;
75     while (p != NULL) {
76         printf("%s %s %.2f\n",p->num,p->name, p->ave);
77         p = p->next;
78     }
79 }
80 void sum2(struct list* q) {
81     struct list *p = q->next;
82     int sum=0;
83     while (p != NULL) {
84         sum=p->c + p->phy + p->e + p->math;
85         printf("%s %s %d %.2f\n",p->num,p->name,sum,p->ave);
86         sum = 0;
87         p = p->next;
88     }
89 }
90 void swap(struct list* p,struct list* q){
91     swop(p->num,q->num);
92     swop(p->name,q->name);
93     int temp;
94     temp=p->c;p->c=q->c;q->c=temp;
95     temp=p->phy;p->phy=q->phy;q->phy=temp;
96     temp=p->e;p->e=q->e;q->e=temp;
97     temp=p->math;p->math=q->math;q->math=temp;
98     float teemp;
99     teemp=p->ave;p->ave=q->ave;q->ave=teemp;
100 }
101 void swop(char pa[],char pb[]){
102     int i=0;
103     while(pa[i]!='\0' || pb[i]!='\0'){
104         swop2(&pa[i],&pb[i]);

```

```

105         i++;
106     }
107 }
108 void swop2(char *a,char *b){
109     char temp;
110     temp=*a;
111     *a=*b;
112     *b=temp;
113 }
114 void upper(struct list* p){
115     for(int i=0;i<all-1;i++){
116         struct list *q=p->next;
117         for(int j=0;j<all-1-i;j++){
118             if(q->ave > q->next->ave){
119                 swap(q,q->next);
120             }
121             q=q->next;
122         }
123     }
124 }
125 void main() {
126     struct list* p=NULL, * h;
127     char c, *p1,ex[15];
128     int u=0;
129     while((c=getchar())!=EOF){
130         if(c=='0') break;
131         else if(c=='1'&&u==0){
132             h=creat();
133             u=1;
134             upper(h);
135         }
136         else if(c=='1'&&u==1){
137             input(h);
138             upper(h);
139         }
140         else if(c=='2') out(h);
141         else if(c=='3'){
142             scanf("%s", &ex);
143             p1=ex;
144             change(h, p1);
145             upper(h);
146         }
147         else if(c=='4') sum1(h);
148         else if(c=='5') sum2(h);
149     }
150 }

```

图 2-2-9 程序代码



```

c:\Users\flans\Desktop\C  x  +  v
1
2
U202054321 Rose 89 94 85 100
U202012345 Jack 99 100 80 96
2
U202054321 Rose 89 94 85 100
U202012345 Jack 99 100 80 96
3
U202054321 1 66
4
U202054321 Rose 86.25
U202012345 Jack 93.75
5
U202054321 Rose 345 86.25
U202012345 Jack 375 93.75
0
-----
--
进程已结束。按任意键关闭窗口...|

```

图 2-2-10 运行结果

(4) 回文字符串是正读和反读都相同的字符串，例如“abba”和“abcba”是回文字符串。设计程序判断输入的任意长度的字符串是否为回文字符串。提示：由于要求字符串长度任意，所以用单链表存储字符串，即判断一个单链表是否为回文链表。

```

5  *****/
6  #include<stdlib.h>
7  #include<stdio.h>
8  void createLinkList(C_NODE **headp, char s[])
9  {
10 /***** BEGIN *****/
11 *headp=(C_NODE*)malloc(sizeof(C_NODE));
12 (*headp)->data=s[0];
13 (*headp)->next=NULL;
14 C_NODE *pre;
15 C_NODE *p;
16 pre=*headp;
17 int length;
18 length=strlen(s);
19 int i=1;
20 while(i<length){
21 p=(C_NODE*)malloc(sizeof(C_NODE));
22 p->data=s[i];
23 pre->next=p;
24 pre=p;
25 i++;
26 }
27 /***** BEGIN *****/
28 }

30 void judgePalindrome(C_NODE *head)
31 {
32 /***** BEGIN *****/
33 C_NODE *p;
34 p=head;
35 int n=0;
36 char str[100];
37 while(p!=NULL){
38 str[n]=p->data;
39 p=p->next;
40 n++;
41 }
42 int i=0;
43 n--;
44 for(i=0;i<n;i++){
45 if(str[i]!=str[n]){printf("false");break;}
46 }
47 if(i==n||i>n) printf("true");
48 /***** BEGIN *****/
49 }

```

图 2-2-11 程序代码

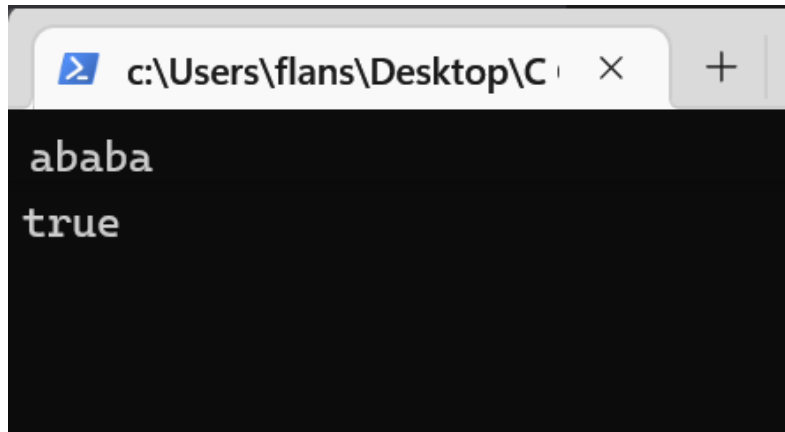


图 2-2-12 运行结果

(5) 利用值栈对逆波兰表达式进行求值。逆波兰表达式从键盘输入，其中的运算符仅包含加、减、乘、除 4 种运算，表达式中的数都是十进制数，用换行符结束输入。由于逆波兰表达式的长度不限，所以值栈要用后进先出链表实现。

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #define MaxSize 199
4
5  struct SNode {
6      int Data[MaxSize];
7      int Top;
8  };
9  typedef struct SNode* Stack;
10
11 Stack CreateStack() {
12     Stack p;
13     p = (Stack)malloc(sizeof(struct SNode));
14     p->Top = -1;
15     return p;
16 }
17
18 void Push(Stack S, int x) {
19     if (S->Top == MaxSize) {
20         printf("Stack Full\n");
21     }
22     else {
23         S->Data[++S->Top] = x;
24     }
25 }
26
27 int Pop(Stack S) {
28     if (S->Top == -1) {
29         printf("Stack is Empty!\n");
30     }
31     else {
32         int t;
33         t = S->Data[S->Top];
34         S->Top--;
35         return t;
36     }
37 }
38
```

```

39 int main() {
40     Stack S;
41     S = CreateStack();
42     char ch;
43     ch = getchar();
44     int a, b, an, t;
45
46     while (ch != EOF) {
47         if (ch >= '0' && ch <= '9') {
48             Push(S, ch - '0');
49             ch = getchar();
50             while (ch >= '0' && ch <= '9') {
51                 t = Pop(S);
52                 Push(S, t * 10 + ch - '0');
53                 ch = getchar();
54             }
55         }
56         else if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
57             a = Pop(S);
58             b = Pop(S);
59             switch (ch) {
60                 case '+': an = b + a; break;
61                 case '-': an = b - a; break;
62                 case '*': an = b * a; break;
63                 case '/': an = b / a; break;
64             }
65             Push(S, an);
66         }
67         else if (ch == '\n') {
68             break;
69         }
70         ch = getchar();
71     }
72     printf("%d", Pop(S));
73 }

```

图 2-2-13 程序代码

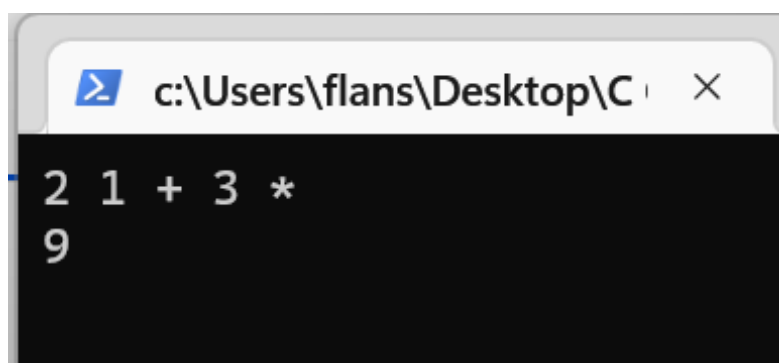


图 2-2-14 运行结果

## 2.4 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，也可以达到接近面向对象语言中“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。

## 参考文献

- [1] 卢萍, 李开, 王多强, 甘早斌. C 语言程序设计典型题解与实验指导, 北京: 清华大学出版社, 2019