華中科技大學

课 程 实 验 报 告

课程名称： C 语言程序设计实验

专业班级：*********

学　　号：**********

姓　　名：EpsilonWin

指导教师：

报告日期：*********

计算机科学与技术学院

目 录

1 实验 6 指针程序设计实验

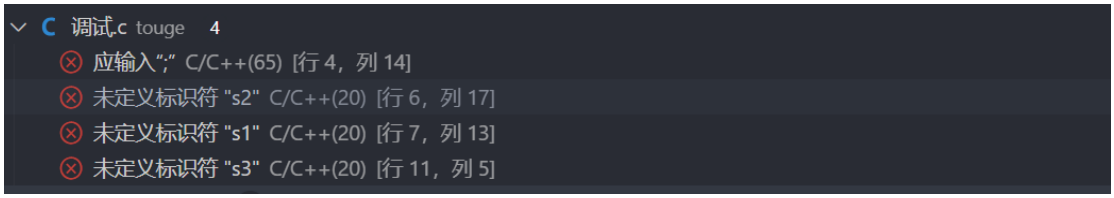
1.1 程序改错与跟踪调试

```
#include<stdio.h>
char* strcopy(char*, const char*);
int main(void) {
    char *s1,  *s2, *s3;
    printf("Input a string:\n");
    scanf("%s", s2);
    strcopy(s1, s2);
    printf("%s\n", s1);
    printf("Input a string again:\n");
    scanf("%s", s2);
    s3 = strcopy(s1, s2);
    printf("%s\n", s3);
    return 0;
}

char* strcopy(char* t, const char* s)
{
    while (*t++=*s++);
    return (t);
}
```

1. Line 4:字符串指针 s1,s2,s3 未初始化，为悬挂指针，无法进行后续的粘贴等操作。
   解决方案：使用 malloc 函数申请储存空间。引用 stdlib.h 库，将申请的空间赋值给三个指针。
2. Line 19:返回的指针 t 储存的地址不是原来字符串的首地址，导致在后面输出结果的时候无法输出正确的字符串。
   解决方案：引入第三个指针，将其赋值为 t 储存的地址，并用这个新的指针的偏移来进行字符串的复制。

调试时的报错：



改正方案：

```c
#include<stdio.h>
#include<stdlib.h>

char* strcopy(char*, const char*);

int main(void) {
    char* s1;
    char* s2;
    char* s3;

    s1 = (char*)malloc(sizeof(char) * 20);
    s2 = (char*)malloc(sizeof(char) * 20);

    printf("Input a string:\n");
    scanf("%s", s2);
    strcopy(s1, s2);
    printf("%s\n", s1);

    printf("Input a string again:\n");
    scanf("%s", s2);
    s3 = strcopy(s1, s2);
    printf("%s\n", s3);

    free(s1);
    free(s2);

    return 0;
}

char* strcopy(char* t, const char* s) {
    char* p = t;
    while (*p++=*s++);
    return (t);
}
```


改正的方案的结果：

```
Input a string:
programming
programming
Input a string again:
language
language
```

1.2 程序完善与修改替换

1.2.1 字符串排序

（1）程序完善：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define N 4
#define MAXLINE 100
/* 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;
    char* s[N], t[50];
    for (i = 0; i < N; i++)
    {
        gets(t);
        s[i] = (char*)malloc(2000);
        strcpy(s[i],t);
    }
    strsort(s,N);
    for (i = 0; i < N; i++)  puts(s[i]);
    return 0;
}
```

输入与输出结果：

```
Python
Java
Golang
C
C
Golang
Java
Python
```

若改为用户可指定输入的数量：

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>


#define N 4
#undef N
#define MAXLINE 100
/* 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 N;
    scanf("%d", &N);
    getchar();
    int i;
    char* s[MAXLINE], t[50];
    for (i = 0; i < N; i++)
    {
        gets(t);
        s[i] = (char*)malloc(2000);
        strcpy(s[i],t);
```

```
    }
    strsort(s,N);
    for (i = 0; i < N; i++)  puts(s[i]);
    return 0;
}
```

输入与输出结果：

```
3
C
Python
Java
C
Java
Python
```

（2）替换方案：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define MAXLINE 1000

void strsort(char** p, int n)
{
    char* p0;
    for (int i = 0; i < n-1; i++)
    {
        for (int j = 0; j < n-i-1; j++)
        {
            if (strcmp(*(p+j),*(p+j+1))>0)
            {
                p0 = *(p + j);
                *(p + j) = *(p + j + 1);
                *(p + j + 1) = p0;
            }

        }
    }
}

int main()
{

    int N;
    scanf("%d", &N);
```

```
    getchar();
    int i;
    char* s[MAXLINE], t[50];
    for (i = 0; i < N; i++)
    {
        gets(t);
        s[i] = (char*)malloc(2000);
        strcpy(s[i], t);
    }
    strsort(s, N);
    for (i = 0; i < N; i++)  puts(s[i]);

    return 0;
}
```

main 函数内仍用数组，但是在排序函数中使用二级指针进行排序，加快程序运行效率。

输入与输出结果：

```
3
Golang
Python
Java
Golang
Java
Python
```

1.2.2 函数指针

（1）程序完善：

```
#include<stdio.h>
#include<string.h>

int main(void)
{
    char* (*p)(char*, 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:
            p = NULL;
            goto down;
        default:
            p = NULL;
            break;
        }
        getchar();
        printf("input the first string please!\n");
        scanf("%s", a);
        printf("input the second string please!\n");
        scanf("%s", b);
        result = p(a, b);
        printf("the result is %s\n", result);
    }
down:
    return 0;
}
```

输入与输出结果：

```
               1 copy string.
               2 connect string.
               3 Parse string.
               4 exit.
               input a number (1-4) please!
1
input the first string please!
the more you learn,
input the second string please!
the result is more
               1 copy string.
               2 connect string.
               3 Parse string.
               4 exit.
               input a number (1-4) please!
input the first string please!
input the second string please!
the result is learn,
               1 copy string.
               2 connect string.
               3 Parse string.
               4 exit.
               input a number (1-4) please!
the more you get.
input the first string please!
input the second string please!
the result is more
               1 copy string.
               2 connect string.
               3 Parse string.
```

```
               4 exit.
               input a number (1-4) please!
input the first string please!
input the second string please!
the result is get.
               1 copy string.
               2 connect string.
               3 Parse string.
               4 exit.
               input a number (1-4) please!
2
input the first string please!
the more you learn,
input the second string please!
the result is themore
               1 copy string.
               2 connect string.
               3 Parse string.
               4 exit.
               input a number (1-4) please!
input the first string please!
input the second string please!
the result is oulearn,
               1 copy string.
               2 connect string.
               3 Parse string.
               4 exit.
               input a number (1-4) please!
the more you get.
input the first string please!
```

```
input the second string please!
the result is hemore
               1 copy string.
               2 connect string.
               3 Parse string.
               4 exit.
               input a number (1-4) please!
input the first string please!
input the second string please!
the result is ouget.
               1 copy string.
               2 connect string.
               3 Parse string.
               4 exit.
               input a number (1-4) please!
3
input the first string please!
www.educoder.net
input the second string please!
.
the result is www
               1 copy string.
               2 connect string.
               3 Parse string.
               4 exit.
               input a number (1-4) please!
4
```


（2）替换方案：

```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

void DeleteBackspace(char* s);

int main()
{
    char* s1 = (char*)malloc(sizeof(char) * 50);
    char* s2 = (char*)malloc(sizeof(char) * 50);
    char* s3;
    int function;

    char* (*functab[])(char*, char*) =
{ NULL,strcpy,strcat,strtok,NULL };
```

```
//使用转移表，用指向函数的指针数组储存函数地址并进行调用

    while (scanf("%d", &function) && function != 4)
    {
        getchar();
        fgets(s1, 50, stdin);
        fgets(s2, 50, stdin);
        //从标准输入流中读取字符串

        DeleteBackspace(s1);
        DeleteBackspace(s2);
        //删除由于调用 fgets 而读取的字符串尾部的回车

        s3 = functab[function](s1, s2);
        //利用指针调用函数

        puts(s3);
    }

    free(s1);
    free(s2);

    return 0;

}

void DeleteBackspace(char*
{
    int i = 0;
    while (*(s + i) != '\n')
    {
        i++;
    }
    *(s + i) = '\0';
}
```


输入与输出结果：

```
1
the more you learn,
the more you get.
the more you get.
2
the more you learn,
the more you get.
the more you learn,the more you get.
3
www.educoder.net
.
www
4
```

1.3 程序设计

1.3.1 指针取出每个字节

（1）解题思路：

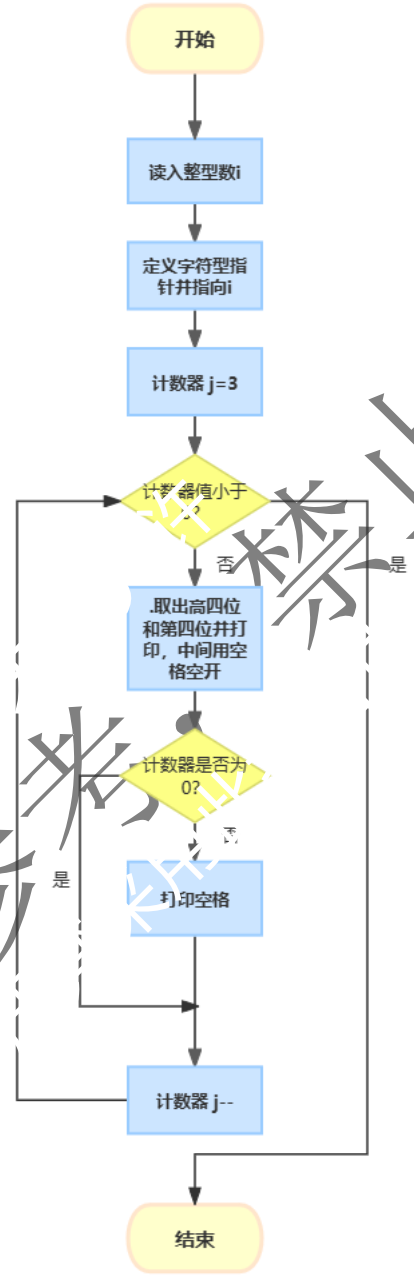


图 1-1 程序设计题 1-1 的流程图

（2）程序源代码

```
#include<stdio.h>

int main(void)
{
```

```
    int i = 0;
    char* p;

    scanf("%d", &i);
    p = (char *)&i;

    for (int j = 3; j >= 0; j--)
    {
        printf("%x %x", *(p+j)>> 4 & 0xf, *(p+j) & 0xf);
        //用指针间接访问后移位运算和与运算来取高位与低位
        if (j!=0)//最后一个字节尾部不需要加空格
        {
            printf(" ");
        }
    }

    return 0;
}
```

1.3.2 删除重复元素

（1）解题思路：

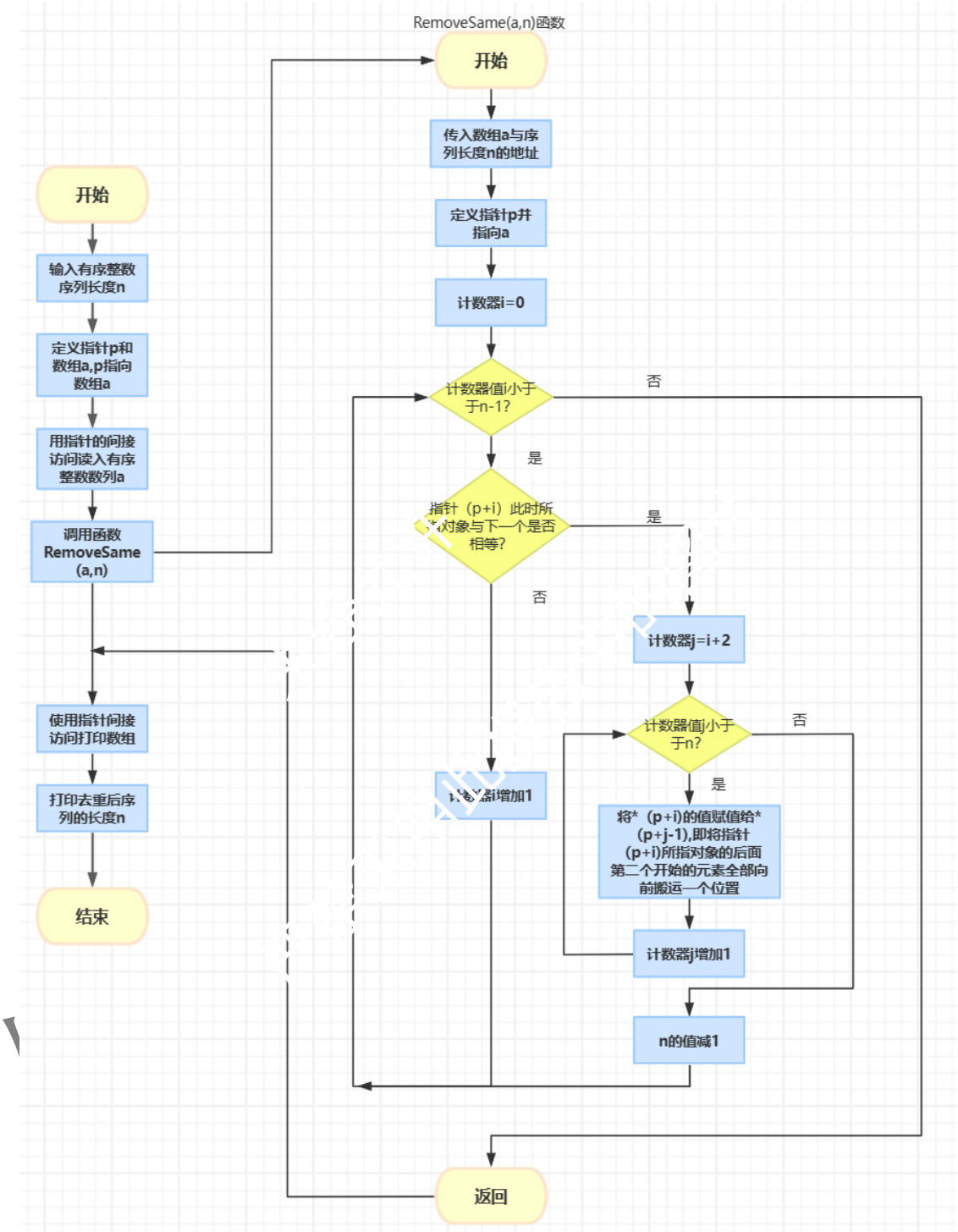


图 1-2 程序设计题 1-2 的流程图

（2）程序源代码：

```
#include<stdio.h>
```

```
void RemoveSame(int* a, int* n)
{
    int* p = a;
    int i = 0;
    int j = 0;
    while (i<(*n)-1)
    {

        //因为序列有序，故判断相邻两个元素是否相等
        //相等元素必相邻
        if (*(p+i+1)!=*(p+i))
        {
            i++;
        }//不相等则指向下一个元素
         //相等则将后面第二个开始的元素都前移一位
        else
        {
            for(j = i + 2; j < *n); j++)
            {
                *(p + j - 1) = *(p + j);
            }
            (*n)--;//前移后数组长度减短
        }
    }
    return;
}

int main(void)
{
    int n;
    scanf("%d", &n);

    int a[20];
    int* p;
    p = a;
    for (int i = 0; i < n; i++)
    {
        scanf("%d", p + i);
    }

    RemoveSame(a,&n);
```

```
    for (int i = 0; i < n; i++)
    {
        printf("%d", *(p + i));
        if (i!=n-1)//若不是最后一个元素，则在元素后方打印空格
        {
            printf(" ");
        }
    }
    printf("\n");


    printf("%d", n);

    return 0;
}
```

1.3.3 旋转图像

（1） 解题思路：

首先定义结构体类型 Matrix,将所有有关数据全部储存在结构体中，便于运算。Matrix 结构中有四个成员，第一个是二维数组 before，用来存放输入的矩阵图像;第二个成员是二维数组 after,用来存放运算后的结果;第三个成员是 hang,用来记录输入的矩阵的行数；第四个成员是 lie,用来储存输入的矩阵的列数。

图 1-3 程序设计题 1-3 的流程图

（2） 程序源代码：

```
#include<stdio.h>

#define MAXLINE 50

typedef struct
{
    int before[MAXLINE][MAXLINE];
    int after[MAXLINE][MAXLINE];
    int hang;
    int lie;
}Matrix;

void CreateMatrix(Matrix* m)//输入矩阵
```

```
{
    for (int i = 0; i < m->hang; i++)
    {
        for (int j = 0; j < m->lie; j++)
        {
            scanf("%d", &(m->before[i][j]));
        }
    }
    return;
}

void TurnMatrix(Matrix* m)//旋转矩阵并储存
{
    for (int i = 0; i < m->hang; i++)
    {
        for (int j = 0; j < m->lie; j++)
        {
            m->after[m->lie - j - 1][i] = m->before[i][j];
        }
    }
}

void PrintAfMatrix(Matrix* m)//打印
{
    for (int i = 0; i < m->lie; i++)
    {
        for (int j = 0; j < m->hang; j++)
        {
            printf("%d", m->after[i][j]);
            if (j!=m->hang-1)
            {
                printf(" ");
            }
        }
        printf("\n");
    }
}

int main(void)
{
    Matrix m;
    scanf("%d%d", &m.hang, &m.lie);
```

```
    CreateMatrix(&m);
    TurnMatrix(&m);

    PrintAfMatrix(&m);

    return 0;
}
```

1.3.4 命令行实现对 N 个整数排序

（1） 解题思路：

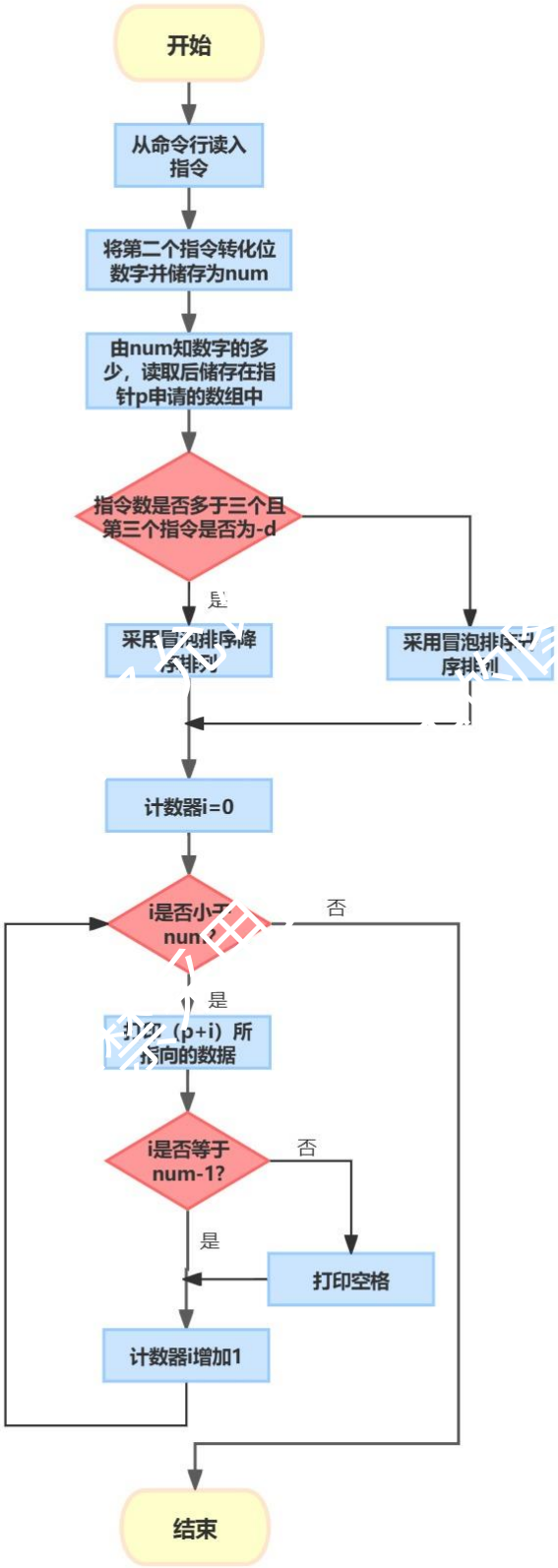


图 1-4 程序设计题 1-4 的流程图

（2） 程序源代码：

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<stdlib.h>

int main(int argc, char *argv[])
{
    int num = atoi(argv[1]);
    //将第二个命令转化为数字

    int* p;
    p = (int*)malloc(sizeof(int) * num);
    //为要排序的数字申请空间

    for (int i = 0; i < num; i++)
    {
        scanf("%d", p + i);
    }
    //读取要排序的数字

    //判断是否由-d 来决定升序还是降序
    if (argc>=3 && strcmp(argv[2], "-d") == 0)
    {
       //冒泡排序
        for (int i = 0; i < num - 1; i++)
        {
            for (int j = 0; j < num - i - 1; j++)
            {
                if (*(p + j) < *(p + j + 1))
                {
                    int t = *(p + j);
                    *(p + j) = *(p + j + 1);
                    *(p + j + 1) = t;
                }
            }
        }
    }
    else
    {
```

```
        //冒泡排序
        for (int i = 0; i < num - 1; i++)
        {
            for (int j = 0; j < num - i - 1; j++)
            {
                if (*(p + j) > *(p + j + 1))
                {
                    int t = *(p + j);
                    *(p + j) = *(p + j + 1);
                    *(p + j + 1) = t;
                }
            }
        }
    }

    //打印数组
    for (int i = 0; i < num; i++)
    {
        printf("%d", *(p + i));
        if (i != num - 1)
        {
            printf(" ");
        }
    }

    return 0;
}
```

1.3.5 删除子串

（1）解题思路：

读取 str 字符串，先将其中检测到的 substr 全部替换为换行符。然后遍历数组，若检测到换行符，将其删除。最后输出字符串。

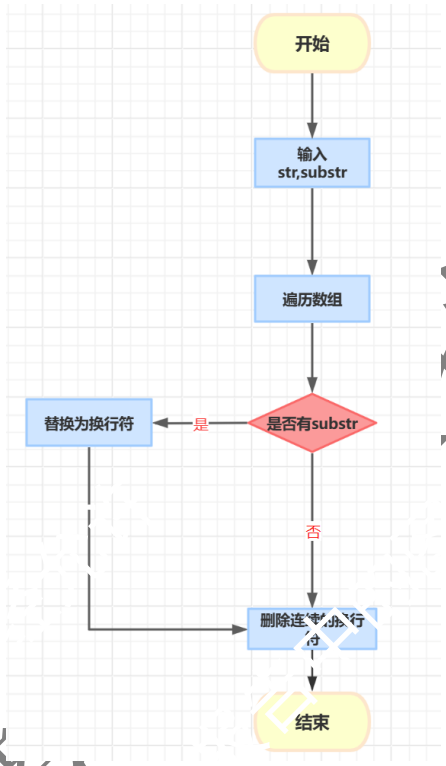


图 1-5 程序设计题 1-3 的流程图

（2）程序源代码：

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>

#define MAX 50

//判断是否是换行符
int isenter(char c)
{
    if (c=='\n')
    {
        return 1;
    }
    else
```

```
    {
        return 0;
    }
}

//计算 str 字符串中换行符的数目便于后续去除时控制循环
int CountEnter(char* str)
{
    int i = 0;
    int cnt = 0;
    while (isenter(*(str+i)))
    {
        cnt++;
        i++;
       //若换行符的下一个为'\0',则不计入，便于后续输出
        if (*(str+i)=='\0')
        {
            cnt--;
        }
    }
    return cnt;
}


int delSubstr(char* str, char* substr)
{
    int len = strlen(substr);
    char* pos ;

    //如果 str 字符串中不存在 substr 字符串，则函数返回 0
    if (strstr(str,substr)==NULL)
    {
        return 0;
    }
    char* p;

    //当 str 字符串中还有 substr 字符串时继续进行替换为换行符的操作
    while ((pos=strstr(str,substr))!=NULL)
    {
        for (p=pos;p - pos < len; p++)
        {
            *p = '\n';
```

```
        }
    }

    //遍历字符串，当 str+i 指向'\0'时停止遍历
    for (int i = 0; *(str+i)!='\0'; i++)
    {
        //判断是否是连续回车，若为连续回车，则说明是要替换的位置
        if (isenter(*(str+i))&&isenter(*(str+i+1)))
        {
            //计算要删除的回车数目
            int m = CountEnter(str + i);
            //通过元素前移覆盖来删除回车
            for ( p = str+i; p-str < strlen(str); p++)
            {
                *p = *(p + m);
            }
        }
    }
    return 1;
}

int main(void)
{
    char str[MAX];
    char substr[MAX];
    int ret = 0;

    //用 fgets 从标准输入读取字符串
    fgets(str, 50, stdin);
    //读取 substr 字符串
    gets(substr);

    //储存是否删除的信息
    ret = delSubstr(str, substr);

    printf("%s", str);
    printf("%d", ret);

    return 0;
}
```

1.3.6 非负整数积

（1）解题思路：

先定义一个结构体 number,其第一个成员为数组 num,用来储存输入的数字与输出的数字；其第二个成员为 length,用来储存数字的位数。

通过函数实现超大数字的输入与输出。输入后，所有数字先右对齐。然后采用分治策略，每次去除乘数的一位，对被乘数的每一位进行乘法运算，并与结果的对应位置相加后储存。最后将结果的每一位进行判断，若有进位，则将进位加到前一位上。

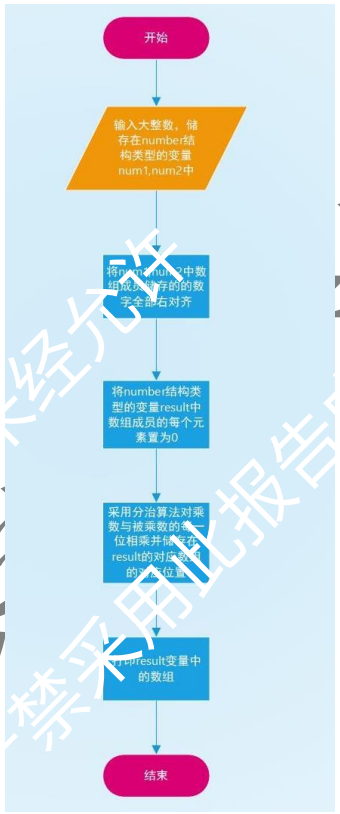


图 1-6 程序设计题 1-6 的流程图

（2）程序源代码：

```
#include<stdio.h>

#define MAX 500

typedef struct
{
    int num[MAX];
```

```
    int length;
}number;

void CreateNum(number* num);
void shift(number* num);
void Initialize(number* result);
void PrintResult(number* result);
void multiply(number* num1, number* num2, number* result);
int judge(int* n, int b);

int main(void)
{
    number num1;
    number num2;
    number result;

    CreateNum(&num1);
    CreateNum(&num2);

    shift(&num1);
    shift(&num2);

    Initialize(&result);

    multiply(&num1, &num2, &result);

    PrintResult(&result);


    return 0;
}

//读取数字并储存数字的位数
void CreateNum(number *num)
{
    char c[MAX];
    num->length = 0;
    int i = 0;
    gets(c);
    while (c[i]!='\0')
    {
        num->num[i] = (int)(c[i] - '0');
```

```
        i++;
        num->length++;
    }
    return;
}

//将所有数字进行右对齐
void shift(number* num)
{
    //计算右对齐每个数字所要移动的位数
    int sft = MAX - num->length;
    for (int i = num->length-1; i >= 0; i--)
    {
        num->num[i + sft] = num->num[i];
    }

//将不用来储存数据的位置全部置为0
    for (int j = 0; j < MAX-num->length; j++)
    {
        num->num[j] = 0;
    }
    return;
}

//将储存结果的数组初始化，所有位置全部置为0
void Initialize(number* result)
{
    for (int i = 0; i < MAX; i++)
    {
        result->num[i] = 0;
    }
}

//对 num1 和 num2 进行乘法并储存在 result 中
void multiply(number* num1, number* num2, number* result)
{
    result->length = 0;
    int weishu = 0;
    for (int i = MAX-1; i >= MAX-num2->length; i--)
    {
        for (int j = MAX-1; j >= MAX-num1->length; j--)
        {
```

```
            //固定乘数的位数，此位对被乘数进行乘法运算并储存
            result->num[MAX-((MAX - j) + (MAX - i) - 1)] +=
num2->num[i] * num1->num[j];
            //判断当前乘完后的位数是否超出了之前记录的最大位数
            if (judge(&weishu, (MAX - j) + (MAX - i) - 1))
            {
                result->length++;
            }
        }
    }

    int j = MAX - 1;
    //取出每位，判断是否有进位，有进位则进位
    while (j>MAX-result->length)
    {
        int tens = result->num[j] / 10;
        int z = result->num[j] % 10;
        result->num[j] = z;
        result->num[j - 1] += tens;
        j--;
    }
    //判断最高位是否要进位，要进位则 result 中记录的长度增加 1 并进位
    if (result->num[MAX-result->length]>=10)
    {
        result->num[MAX - result->length - 1] = result->num[MAX -
result->length] / 10;
        result->num[MAX - result->length] %= 10;
        result->length++;
    }
    return;
}

//打印结果
void PrintResult(number* result)
{
    for (int i = MAX - result->length; i <= MAX-1; i++)
    {
        printf("%d", result->num[i]);
    }
}

//判断现在的位数是否超出记录的最大位数
```

```
//若超出，则记录其位新的最大位数，并返回 1；反之返回 0；
int judge(int* n, int b)
{
    if (b<=*n)
    {
        return 0;
    }
    if (b>*n)
    {
        *n = b;
        return 1;
    }
}
```

1.3.7 函数调度

（1）解题思路：

使用指针读取字符来调用函数，利用 switch-case 语句来调用函数。

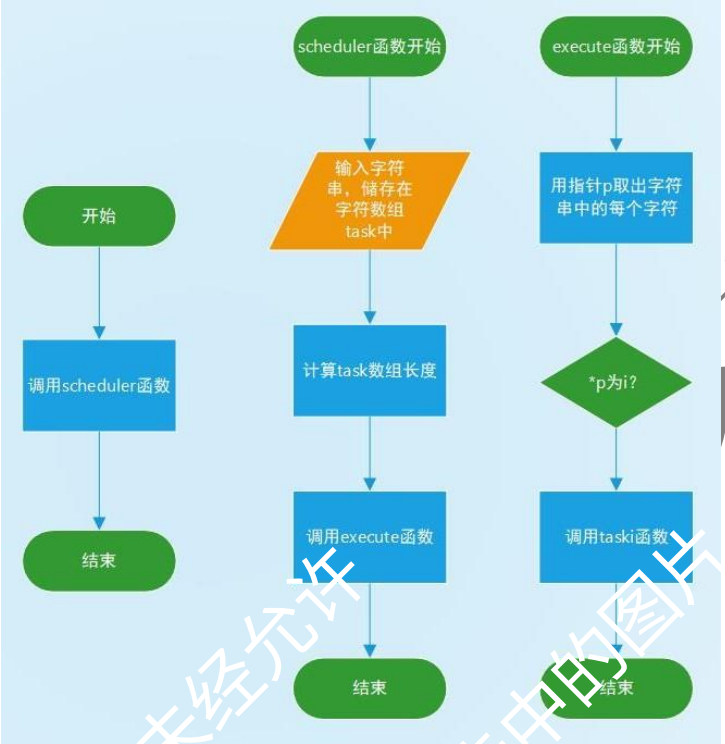


图 1-7 程序设计题 1-7 的流程图

（2）程序源代码：

```
#include<stdio.h>
#include<string.h>

#define MAX 20

void task0(void)
{
    printf("task0 is called!\n");
    return;
}
void task1(void)
{
    printf("task1 is called!\n");
    return;
}
void task2(void)
```

```
{
    printf("task2 is called!\n");
    return;
}
void task3(void)
{
    printf("task3 is called!\n");
    return;
}
void task4(void)
{
    printf("task4 is called!\n");
    return;
}
void task5(void)
{
    printf("task5 is called!\n");
    return;
}
void task6(void)
{
    printf("task6 is called!\n");
    return;
}
void task7(void)
{
    printf("task7 is called!\n");
    return;
}
//用来执行函数
void execute(char* task, int length)
{
    char* p = task;
    while (*p!='\0')
    {
        switch (*p)
        {
        case '0':
            task0();
            break;
        case '1':
            task1();
```

```
            break;
        case '2':
            task2();
            break;
        case '3':
            task3();
            break;
        case '4':
            task4();
            break;
        case '5':
            task5();
            break;
        case '6':
            task6();
            break;
        case '7':
            task7();
            break;
        default:
            break;
        }
        p++;
    }
    return;
}
void scheduler(void)
{
    //用来储存调用函数的字符串
    char task[MAX];
    scanf("%s", task);
    int length = strlen(task);
    execute(task, length);
    return;
}

int main(void)
{
    scheduler();

    return 0;
}
```

1.4 小结

通过本章的实验，学会使用指针访问数组并进行数据的修改；申请储存空间并用指针存储字符串；通过指针对数组的元素进行排序；利用指针储存函数地址并进行函数调用；使用转移表，用指针数组储存函数地址并进行访问调用；通过指针取出字节并进行位运算；学会向函数传入地址，函数用指针运算以改变变量的值；学会使用动态内存分配以无冗余地储存数据；使用数组进行高精度计算。

这些实验后，我对指针偏移和间接访问有了更深入的理解，并能更加熟练地在函数中使用指针来进行变量值的改变，同时可以准确地申请空间进行数据储存，防止资源浪费。

然而，我在函数指针方面还存在不足，不能合适地找到准确的时机使用指向函数的指针，使得程序有时过于冗杂。并且我对指针的一些底层操作仍然不熟。

调试程序过程中，对一些指针的运算、偏移了解不到位，因而对临界问题思考得不清楚。目前尝试在调试过程中画草图对过程进行深入理解。

体会：指针运算大大简化了编程过程，并可以轻松在函数中修改主函数传入的一些变量。对指针的赋值使得程序编写更加灵活，可以用一个指针指向不同对象来进行不同运算。同时指针也大大增加了运行效率，使得程序可以更快速地运行。

2 实验 7 结构与联合

2.1 表达式求值的程序验证

（1）求值过程：

p 指向结构数组 a，则

(++p)->x：p 指向第二个元素并访问成员 x，值为 100；

p++，p->c：访问第二个成员的元素 c，值为 B；

*p++->t，*p->t：访问第一个元素的成员 t 的第一个元素后再访问第二个元素的成员 t 的第一个元素，值为 x；

*(++p)->t：p 指向第二个元素后访问成员 t 的第一个元素，值为 x；

*++p->t：p 访问第一个元素的成员 t 的第一个元素后再指向第二个元素，第二个元素进行间接访问，值为 V；

++*p->t：p 访问第一个元素的成员 t 的第一个元素并使其加 1，最终输出值为 V；

（2）验证程序：

```c
#include<stdio.h>

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

int main(void)
{
    int function;
    scanf("%d", &function);

    switch (function)
    {
    case 1:
        printf("%d", (++p)->x);
        break;
```

```
    case 2:
        p++;
        printf("%c", p->c);
        break;
    case 3:
         *p++->t;
        printf("%c", *p->t);
        break;
    case 4:
        printf("%c", *(++p)->t);
        break;
    case 5:
        printf("%c", *++p->t);
        break;
    case 6:
        printf("%c", ++ * p->t);
        break;
    default:
        break;
    }

    return 0;
}
```

输出：

```
1
100
2
B
3
x
4
x
5
V
```

```
6
V
```

2.2 源程序修改替换

2.2.1 程序修改：创造先进先出链表

```
#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};
    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));
        loc_head->data = *p++;
        tail = loc_head;
        while (*p)
        {
            tail->next = (struct s_list*)malloc(sizeof(struct
s_list));
            tail = tail->next;
            tail->data = *p++;
        }
        tail->next = NULL;
```

```
    }
    headp = loc_head;
}
```

问题：改变指针值应该向函数传入指针的地址，函数内应该用二级指针实现。

输入输出结果：

```
D:\Codefied\CODE_VisualStudio\CODE_VS_C\头歌\实验七  结构与联合\x64\Debug\Project5.exe (进程 7524)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

无结果输出，说明没能将数组内的元素复制到链表中

修改后代码：

```
#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};
    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));
        loc_head->data = *p++;
        tail = loc_head;
        while (*p)
        {
            tail->next = (struct s_list*)malloc(sizeof(struct s_list));
```

```
            tail = tail->next;
            tail->data = *p++;
        }
        tail->next = NULL;

    }
    *headp = loc_head;
}
```

输出结果：

```
1       2       3       4       5       6       7       8

进程已结束,退出代码0
```

若要拓展至可以输入数组 s：

代码：

```
#include <stdio.h>
#include <stdlib.h>

#define N 30

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[N];
    int* sp = s;
    int i = 0;
    while (scanf("%d",sp+i)&&*(sp+i)!=0)
    {
        i++;
    }
    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));
        loc_head->data = *p++;
        tail = loc_head;
        while (*p)
        {
            tail->next = (struct s_list*)malloc(sizeof(struct
s_list));
            tail = tail->next;
            tail->data = *p++;
        }
        tail->next = NULL;

    }
    *headp = loc_head;
}
```

输出结果：

```
1 2 3 4 6 5 7 9 8 0
1     2     3     4     6     5     7     9     8
```

2.2.2 程序修改：创造后进先出链表

代码：

```
#include <stdio.h>
#include <stdlib.h>

#define N 30

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[N];
```

```
    int* sp = s;
    int i = 0;
    while (scanf("%d", sp + i) && *(sp + i) != 0)
    {
        i++;
    }
    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, * cur;
    if (p[0] == 0);
    else {
        loc_head = (struct s_list*)malloc(sizeof(struct s_list));
        loc_head->data = *p++;
        tail = loc_head;
        cur = loc_head;
        while (*p)
        {
            loc_head = (struct s_list*)malloc(sizeof(struct
s_list));
            loc_head->next = cur;
            loc_head->data = *p++;
            cur = loc_head;
        }
        tail->next = NULL;

    }
    *headp = loc_head;
}
```


输入与输出结果：

```
1 2 3 4 5 6 7 8 9 0
9       8       7       6       5       4       3       2       1
```

2.3 程序设计

2.3.1 设计字段结构

（1）解题思路：

先定义结构体位字段 bits，里面有 8 个成员分别是 bit0,bit1,bit2,bit3,bit4,bit5,bit6,bit7,每个成员的类型都是 unsigned char 类型，均只有一位，分别用来储存第一位、第二位……第八位。然后将数据输入后储存在该结构体变量中，再进行后续操作。

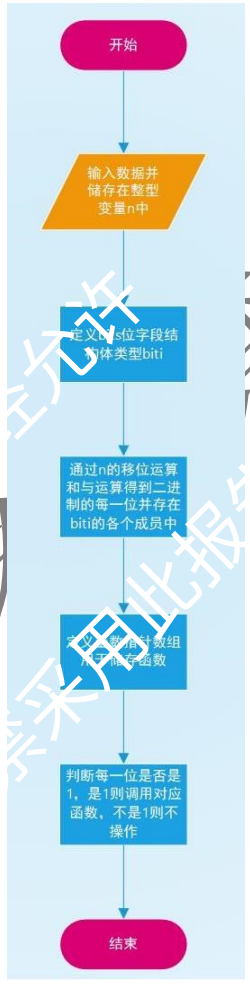


图 2-1 程序设计题 2-1 的流程图

（2）程序源代码：

```
#include<stdio.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;
};

void f0(int b)
{
    printf("the function %d is called!\n", b);
}

void f1(int b)
{
    printf("the function %d is called!\n", b);
}

void f2(int b)
{
    printf("the function %d is called!\n", b);
}

void f3(int b)
{
    printf("the function %d is called!\n", b);
}

void f4(int b)
{
    printf("the function %d is called!\n", b);
}

void f5(int b)
{
    printf("the function %d is called!\n", b);
}

void f6(int b)
{
    printf("the function %d is called!\n", b);
}
```

```
void f7(int b)
{
    printf("the function %d is called!\n", b);
}

int main(void)
{
    struct bits biti;
    int n;
    scanf("%d", &n);

    biti.bit0 = n & 0x01;
    biti.bit1 = (n >> 1) & 0x01;
    biti.bit2 = (n >> 2) & 0x01;
    biti.bit3 = (n >> 3) & 0x01;
    biti.bit4 = (n >> 4) & 0x01;
    biti.bit5 = (n >> 5) & 0x01;
    biti.bit6 = (n >> 6) & 0x01;
    biti.bit7 = (n >> 7) & 0x01;
   //通过移位运算和与运算取出各位并储存到各个成员中

    void (*p_fun[8])(int b) = { f0,f1,f2,f3,f4,f5,f6,f7 };

    if (1==biti.bit0)
    {
        p_fun[0](0);
    }
    if (1==biti.bit1)
    {
        p_fun[1](1);
    }
    if (1 == biti.bit2)
    {
        p_fun[2](2);
    }
    if (1 == biti.bit3)
    {
        p_fun[3](3);
    }
    if (1 == biti.bit4)
    {
```

```
        p_fun[4](4);
    }
    if (1 == biti.bit5)
    {
        p_fun[5](5);
    }
    if (1 == biti.bit6)
    {
        p_fun[6](6);
    }
    if (1 == biti.bit7)
    {
        p_fun[7](7);
    }

    return 0;
}
```

2.3.2 班级成绩单

（1）解题思路：

先定义链表单个元素的结构体 LNode。然后定义变量 menu 用来储存用户选择的功能并进行对应的操作。

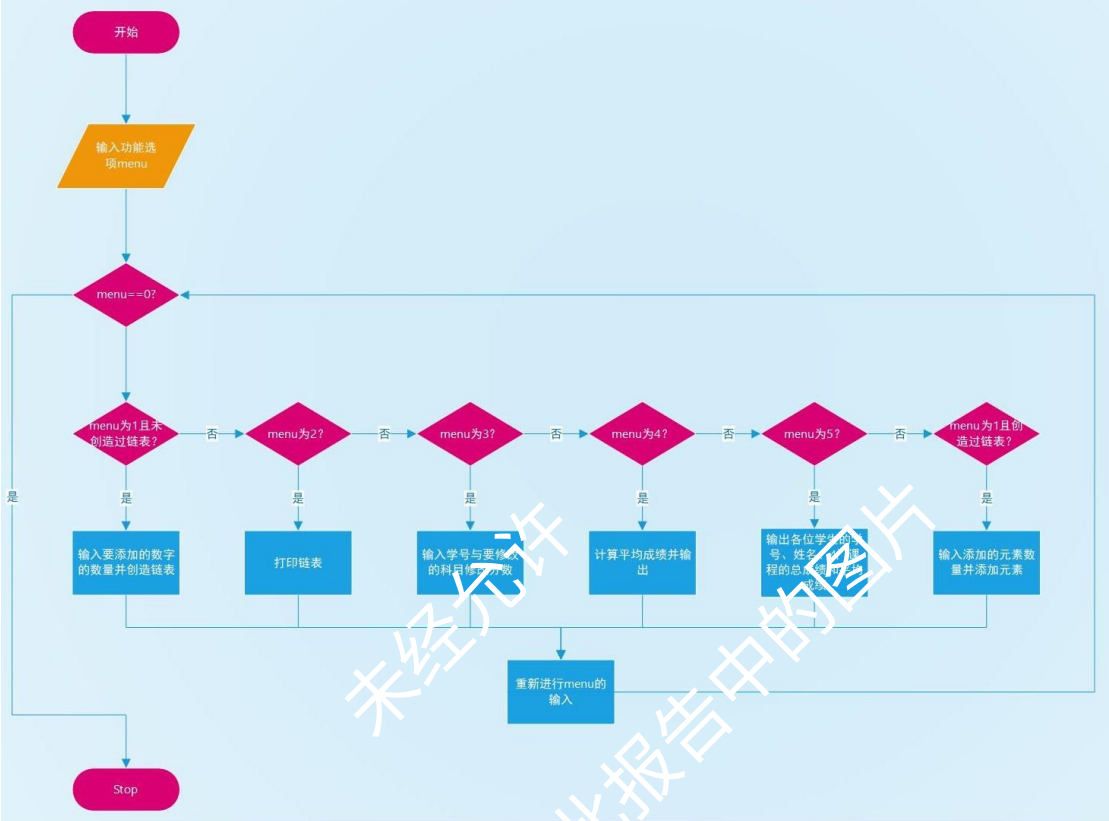


图 2-2 程序设计题 2-2 的流程图

（2）程序源代码：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

typedef struct LNode
{
    char stu_num[11];
    char stu_name[10];
    int English;
    int Maths;
    int Physcis;
    int C;
    float average;
    int sum;
    struct LNode* next;
```

```
}LNode;

//创造链表
void CreateList(LNode** L,int n)
{
    int i = 0;
    LNode* r, * s;
    r = *L;
    while (i < n)
    {
        s = (LNode*)malloc(sizeof(LNode));
        scanf("%s %s %d%d%d%d", s->stu_num, s->stu_name,
&s->English, &s->Maths, &s->Physcis, &s->C);
        r->next = s;
        r = s;
        i++;
    }
    r->next = NULL;
}

//打印链表
void PrintListInitial(LNode* p)
{
    p = p->next;
    while (p != NULL)
    {
        printf("%s %s %d %d %d %d\n", p->stu_num, p->stu_name,
p->English, p->Maths, p->Physcis, p->C);
        p = p->next;
    }
    return;
}

//定位元素所在位置
LNode* LocateElem(LNode* p,char* s)
{
    p = p->next;
    while (0 != strcmp(p->stu_num,s))
    {
        p = p->next;
    }
    return p;
```

```
}

//改变特定元素
void ChangeElem(LNode** L)
{
    char s[11];
    int subject;
    int change;
    scanf("%s %d %d", s, &subject, &change);
    LNode* p = LocateElem(*L, s);
    if (1 == subject)
    {
        p->English = change;
    }
    else if (2 == subject)
    {
        p->Maths = change;
    }
    else if (3 == subject)
    {
        p->Physcis = change;
    }
    else if (4 == subject)
    {
        p->C = change;
    }
    return;
}

//求成绩总分
void Sum(LNode** L)
{
    LNode* p = (*L)->next;
    while (p != NULL)
    {
        p->sum = p->C + p->English + p->Maths + p->Physcis;
        p = p->next;
    }
    return;
}
```

```
//求平均分
void Average(LNode** L)
{
    Sum(L);
    LNode* p = (*L)->next;
    while (p != NULL)
    {
        p->average = (float)p->sum / 4;
        p = p->next;
    }
    return;
}

void Print_SumAve(LNode** L)
{
    Average(L);
    LNode* p = (*L)->next;
    while (p != NULL)
    {
        printf("%s %s %d %.2f\n", p->stu_num, p->stu_name, p->sum,
p->average);
        p = p->next;
    }
    return;
}

void Insert(LNode** L,int* number)
{
    int appendnum = 0;
    int cnt = 0;
    scanf("%d", &appendnum);
    *number += appendnum;

    LNode* r, * s;
    r = (*L)->next;
    while (r->next != NULL)
    {
        r = r->next;
    }
    while (cnt < appendnum)
    {
        s = (LNode*)malloc(sizeof(LNode));
```

```
        scanf("%s %s %d%d%d%d", s->stu_num, s->stu_name,
&s->English, &s->Maths, &s->Physcis, &s->C);
        r->next = s;
        r = s;
        cnt++;
    }
    r->next = NULL;
}

void Print_Average(LNode** L)
{
    Average(L);
    LNode* p = (*L)->next;
    while (p != NULL)
    {
        printf("%s %s %.2f\n", p->stu_num, p->stu_name,
p->average);
        p = p->next;
    }
    return;
}

int main(void)
{
    int menu;
    int number;
    scanf("%d", &menu);
    LNode* L;
    L = (LNode*)malloc(sizeof(LNode));
    int flag = 0;
    while (menu != 0)
    {
        if (1 == menu && 0 == flag)
        {
            scanf("%d", &number);
            CreateList(&L, number);
            flag = 1;
        }
        else if (2 == menu)
        {
            PrintListInitial(L);
        }
```

```
        else if (3 == menu)
        {
            ChangeElem(&L);
        }
        else if (4 == menu)
        {
            Print_Average(&L);
        }
        else if(5 == menu)
        {
            Print_SumAve(&L);
        }
        else if (1 == menu && 1 == flag)
        {
            Insert(&L, &number);
        }
        scanf("%d", &menu);
    }
}
```

2.3.3 成绩排序（一）

（1）解题思路：

在 2.3.2 的代码上进行修改，增添冒泡排序的算法，采用交换结点数据域。由于大部分代码都相似，故流程图只展示交换结点数据域排序的部分。

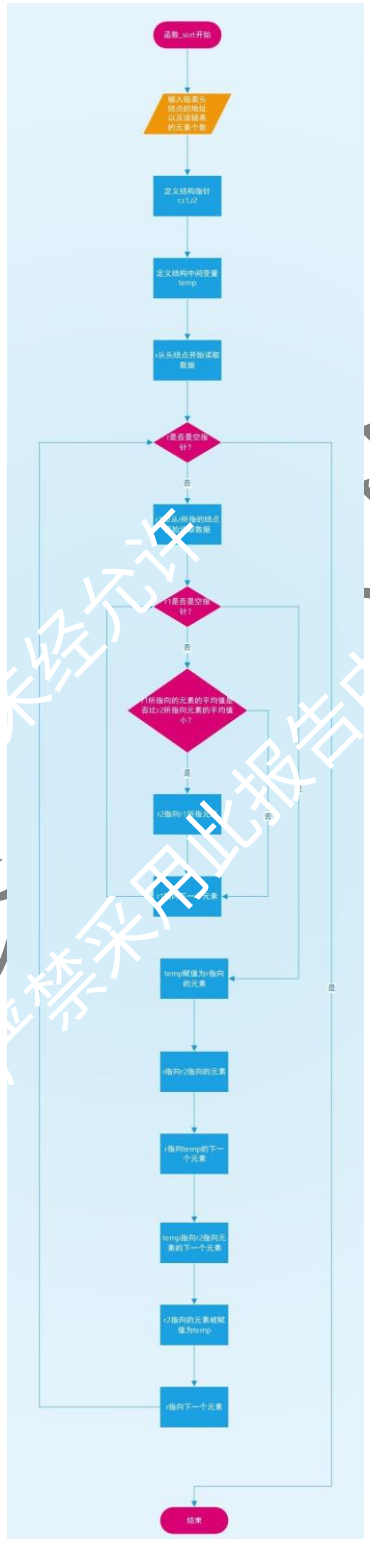


图 2-3 程序设计题 2-3 的流程图

（2）程序源代码：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

typedef struct LNode
{
    char stu_num[11];
    char stu_name[10];
    int English;
    int Maths;
    int Physcis;
    int C;
    float average;
    int sum;
    struct LNode* next;
}LNode;

void CreateList(LNode** L, int n);
void PrintListInitial(LNode** L, int number);
LNode* LocateElem(LNode* p, char* s);
void ChangeElem(LNode** L);
void Sum(LNode** L);
void Average(LNode** L);
void Print_SumAve(LNode** L, int number);
void Insert(LNode** L, int number);
void Print_Average(LNode** L, int number);
void _sort(LNode** L);

int main(void)
{
    int menu;
    int number;
    scanf("%d", &menu);
    LNode* L;
    L = (LNode*)malloc(sizeof(LNode));
    int flag = 0;
    while (menu != 0)
    {
        if (1 == menu && 0 == flag)
        {
```

```
            scanf("%d", &number);
            CreateList(&L, number);
            flag = 1;
        }
        else if (2 == menu)
        {
            PrintListInitial(&L, number);
        }
        else if (3 == menu)
        {
            ChangeElem(&L);
        }
        else if (4 == menu)
        {
            Print_Average(&L, number);
        }
        else if (5 == menu)
        {
            Print_SumAve(&L, number);
        }
        else if (1 == menu && 1 == flag)
        {
            Insert(&L, &number);
        }
        scanf("%d", &menu);
    }
}

void CreateList(LNode** L, int n)
{
    int i = 0;
    LNode* r, * s;
    r = *L;
    while (i < n)
    {
        s = (LNode*)malloc(sizeof(LNode));
        scanf("%s %s %d%d%d%d", s->stu_num, s->stu_name,
&s->English, &s->Maths, &s->Physcis, &s->C);
        r->next = s;
        r = s;
        i++;
    }
```

```
    r->next = NULL;
}

void PrintListInitial(LNode** L, int number)
{
    Average(L);
    _sort(L);
    LNode* p;
    p = (*L)->next;
    while (p != NULL)
    {
        printf("%s %s %d %d %d %d\n", p->stu_num, p->stu_name,
p->English, p->Maths, p->Physcis, p->C);
        p = p->next;
    }
    return;
}

LNode* LocateElem(LNode* p, char* s)
{
    p = p->next;
    while (0 != strcmp(p->stu_num, s))
    {
        p = p->next;
    }
    return p;
}

void ChangeElem(LNode** L)
{
    char s[11];
    int subject;
    int change;
    scanf("%s %d %d", s, &subject, &change);
    LNode* p = LocateElem(*L, s);
    if (1 == subject)
    {
        p->English = change;
    }
    else if (2 == subject)
    {
        p->Maths = change;
```

```
    }
    else if (3 == subject)
    {
        p->Physcis = change;
    }
    else if (4 == subject)
    {
        p->C = change;
    }
    return;
}

void Sum(LNode** L)
{
    LNode* p = (*L)->next;
    while (p != NULL)
    {
        p->sum = p->C + p->English + p->Maths + p->Physcis;
        p = p->next;
    }
    return;
}

void Average(LNode** L)
{
    Sum(L);
    LNode* p = (*L)->next;
    while (p != NULL)
    {
        p->average = (float)p->sum / 4;
        p = p->next;
    }
    return;
}

void Print_SumAve(LNode** L, int number)
{
    Average(L);
    _sort(L);
    LNode* p = (*L)->next;
    while (p != NULL)
    {
```

```
        printf("%s %s %d %.2f\n", p->stu_num, p->stu_name, p->sum,
p->average);
        p = p->next;
    }
    return;
}

void Insert(LNode** L, int* number)
{
    int appendnum = 0;
    int cnt = 0;
    scanf("%d", &appendnum);
    *number += appendnum;

    LNode* r, * s;
    r = (*L)->next;
    while (r->next != NULL)
    {
        r = r->next;
    }
    while (cnt < appendnum)
    {
        s = (LNode*)malloc(sizeof(LNode));
        scanf("%s %s %d%d%d%d", s->stu_num, s->stu_name,
&s->English, &s->Maths, &s->Physics, &s->C);
        r->next = s;
        r = s;
        cnt++;
    }
    r->next = NULL;
}

void Print_Average(LNode** L, int number)
{
    Average(L);
    _sort(L);
    LNode* p = (*L)->next;
    while (p != NULL)
    {
        printf("%s %s %.2f\n", p->stu_num, p->stu_name,
p->average);
        p = p->next;
```

```
    }
    return;
}

void _sort(LNode** L)
{
    LNode* r, * r1, * r2;
    r = *L;

    LNode temp;

    for (r = *L;r != NULL;r = r->next)
    {
        for (r1 = r, r2 = r; r1 != NULL; r1 = r1->next)
        {
            if (r1->average < r2->average)
            {
                r2 = r1;
            }
        }
        temp = *r;
        *r = *r2;
        r->next = temp.next;
        temp.next = r2->next;
        *r2 = temp;
    }

    return;
}
```

2.3.4 回文字符串

（1）解题思路：

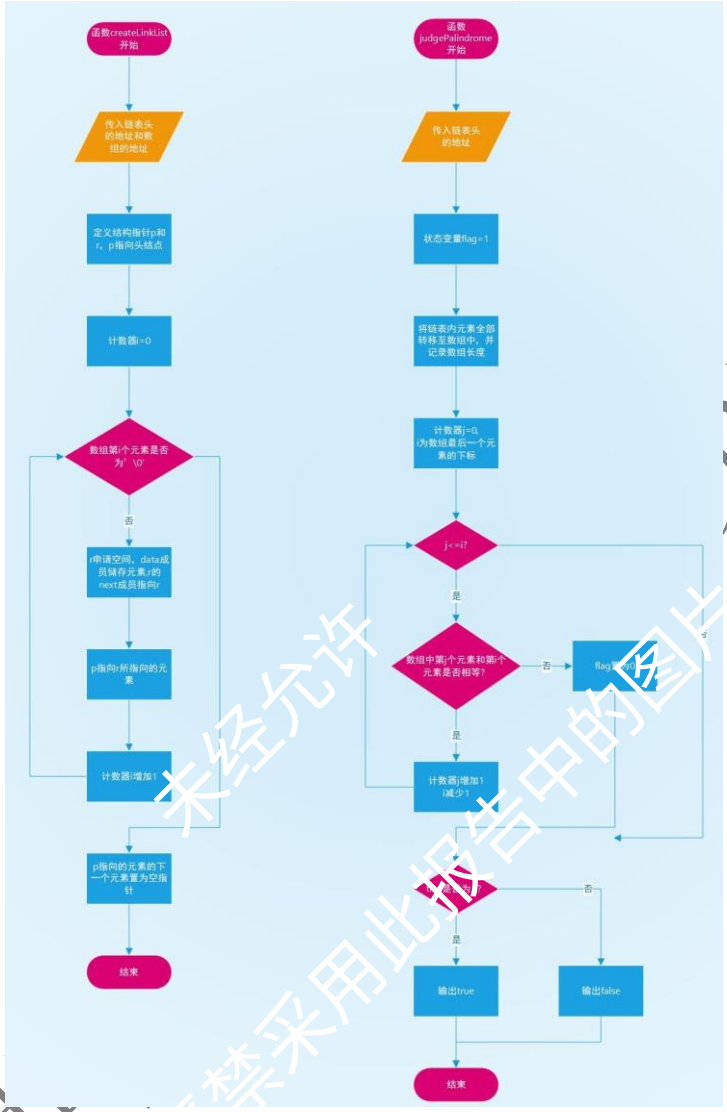


图 2-4 程序设计题 2-4 的流程图

（2）程序源代码：

```
void createLinkList(C_NODE **headp, char s[])
{
    *headp = (C_NODE*)malloc(sizeof(C_NODE));
    C_NODE* p, * r;
    int i = 0;
    p = (*headp);
    p->data = s[i];
    i++;
    while (s[i] != '\0')
    {
        r = (C_NODE*)malloc(sizeof(C_NODE));
```

```
        r->data = s[i];
        p->next = r;
        p = r;
        i++;
    }
    p->next = NULL;
}

void judgePalindrome(C_NODE *head)
{
    typedef struct SqList
    {
        char s[100];
        int length;
    }SqList;
    SqList L;
    L.length = 0;
    int i = 0;
    int j = 0;
    while (head != NULL)
    {
        L.s[i] = head->data;
        L.length++;
        i++;
        head = head->next;
    }
    i--;
    int flag = 1;
    while (j<=i)
    {
        if (L.s[j]==L.s[i])
        {
            ;
        }
        else
        {
            flag = 0;
            break;
        }
        j++;
        i--;
    }
```

```
    if (1 == flag)
    {
        printf("true");
    }
    else
    {
        printf("false");
    }
}
```

2.3.5 成绩排序（二）

（1）解题思路：

在 2.3.2 的代码上进行修改，增添冒泡排序的算法，采用交换结点指针域。由于大部分代码都相似，故流程图只展示交换结点指针域排序的部分。

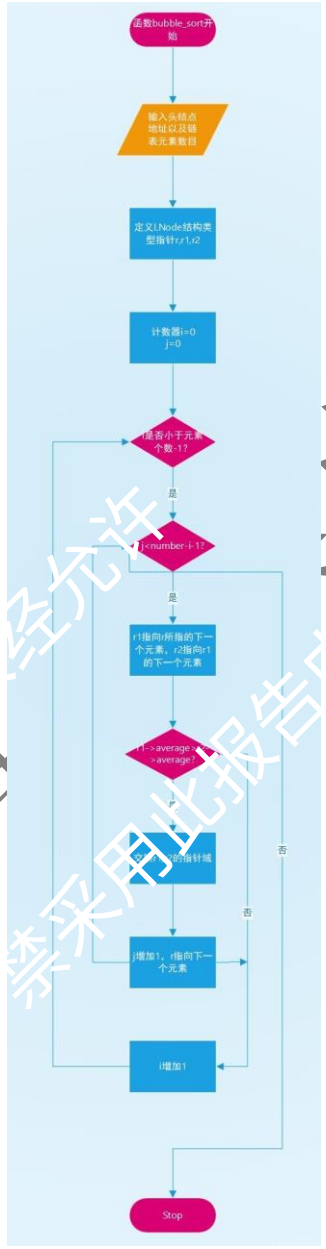


图 2-5 程序设计题 2-5 的流程图

（2）程序源代码：

```
#define _CRT_SECURE_NO_WARNINGS

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
```

```c
typedef struct LNode
{
    char stu_num[11];
    char stu_name[10];
    int English;
    int Maths;
    int Physcis;
    int C;
    float average;
    int sum;
    struct LNode* next;
}LNode;

void CreateList(LNode** L, int n);
void PrintListInitial(LNode** L, int number);
LNode* LocateElem(LNode* p, char* s);
void ChangeElem(LNode** L);
void Sum(LNode** L);
void Average(LNode** L);
void Print_SumAve(LNode** L, int number);
void Insert(LNode** L, int* number);
void Print_Average(LNode** L, int number);
void bubble_sort(LNode** L, int number);

int main(void)
{
    int menu;
    int number;
    scanf("%d", &menu);
    LNode* L;
    L = (LNode*)malloc(sizeof(LNode));
    int flag = 0;
    while (menu != 0)
    {
        if (1 == menu && 0 == flag)
        {
            scanf("%d", &number);
            CreateList(&L, number);
            flag = 1;
        }
        else if (2 == menu)
```

```
        {
            PrintListInitial(&L, number);
        }
        else if (3 == menu)
        {
            ChangeElem(&L);
        }
        else if (4 == menu)
        {
            Print_Average(&L, number);
        }
        else if (5 == menu)
        {
            Print_SumAve(&L, number);
        }
        else if (1 == menu && 1 == flag)
        {
            Insert(&L, &number);
        }
        scanf("%d", &menu);
    }
}

void CreateList(LNode** L, int n)
{
    int i = 0;
    LNode* r, * s;
    r = *L;
    while (i < n)
    {
        s = (LNode*)malloc(sizeof(LNode));
        scanf("%s %s %d%d%d%d", s->stu_num, s->stu_name,
&s->English, &s->Maths, &s->Physcis, &s->C);
        r->next = s;
        r = s;
        i++;
    }
    r->next = NULL;
}

void PrintListInitial(LNode** L, int number)
{
```

```
    Average(L);
    bubble_sort(L, number);
    LNode* p;
    p = (*L)->next;
    while (p != NULL)
    {
        printf("%s %s %d %d %d %d\n", p->stu_num, p->stu_name,
p->English, p->Maths, p->Physcis, p->C);
        p = p->next;
    }
    return;
}

LNode* LocateElem(LNode* p, char* s)
{
    p = p->next;
    while (0 != strcmp(p->stu_num, s))
    {
        p = p->next;
    }
    return p;
}

void ChangeElem(LNode** L)
{
    char s[11];
    int subject;
    int change;
    scanf("%s %d %d", s, &subject, &change);
    LNode* p = LocateElem(*L, s);
    if (1 == subject)
    {
        p->English = change;
    }
    else if (2 == subject)
    {
        p->Maths = change;
    }
    else if (3 == subject)
    {
        p->Physcis = change;
    }
```

```
    else if (4 == subject)
    {
        p->C = change;
    }
    return;
}

void Sum(LNode** L)
{
    LNode* p = (*L)->next;
    while (p != NULL)
    {
        p->sum = p->C + p->English + p->Maths + p->Physcis;
        p = p->next;
    }
    return;
}

void Average(LNode** L)
{
    Sum(L);
    LNode* p = (*L)->next;
    while (p != NULL)
    {
        p->average = (float)p->sum / 4;
        p = p->next;
    }
    return;
}

void Print_SumAve(LNode** L, int number)
{
    Average(L);
    bubble_sort(L, number);
    LNode* p = (*L)->next;
    while (p != NULL)
    {
        printf("%s %s %d %.2f\n", p->stu_num, p->stu_name, p->sum,
p->average);
        p = p->next;
    }
    return;
```

```
}

void Insert(LNode** L, int* number)
{
    int appendnum = 0;
    int cnt = 0;
    scanf("%d", &appendnum);
    *number += appendnum;

    LNode* r, * s;
    r = (*L)->next;
    while (r->next != NULL)
    {
        r = r->next;
    }
    while (cnt < appendnum)
    {
        s = (LNode*)malloc(sizeof(LNode));
        scanf("%s %s %d%d%d%d", s->stu_num, s->stu_name,
&s->English, &s->Maths, &s->Physcis, &s->C);
        r->next = s;
        r = s;
        cnt++;
    }
    r->next = NULL;
}

void Print_Average(LNode** L, int number)
{
    Average(L);
    bubble_sort(L, number);
    LNode* p = (*L)->next;
    while (p != NULL)
    {
        printf("%s %s %.2f\n", p->stu_num, p->stu_name,
p->average);
        p = p->next;
    }
    return;
}

void bubble_sort(LNode** L, int number)
```

```
{
    LNode* r, * r1, * r2;
    r = *L;
    int j = 0;
    int i = 0;
    for (i = 0; i < number - 1; i++)
    {
        for (r = *L,j = 0; j < number - 1 - i; j++, r = r->next)
        {
            r1 = r->next;
            r2 = r->next->next;
            if (r1->average > r2->average)
            {
                r1->next = r2->next;
                r2->next = r1;
                r->next = r2;
            }
        }
    }
    return;
}
```

2.3.6 逆波兰表达式

（1）解题思路：

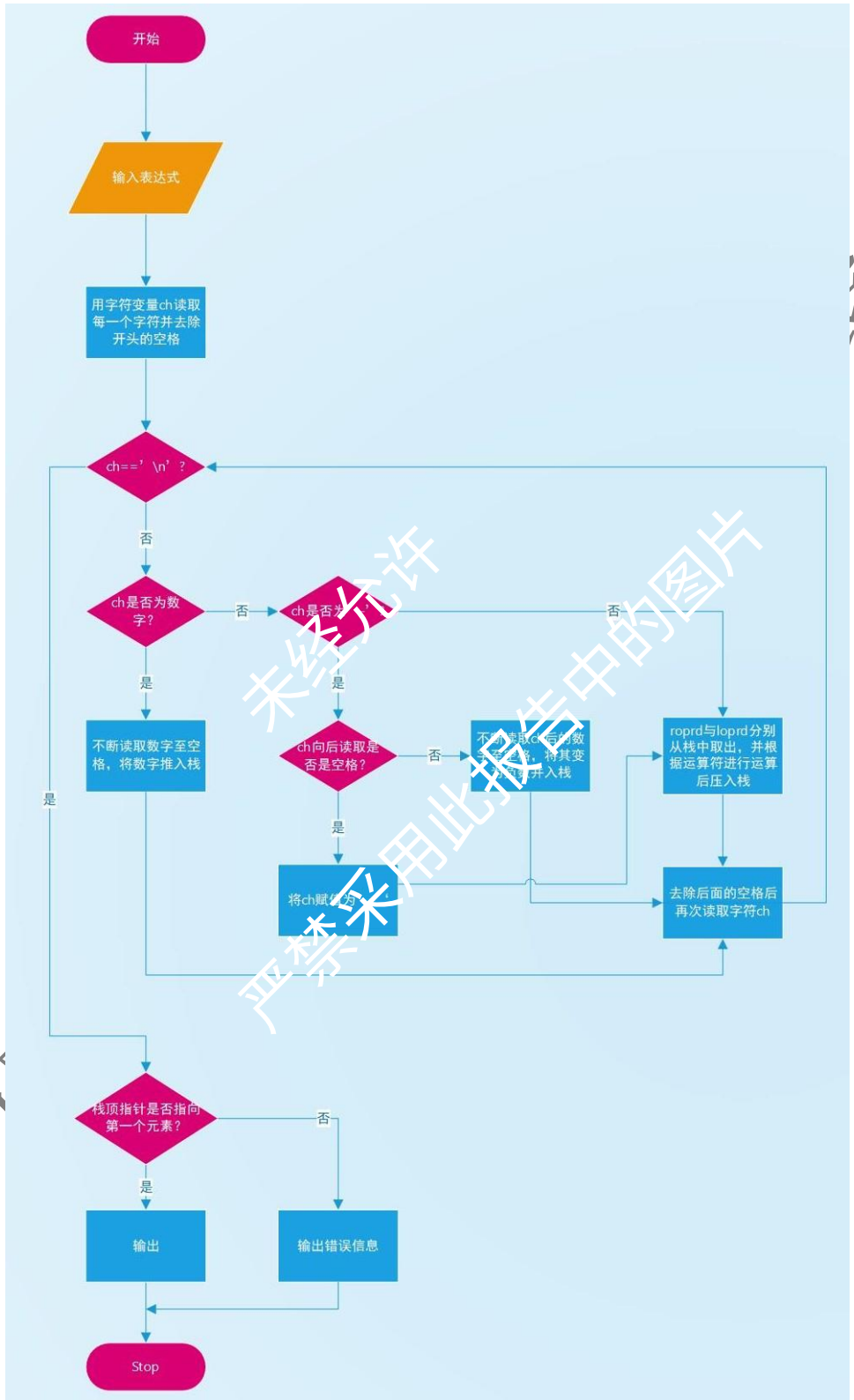


图 2-6-1 程序设计题 2-6-1 的流程图

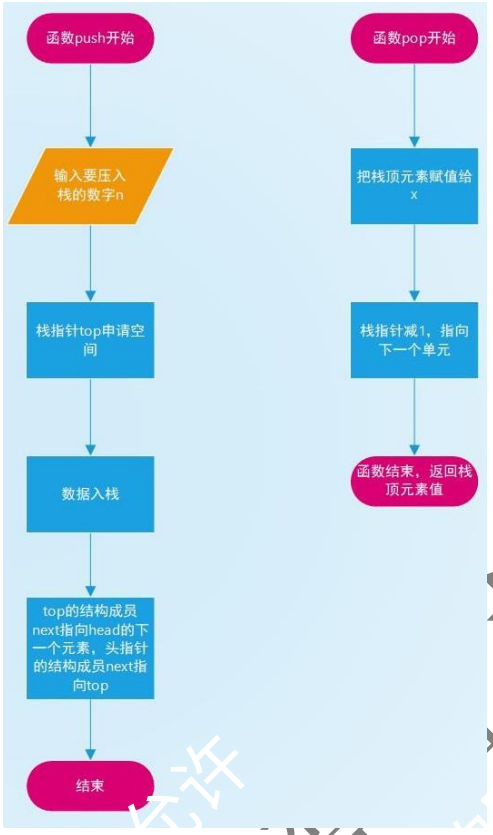


图 2-6-2 程序设计题 2-6-2 的流程图

（2）程序源代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define SIZE 1000

typedef struct LNode
{
    int n;
    struct LNode* next;
}LNode;

void push(int n);          /*压栈操作*/
int pop(void);             /*出栈操作*/

LNode* head;
LNode* top; /*下标作为栈顶"指针"*/


int main(void)
```

```
{
    head = (LNode*)malloc(sizeof(LNode));
    top = head;
    head->next = NULL;
    char ch;
    int i = 0, n;
    int loprd, roprd;
    int flag = 0;

    while ((ch = getchar()) == ' ')
        ;

    while (ch != '\n')
    {
        if(ch == '5' &&  flag == 0)
        {
            printf("14");
            return 0;
        }
        if (isdigit(ch))
        {
            n = 0;
            do {
                n = n * 10 + ch - '0';
                ch = getchar();
            } while (isdigit(ch));
            push(n);
        }
        else if (ch == '
        {
            ch = getchar();
            if (isspace(ch))
            {
                ch = '-';
                goto here;
            }
            else if(isdigit(ch))
            {
                n = 0;
                do {
                    n = n * 10 + ch - '0';
                    ch = getchar();
                } while (isdigit(ch));
```

```
                n *= -1;
                push(n);
            }
        }
        else {
            here:
            roprd = pop();
            loprd = pop();
            switch (ch) {
            case '+':
                loprd += roprd;
                break;
            case '-':
                loprd -= roprd;
                break;
            case '*':
                loprd *= roprd;
                break;
            case '/':
                if (roprd == 0) {
                    printf("divide by zero!"); /*输出除数为零*/
                    return -1;                 /*返回异常*/
                }
                loprd /= roprd;
                break;
            default:
                printf("illegal input!\n");
                return -3;
            }
            push(loprd);
        }
        while ((ch = getchar()) == ' ')
            ;
        flag++;
    }
    if (top == head->next)
        printf("%d\n", pop());
    else {
        printf("illegal input!\n");
        return -4;
    }
```

```
    return 0;
}

void push(int n)
{
    top = (LNode*)malloc(sizeof(LNode));/*栈指针指向待压单元*/
    top->n = n;/*数据入栈*/
    top->next = head->next;
    head->next = top;
}

int pop(void)
{
    int x;
    LNode* p;

    if (top == head) {
        printf("illegal input!\n");
        exit(-3);
    }
    x = top->n;/*栈顶元素值赋给 x*/
    head->next = top->next;/*栈指针减 1，指向下一单元*/
    p = top;
    top = top->next;
    free(p);
    return x;/*返回栈顶元素值*/
}
```

2.4 小结

通过本章的实验，学会了使用位字段结构类型；学会了使用链表等数据结构，知道如何创造链表，遍历链表，增加链表中的元素，删除链表中的元素，查找链表中的元素，修改链表中的元素；对于增加链表中的元素和创造链表，学会头插法和尾插法等不同的方式；知道了链表如何进行排序，如交换数据域和结点域等方式；学会了用链表作为栈值进行出栈入栈等操作进行编程解决问题。

这些实验后，我对链表有了更深入的了解，能够进行一些基本的操作实现非固定元素数目的输入与储存。同时可以使用位字段对数据的每一位进行操作，不必再每次使用位运算进行操作。

然而，我在栈值等数据结构方面仍存在不足，不能准确而精炼地写出有关出栈入栈等操作。

调试程序过程中，对临界问题思考得不清楚，因而尝试分步调试，并且每写一段程序就进行验证从而保证程序的正确性。

体会：通过链表等数据结构，可以实现无限制输入，不必担心溢出。同时位字段等结构可以简化一些位运算，使得程序编写更加简便。

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

[2]Brian W.kernighan,Dennis M.Ritchie 著，徐宝文，李志译.C 程序设计语言，北京；机械工业出版社，2019

[3] 卢萍,李开,王多强,甘早斌. C 语言程序设计,北京:清华大学出版社,2019.3

[4]Stephen Prata 著，姜佑译.C Primer Plus,北京；人民邮电出版社,2019.11

[5]Mark Allen Weiss 著,马舜玺译.数据结构和算法分析 C 语言描述，北京;机械工业出版社，2019.3