華中科技大学

课 程 实 验 报 告

课程名称： C 语言程序设计实验

专业班级： 计算机科学与技术 XXXX 班

学　　号： UXXXX

姓　　名： XXXX

指导教师： XXXX

报告日期： XXXX

计算机科学与技术学院

目 录

1 实验 6 指针程序设计实验

1.1 程序改错与跟踪调试

```
#include <stdio.h>
char * strcopy(char *, const char *);
int main(void)
{
    char *s1,*s2,*s3;
    改正：char s1[20],s2[20],*s3
    // printf("Input a string:\n");
    scanf("%s",s2);
    strcopy(s1,s2);
    printf("%s\n",s1);
    // printf("Input a string again:\n",s2);
    scanf("%s",s2);
    s3=strcopy(s1,s2);
    printf("%s\n",s3);
    return 0;
}
/*将字符串 s 复制给字符串 t，并且返回串 t 的首地址*/
char * strcopy(char *t,const char *s)
{
    改正：char *p=t;
    while(*t++=*s++);
    return (t);
    改正：return p;
}
```

第一处错误：s1，s2 若声明为两个指向 char 型数据的指针，则不能通过 scanf 对其进行写入操作，因此应声明为两个字符串数组，用来保存输入的字符串。

第二处错误：字符串复制后，应返回第一个字符串的首地址，而指针 t 此时已经移到尾端，因此应在字符串复制前声明一个字符型指针用来保存第一个字符串的首地址。

1.2 程序完善与修改替换

1)字符串排序

程序完善：

```
/* 实验 6 程序完善与修改替换第（1）题源程序：字符串升序排序 */
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
// #define N 4
/* 对指针数组 s 指向的 size 个字符串进行升序排序 */
void strsort ( char *s[ ],int size )
{
    char *temp;
    int i, j ;
    for(i=0; i<size-1; i++)
        for(j=0; j<size-i-1; j++)
            if(strcmp(s[j],s[j+1])>0){
                temp=s[j];
                s[j]=s[j+1];
                s[j+1]=temp;
            }
 }

int main( )
{
    int i;
    int N;
    scanf("%d",&N);
    char *s[N], t[50];
    getchar();
    for(i=0;i<N;i++)
{
        gets(t);
       s[i] = (char *)malloc(strlen(t)+1);
       strcpy(s[i],t);
    }
    strsort(s,N);
    for(i=0;i<N;i++)  printf("%s\n",s[i]);
    return 0
}
```

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
/* 对指针数组s指向的size个字符串进行升序排序 */
void strsort ( char *s[ ],int size )
{
    char *temp;
    int i, j ;
    for(i=0; i<size-1; i++)
        for(j=0; j<size-i-1; j++)
            if (strcmp(s[j],s[j+1])>0)  {
                temp=s[j];
                s[j]=s[j+1];
                s[j+1]=temp;
            }
}

int main( )
{
    int i;
    int N;
    scanf("%d",&N);
    char *s[N], t[50];
    getchar();
    for(i=0;i<N;i++)
{
       gets(t);
       s[i] = (char *)malloc(strlen(t)+1);
       strcpy(s[i],t);
    }
    strsort(s,N);
    for(i=0;i<N;i++)  printf("%s\n",s[i]);
    return 0;
}
```

```
D:\devc++\实验六.exe
3
C
python
java
C
java
python

--------------------------------
Process exited after 14.76 s
请按任意键继续. . .
```

修改替换：

将 strsort 函数声明中的参数 char *s[]改为二级指针 char **s，函数中所有 s[i]改为*(s+i)。

2)函数指针

程序完善：

```c
#include<stdio.h>
#include<string.h>
int main(void)
{
    char *(*p)(char *,const char *);
    char a[80],b[80],* result;
    int choice;
    while(1)
    {
        do{
            // printf("\t\t1 copy string.\n");
            // printf("\t\t2 connect string.\n");
            // printf("\t\t3 Parse string.\n");
```

```
            // printf("\t\t4 exit.\n");
            // printf("\t\tinput a number(1-4) please!\n");
            scanf("%d",&choice);
        } while(choice<1 || choice>4);
        switch(choice)
        {
            case 1: p=strcpy; break;
            case 2: p=strcat; break;
            case 3: p=strtok; break;
            case 4: goto down;
        }
        getchar();
        // printf("input the first string please!\n");
        gets(a);
        // printf("input the second string please!\n");
        gets(b);
        result=p(a,b);
        printf("%s\n",result);
    }
down:
    return 0;
}
```

```
#include<string.h>
int main(void)
{
    char *(*p)(char *,const char *);
    char a[80],b[80],* result;
    int choice;
    while(1)
    {
        do{
//          printf("\t\t1 copy string.\
//          printf("\t\t2 connect string
//          printf("\t\t3 Parse string.
//          printf("\t\t4 exit.\n");
//          printf("\t\tinput a number(
            scanf("%d",&choice);
        } while(choice<1 || choice>4);
        switch(choice)
        {
            case 1: p=strcpy; break;
            case 2: p=strcat; break;
            case 3: p=strtok; break;
            case 4: goto down;
        }
        getchar();//读取回车
//      printf("input the first string
        gets(a);//gets函数从输入流中读取
//      printf("input the second string
        gets(b);
        result= p(a,b);
        printf("the result is: %s\n",result);
    }
down:
    return 0;
}
```

```
D:\devc++\实验六.exe
1
i like c programming
i don't like c programming
the result is: i don't like c programming
2
i am
wjl
the result is: i am wjl
3
www.hust.edu.cn
.
the result is: www
4

--------------------------------
Process exited after 53.35 seconds with retu
请按任意键继续. . .
```

修改替换：

用函数指针数组来指向多个同类的函数，不再使用 switch 语句。

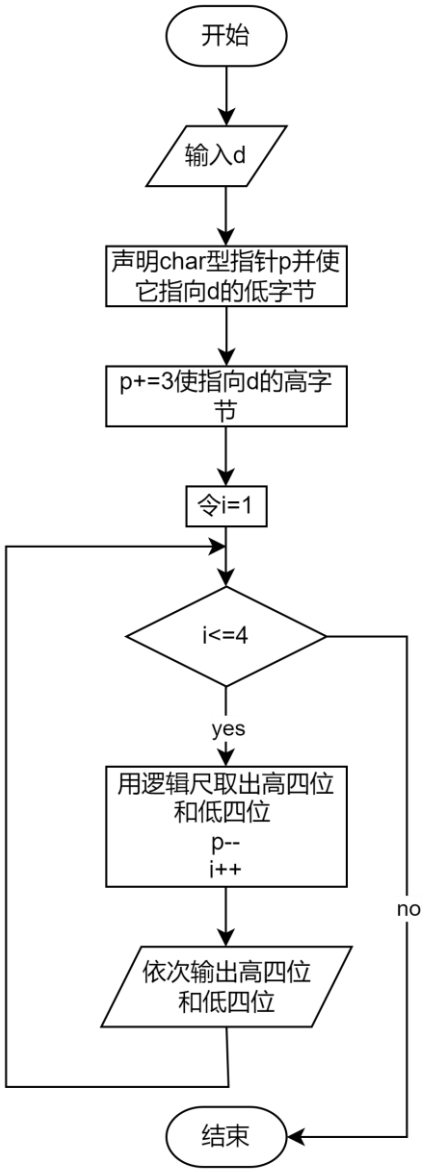
```
#include<stdio.h>
#include<string.h>
int main(void)
{
    char *(*p[3])(char *,const char *);
    char a[80],b[80],* result;
    int choice;
    while(1)
    {
        do{
          scanf("%d",&choice);
        } while(choice<1 || choice>4);
//      switch(choice)
//      {
//          case 1: p=strcpy; break;
//          case 2: p=strcat; break;
//          case 3: p=strtok; break;
//          case 4: goto down;
//      }
        if(choice==4) goto down;
        p[0]=strcpy;
        p[1]=strcat;
        p[2]=strtok;
        getchar();
        gets(a);
        gets(b);
        result= p[choice-1](a,b);
        printf("%s\n",result);
    }
down:
    return 0;
}
```

1.3 程序设计

1. 通过指针取出字节

1)解题思路：声明一个 char 型指针指向整型变量的低字节，再给 p 加 3 使它指向整型变量的高字节，依次间访 p 取出一个字节，再设计两个逻辑尺 0xf0

和 0x0f 取出该字节的高四位和低四位，注意每一轮结束后 p 应减 1 指向下一个字节，共进行四轮。



2)源代码：

```
#include <stdio.h>
int main()
{
    int d;
    scanf("%d",&d);
    char *p;
    p=(char *)&d;
    p+=3;
    for(int i=1;i<=4;i++){
        char a=*p;
```

```
        printf("%x %x ",(a&0xf0)>>4,a&0x0f);
        p--;
    }
    return 0;
}
```

2. 删除重复元素

1)解题思路：编写 RemoveSame 函数，将输入的有序整数数组通过参数从主函数传入 RemoveSame 函数，设计第一层循环，先假设每个数都不重复即 key=1，再将这个数与前面查到的 sum 个不重复的数比较，若等于其中之一，则使 key=0。若 key=1 就让 a[sum]等于这个数(将查到的 sum 个互不重复的数放到数组最前面)，最后返回 sum。

2)源代码：

```
#include <stdio.h>
#include <stdlib.h>
 int RemoveSame(int a[], int n);//给出数组和数组的长度
int main(void)
{
    int n1,n2, a[100];
    scanf("%d", &n1);
    for (int i=0;i<n1;i++) scanf("%d", &a[i]);
    n2=RemoveSame(a, n1);
    for (int i=0;i<n2-1;i++) printf("%d ",a[i]);
    printf("%d\n%d\n", a[n2-1],n2);
    return 0;
}
int RemoveSame(int a[], int n)
{
    int flag,sum=0;
    for (int i=0;i<n;i++)
    {
        flag=1;//假设开始没有重复的
        for (int j=0;j<sum;j++)//与已经查到的没重复的一一比较
            if (*(a+i)==*(a+j)) {
                falg=0;
                break;
            }//如果相等，flag 标识为 0 代表有重复的，后面 sum 不累加
```

```
            if (flag) {
            a[sum] = *(a+i);
            sum++;
            }
    }
    return sum;
}
```

3.旋转图像

1)解题思路：

声明一个 n*m 的二维数组，主函数从第 0 行开始，每行输入 m 个数，共输入 n 行。子函数从第 m-1 列的第一个元素开始，依次输出这一列的 n 个元素，注意最后一个元素后应加上回车，再输出前一列直至第 0 列，这样就能实现矩阵逆时针旋转 90 度。

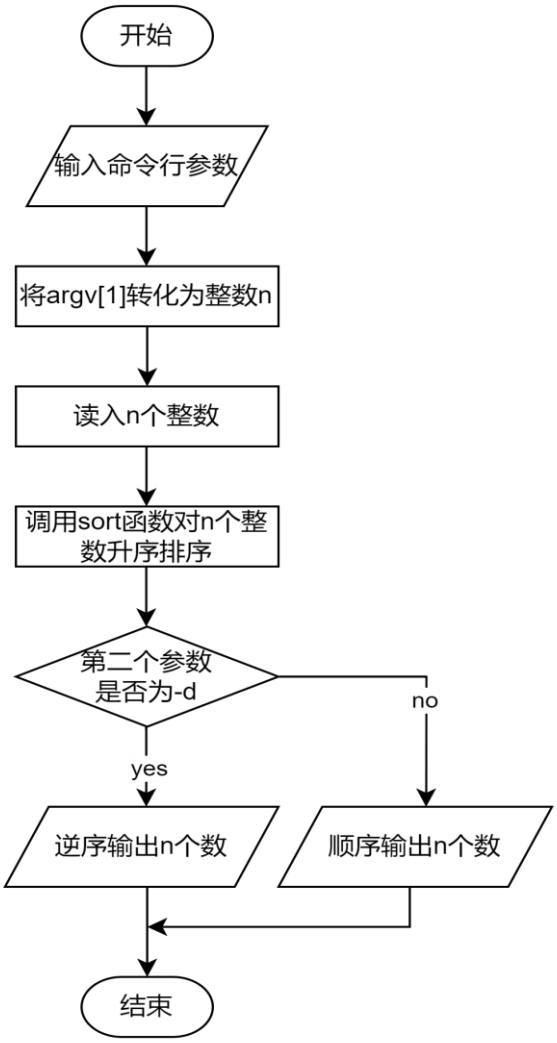
2)源代码：

```
#include <stdio.h>
int main()
{
    int n,m;
    scanf("%d %d",&n,&m);
    int a[n][m];
    int *p=&a[0][0];
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            scanf("%d",p+i*m+j);
        }
    }
    for(int j=m-1;j>=0;j--){
        for(int i=0;i<n;i++){
        if(i!=n-1) printf("%d ",*(p+i*m+j));
        else printf("%d\n",*(p+i*m+j));
        }
    }
    return 0;
}
```

4.命令行实现对 N 个整数排序

1)解题思路：命令行第一个参数为文件名，所以先读取命令行第二个参数

argv[1]并调用 atoi 函数将其转为 int 型整数并赋值给 n，再读入 n 个整数并调用 sort 冒泡排序函数对这 n 个整数进行升序排序。接下来判断命令行是否有-d 参数，有则逆序输出，否则正序输出。



2)源代码：

```
#include <stdio.h>
#include <stdlib.h>
void sort(int *a,int n);
int main(char argc,char *argv[])
{
    int n;
    n=atoi(argv[1]);
    int a[n];
    for(int i=0;i<n;i++) scanf("%d",a+i);
    sort(a,n);
    if(argc==2){
        for(int i=0;i<n;i++) printf("%d ",*(a+i));
```

```
    }
    else if(*argv[2]=='-'){
        for(int i=n-1;i>=0;i--) printf("%d ",*(a+i));
    }
    return 0;
}
void sort(int *a,int n){
    for(int i=0;i<n-1;i++){
        for(int j=0;j<n-i-1;j++){
            if(*(a+j)>*(a+j+1)){
                int temp;
                temp=*(a+j);
                *(a+j)=*(a+j+1);
                *(a+j+1)=temp;
            }
        }
    }
}
```

5.删除子串

1)解题思路：主函数中输入两个字符串，再调用 delsubstr 函数。delsubstr 函数中，先令 flag=0，再调用 strstr 函数查找子串，若 strstr 函数返回的不是空指针 NULL 则说明有子串，令 flag=1。用 char 型指针 p 来保存查找到的子串地址，len 表示子串长度，strcpy(p,p+len)表示将查找到的子串后的字符串移到 p 所指的地址，即为删除子串。注意每一次删除后应使指针 a 移到 p 所指的地址，否则会在新串中继续查找。(输入 aabcbccabc/abc 应输出 abcc 而非 c)

2)源代码：

```
#include <stdio.h>
#include <string.h>
int delsubstr(char *a,const char *b);
int main()
{
    char str[100],substr[100];
    gets(str);
    gets(substr);
    int n=delsubstr(str,substr);
    printf("%s\n%d",str,n);
```

```
    return 0;
 }
int delsubstr(char *a,const char *b)
{
     char *p;
     int flag=0;
     while((p=strstr(a,b))!=NULL){
        int len=strlen(b);
        strcpy(p,p+len);
         flag=1;
         a=p;
     }
     return flag;
 }
```

6. 非负整数积

1)解题思路：先声明三个全局数组(自动初始化为 0)分别保存两个乘数和其积的每位数。读入两个乘数并存入字符串数组中，再将每一位转化为对应的整形数据存入 a1[] a2[]中，注意字符串的首个元素是最高位，所以应逆序存入。a3[i+j]+=a1[i]*a2[j]表示将对应位数的乘积加到 a3[]的相应位，再处理 a3[]的进位问题，最后不加前导 0 逆序输出即可。

2)源代码：

```
#include <stdio.h>
#include <string.h>
int a1[200],a2[200],a3[400];
int main()
{
    char str1[200],str2[200];
    scanf("%s%s",str1,str2);
    int l1=strlen(str1),l2=strlen(str2);
    for(int i=0;i<l1;i++){
        a1[i]=str1[l1-i-1]-'0';
    }
    for(int j=0;j<l2;j++){
        a2[j]=str2[l2-j-1]-'0';
    }
    for(int i=0;i<l1;i++){
```

```
        for(int j=0;j<l2;j++){
            a3[i+j]+=a1[i]*a2[j];
        }
    }
    for(int k=0;k<l1+l2;k++){
        if(a3[k]>9){
            a3[k+1]+=a3[k]/10;
            a3[k]%=10;
        }
    }
    int flag;
    for(int i=l1+l2+1;;i--){
        if(a3[i]){
            flag=i;
            break;
        }
    }
    for(int i=flag;i>=0;i--) printf("%d",a3[i]);
    return 0;
 }
```

7.函数调度

1)解题思路：主函数中读入字符串 s，并将 s 及其长度 len 传给 scheduler 函数。在 scheduler 函数中，声明一个长度为 len 的函数指针数组 p，依次读取字符串 s 的每个元素并将其转为整数 a，令 p[i]指向 task a。最后调用 execute 函数执行 len 个指令。

2)源代码：

```
#include<stdio.h>
#include<string.h>
void task0(void)
{
    printf("task0 is called!\n");
 }
void task1(void)
{
    printf("task1 is called!\n");
 }
 void task2(void)
{
    printf("task2 is called!\n");
 }
```

```
 void task3(void)
{
    printf("task3 is called!\n");
 }
 void task4(void)
{
    printf("task4 is called!\n");
 }
 void task5(void)
{
    printf("task5 is called!\n");
 }
 void task6(void)
{
    printf("task6 is called!\n");
 }
 void task7(void)
{
    printf("task7 is called!\n");
 }


void scheduler(char *s,int n);
void execute(void (*p[])(),int n);
int main()
{
    char s[20];
    scanf("%s",s);
    int len=strlen(s);
    scheduler(s,len);
    return 0;
}
void scheduler(char *s,int len)
{
    void (*p[len])(void);
    int a;
    for(int i=0;i<len;i++){
        a=*(s+i)-'0';
        switch(a){
            case 0:p[i]=task0; break;
            case 1:p[i]=task1; break;
            case 2:p[i]=task2; break;
```

```
            case 3:p[i]=task3; break;
            case 4:p[i]=task4; break;
            case 5:p[i]=task5; break;
            case 6:p[i]=task6; break;
            case 7:p[i]=task7; break;
        }
    }
    execute(p,len);
}
void execute(void (*p[])(void),int len)
{
    for(int i=0;i<len;i++)
    p[i]();
}
```

1.4 小结

通过本次实验，我对指针有了更加深刻的理解，学会了用指针处理数组和字符串的一些问题，也学会了指针数组，指针函数，函数指针，函数指针数组的声明和使用，并且学会了用命令行对 main 函数传递参数。

2 实验 7 结构与联合

2.1 表达式求值的程序验证

```
    char u[]="UVWXYZ";
    char v[]="xyz";
    struct T{
        int x;
        char c;
        char *t;
}a[]={{11,'A',u},{100,'B',v}},*p=a;
```

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)计算过程：

1. ++p 使 p 指向 a[1]，再访问出 a[1]的 x 变量，值为 100
2. p++后有序列点，因此 p 指向 a[1],再访问 a[1]的 c 变量，值为 'B'
3. 前一个表达式先访问出 a[0]中 t 成员所指的字符串的首元素，再使 p 加一指向 a[1]；后一个表达式访问出 a[1]中 t 成员所指的字符串的首元素，为 'x'；而逗号表达式的值与最后一个表达式的值相同，因此值为 'x'
4. ++p 使 p 指向 a[1],而->的优先级高于*,因此先访问 a[1]的 t 成员，再取出 t 成员所指字符串的首元素，值为 'x'
5. p->t 访问 a[0]的 t 成员，*的优先级高，因此先取出 t 成员所指字符串的首元素，为 'U'，再执行自增操作，值为 'V'
6. p->t 访问 a[0]的 t 成员，*的优先级高，因此先取出 t 成员所指字符串的首元素，为 'U'，再执行自增操作，值为 'V'

2)源代码：

```c
#include<stdio.h>
int main()
{
    char u[]="UVWXYZ",v[]="xyz";
    struct T{
        int x;
        char c;
        char *t;
    } a[]={{11,'A',u},{100,'B',v}},*p=a;
    int num;
    scanf("%d",&num);
```

```
    switch(num){
        case 1:printf("%d",(++p)->x); break;
        case 2:printf("%c",(p++,p->c)); break;
        case 3:printf("%c",(*p++->t,*p->t)); break;
        case 4:printf("%c",*(++p)->t); break;
        case 5:printf("%c",*++p->t); break;
        case 6:printf("%c",++*p->t); break;
    }
    return 0;
}
```

2.2 源程序修改替换

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：

```
#include <stdio.h>
#include <stdlib.h>
struct s_list{
    int data;
    struct s_list *next;
};
void create_list(struct s_list *headp,int *p);
int main(void){
    struct s_list *head=NULL,*p;
    // int s[]={1,2,3,4,5,6,7,8,0};
    int s[50];
    int x;
    int i=0;
    do{
        scanf("%d",&x);
        s[i++]=x;
    }while(x);
    create_list(head,s);
    p=head;
    while (p){
```

```
        printf("%d\t",p->data);
        p=p->next;
    }
    printf("\n");
    return 0;
}



void create_list(struct s_list *headp,int *p){
    struct s_list *loc_head=NULL,*tail;
    if(p[0]==0);
    else {
        loc_head=(struct s_list *)malloc(sizeof(struct s_list));
           //建立头结点
        loc_head->data=*p++;//数据存入头结点数据域
        tail=loc_head;//尾指针指向链尾
        while (*p)
        {
            tail->next=(struct s_list *)malloc(sizeof(struct
s_list));       //建立新节点
            tail=tail->next;//尾指针指向链尾
            tail->data=*p++;
        }
        tail->next=NULL;//尾节点指针域置空

    }
    headp=loc_head;
}
```

程序改错：

a) 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);

b) 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);

c) 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)

d） 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;

（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。

```
#include <stdio.h>
#include <stdlib.h>
struct s_list{
    int data;
    struct s_list *next;
};
struct s_list *create_list(int *p);
int main(void){
    struct s_list *head=NULL,*p;
// int s[]={1,2,3,4,5,6,7,8,0};
    int s[50];
    int x;
    int i=0;
    do{
        scanf("%d",&x);
        s[i++]=x;
    }while(x);
    p=create_list(s);
    while (p){
        printf("%d\t",p->data);
        p=p->next;
    }
    printf("\n");
    return 0;
}
/*建立先进后出链表(头插法)*/
struct s_list *create_list(int *p){
    struct s_list *head=NULL,*q;
    while(*p){
        q=(struct s_list *)malloc(sizeof(struct s_list));
        //建立新结点
        q->data=*p;  //数据存入新节点数据域中
        q->next=head;  //新节点指向原首节点
        head=q;  //新节点为首节点
        p++;
```

```
    }
    return head;
}
```

此处建立先进后出链表也可同样通过二级指针参数返回头指针值

2.3 程序设计

1.设计字段结构

1)解题思路：声明一个函数指针数组 p 分别指向 f0 到 f7 八个函数；声明一个长度为一个字节的字段类型 bits;声明一个联合类型 w8,其成员为无符号 char 型数据及 bits 类变量 bit，她们共享一个内存单元。在主函数中声明一个 w8 型变量 w，读入一个无符号 char 型数据并把它赋值给 w，再依次判断 bit 的每个成员是否为 1，是的话则调用相应函数。

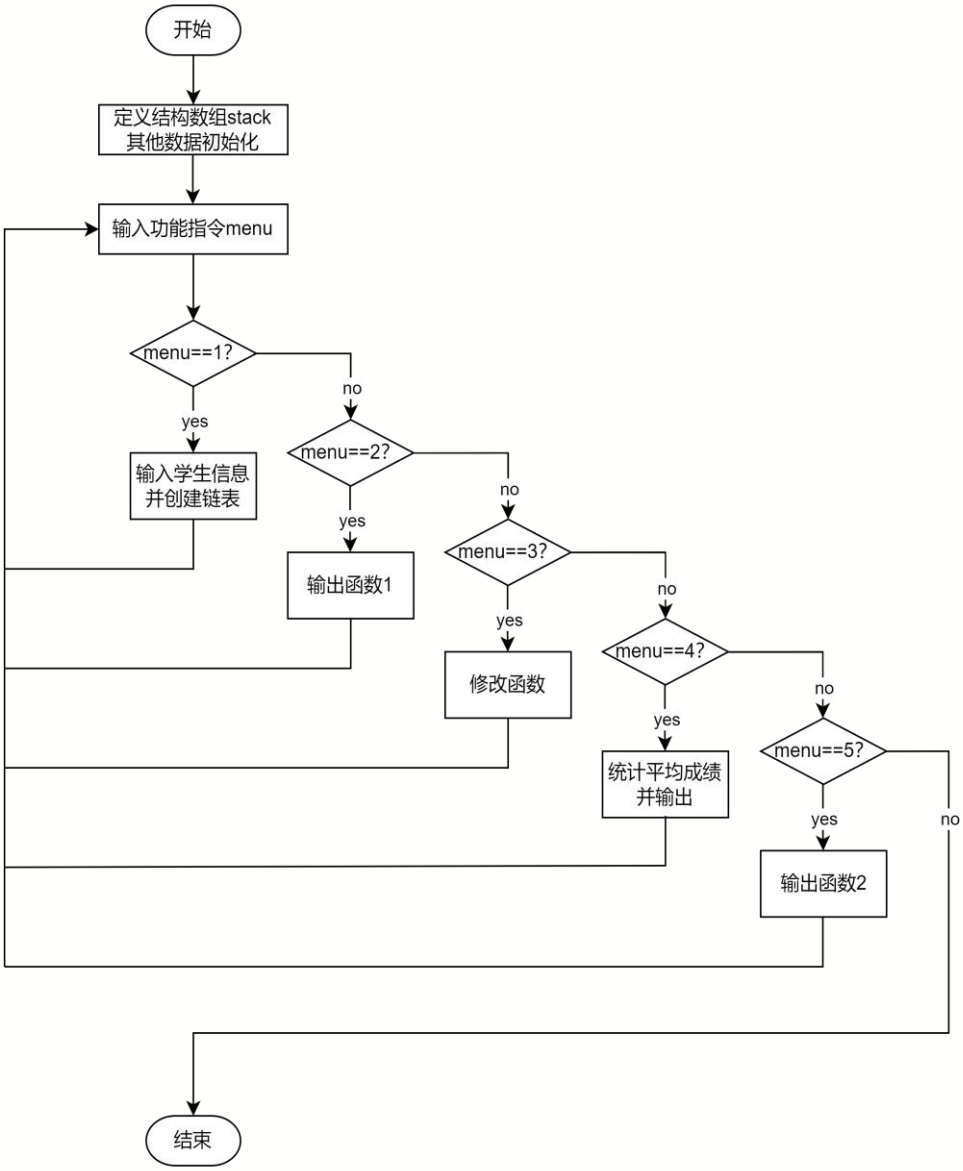
2)源代码：

```
//字段结构
 #include <stdio.h>
 #include <stdlib.h>
 struct bits{
    unsigned char bit0: 1;
    unsigned char bit1: 1;
    unsigned char bit2: 1;
    unsigned char bit3: 1;
    unsigned char bit4: 1;
    unsigned char bit5: 1;
    unsigned char bit6: 1;
    unsigned char bit7: 1;
 };
 union w8{
    unsigned char x;
    struct bits bit;
 };
 void f0(int a){
    printf("the function %d is called!\n",a);
 }
 void f1(int a){
    printf("the function %d is called!\n",a);
 }
 void f2(int a){
    printf("the function %d is called!\n",a);
 }
 void f3(int a){
```

```
    printf("the function %d is called!\n",a);
}
void f4(int a){
    printf("the function %d is called!\n",a);
}
void f5(int a){
    printf("the function %d is called!\n",a);
}
void f6(int a){
    printf("the function %d is called!\n",a);
}
void f7(int a){
    printf("the function %d is called!\n",a);
}
int main()
{
    unsigned char b;
    scanf("%d",&b);
    union w8 w={b};
    void (*p[8])(int a)={f0,f1,f2,f3,f4,f5,f6,f7};
    if(w.bit.bit0==1) p[0](0);
    if(w.bit.bit1==1) p[1](1);
    if(w.bit.bit2==1) p[2](2);
    if(w.bit.bit3==1) p[3](3);
    if(w.bit.bit4==1) p[4](4);
    if(w.bit.bit5==1) p[5](5);
    if(w.bit.bit6==1) p[6](6);
    if(w.bit.bit7==1) p[7](7);
    return 0;
}
```

2.班级成绩单

1)解题思路：声明一个 stu 结构类型存放每个学生的各项信息；主函数中循环读入功能选项并调用相应函数直至 0 结束；用变量 i 保存输入数据的次数，以便创建链表，若 i 为 1，则从首节点开始创建先进先出链表，若 i 大于 1，则在上轮链表的尾指针 tail 后继续插入。



2)源代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct stu{
    char number[20];//学号
    char name[20];//姓名
    int english;//英语成绩
    int math;//数学成绩
    int physics;//物理成绩
    int c_code;//C 语言成绩
    int sum;//总成绩
    double ave;//平均成绩
    struct stu *next;
```

```
};
struct stu *head,*tail;
int n;
void input(int i);
void output1(void);
void modify(void);
void average(void);
void output2(void);
int main()
{
    int menu;
int i = 0; //i 表示第几次输入
    do {
        scanf("%d", &menu);
        switch (menu) {
        case 1: i++; input(i); break;
        case 2: output1(); break;
        case 3: modify(); break;
        case 4: average(); break;
        case 5: output2(); break;
        }
    } while (menu != 0);
    return 0;
}
/*建立先进先出链表*/
void input(int i)
{
    scanf("%d",&n);
    /*若为首次输入，创建头节点*/
    if(i==1)
    {
        head=(struct stu *)malloc(sizeof(struct stu));
        scanf("%s%s%d%d%d%d",head->number,head->name,&head->eng
lish,&head->math,&head->physics,&head->c_code);
        head->sum=head->english+head->math+head->physics+head->
c_code;
        head->ave=1.0*head->sum/4;
        tail=head;
    }
        /*依次向后创建节点*/
        int start;
        if(i==1) start=1; //首次输入
```

```
        else start=0;  //非首次输入，在上轮 tail 指针后继续插入
        for(int j=start;j<n;j++)
        {
            tail->next=(struct stu *)malloc(sizeof(struct stu));
            tail=tail->next;
            scanf("%s%s%d%d%d%d",tail->number,tail->name,&tail-
>english,&tail->math,&tail->physics,&tail->c_code);
            tail->sum=tail->english+tail->math+tail->physics+ta
il->c_code;
            tail->ave=1.0*tail->sum/4;
        }
        tail->next=NULL;
}

void output1(void)
{
    struct stu *p=head;
    while(p!=NULL){
        printf("%s %s %d %d %d %d\n",p->number,p->name,p->engli
sh,p->math,p->physics,p->c_code);
        p=p->next;
    }
}

void modify(void)
{
    char s[20];//被修改学生的学号
    scanf("%s",s);
    int num,grade;//被修改科目及成绩
    scanf("%d%d",&num,&grade);
    struct stu *p=head;
    while(p!=NULL){
        if(strcmp(s,p->number)==0) break;
        p=p->next;
    }
    switch(num){
        case 1: p->english=grade; break;
        case 2: p->math=grade; break;
        case 3: p->physics=grade; break;
        case 4: p->c_code=grade; break;
    }
    /*修改成绩后应重新计算总成绩和平均成绩*/
```

```
    p->sum=p->english+p->math+p->physics+p->c_code;
    p->ave=1.0*p->sum/4;
}

void average(void)
{
    struct stu *p=head;
    while(p!=NULL){
        printf("%s %s %.2lf\n",p->number,p->name,p->ave);
        p=p->next;
    }
}

void output2(void)
{
    struct stu *p=head;
    while(p!=NULL){
        printf("%s %s %d %.2lf\n",p->number,p->name,p->sum,p->a
ve);
        p=p->next;
    }
}
```

3. 成绩排序(一)

1)解题思路：为了方便排序，用 total 变量记录总人数；为了方便交换节点数据域，将学生的信息在再声明为一个 information 结构类型，stu 中采用嵌套的结构类型；sort 函数用于按平均成绩进行冒泡排序。

2)源代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
/*每位学生的信息*/
struct information {
    char number[20];
    char name[20];
    int english;
    int math;
    int physics;
    int c_code;
    int sum;
    double ave;
```

```
};
struct stu {
    struct information data;
    struct stu* next;
};
struct stu* head, * tail;
int n, total = 0;
void input(int i);
void output1(void);
void modify(void);
void average(void);
void output2(void);
void sort(void);
int main()
{
    int menu;
    int i = 0;
    do {
        scanf("%d", &menu);
        switch (menu) {
        case 1: i++; input(i); break;
        case 2: output1(); break;
        case 3: modify(); break;
        case 4: average(); break;
        case 5: output2(); break;
        }
    } while (menu != 0);
    return 0;
}

/*建立先进先出链表*/
void input(int i)
{
    scanf("%d",&n);
    total+=n;
    /*若为首次输入，则用尾插法创建链表*/
    if (i==1)
    {
        /*创建头节点*/
        head=(struct stu*)malloc(sizeof(struct stu));
```

```
        scanf("%s%s%d%d%d%d",head->data.number,head->data.name,
&head->data.english,&head->data.math,&head->data.physics,&head
->data.c_code);

        head->data.sum = head->data.english + head->data.math +
head->data.physics + head->data.c_code;

        head->data.ave = 1.0 * head->data.sum / 4;

        tail=head;
    }
        int start;
        if(i==1) start=1;
        else start =0;
         for (int j=start;j<n;j++)
        {
            tail->next=(struct stu*)malloc(sizeof(struct stu));

            tail=tail->next;

            scanf("%s%s%d%d%d%d",tail->data.number,tail->data.n
ame,&tail->data.english,&tail->data.math,&tail->data.physics,&
tail->data.c_code);

            tail->data.sum = tail->data.english + tail->data.math
+ tail->data.physics + tail->data.c_code;

            tail->data.ave = 1.0 * tail->data.sum / 4;
        }
        tail->next=NULL;
    /*调用排序函数，按平均成绩升序排序*/
    sort();
}

void output1(void)
{
    struct stu* p = head;
    while (p != NULL) {
        printf("%s %s %d %d %d %d\n", p->data.number, p->data.name,
p->data.english, p->data.math, p->data.physics, p->data.c_code);
        p = p->next;
    }
```

```
}

void modify(void)
{
    char s[20];
    scanf("%s", s);
    int num, grade;
    scanf("%d%d", &num, &grade);
    struct stu* p = head;
    while (p != NULL) {
        if (strcmp(s, p->data.number) == 0) break;
        p = p->next;
    }
    switch (num) {
    case 1: p->data.english = grade; break;
    case 2: p->data.math = grade; break;
    case 3: p->data.physics = grade; break;
    case 4: p->data.c_code = grade; break;
    }
    /*修改成绩后应重新计算总成绩和平均成绩并排序*/
    p->data.sum = p->data.english + p->data.math + p->data.physics
+ p->data.c_code;
    p->data.ave = 1.0 * p->data.sum / 4;
    sort();
}

void average(void)
{
    struct stu* p = head;
    while (p != NULL) {
        printf("%s %s %.2lf\n", p->data.number, p->data.name,
p->data.ave);
        p = p->next;
    }
}

void output2(void)
{
    struct stu* p = head;
    while (p != NULL) {
        printf("%s %s %d %.2lf\n", p->data.number, p->data.name,
p->data.sum, p->data.ave);
```

```
            p = p->next;
        }
    }

/*冒泡排序：按平均成绩升序排序*/
void sort(void)
{
    /*total 表示总人数*/
    struct stu* p = head;
    struct information temp;
    for (int i = 1; i < total; i++) {
        p = head;       //共进行 total-1 轮，每一轮 p 应从 head 开始
        for (int j = 0; j < total - i; j++) {
            if (p->data.ave > p->next->data.ave) {
                temp = p->data;
                p->data = p->next->data;
                p->next->data = temp;
            }
            p = p->next;
        }
    }
}
```

4. 回文字符串

1)解题思路：分别建立一个先进先出链表和一个先进后出链表，然后逐个输出链表内容并比较。

2)源代码：

```
#include <stdlib.h>
#include <stdio.h>
/*定义节点*/
typedef struct c_node{
    char data;
    struct c_node *next;
} C_NODE;
void createLinkList(C_NODE **headp,C_NODE **headp1,char s[]);
void judgePalindrome(C_NODE *head,C_NODE *head1);
int main(){
    char s[100];
    scanf("%s",s);
    C_NODE *head=NULL,*head1=NULL;
    createLinkList(&head,&head1,s);
```

```
    judgePalindrome(head,head1);
    return 0;
}
void createLinkList(C_NODE **headp,C_NODE **headp1,char s[])
{
    /*建立先进先出链表，头指针 loc_head 传给 head*/
    char *p=s;
    C_NODE *loc_head=NULL,*tail;
    loc_head=(C_NODE *)malloc(sizeof(C_NODE));
    loc_head->data=*p++;
    tail=loc_head;
    while (*p!='\0'){
        tail->next=(C_NODE *)malloc(sizeof(C_NODE));
        tail=tail->next;
        tail->data=*p++;
    }
    tail->next=NULL;
    *headp=loc_head;

    /*建立先进后出链表，头指针 loc_head1 传给 head1*/
    p=s;
    C_NODE *loc_head1=NULL,*q;
    while(*p!='\0'){
        q=(C_NODE *)malloc(sizeof(C_NODE));
        q->data=*p++;
        q->next=loc_head1;
        loc_head1=q;
    }
    *headp1=loc_head1;
}
void judgePalindrome(C_NODE *head,C_NODE *head1)
{
    C_NODE *m,*n;
    int flag=1;
    for(m=head,n=head1;m!=NULL&&n!=NULL;m=m->next,n=n->next){
        if(m->data!=n->data){
            flag=0;
            break;
        }
    }
    if(flag==1) printf("true");
    else printf("false");
```

```
}
```

5.成绩排序(二)

1)解题思路：其余代码与成绩排序(一)相同，只需将 sort 函数修改为交换指针域的排序方法。而头节点前没有节点，因此每轮循环应单独判断头节点与下一个节点相应数据的大小

2)源代码：

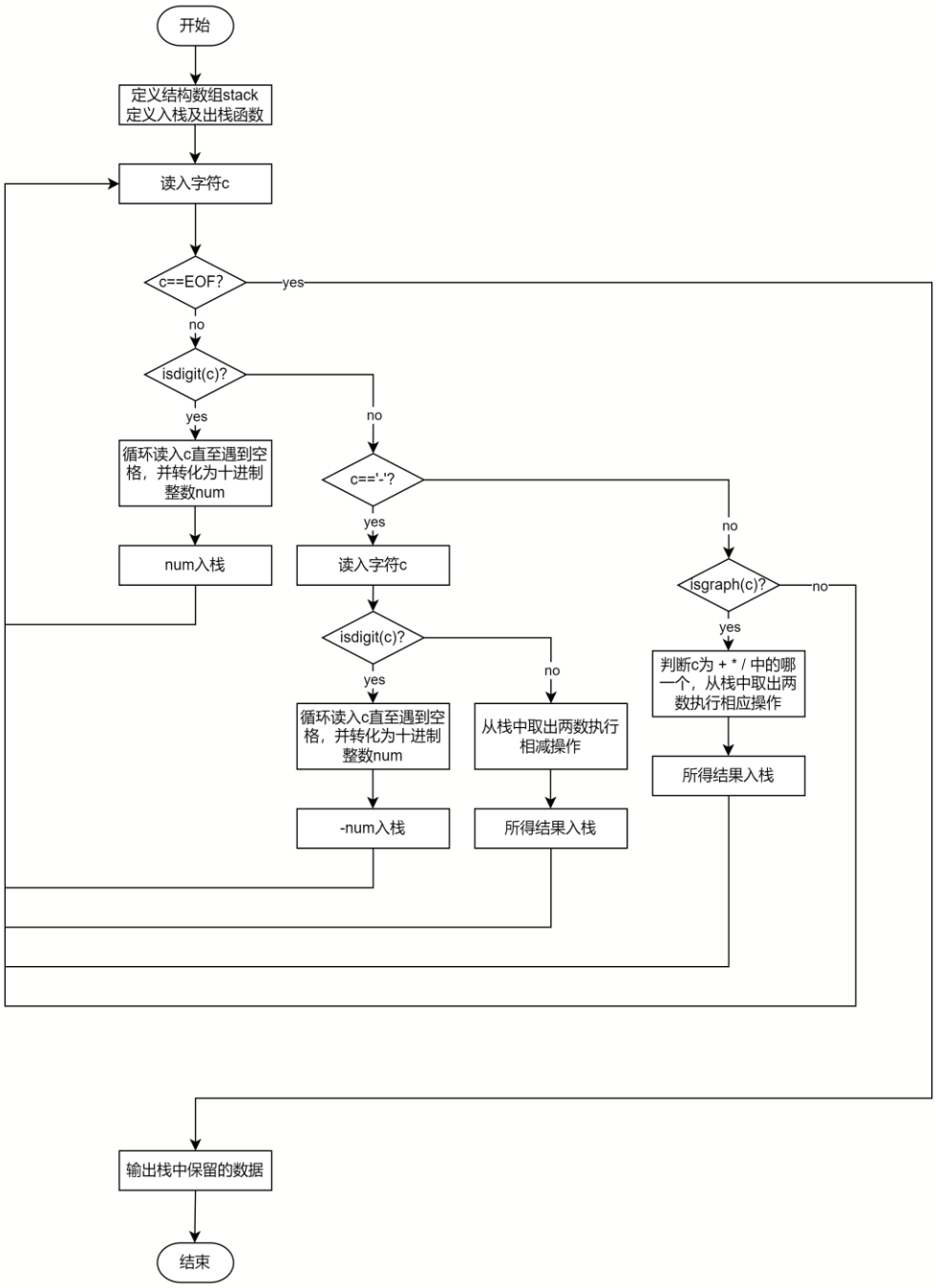
```
/*交换指针域的冒泡排序*/
void sort(void)
{
    struct stu *p,*last,*q;
    for (int i = 1; i < total; i++) {
        /*头节点特殊，应单独判断*/
        if(head->data.ave > head->next->data.ave ){
            q=head->next;
            head->next=head->next->next;
            q->next=head;
            head=q;
        }
            p=head;
           //共进行 total-1 轮,每一轮 p 应从 head 开始
        for (int j = 1; j < total - i; j++) {
            last=p;
            p=p->next;
            if(p->data.ave > p->next->data.ave){
                last->next=p->next;
                p->next=p->next->next;
                last->next->next=p; //交换节点
            }
            p=last->next;//p 回退一格
        }
    }
}
```

6. 逆波兰表达式

1)解题思路：本题借助值栈求解。依次检索输入的字符串，若为数字，则遇到空格停止并将其转化为十进制整数存入值栈（负数应特殊判断）；若检索到的为+ - * / 四个运算符，则从值栈中取出两个数字进行相应运算并将所得结果存入值栈；直至检索到换行符程序结束，此时值栈中只剩一个数，该数为所求结果。

值栈可用先进后出链表实现，入栈即为创建节点并使这个节点成为首节点，出栈即为取出头节点数据域并使下一个节点成为首节点同时释放原首节点。

值栈也可用线性数组模拟，以下代码给出两种不同解法。



2)源代码：

```
/*利用先进后出链表求解逆波兰表达式*/
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
struct stack{
    int data;
    struct stack *next;
```

```
};
struct stack *head=NULL;
/*入栈*/
void push(int x)
{
    struct stack *p=(struct stack *)malloc(sizeof(struct stack));
    p->data=x;
    p->next=head;//指向原首节点
    head=p;//新节点成为首节点
}
/*出栈*/
int out(void)
{
    int x=head->data;
    struct stack *p=head;
    head=head->next;//回退一格
    free(p);
    return x;
}
int main()
{
    int x,num,temp;
    char c;
    while((c=getchar())!=EOF)
    {
        /*检索到数字*/
        if(isdigit(c))
        {
            num=0;
            do{
                num=num*10+c-'0';
                c=getchar();
            }while(c!=' ');
            push(num);//转化为十进制整数并入栈
        }
        /*检索到‘-’*/
        else if(c=='-')
        {
            c=getchar();
            /*若‘-’后为数字，则表示一个负数*/
            if(isdigit(c))
            {
```

```
                num=0;
                do{
                    num=num*10+c-'0';
                    c=getchar();
                }while(c!=' ');
                push(0-num);
            }
            /*否则表示减号,执行相减操作*/
            else
            {
                temp=out();
                push(out()-temp);
            }
        }
        /*检索到除数字和‘-’外的其他非空格字符*/
        else if(isgraph(c))
        {
            switch(c)
            {
                case '+': push(out()+out()); break;
                case '*': push(out()*out()); break;
                case '/': temp=out();
                          push(out()/temp);
                          break;
            }
        }
    }
    printf("%d",out());
    return 0;
}

/*利用线性数组求解逆波兰表达式*/
/*只需将入栈和出栈操作用数组模拟即可*/
int stack[100];
int *top=stack;
/*入栈*/
void push(int x)
{
    *top++=x;
}
/*出栈*/
int out(void)
```

```
{
    return *--top;
}
```

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。

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