

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： CSE202003**

**学 号： U202012061**

**姓 名： 王祖鹏**

**指导教师： 张云鹤**

**报告日期： 2020年10月17日**

**网络空间安全学院**

**目 录**

[实验1 表达式和标准输入输出实验 4](#_Toc60906423)

[**1.1 实验目的** 4](#_Toc60906424)

[**1.2 实验内容及要求** 4](#_Toc60906425)

[**1.3 实验小结** 17](#_Toc60906426)

[实验2 流程控制实验 17](#_Toc60906427)

[**2.1 实验目的** 17](#_Toc60906428)

[**2.2 实验内容及要求** 17](#_Toc60906429)

[**1.3 实验小结** 34](#_Toc60906430)

[实验3 函数与程序结构实验 34](#_Toc60906431)

[**3.1实验目的** 34](#_Toc60906432)

[**3.2实验内容** 35](#_Toc60906433)

[**3.3 实验小结** 46](#_Toc60906434)

[实验4 编译预处理实验 47](#_Toc60906435)

[**4.1实验目的** 47](#_Toc60906436)

[**4.2实验内容** 47](#_Toc60906437)

[**4.3实验小结** 58](#_Toc60906438)

[实验5 数组实验 58](#_Toc60906439)

[**5.1 实验目的** 58](#_Toc60906440)

[**5.2 实验内容及要求** 59](#_Toc60906441)

[**5.3 实验小结** 73](#_Toc60906442)

[实验6 指针实验 73](#_Toc60906443)

[**6.1、实验目的** 73](#_Toc60906444)

[**6.2、实验题目及要求** 74](#_Toc60906445)

[**6.3、实验小结** 93](#_Toc60906446)

[实验7 结构与联合实验 93](#_Toc60906447)

[**7.1实验目的** 93](#_Toc60906448)

[**7.2实验题目及要求** 93](#_Toc60906449)

[**7.3实验小结** 108](#_Toc60906450)

[实验8 文件操作实验 108](#_Toc60906451)

[**8.1实验目的** 108](#_Toc60906452)

[**8.2实验题目及要求** 109](#_Toc60906453)

[**8.3实验小结** 115](#_Toc60906454)

[参考文献 116](#_Toc60906455)

**实验1 表达式和标准输入输出实验**

**1.1 实验目的**

（1）熟练掌握各种运算符的运算功能，操作数的类型，运算结果的类型及运算过程中的类型转换，重点是C语言特有的运算符，例如位运算符，问号运算符，逗号运算符等；熟记运算符的优先级和结合性。

（2）掌握getchar, putchar, scanf 和printf 函数的用法。

（3）掌握简单C程序的编写方法。

（4）熟悉C语言程序的开发环境，并学会调试程序的方法。

**1.2 实验内容及要求**

**1、程序改错与跟踪调试**

下面的实验1-1程序用来完成以下任务：

（1）输入华氏温度f，将它转换成摄氏温度c后输出。

（2）输入圆的半径值ｒ，计算并输出圆的面积ｓ。

（3）将ｋ的高字节作为结果的低字节，ｐ的高字节作为结果的高字节，拼成一个新的整数后输出。

在这个程序中存在若干语法和逻辑错误，要求先编译程序改正语法错误，再采用单步执行的方式调试程序找出逻辑错误。在单步执行程序的过程中，观察以下变量值：

1. 执行完c = 5/9 \* (f-32)，c的值为多少？
2. 执行完scanf(“%f”, &r)，r的值为多少？
3. 执行完newint = p&0xff00|k>>8， newint的值是多少？表达式k>>8的值是多少？

根据观察结果分析代码并修改程序，使之能够正确完成指定任务。

/\*实验1-1程序改错与跟踪调试题源程序\*/

1 #include<stdio.h>

2 #define PI 3.14159;

3 int main( void )

4 {

5 int f ;

6 short p, k ;

7 double c, r, s ;

/\* 任务1 \*/

8 printf("Input Fahrenheit: " ) ;

9 scanf(“%d”, f ) ;

10 c = 5/9 \* (f-32) ;

11 printf( “ \n %d (F) = %.2f (C)\n\n ”, f, c ) ;

/\* 任务2 \*/

12 printf("input the radius r:");

13 scanf("%f", &r);

14 s = PI \* r \* r;

15 printf("\nThe acreage is %.2f\n\n",&s);

/\* 任务3 \*/

16 k = 0xa1b2, p = 0x8432;

17 newint = p&0xff00|k>>8;

18 printf("new int = %#x\n\n",newint);

19 return 0;

20 }

**解答：**

（1）错误修改：

1) 第2行的符号常量定义后不能有分号，正确形式为：

#define PI 3.14159

2) 第6行的short定义错误，正确形式为：

unsigned short p,k

3）第9行的f前未加地址符，正确形式为：

&f

4）第10行5/9错误，正确形式为：

5.0/9

5）第13行f错误，正确形式为：

lf

6）第15行s前不加地址符，正确形式为：

s

7）第17行newint未定义，正确形式为：

int newint

（2）错误修改后运行结果：

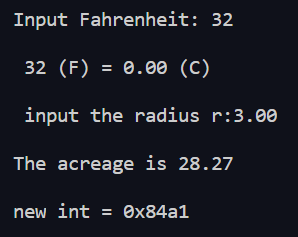


图1-1 改错后的运行结果

**2、程序分析与修改替换**

下面的实验1-2程序用“更相减损”法求m与n的最大公约数。

/\* 实验1-2程序分析与修改题源程序 \*/

#include <stdio.h>

int main( )

{

int m, n, k, p, i, d;

printf("input m, n \n");

scanf("%d%d", &m, &n);

if (m<n) /\* 交换m和n \*/

{

int t;

t = m;

m = n;

n = t;

}

k = 0;

while (m%2 == 0 && n%2 == 0) /\* m和n均为偶数 \*/

{

m /= 2; /\* 用2约简m和n \*/

n /= 2;

k++;

}

for (p = 1; i = 0; i<k; i++) p \*= 2; /\* 求p=2k \*/

while((d=m-n)!=n)

{

if(d>n) m = d;

else

{

m = n;

n = d;

}

}

d \*=p;

printf("the greatest common divisor : %d", d);

return 0;

}

1. 分析程序采用的“更相减损”法的算法步骤。
2. 按以下要求对源程序进行优化，提高程序的执行效率。

将交换m和n、判断一个整数是否是偶数、用2约简m和n、求p=2k等操作改成用位运算实现，并且m和n的交换不能够使用中间变量t，需要删除声明t的语句。

1. 将else后的复合语句改用一条表达式语句。

**解答：**

(1) int t;

t = m;

m = n;

n = t;

替换后的程序如下所示：



图1-2 第一个替换处

(2) while (m%2 == 0 && n%2 == 0) /\* m和n均为偶数 \*/

{

m /= 2; /\* 用2约简m和n \*/

n /= 2;

k++;

}

for (p = 1; i = 0; i<k; i++) p \*= 2; /\* 求p=2k \*/

替换后的程序如下所示：



图1-3 第二处替换

(3) m = n;

n = d;

替换后的程序如下所示：



图1-4 第三处替换

运行结果：

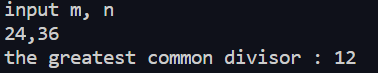


图1-5 替换之后的运行结果

**3、程序设计**

（1）输入字符ｃ，如果ｃ是大写字母，则将ｃ转换成对应的小写，否则ｃ的值不变，输入Ctrl+Z程序结束。要求①用条件表达式；②字符的输入输出用getchar和putchar函数。程序应能循环接受用户的输入，直至输入Ctrl+Z程结束。例如：

A （键盘输入）

a

^Z （键盘输入）

**解答：**

1. 算法流程如图1.1所示。



图1-6 程序一的流程图

2）源程序清单



图1-7 程序一的源代码

3）测试

（a） 测试数据：

A，m，ctrl z

（b） 对应测试数据的运行结果截图

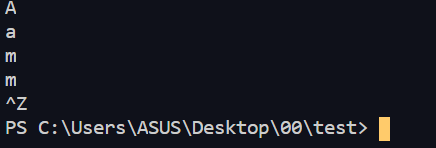


图1-8 程序一的测试结果

（2）编写一个程序，输入无符号短整数x，ｍ，ｎ（0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ）,取出x从第ｍ位开始向左的ｎ位（ｍ从右至左编号为0～15），并使其向左端（第15位）靠齐。

**解答：**

1. 解题思路：



图1-9 程序二的流程图

2）程序清单



图1-10 程序二的源代码

3）测试

（a） 测试数据：

0xffb3，4，9

0xffb3，8，9

（b） 对应测试数据的运行结果截图

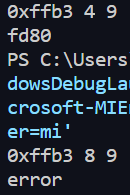


图1-11 程序二的运行结果

（3）IP地址通常是4个用句点分隔的小整数（即点分十进制），但这些地址在机器中是用一个无符号长整型数表示的。例如3232235876，其机内二进制表示就是11000000 10101000 00000001 01100100，按照8位一组用点分开，该IP地址就写成192.168.1.100。

读入无符号长整型数表示的互联网IP地址，对其译码，以常见的点分十进制形式输出。要求循环输入和输出，直至输入Ctrl+Z结束。

**解答：**

1）解题思路：



图1-12 程序三的流程图

2）程序清单



图1-13 程序三的源代码

3）测试

（a） 测试数据：

123456789

Ctrl z

（b） 对应测试数据的运行结果截图

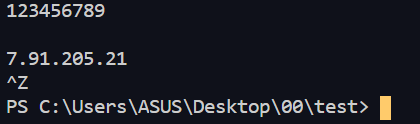


图1-14 程序三运行结果

**4. 选做题**

某加密算法对数据按字节进行加密，具体为：对字节的8个二进制位从右向左用0～7编号，先将0、2、4位分别与1、3、5位两两对应交换，接着对0～5位进行循环左移（左边移出的位接在右边），循环左移的位数有6、7两位的值决定。例如，若6、7位组成的二进制数为01，则将0～5位左移1位，最后得到加密结果，如图1-1所示。

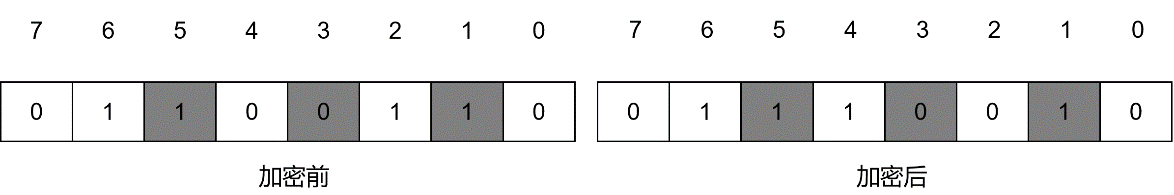


图1-1 加密示意图

输入一行明文字符串，按该算法进行加密后输出密文。例如，输入“abcd”，则输出“dbfp”。

1. 解题思路：

1.定义字符变量c，以及一个数组x。

2.输入字符串，将字符串中的每个字符按序存入x。

3.定义变量i=0，当i小于数组长度时，执行如下循环，执行一次，并i++

4.循环体：定义a，b，k，k1；用c读取x[i];用a，b，k分别和逻辑尺运算，取出所需位数，然后拼接，k1=k>>6。根据题目要求，这里还需要内嵌循环。

5.定义变量j=1，当j小于等于k1时，执行如下循环，并j++

6.循环体：定义m，n；m，n分别和逻辑尺运算取数并移位，最后拼接m，n，k得到转换之后的字符。

流程图如下：



图1-15 选做题流程图

2）程序清单



图1-16 选做题源代码

3）测试

（a） 测试数据：

Abcd

（b） 对应测试数据的运行结果截图



图1-17 选做题运行结果

**5. 自设题**

给定一个正整数，要求输出反转各位数字之后的结果。例如输入1234，输出4321

1. 解题思路：



图1-18 自设题流程图

1. 程序清单



图1-19 自设题源代码

3）测试

（a） 测试数据：

123456789

（b） 对应测试数据的运行结果截图

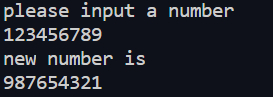


图1-20 自设题运行结果

**1.3 实验小结**

在实验过程中，难免会遇到很多问题，犯很多错误。但是，在纠正错误的时候，会学到很多知识。比如语法错误很细微，需要仔细观察才能发现。还有逻辑错误，需要不断调试才能找到。当然，找到错误之后还要不断分析调试以及优化，最后才能设计出一个完整的程序。在这样不断的试错过程中，可以有效地提升自己的编程能力。

**实验2 流程控制实验**

**2.1 实验目的**

（1）掌握复合语句、if语句、switch语句的使用，熟练掌握for、while、do-while三种基本的循环控制语句的使用，掌握重复循环技术，了解转移语句与标号语句。

（2）练习循环结构for、while、do-while语句的使用。

（3）练习转移语句和标号语句的使用。

（4）使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

**2.2 实验内容及要求**

**1．程序改错**

下面的实验2-1程序是合数判断器（合数指自然数中除了能被1和本身整除外，还能被其它数整除的数），在该源程序中存在若干语法和逻辑错误。要求对该程序进行调试修改，使之能够正确完成指定任务。

/\* 实验2-1改错题程序：合数判断器\*/

1 #include <stdio.h>

2 int main( )

3{

4 int i, x, k, flag = 0;

5 printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

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

7 for(i=2,k=x>>1;i<=k;i++)

8 if (!x%i) {

9 flag = 1;

10 break;}

11 if(flag=1) printf("%d是合数", x);

12 else printf("%d不是合数", x);

13 }

14 return 0;

15 }

**解答：**

（1）错误修改：

1）第7行for里面加语句flag=0

2）第8行！后加括号！（x%i）

3）第11行=改为==

**2．程序修改替换**

/\* 实验2-1修改题程序：合数判断器\*/

1 #include <stdio.h>

2 int main( )

3{

4 int i, x, k, flag = 0;

5 printf("本程序判断合数，请输入大于1的整数，以Ctrl+Z结束\n");

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

7 for(i=2,k=x>>1;i<=k;i++)

8 if (!x%i) {

9 flag = 1;

10 break;}

11 if(flag=1) printf("%d是合数", x);

12 else printf("%d不是合数", x);

13 }

14 return 0;

15 }

（1）修改实验2-1程序，将内层两出口的for循环结构改用单出口结构，即不允许使用break、goto等非结构化语句。

**解答：**

1）删除第10行break语句

2）第12行后加flag=0；

3）第7行改为i<=k&&flag==0

（2）修改实验2-1程序，将for循环改用do-while循环。

**解答：**

1）修改后的程序

do

{

flag = 0; //刷新flag

if (!(x % i))

{

flag = 1;

break;

}

else

{

flag = 0;

}

i++;

} while (i <= k);

}

else

{

flag = 0;

}

if (flag == 1)

printf("%d is he su\n", x);

else

printf("%d is not he su\n", x);

}

return 0;}

2）测试

（a）测试数据

2 3 4 6 8 9

运行结果：

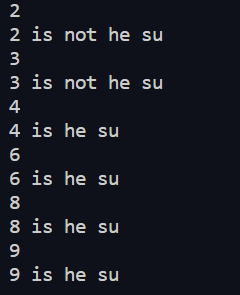


图2-1 do-while运行结果

（3）修改实验2-1程序，将其改为纯粹合数求解器，求出所有的3位纯粹合数。一个合数去掉最低位，剩下的数仍是合数；再去掉剩下的数的最低位，余留下来的数还是合数，这样反复，一直到最后剩下一位数仍是合数，这样的数称为纯粹合数。

**解答：**

1）流程图



图2-2 纯粹和数流程图

2）源代码

for (x = 100; x <= 999; x++)

{

for (i = 2, flag = 0, k = x >> 1; i <= k; i++) //保证flag每次刷新

{

if (!(x % i))

{

flag = 1;

break;

}

}

3）测试

（a）测试数据

100-999

（b）运行结果

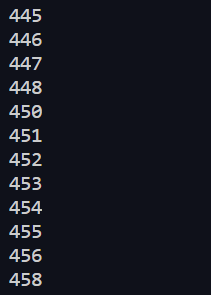


图2-3 纯粹合数部分运行结果

**3．程序设计**

（1） 假设工资税金按以下方法计算：x ＜ 1000元，不收取税金；1000 ≤ x ＜ 2000，收取5%的税金；2000 ≤ x ＜ 3000，收取10%的税金；3000 ≤ x ＜ 4000，收取15%的税金；4000 ≤ x ＜ 5000，收取20%的税金；x＞5000，收取25%的税金。（注意税金的计算按照阶梯计税法，比如，工资为4500，那么税金=1000\*5% + 1000\*10% + 1000\*15% + 501\*20%）。编写一个程序,输入工资金额，输出应收取税金额度，要求分别用if语句和switch语句来实现。

**解答：**

1. 流程图



图2-4 程序3.1流程图

1. 源代码

#include <stdio.h>

int main()

{

double x;

scanf("%lf", &x);

if (0 <= x && x < 1000)

printf("%.2f\n", x);

switch ((int)((int)x / 1000))

{

case 0:

printf("%.2f", x);

break;

3）运行结果

1. 测试数据

4500 6200

1. 对应测试数据运行结果截图

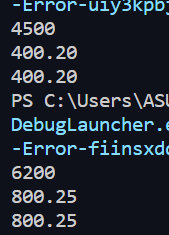


图2-5 程序3-1运行结果

（2）将输入的正文复制到输出，复制过程中将每行一个以上的空格字符用一个空格代替。

1）流程图



图2-6 程序3-2流程图

2）源代码

#include <stdio.h>

#define COPY 0

#define START 1

int main()

{

char c;

int state;

state = COPY;

while ((c = getchar()) != EOF)

switch (state)

{

case COPY:

if (c == ' ')

{

putchar(c);

state = START;

break;

}

else

{

putchar(c);

break;

}

case START:

if (c == ' ')

{

break;

}

else

{

putchar(c);

state = COPY;

break;

}

}

return 0;

}

1. 测试
2. 测试数据

Qwe qwe qwe

1. 对应运行结果截图



图2-7 程序3-2运行结果

1. 打印如下的杨辉三角形。

1 /\*第0行 \*/

1 1 /\*第1行 \*/

1 2 1 /\*第2行 \*/

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

第i行第j列位置的数据值可以由组合表示，而的计算如下：

|  |  |
| --- | --- |
|  | (i=0,1,2,…) |
|  | (j=0,1,2,3,…,i) |

根据以上公式，采用顺推法编程，输出金字塔效果的杨辉三角形。特别要注意空格的数目，一位数之间是3个空格，两位数之间有2个空格，3位数之间只有一个空格。

**解答：**

1. 流程图



图2-8 程序3-3流程图

2）源代码

#include <stdio.h>

int main()

{

int n;

scanf("%d", &n);

int i = 0, j = 0, a[n][n];

for (i = 0; i < n; i++)//存入第一个和最后一个1

{

a[i][0] = 1;

a[i][i] = 1;

}

for (i = 2; i < n; i++)//填满空隙

{

for (j = 1; j < i; j++)

{

a[i][j] = a[i - 1][j - 1] + a[i - 1][j];

}

}

for (i = 0; i < n; i++)//调整位置

{

for (j = 0; j < 2 \* (n - i); j++)/\*一位数字和一个空格占位相同，定义数组长度为4\*/

{ /\*，第0行应该移位 2\*n，后面依次递减两位,最后一行即n-1行是2\*/

printf(" ");

}

for (j = 0; j <= i; j++)

{

printf("%-4d", a[i][j]);

}

printf("\n");

}

return 0;}

3）测试

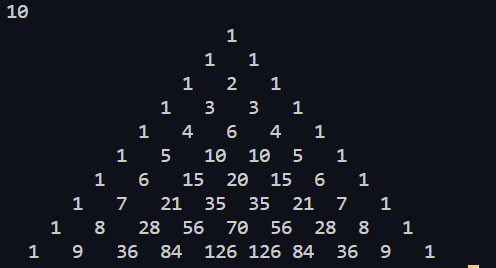


图2-9 程序3-3运行结果

（4）625这个数很特别，625的平方等于390625，其末3位也是625。请编程输出所有这样的3位数：它的平方的末3位是这个数本身。

1）流程图



图2-10 程序3-4流程图

2）源代码

#include <stdio.h>

int main()

{

int x, y;

for (x = 100; x <= 999; x++)

{

y = x \* x;

y = y % 1000;

if (x == y)

{

printf("%d\n", x);

}

}

return 0;

}

3）测试



图2-11 程序3-4运行结果

**4. 选做题**

（1）判断给定的字符串是否是合法的C整型常量，如果是，则输出Yes；不是，则输出No。例如，0xabL是C整形常量，而092不是C整形常量。要求程序能够循环接受用户的输入，每行输入一个字符串，给出判定结果，直至输入Ctrl+Z结束。

**解答：**

1. 思路

特判各种情况

2）源代码

if (!(s[len - 1] == 'U' || s[len - 1] == 'L'))

{

if (s[0] == '0' && s[1] == 'X')

{

if (len == 2)

{

printf("NO\n"); //特判0x是否合法

goto START;

}

3）测试

（a）测试数据

0xadfL，-0x45u

（b）对应运行结果

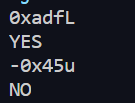


图2-12 程序4-1运行结果

（2）输入正整数x（2⩽x⩽79），输出所有形如abcde/fghij=x的表达式，其中a～j由不同的数字0～9组成。例如：x=32时，输出为：75168/02349=32。

**解答：**1）流程图



2）源代码

for (x = 12345; x <= 98765; x++)

{

int a[10] = {0};//初始化数组a

flag = 1;

z = x;//存储x的数值

if (x % n == 0)//可以整除

{

y = x / n;//取除数，注意可能没有5位，但之前已经初始化为0

for (i = 4; i >= 0; i--)//装入数组a

{

a[i] = y % 10;//个位数放到a4，以此类推，最高位放到a0

y = y / 10;

}

}

3）测试

（a）测试数据

6，12

（b）对应运行结果截图

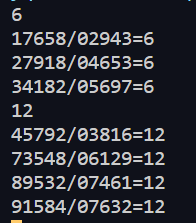


图2-14 程序4-2运行结果

**5.自设题**

已知正整数n，问从1-n任选3个数他们的最小公倍数最大可以是多少

其中3<=n<=100

**解答：**

1. 流程图



1. 源代码

#include <stdio.h>

int main()

{

int n;

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

{

long long t; //用于偶数判断记录(n-1)\*(n-2)\*(n-3)

if (n % 2 == 0) //偶数判断

{

t = (n - 1) \* (n - 2) \* (n - 3);

int j = 3;

while (n % j == 0)

{ //处理 3 ，公约数3 5 7 9...

j++; //跳过偶数

j++;

}

long long k = n \* (n - 1) \* (n - j); //获取到n, n-j无最小公倍数是时的值k

if (t > k)

{ //处理 标记3

printf("%lld\n", t);}

else

{printf("%lld\n", k);}

}

else //奇数判断

{

printf("%lld\n", n \* (n - 1) \* (n - 2));

}

}

}

1. 测试
2. 测试数据

6，20

1. 对应运行结果截图

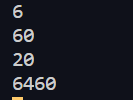


图2-15 自设题运行结果

**1.3 实验小结**

在本次实验中，我遇到了许多的困难，也出现了很多的错误。各种嵌套语句对我来说是一个不小的挑战。但是当你一步步做下去的时候，你就会发现编程语言的魅力。当我沉浸于此的时候，我就已经知道只要热爱，就没有什么困难可以阻挡学习的步伐。

**实验3 函数与程序结构实验**

**3.1实验目的**

（1）熟悉和掌握函数的定义、声明；函数调用与参数传递，函数返回值类型的定义和返回值使用。

（2）熟悉和掌握不同存储类型变量的使用。

（3）练习使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

**3.2实验内容**

**1．程序改错题**

下面是计算s=1!+2!+3!+…+n!的源程序(n<20)。在这个源程序中存在若干语法和逻辑错误。要求对该程序进行调试修改，使之能够输出如下结果：

k=1 the sum is 1

k=2 the sum is 3

k=3 the sum is 9

……

k=20 the sum is 2561327494111820313

/\*实验3-1改错题程序：计算s=1!+2!+3!+…+n!\*/

1 #include <stdio.h>

2 int main(void)

3 {

4 int k;

5 for(k=1;k<=20;k++)

6 printf("k=%d\tthe sum is %ld\n",k,sum\_fac(k));

7 return 0;

8 }

9 long sum\_fac(int n)

10 {

11 long s=0;

12 int i,fac;

13 for(i=1;i<=n;i++)

14 fac\*=i;

15 s+=fac;

16 return s;

17 }

**解答：**

（1）

1. 第一行之后加入声明函数语句
2. long型函数和变量均改为long long型
3. 第11行定义s放到第8行之后
4. 第12行定义fac初始值为1

（2）错误修改后的运行结果

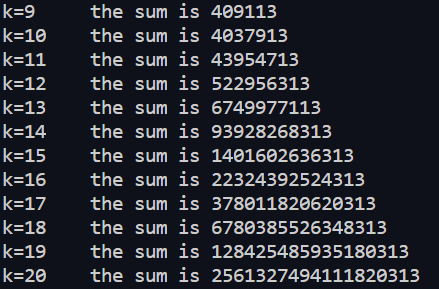


图3-1 程序1修改后运行结果

**2．程序修改替换题**

（1）根据将实验3-1改错题程序中sum\_fac函数修改为一个递归函数，用递归的方式计算。



**解答：**

替换后的sum\_ fac函数如下：

long long sum\_fac(int m)

{

for (i = 1; i <= m; i++)

{

fac \*= i;

}

s = fac+sum\_fac(m - 1);

}

return s;

}

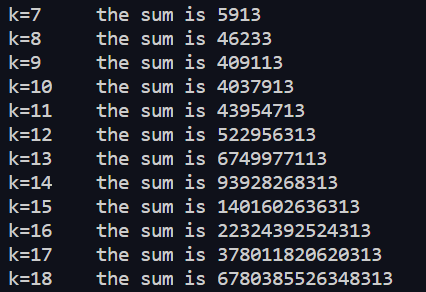
****

图3-2 程序2-1运行结果

（2）下面是计算的源程序，其中x是浮点数，n是整数。从键盘输入x和n，然后计算s的值。修改该程序中的sum和fac函数，使之计算量最小。



/\*实验3-2程序修改替换第(2)题程序：根据公式计算 s\*/

double sum(double x,int n)

{

int i;

double z=1.0;

for(i=1;i<=n;i++)

{

z=z+mulx(x,i)/fac(i);

}

return z;

}

double mulx(double x,int n)

{

int i;

double z=1.0;

for(i=0;i<n;i++)

{

z=z\*x;

}

return z;

}

long fac(int n)

{

int i;

long h=1;

for(i=2;i<=n;i++)

{

h=h\*i;

}

return h;

}

int main()

{

double x;

int n;

printf("Input x and n:");

scanf("%lf%d",&x,&n);

printf("The result is %lf:",sum(x,n));

return 0;

}

**解答：**

替换之后的sum和fac函数如下：

double sum(double x, int n)

{

c = sum(x, n - 1) + mulx(x, n) / fac(n);

long fac(int n)

c = fac(n - 1) \* n;

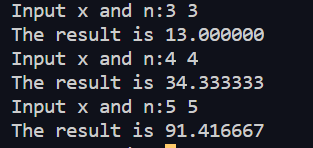


图3-3 程序2-2运行结果

**3．跟踪调试题**

下面是计算fabonacci数列前n项和的源程序，现要求单步执行该程序，在watch窗口中观察Ik,sum,n值。具体操作如下：

（1）设输入5，观察刚执行完“scanf("%d",&k);”语句时，sum、k的值是多少？

（2）在从main函数第一次进入fabonacci函数前的一刻，观察各变量的值是多少？返回后光条停留在哪个语句上？

（3）在从main函数第一次进入fabonacci函数后的一刻，观察光条从main函数“sum+=fabonacci(i);”语句调到了哪里？

（4）在fabonacci函数内部单步执行，观察函数的递归执行过程。体会递归方式实现的计算过程是如何完成数计算的，并特别注意什么时刻结束递归，然后直接从第一个return语句返回到了哪里？

（5）在fabonacci函数递归执行过程中观察参数n的变化情况，并回答为什么k、sum在fabonacci函数内部不可见？

**/\*实验3-3跟踪调试题程序：**计算fabonacci数列前n项和**\*/**

#include<stdio.h>

int main(void)

{

int i,k;

long sum=0,fabonacci(int n);

printf("Inut n:");

scanf("%d",&k);

for(i=1;i<=k;i++){

sum+=fabonacci(i);

printf("i=%d\tthe sum is %ld\n",i,sum);

}

return 0;

}

long fabonacci(int n)

{

if(n==1 || n==2)

return 1;

else

return fabonacci(n-1)+fabonacci(n-2);

}

**解答：**

1）

****

图3-4 程序3（1）调试结果

2）

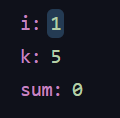


图3-5 程序3（2）调试结果

3）

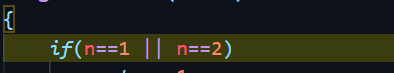


图3-6 程序3（2）调试结果

4）



图3-7 程序3（4）调试结果

5）

k和sum的作用域是main函数

**4．程序设计**

（1）编程验证歌德巴赫猜想：一个大于等于4的偶数都是两个素数之和。要求设计一个函数对其形参n验证哥德巴赫猜想，并以“n=n1+n2”的形式输出结果。例如：n=6，输出“6=3+3”。main函数循环接收从键盘输入的整数n，如果n是大于或等于4的偶数，调用上述函数进行验证。

**解答：**

1）流程图



图**3-8** 程序4-1流程图

2）源代码

void GDBH(int n)

{

int i, x, y;

for (i = 2; i < n; i++)

{

x = SUSHU(i);

if (x == 0)

continue;

y = SUSHU(n - x);

if (y == 0)

continue;

if(x<=(n-x))

printf("%d=%d+%d\n", n, x, y);

//break;

}

}

3）测试

（a）测试数据

56，24

（b）运行结果

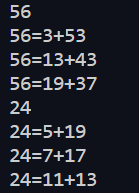
****

图3-9 程序4-1对应数据运行结果

（2）完全数（Perfect number），又称完美数或完备数，特点是它的所有真因子（即除了自身以外的约数，包括1）之和恰好等一它本身。例如6=1+2+3，28=1+2+4+7+14等。编程寻找108以内的所有完全数。要求设计一个函数，判定形参n是否为完全数，如果是，则以n的真因子之和的形式输出结果，例如“6=1+2+3”；否则，输出“not a perfect number”，例如“5 is not a perfect number”。

在main函数中调用该函数求108以内的所有完全数。

**解答：**

1）流程图



图3-10 程序4-2流程图

2）源代码

void KUAISU(long long n)

{

for (int i = 2; i < 20; i++)

{

SUSHU(pow(2, i) - 1);

if (SUSHU(pow(2, i) - 1) == 0)

continue;

n = pow(2, i - 1) \* (pow(2, i) - 1);

printf("%lld=1", n);

for (long long j = 2; j <= n/2; j++)

{

if (n % j == 0)

printf("+%lld", j);

if (j == n/2 )

printf("\n");

}

}

}

3）测试

（a）测试数据

无

（b）运行结果

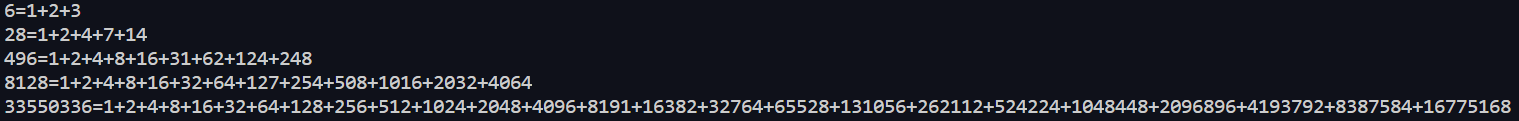
****

图3-11 程序4-2运行结果

（3）自幂数是指一个n位数，它的每个位上的数字的n次幂之和等于它本身。水仙花数是3位的自幂数，除此之外，还有4位的四叶玫瑰数、5位的五角星数、6位的六合数、7位的北斗星数、8位的八仙数等。编写一个函数，判断其参数n是否为自幂数，如果是，则返回1；否则，返回0。main函数能反复接收从键盘输入的整数k，k代表位数，然后调用上述函数求k位的自幂数，输出所有k位自幂数，并输出相应的信息，例如“3位的水仙花数共有4个153，370，371，407”。当k=0时程序结束执行。

**解答：**

1）流程图



图3-12 程序4-3流程图

2）源代码

void ZIMISHU(int n)

{

long long a = 0, b = 0, c = 0, d = 0, num = 0, s = 0, q = 0, q1;

long long A[10] = {0};

long long B[10] = {0};

a = pow(10, n - 1);

b = pow(10, n);

for (; a < b; a++)

{

num = a;

while (a > 0)

{

A[c] = a % 10;

a = a / 10;

c++;

}

3）测试

（a）测试数据

4，5，6

（b）运行结果

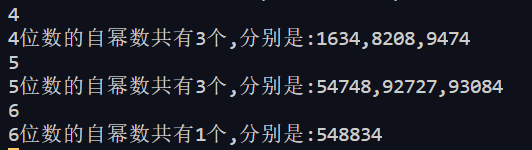
****

图3-13 对应测试数据运行结果

**5．自设题**

因为151即是一个质数又是一个回文数(从左到右和从右到左是看一样的)，所以 151 号是回文质数。  
写一个程序来找出范围[a,b](5 <= a < b <= 100,000,000)间的所有回文质数

**解答：**

1. 思路



图3-14 自设题流程图1

1. 源代码

int huiwen(int k) //判断回文数

{

int a[10], i = 0, j;

while (k > 0)

{

a[i] = k % 10;

k /= 10;

i++;

}

for (j = 0; j < i; j++)

if (a[j] != a[i - j - 1])

return 0;

return 1;

}

1. 测试
2. 测试数据

10 10000

1. 对应数据运行结果截图

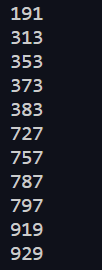


图3-16 自设题运行结果

**3.3 实验小结**

在本次实验中，我学会了如何声明以及调用函数，还有变量的存储类型以及他的作用域和生命周期。在实验过程中，有许多细节仍需要修改。Bug总是有许多，单步调试可以帮助我更好地分析程序以及修改bug。

**实验4 编译预处理实验**

**4.1实验目的**

（1）掌握文件包含、宏定义、条件编译和assert宏的使用；

（2）练习使用集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

（3）熟悉多文件编译技术

**4.2实验内容**

**1．程序改错**

下面是用宏来计算平方差、交换两数的源程序.在这个源程序中存在若干错误，要求对该程序进行调试修改，使之能够正确完成指定任务。

/\*实验4-1改错与跟踪调试题程序：计算平方差、将换两数\*/

1 #include<stdio.h>

2 #define SUM a+b

3 #define DIF a-b

4 #define SWAP(a,b) a=b,b=a

5 int main()

6 {

7 int a,b;

8 printf("Input two integers a, b:");

9 scanf("%d%d", &a,&b);

10 printf("\nSUM=%d\n the difference between square of a and square of b is:%d",SUM, SUM\*DIF);

11 SWAP(a,b);

12 printf("\nNow a=%d,b=%d\n",a,b);

13 return 0;

14 }

**解答：**

1. 错误修改
2. 第2行a+b改为（a+b）
3. 第3行a-b改为（a-b）
4. 第4行a=b,b=a改为t=a,a=b,b=t
5. 第7行加上int t;
6. 修改后运行截图

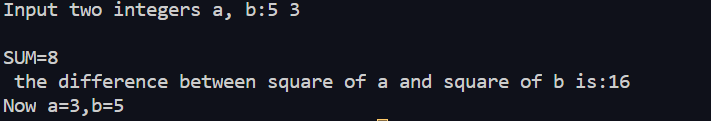


图4-1 程序修改后运行结果

**2．程序修改替换**

下面是用函数实现求三个数中最大数、计算两浮点数之和的程序。在这个源程序中存在若干语法和逻辑错误。

要求：（1）对这个例子程序进行调试修改，使之能够正确完成指定任务；

（2）用带参数的宏替换函数max，来实现求最大数的功能。

/\*实验4-2程序修改替换题程序\*/

#include<stdio.h>

int main(void)

{

int a, b, c;

float d, e;

printf("Input three integers:");

scanf("%d %d %d",&a,&b,&c);

printf("\nThe maximum of them is %d\n",max(a,b,c));

printf("Input two floating point numbers:");

scanf("%f %f",&d,&e);

printf("\nThe sum of them is %f\n",sum(d,e));

return 0;

}

int max(int x, int y, int z)

{

int m=z;

if (x>y)

if(x>z) m=x;

else

if(y>z) m=y;

return m;

}

float sum(float x, float y)

{

return x+y;

}

**解答：**

1）错误修改

（a）提前声明函数

（b）If和else后加大括号

2）修改后运行截图

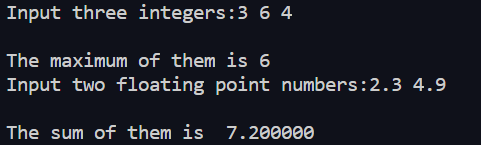


图4-2 程序修改后运行结果

（2）

1）替换后的程序

#define max(a,b,c) (a)>(b)?((a)>(c)?(a):(c)):((b>c)?(b):(c))

2）运行结果截图

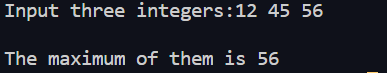


图4-3 程序修改后运行结果

**3．跟踪调试**

下面程序利用R计算圆的面积s，以及面积s的整数部分。现要求：

（1）修改程序，使程序编译通过且能运行；

（2）单步执行。进入函数integerl\_fraction时，watch窗口中x为何值？在返回main时, watch窗口中i为何值？

（3）修改程序，使程序能输出面积s值的整数部分（要求四舍五入），不会输出错误信息assertion failed。

/\*实验4-3跟踪调试题程序利用R计算圆的面积s\*/

#define R

int main(void)

{

float r, s;

int s\_integer=0;

printf ("Input a number: ");

scanf("%f",&r);

#ifdef R

s=3.14159\*r\*r;

printf("Area of round is: %f\n",s);

s\_integer=integer\_fraction(s);

assert((s-s\_integer)<0.5);

printf("The integer fraction of area is %d\n", s\_integer);

#endif

return 0;

}

int integer\_fraction(float x)

{

int i=x;

return i;

}

**解答：**

（1）

1 ）修改后的程序

#define R

#include<assert.h>

#include<stdio.h>//1

int integer\_fraction(float x);//2

）运行结果

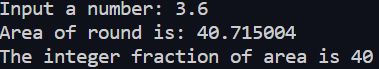


图4-4 程序修改后运行结果

（2）以输入r=3.6为例

进入函数时：



图4-5 变量x的值

返回main时：



图4-6 变量i的值

（3）

1）修改后程序

#define R

#include<assert.h>

#include<stdio.h>//1

int integer\_fraction(float x);//2

int integer\_fraction(float x)

{

int i=x;

if((x-(int)x)<0.5)

return i;

else

return i+1;

}

2）运行结果

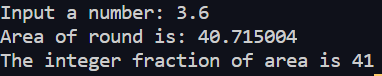


图4-7 程序运行结果

**4．程序设计**

（1）三角形的面积是，其中，a,b,c为三角形的三边，要求编写程序用带参数的宏来计算三角形的面积。定义两个带参数的宏，一个用来求s，另一个用来求area。



**解答：**

1. 流程图



1. 源代码

#include <math.h>

#include <stdio.h>

#define max(a, b, c) (a) >= (b) ? ((a) >= (c) ? (a) : (c)) : ((b >= c) ? (b) : (c))

#define min(a, b, c) (a) <= (b) ? ((a) <= (c) ? (a) : (c)) : ((b <= c) ? (b) : (c))

#define mid(a, b, c) (a) >= (b) ? ((a) <= (c) ? (a) : ((b) >= (c) ? (b) : (c))) : ((a) >= (c) ? (a) : ((b) <= (c) ? (b) : (c)))

#define s(a, b, c) ((double)(a + b + c) / 2.0)

#define area(a, b, c, s) sqrt(s \*(s - a) \* (s - b) \* (s - c))

3）测试

（a）测试数据

1，2，3

6，9，12

（b）对应数据运行截图

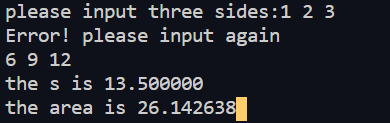


图4-8 程序运行结果

（2）用条件编译方法来编写程序。输入一行英文字符序列，可以任选两种方式之一输出：一为原文输出；二为变换字母的大小写后输出。例如小写‘a’变成大写‘A’，大写‘D’变成小写‘d’，其他字符不变。用#define命令控制是否变换字母的大小写。例如，#define CHANGE 1 则输出变换后的文字，若#define CHANGE 0则原文输出。

**解答：**

1. 流程图



1. 源代码

#include <stdio.h>

#include <string.h>

#define CHANGE 1

#define n 100

#if CHANGE

for (int i = 0; i < len; i++)

{

s[i] = change(s[i]);

}

printf("%s", s);

#else

printf("%s", s);

#endif

}

1. 测试
2. 测试数据

asdfQWER@#&

1. 对应数据运行结果截图



图4-9 程序运行结果

**5．选做题**

假设一个C程序由file1.c和file2.c两个源文件及一个file.h头文件组成，file1.c、file2.c和file.h的内容分别如下所述。试编辑该多文件C程序，补充file.h头文件内容，然后编译和链接。然后运行最后生成的可执行文件。

/\*源文件file1.c的内容\*/

#include "file.h"

int x,y; /\* 外部变量的定义性说明 \*/

char ch; /\* 外部变量的定义性说明 \*/

int main(void)

{

x=10;

y=20;

ch=getchar();

printf("in file1 x=%d,y=%d,ch is %c\n",x,y,ch);

func1();

return 0;

}

/\*源文件file2.c的内容为：\*/

#include "file.h"

void func1(void)

{

x++;

y++;

ch++;

printf("in file2 x=%d,y=%d,ch is %c\n",x,y,ch);

}

**解答：**

1. 思路

在file.h文件中使用条件编译

1. 经过编辑的代码

#include<stdio.h>

#ifndef \_FILE\_H

#define \_FILE\_H

#include"file2.c"

#else

extern int x, y;

extern char ch;

#endif

1. 测试
2. 测试数据

a，D

1. 对应数据运行结果

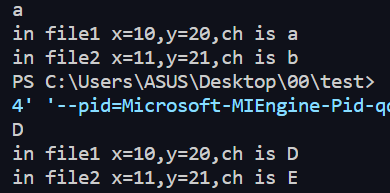


图4-10 程序运行结果

**6.自设题**

问题描述：给定n种不同价值，不同重量的物品和一背包。物品i的重量为wi，价值为vi，背包容量为c。选择装入背包的物品，使物品的总价值最大。

解答：

1. 思路
2. 最优解结构

设（x1，x2，…，xn）是背包问题的一个最优解，则（x2，…，xn）是子问题“将物品2，3，…，n放入容量为c-w1x1的背包”的一个最优解。子问题可描述如下：

其最优值定义为m（i，j）

1. 最优值递归定义

第i件物品可选择放或不放。若放，问题转化为m（i+1，j-wi）；不放，问题转换为m（i+1，j）。可建立如下的递归式：

边界：

1. 计算最优值

需要一个二维数组d存储所有子问题的最优值，将m（i，j）存入d（i，j），初始时，将d中所有元素设置为-1。可不用重复计算

1. 构造最优解

步骤如下：

1. i=1，从m（1，c）开始，决策物品1
2. 如果m（i，c）=m（i+1，c），则x[i]=0；否则，x[i]=1。此时，c=c-w[i]
3. 决策下一个物品，i=i+1，重复步骤b，直到i=n-1
4. 决策最后一个物品n，直接由m（n，c）是否为0确定x[n]
5. 源代码

#include <stdio.h>

#define N 5

#define C 10

#define max(a, b) ((a) > (b) ? (a) : (b))

int m(int i, int j);

void printans(int c, int n);

int w[N + 1] = {0, 2, 3, 6, 4, 5}; //重量

int v[N + 1] = {0, 6, 3, 5, 4, 2}; //价值

int d[N + 1][C + 1]; //子问题最优值

int x[N + 1]; //最优方案

//记忆化搜索计算m(i,j)

int m(int i, int j)

{

if (d[i][j] >= 0)

return d[i][j]; //避免重复

if (i == N+1)

return (d[i][j] = (j >= w[i] ? v[i] : 0)); //已到终点

if (j < w[i])

return (d[i][j] = m(i + 1, j));

//放不下第i个物品，继续搜索

return (d[i][j] = max(m(i + 1, j), m(i + 1, j - w[i]) + v[i])); //放得下，进入递归比较

}

//构造最优值并输出

1. 测试
2. 测试数据

N=5，c=10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| w | 2 | 2 | 3 | 5 | 4 |
| v | 6 | 3 | 6 | 4 | 1 |

表4-1 自设题测试数据

1. 运行结果截图



图4-11 程序运行结果

**4.3实验小结**

在这次实验中，我学会了预编译处理。尤其是在处理多文件c程序时，预编译具有明显的优势。在修改常量时，使用宏定义就很方便。另外，使用条件编译可以让程序有更多的可选择性。

**实验5 数组实验**

**5.1 实验目的**

（1）掌握数组的说明、初始化和使用。

（2）掌握一维数组作为函数参数时实参和形参的用法。

（3）掌握字符串处理函数的设计，包括串操作函数及数字串与数之间转换函数实现算法。

（4）掌握基于分治策略的二分查找算法和选择法排序算法的思想，以及相关算法的实现。

**5.2 实验内容及要求**

**1、 源程序改错与跟踪调试**

在下面所给的源程序中，函数strcate(t,s)的功能是将字符串s连接到字符串t的尾部；函数strdelc(s,c)的功能是从字符串s中删除所有与给定字符c相同的字符，程序应该能够输出如下结果：

Programming Language

ProgrammingLanguage Language

ProgramingLnguage

跟踪和分析源程序中存在的问题，排除程序中的各种逻辑错误，使之能够输出正确的结果。

1. 单步执行源程序。进跟踪进入strcate时，观察字符数组t和s中的内容，分析结果是否正确。当单步执行光条刚落在第二个while语句所在行时，i为何值？t[i]为何值？分析该结果是否存在问题。当单步执行光条落在strcate函数块结束标记即右花括号“}”所在行时，字符数组t和s分别为何值？分析是否实现了字符串连接。

（2） 跟踪进入函数strdelc时，观察字符数组s中的内容和字符c的值，分析结果是否正确。单步执行for语句过程中，观察字符数组s, j和k值的变化，分析该结果是否存在问题。当单步执行光条落在strdelc函数块结束标记“}”所在行时，字符串s为何值？分析是否实现了所要求的删除操作。

/\*实验5-1程序改错与跟踪调试题程序\*/

#include<stdio.h>

void strcate(char [],char []);

void strdelc(char [],char );

int main(void)

{

char a[]="Language", b[30]="Programming";

printf("%s %s\n", b,a);

strcate(b,a); printf("%s %s\n",b,a);

strdelc(b, 'a'); printf("%s\n",b);

return 0;

}

void strcate(char t[],char s[])

{

int i = 0, j = 0;

while(t[i++]) ;

i=i-1;

while((t[i++] = s[j++] )!= '\0');

}

void strdelc(char s[], char c)

{

int j,k;

for(j=k=0; s[k] != '\0'; j++)

if(s[j] != c) s[k++] = s[j];

}

解答：

1. I是12，t[i]是76L，不合适，溢出。未实现。T80s76，未实现。
2. 存在问题。S为0，未实现。
3. 修改如下：见源程序高亮字体。
4. 运行结果截图

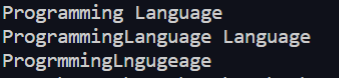


图5-1 程序运行结果

**2、 源程序完善和修改替换**

(1) 下面的源程序用于求解瑟夫问题：M个人围成一圈，从第一个人开始依次从1至N循环报数，每当报数为N时报数人出圈，直到圈中只剩下一个人为止。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。

#include<stdio.h>

#define M 10

#define N 3

int main(void)

{

int a[M], b[M]; /\* 数组a存放圈中人的编号，数组b存放出圈人的编号 \*/

int i, j, k;

for(i = 0; i < M; i++) /\* 对圈中人按顺序编号1—M \*/

a[i] = i + 1;

for(i = M, j = 0; i > 1; i--){

/\* i表示圈中人个数，初始为M个，剩1个人时结束循环；j表示当前报数人的位置 \*/

for(k = 1; k <= N; k++) /\* 1至N报数 \*/

if(++j > i - 1) j = 0;/\* 最后一个人报数后第一个人接着报，形成一个圈 \*/

b[M-i] = j ? \_b[M-1]=a[j-1]\_:\_ b[M-1]=a[i-1]\_\_; /\* 将报数为N的人的编号存入数组b \*/

if(j)

for(k = --j; k < i; k++) /\* 压缩数组a，使报数为N的人出圈 \*/

\_\_\_a[k]=a[k+1]\_\_\_\_\_\_\_\_\_\_\_;

}

for(i = 0;i < M-1; i++) /\* 按次序输出出圈人的编号 \*/

printf(“%6d”, b[i]);

printf(“%6d\n”, a[0]); /\* 输出圈中最后一个人的编号 \*/

return 0;

}

②上面的程序中使用数组元素的值表示圈中人的编号，故每当有人出圈时都要压缩数组，这种算法不够精炼。如果采用做标记的办法，即每当有人出圈时对相应数组元素做标记，从而可省掉压缩数组的时间，这样处理效率会更高一些。请采用做标记的办法修改程序，并使修改后的程序与原程序具有相同的功能。

解答：

1. 思路：

1.定义两个数组，一个存出圈人的编号，另一个做标记。

2.大循环循环一次出圈同时一个标记一个。

3.小循环开始报数跳过被标记的元素，报到N结束。

4.小循环结束后进行存号和标记操作

5.大循环结束后进行循环打印操作。

2）源代码

#include <stdio.h>

#define M 3

#define N 3

int main(void)

{

int a[M], b[M + 1] = {0}; //a数组存出圈人的编号，b数组标记

int i, j, k;

for (i = M, j = 1; i >= 1; i--) //每次标记一个人，直到剩一个人时结束循环

{

for (k = 1; k <= N;) //未标记的人报数

{

if (!b[j]) //跳过标记

{

k++; //有标记，不自加

}

if (++j > M) //j=M时，j=1重新循环

{

j = 1;

}

}

if (j == 1)

b[M] = 1;

else

b[j - 1] = 1; //标记出圈人，

a[M - i] = (j == 1 ? M : (j - 1)); //j=1说明到恰好是第M个人报到N，否则是编号为j-1的人

}

for (i = 0; i < M - 1; i++) /\* 按次序输出出圈人的编号 \*/

printf("%6d", a[i]);

printf("\n%6d", j - 1); /\* 输出圈中最后一个人的编号 \*/

return 0;

}

1. 测试
2. 测试数据

M=12，N=3

1. 运行结果



图5-2 程序运行结果

**3、 程序设计**

（1）输入一个整数，将它在内存中二进制表示的每一位转化成对应的数字字符并且存放到一个字符数组中，然后输出该整数的二进制表示。

解答：

1. 流程图：



图5-3 程序3-1流程图

1. 源代码

void transform(int x,char a[])

{

int i = 1, j = 0;

a[0] = '\0';

while(x!=0)

{

a[i++] = x % 2+48;

x = x / 2;

}

char b[100];

for (j = 0; j < i;j++)

{

b[j] = a[i - j-1];

}

printf("%s", b);

}

1. 测试
2. 测试数据

12，96，486

1. 运行结果截图

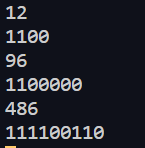


图5-4 程序3-1运行结果

（2）编写一个C程序，要求采用模块化程序设计思想，将相关功能用函数实现，并提供菜单选项。该程序具有以下功能：

①输入n个学生的姓名和C语言课程的成绩。

②将成绩按从高到低的次序排序，姓名同时进行相应调整。

③输出所有学生的姓名和C语言课程的成绩。

解答：

1. 流程图



图5-5 程序3-2流程图

1. 源代码

int input(void);

void dictionarysort(int len);

void compare(char a[], char b[][m], int len);

void output(int len, int a[], char[][m]);

void bubblesort(int a[], int len);

void quicksort(int a[], int left, int right);

int part(int a[], int left, int right);

void swap(int a[], int i, int j);

int input(void)

{

int i = 0;

printf("please input students' mark:\ninput 'ctrl^z' to finish\n");

while (scanf("%d", &mark[i++]) != EOF)

;

printf("please input students' name:\n");

for (int j = 0; j < i - 1; j++)

gets(name[j]);

return i - 1;

}

void output(int len, int a[], char b[][m])

{

for (int i = len - 1; i >= 0; i--)

{

printf("%s:", b[i]);

printf("%d\n", a[i]);

}

}

void bubblesort(int a[], int len)

{

for (int i = 0; i < len - 1; i++)

{

for (int j = i + 1; j < len; j++)

{

if (a[i] > a[j])

swap(a, i, j);

}

}

}

void dictionarysort(int len)

{

int i, j;

for (i = 0; i < len - 1; i++)

{

if (mark[i] == mark[i + 1])

{

for (j = 0; j < m; j++)

{

if (name[i][j] < name[i + 1][j])

{

char t[m];

strcpy(t, name[i]);

strcpy(name[i], name[i + 1]);

strcpy(name[i + 1], t);

break;

}

}

}

}

}

1. 测试
2. 测试数据

Mark：12 45 23 34 56 34

Name：mike lee juicy tyler lucy steven

1. 运行结果

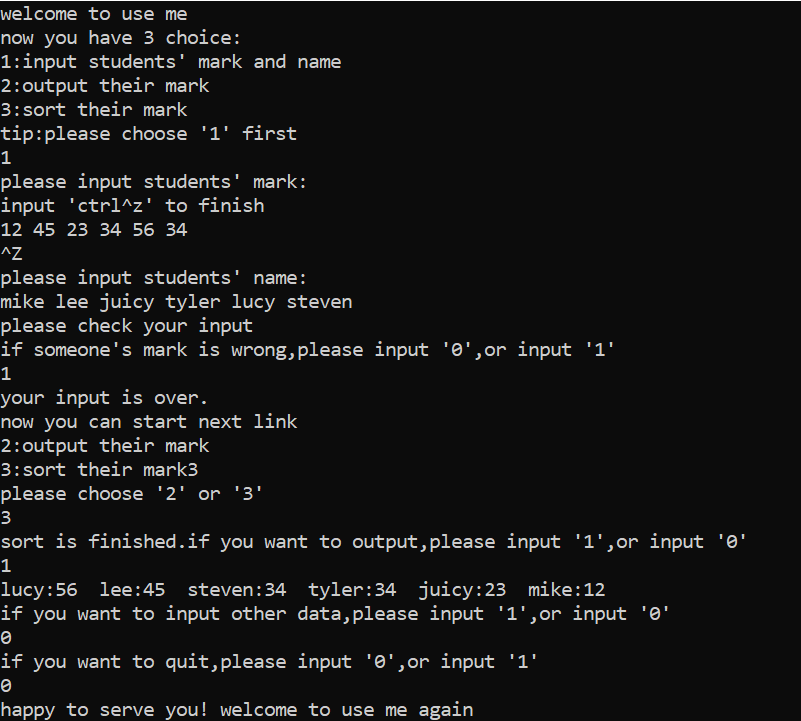


图5-6 程序3-2运行结果

（3）对程序设计第（2）题的程序增加查找功能：输入一个C语言课程成绩值，用二分查找进行搜索。如果查找到有该成绩，则输出该成绩学生的姓名和C语言课程的成绩；否则，输出提示“not found!”。

1）流程图

见上题

图5-7 程序3-3流程图

2)源代码

int binarysearch(int a[], int x, int len)

{

int front = 0, back = len - 1, middle;

while (front <= back)

{

middle = (front + back) / 2;

if (x < a[middle])

back = middle - 1;

else if (x > a[middle])

front = middle + 1;

else

{

int temp = middle;

while (middle >= 0)

{

middle--;

if (x != a[middle])

break;

}

left1 = middle + 1;

middle = temp;

while (middle <= len - 1)

{

middle++;

if (x != a[middle])

break;

}

right1 = middle - 1;

return 0;

}

}

return -1;

}

1. 测试
2. 测试数据

Mark：62 23 97 56 73 45 63 85 56 34 56

Name：ljl msk nyx xmz yyl zhm zjw zxz czt hsw hyf

1. 运行结果

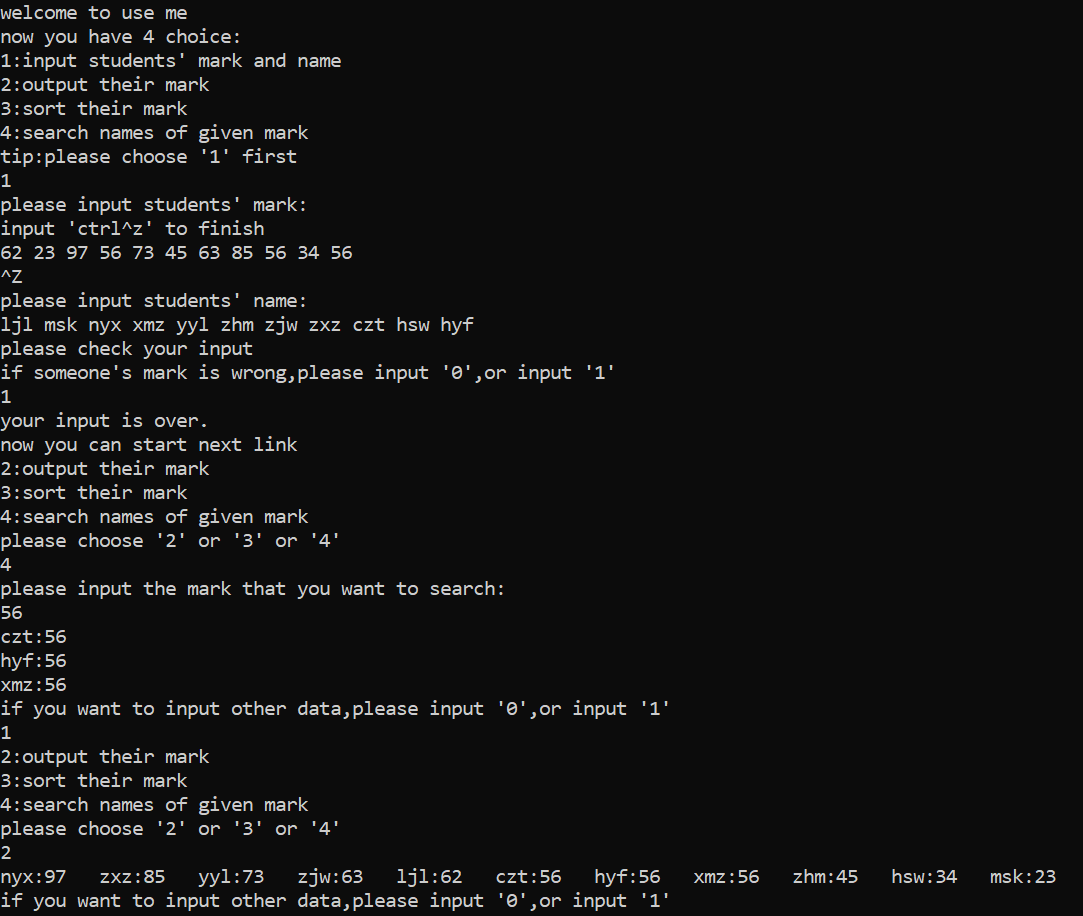


图5-8 程序3-3运行结果

**4、 程序设计选做题**

编写并上机调试运行能实现以下功能的函数和程序。

编写函数strnins(s,t,n),其功能是：可将字符数组 t中的字符串插入到字符数组 s中字符串的第n个字符的后面。解答：

1. 流程图



图5-9 选做题流程图

1. 源代码

//编写函数strnins(s,t,n),其功能是：可将字符数组 t中的字符串插入到字符数组 s中字符串的第n个字符的后面。

#include <stdio.h>

#include <string.h>

#define m 100

void strnins(char s[], char t[], int n)

{

int i = 0, j = 0;

int len = strlen(t);

int len1 = strlen(s);

for (i = len1; i >= n; i--)

{

s[i + len] = s[i];

}

for (i = n, j = 0; j < len; i++, j++)

{

s[i] = t[j];

}

s[len1 + len + 1] = '\0';

printf("%s", s);

}

1. 测试
2. 测试数据

S：qwerty

T：asdf

N：3

1. 运行结果

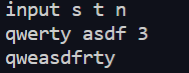


图5-10 选做题运行结果

**5、 自设题**

题目：在8×8格的国际象棋上摆放8个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。

**解答：**

1. 思路

dfs深度优先搜索

放一个皇后，调用检查函数检查一次

数组中存入每一行皇后的位置

依次输出8\*8的图，皇后用#表示

每输出一个正确结果，计数加一，最后输出总数

1. 源代码

#include <stdio.h>

int queens[8]={0},counts=0;

int check(int line,int list){

void eight\_queen(int line){

//在数组中为0-7列

for (int list=0; list<8; list++) {

//对于固定的行列，检查是否和之前的皇后位置冲突

if (check(line, list)) {

//不冲突，以行为下标的数组位置记录列数

queens[line]=list;

//如果最后一样也不冲突，证明为一个正确的摆法

if (line==7) {

//统计摆法的counts加1

counts++;

//输出这个摆法

print();

//每次成功，都要将数组重归为0

queens[line]=0;

return;

}

//继续判断下一样皇后的摆法，递归

eight\_queen(line+1);

//不管成功失败，该位置都要重新归0，以便重复使用。

queens[line]=0;

}

}

}

1. 测试

运行结果截图：

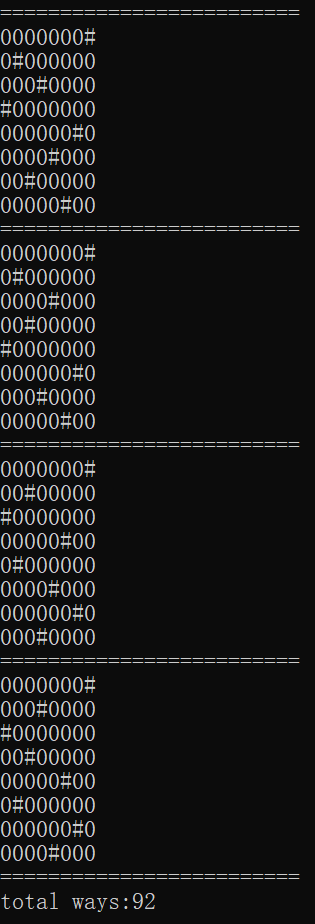


图5-11 自设题运行结果

**5.3 实验小结**

在本次实验中，我学会了如何定义、使用数组，以及数组在内存空间的存储形式。数组作为一种基本数据结构，可以实现许多操作，比如字符串的查找、复制、插入等等。在使用数组时，尤其要注意溢出的情况，因为溢出时程序并不会报错。还学了几个简单的算法，二分查找和选择排序。

**实验6 指针实验**

**6.1、实验目的**

（1）熟练掌握指针的说明、赋值、使用。

（2）掌握用指针引用数组的元素，熟悉指向数组的指针的使用。

（3）熟练掌握字符数组与字符串的使用，掌握指针数组及字符指针数组的用法。

（4）掌握指针函数与函数指针的用法。

（5）掌握带有参数的main函数的用法。

**6.2、实验题目及要求**

**1、源程序改错题**

在下面所给的源程序中，函数strcopy(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

#include<stdio.h>

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

int main(void)

{

char \*s1, \*s2, \*s3=NULL,a[20],b[20];

s1=a,s2=b;

printf("Input a string:\n", s2);

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)

{

Int ans;

while(\*t++ = \*s++)ans++;

while(ans--)t--;

return (t);

}

解答：

1. 修改见高亮处
2. 运行结果截图：

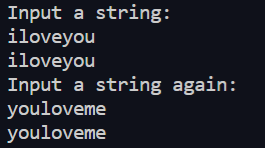


图6-1 程序运行结果

**2、源程序完善、修改替换题**

（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。

#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-j-1; j++)

if (\_\_\*s[j]>\*s[j+1]\_\_\_\_\_\_\_\_\_)

{

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(strlen(t)+1);

strcpy(\_s[i]=t\_\_\_\_\_);

}

strsort(\_\_s,N\_\_\_\_\_\_);

for (i=0; i<N; i++) puts(s[i]);

return 0;

}

解答：

1）修改见高亮处

2）运行结果截图：

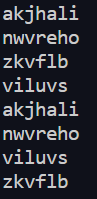


图6-2 程序运行结果

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。

解答：代码如下：

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 (\*\*(s+j)>\*\*(s+j+1))

{

temp = \*(s+j);

\*(s+j)=\*(s+j+1);

\*(s+j+1) = temp;

}

}

}

}

1. 运行结果截图：

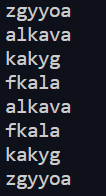


图6-3 程序运行结果

（2）下面源程序通过函数指针和菜单选择来调用库函数实现字符串操作；串复制strcpy、串连接strcat或串分解strtok。

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，使之能按照要求输出下面结果：

1 copy string.

2 connect string.

3 parse string.

4 exit.

input a number (1-4) please!

2↙ （键盘输入）

input the first string please!

the more you learn,↙ （键盘输入）

input the second string please!

the more you get. ↙ （键盘输入）

the result is the more you learn, the more you get.

# include<stdio.h>

# include<string.h>

int main (void)

{

\_\_\_\_\_char \*(\*p)(char \*,const char \*)=NULL\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

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");

\_\_\_\_\_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;

}

解答：

1. 见高亮处
2. 运行结果截图

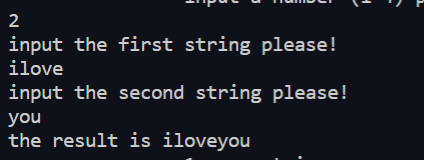


图6-4 程序运行结果

②函数指针的一个用途是用户散转程序，即通过一个转移表（函数指针数组）来实现多分枝函数处理，从而省去了大量的if语句或者switch语句。转移表中存放了各个函数的入口地址（函数名），根据条件的设定来查表选择执行相应的函数。请使用转移表而不是switch语句重写以上程序。

解答：

1. 代码如下：

# include<stdio.h>

# include<string.h>

int main (void)

{

char \*(\*p[4])(char \*, const char \*);

char a[80], b[80], \*result = NULL;

p[1] = strcpy;

p[2] = strcat;

p[3] = strtok;

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);

if(choice==4)

goto down;

getchar();

printf("input the first string please!\n");

scanf("%s",a);

printf("input the second string please!\n");

scanf("%s",b);

result = p[choice](a, b);

printf("the result is %s\n", result);

}

down:

return 0;

}

1. 运行结果截图：

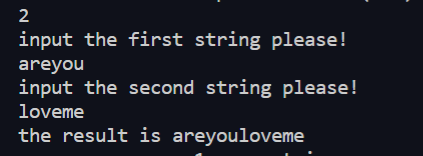


图6-5 程序运行结果

**3、跟踪调试题**

请按下面的要求对源程序进行操作，并回答问题和排除错误。

（1）单步执行。进入strcpy时watch窗口中s为何值？返回main时, watch窗口中s为何值？

（2）排除错误，使程序输出结果为：there is a boat on the lake.

#include "stdio.h"

char \*strcpy(char \*,char \*);

void main(void)

{

char a[20],b[60]="there is a boat on the lake.";

printf("%s\n",strcpy(a,b));

}

char \*strcpy(char \*s,char \*t)

{

Int ans;

while(\*s++=\*t++)

ans++;

while(ans--)

s--;

return (s);

}

解答：

1. 进入strcpy时：



图6-6 调试结果

返回main时：



图6-7 调试结果

1. 修改见高亮
2. 运行结果截图



图6-8 程序运行结果

**4、编程设计题**

（1）一个长整型变量占4个字节，其中每个字节又分成高4位和低4位。试从该长整型变量的高字节开始，依次取出每个字节的高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。

（2）旋转是图像处理的基本操作，编程实现一个将一个图像逆时针旋转90°。提示：计算机中的图像可以用一个矩阵来表示，旋转一个图像就是旋转对应的矩阵。将旋转矩阵的功能定义成函数（这个函数的形参必须有一个是指向数组元素的指针，其它形参随意），通过使用指向数组元素的指针作为参数使该函数能处理任意大小的矩阵。要求在main函数中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°后的矩阵。例如，输入：

2 3

1 5 3

3 2 4

则输出：

3 4

5 2

1 3

（3）输入n行文本，每行不超过80个字符，用字符指针数组指向键盘输入的n行文本，且n行文本的存储无冗余，删除每一行中的前置空格（' '）和水平制表符（'\t'）。要求：将删除一行文本中前置空格和水平制表符的功能定义成函数，在main函数中输出删除前置空格符的各行。

（4）编写8个任务函数，一个scheduler调度函数和一个execute执行函数。仅在main函数中调用scheduler函数，scheduler函数要求用最快的方式调度执行用户指定的任务函数。

①先设计task0, task1, task2, task3, task4, task5, task6, task7共8个任务函数，每个任务函数的任务就是输出该任务被调用的字符串。例如，第0个任务函数输出“task0 is called!”，第1个任务函数输出“task1 is called!”，以此类推。

②scheduler函数根据键盘输入的数字字符的先后顺序，一次调度选择对应的任务函数。例如，输入：1350并回车，则scheduler函数一次调度选择task1, task3, taks5, task0，然后以函数指针数组和任务个数为参数将调度选择结果传递给execute函数并调用execute函数。

③execute函数根据scheduler函数传递的指针数组和任务个数为参数，按照指定的先后顺序依此调用执行选定的任务函数。

例如，当输入13607122并回车，程序运行结果如下：

task1 is called!

task3 is called!

task6 is called!

task0 is called!

task7 is called!

task1 is called!

task2 is called!

task2 is called!

1. 解答：
2. 流程图



图6-9 程序4-1流程图

1. 源代码

#include<stdio.h>

int main(void)

{

unsigned long x=564767342;

unsigned char \*p=(unsigned char\*)&x;

p += 3;

for (int i = 4; i > 0;i--)

{

unsigned int temp1,temp2;

temp1 =temp2= \*p;

printf("%x %x\n", temp1>>4,temp2&0xf);

--p;

}

return 0;

}

1. 测试
2. 测试数据

564767342

1. 运行结果截图

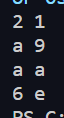


图6-10 程序运行结果

1. 解答：
2. 流程图



图6-11 程序4-2流程图

1. 源代码

#include <stdio.h>

void rotate(int \*p, int n, int m)

{

int b[m + 1][n + 1];

for (int i = 1; i < n + 1; i++)

{

for (int j = 1; j < m + 1; j++)

b[m - j + 1][i] = \*(p + (i - 1) \* (m + 1) + (j - 1));

}

for (int i = 1; i < m + 1; i++)

{

for (int j = 1; j < n + 1; j++)

printf("%d ", b[i][j]);

printf("\n");

}

}

3）测试

1. 测试数据

2，3

1，2，3

4，5，6

1. 运行结果截图

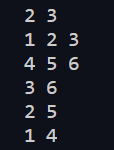


图6-12 程序运行结果

1. 解答：
2. 流程图



图6-13 程序4-3流程图

1. 源代码

#include <stdio.h>

#include<string.h>

#include<stdlib.h>

#define N 100

char \*delete (char \*p);

int main(void)

{

char \*p[N];

char s[80];

int i = 0, ans = 0;

while(gets(s))

{

p[i] = (char \*)malloc(strlen(s)+1);

strcpy(p[i], s);

p[i] = delete (p[i]);

i++;

ans++;

}

for (int i = 0; i < ans; i++)

printf("%s\n", p[i]);

}

1. 测试
2. 测试数据

jhykuf hjhv

mgndt jjgy

1. 运行结果截图



图6-14 程序运行结果

1. 解答：
2. 流程图



图6-15 程序4-4流程图

1. 源代码

#include<stdio.h>

void scheduler(char \*p);

void execute(void (\*p[])(void), int n);

int main(void)

{

char s[100];

scanf("%s", s);

scheduler(s);

}

void scheduler(char \*p)

{

int n = strlen(p);

void (\*pr[n])(void);

for (int i = 0; i < n;i++)

{

int temp = \*(p + i) - '0';

switch(temp)

{

case 0:pr[i] = task0;break;

}

}

execute(pr, n);

}

void execute(void (\*p[])(void), int n)

{

for (int i = 0; i < n;i++)

{

p[i]();

}

}

void task0(void)

{

printf("task0 is called!\n");

}

1. 测试
2. 测试数据

1433223

1. 运行结果截图

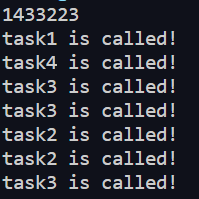


图6-16 程序运行结果

**5、选做题**

（1）设有N位整数和M位小数（N=20，M=10）的数据a,b。编程计算a+b并输出结果。(M、N的值可固定为N=20，M=10，也可设计为可变值。)

如：12345678912345678912.1234567891 + 98765432109876543210.0123456789

（2）编写使用复杂声明char \*(\*p[2])(const char \*,const char \*);的程序。

提示：p中元素可为strstr、strcpy等函数名。

（3）指定main函数的参数

在IDE（比如DevC++）中，选择“**运行**”｜“**参数**”菜单，在 “**传递给主程序的参数**”文本框中输入main函数的参数arg1 arg2 arg3，只输入命令行中文件名后的参数，文件名不作为参数输入，参数间以空格隔开。编写程序在命令行输出这三个参数。（注意不同IDE输入参数的方式不相同，可参考各个IDE的使用手册。）

1. 解答：
2. 流程图



图6-17 程序5-1流程图

1. 源代码

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define M 10

#define N 20

int main(void)

{

for (int i = M + N + 1; i >= 0; i--)

{

if (i == M + N + 1)

{

s3[i] = ((\*p1 - '0') + (\*p2 - '0')) % 10 + '0';

temp = ((\*p1 - '0') + (\*p2 - '0')) / 10;

p1--;

p2--;

continue;

}

if (i == N + 1)

{

s3[i] = '.';

p1--;

p2--;

continue;

}

if (i == 0)

{

s3[i] = temp + '0';

continue;

}

s3[i] = ((\*p1 - '0') + (\*p2 - '0')) % 10 + temp + '0';

temp = ((\*p1 - '0') + (\*p2 - '0')) / 10;

p1--;

p2--;

}

if (s3[0] == '0')

p3++;

printf("%s", p3);

return 0;

}

1. 测试
2. 测试数据

99999999999999999999.9999999999

99999999999999999999.9999999999

1. 运行结果

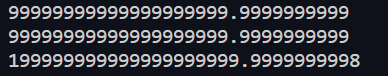


图6-18 程序运行结果

1. 解答：
   1. 思路

函数实现字符串的复制和连接作用

2）源代码

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int main(void)

{

char \*(\*p[2])(char \*, const char \*);

p[0] = strcpy;

p[1] = strcat;

char \*pr1,\*pr2,s[100];

pr1=(char \*)malloc(strlen(s)+1);

pr2=(char \*)malloc(strlen(s)+1);

scanf("%s", s);

p[0](pr1, s);

printf("%s\n", pr1);

scanf("%s", s);

p[0](pr2, s);

printf("%s", p[1](pr1, pr2));

return 0;

}

3）运行结果

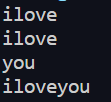


图6-19 程序运行结果

1. 解答：
   1. 思路

直接改参数

2）源代码

#include <stdio.h>

int main(int argc, char \*argv[])

{

printf("%s %s %s", argv[1], argv[2], argv[3]);

return 0;

}

3）运行结果



图6-20 程序运行结果

**6.3、实验小结**

通过指针的学习，我了解到了c语言中变量以及函数的内存。通过指针的使用，可以快速访问内存地址。数组和指针的互通可以灵活处理各种情况。复杂类型说明可以嵌套各种结构，使得c语言富有变化。

**实验7 结构与联合实验**

**7.1实验目的**

1．通过实验，熟悉和掌握结构的说明和引用、结构的指针、结构数组、以及函数中使用结构的方法。

2．通过实验，掌握动态储存分配函数的用法，掌握自引用结构，单向链表的创建、遍历、结点的增删、查找等操作。

3．了解字段结构和联合的用法。

**7.2实验题目及要求**

**1．表达式求值的程序验证题**

设有说明：

char u[]="UVWXYZ";

char v[]="xyz";

struct T{

int x; char c; char \*t;

}a[]={{11,ˊAˊ,u},{100, ˊBˊ,v}},\*p=a;

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

解答：

1）程序

#include<stdio.h>

int main(void)

{

char u[]="UVWXYZ";

char v[]="xyz";

struct T{

int x;

char c;

char \*t;

}a[]={{11,'A',u},{100, 'B',v}},\*p=a;

//printf("%d", (++p)->x);

//p++;

//printf("%c",p->c);

//\*p++->t;

//printf("%c", \*p->t);

//printf("%c", \*(++p)->t);

//printf("%c", \*++p->t);

printf("%c", ++\*p->t);

return 0;

}

2）运行结果

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **表达式** | **计算值** | **验证值** |
| 1 | (++p)->x | 100 | 100 |
| 2 | p++,p->c | B | B |
| 3 | \*p++->t,\*p->t | x | x |
| 4 | \*(++p)->t | x | x |
| 5 | \*++p->t | V | V |
| 6 | ++\*p->t | V | V |

表7-1 表达式计算及验证

**2．源程序修改替换题**

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

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

源程序如下：

#include "stdio.h"

#include "stdlib.h"

struct s\_list{

int data; /\* 数据域 \*/

struct s\_list \*next; /\* 指针域 \*/

} ;

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

void main(void)

{

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

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

create\_list(head,s); /\* 创建新链表 \*/

p=head; /\*遍历指针p指向链头 \*/

while(p){

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

p=p->next; /\*遍历指针p指向下一结点 \*/

}

printf("\n");

}

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

{

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

if(p[0]==0) /\* 相当于\*p==0 \*/

;

else { /\* loc\_head指向动态分配的第一个结点 \*/

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

loc\_head->data=\*p++; /\* 对数据域赋值 \*/

tail=loc\_head; /\* tail指向第一个结点 \*/

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

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

tail=tail->next; /\* tail指向新创建的结点 \*/

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

}

tail->next=NULL; /\* 对指针域赋NULL值 \*/

}

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

}

解答：

1）错误见高亮处

2）修改

第一处：

struct s\_list \*head = (struct s\_list \*)malloc(sizeof(struct s\_list))

第二处：

headp->data = loc\_head->data;

headp->next = loc\_head->next;

3）运行结果截图



图7-1程序运行结果

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

解答：

1. 修改后函数如下

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

{

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

if (p[0] == 0) /\* 相当于\*p==0 \*/

;

else

{ /\* loc\_head指向动态分配的第一个结点 \*/

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

loc\_head->data = \*p++; /\* 对数据域赋值 \*/

loc\_head->next = NULL; /\* tail指向第一个结点 \*/

while (\*p)

{ /\* tail所指结点的指针域指向动态创建的结点 \*/

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

tail->data = \*p++; /\* tail指向新创建的结点 \*/

tail->next = loc\_head; /\* 向新创建的结点的数据域赋值 \*/

loc\_head = tail;

}

//tail->next = NULL; /\* 对指针域赋NULL值 \*/

}

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

headp->next = loc\_head->next;

}

1. 运行结果



图7-2 程序运行结果

**3．程序设计题**

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

void f0( int n)

{

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

}{

解答：

1. 流程图



图7-3 程序3-1流程图

2）源代码

#include<stdio.h>

struct bits

{

unsigned int bit0 : 1, bit1 : 1, bit2 : 1, bit3 : 1, bit4 : 1, bit5 : 1, bit6 : 1, bit7 : 1; /\* data \*/

};

void f0( int n)

{

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

}

int main(void)

{

struct bits bit = {0};

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

unsigned int c;

scanf("%d", &c);

for (int i = 0; i < 8;i++,c>>= 1){

if (c&1)

{

switch(i)

{

case 0:bit.bit0 = 1;p\_fun[0](0);break;

3）测试

1. 测试数据

255

1. 运行结果

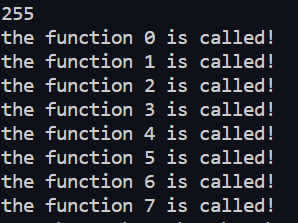


图7-4 程序运行结果

（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语言程序设计四门课程的成绩。用函数编程实现下列功能：

(1) 输入每个学生的各项信息。

(2) 输出每个学生的各项信息。

(3) 修改指定学生的指定数据项的内容。

(4) 统计每个同学的平均成绩（保留2位小数）。

(5) 输出各位同学的学号、姓名、四门课程的总成绩和平均成绩。

解答：

1. 解题思路
   * + 1. 读入数据，存入链表
       2. 遍历链表，输出数据
       3. 遍历链表，修改数据
       4. 遍历链表，统计数据
2. 源代码

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

typedef struct Score

{

char number[10];

char name[10]; /\* data \*/

int math;

int physic;

int cp;

struct Score \*next;

} score;

void input(score \*\*head)

{

int ans = 0;

score \*loc\_head = (score \*)malloc(sizeof(score)), \*tail, \*q;

printf("input 'ctrl z' to finish\n");

scanf("%s%s%d%d%d", loc\_head->number, loc\_head->name, &loc\_head->math, &loc\_head->physic, &loc\_head->cp);

q = tail = loc\_head;

while (1)

{

if (ans > 0)

{

q = q->next;

}

tail->next = (score \*)malloc(sizeof(score));

tail = tail->next;

if (scanf("%s", tail->number) == EOF)

break;

scanf("%s%d%d%d", tail->name, &tail->math, &tail->physic, &tail->cp);

ans++;

}

q->next = NULL;

\*head = loc\_head;

}

void change(score \*head)

{

char s[10],m[10],math[]={"math"},physic[]={"physic"},cp[]={"cp"};

int new,flag=0,flag1=1;

printf("input name and course and new data\n");

scanf("%s%s%d", s,m, &new);

while(head)

{

if(strcmp(s,head->name)==0)

{

if(strcmp(m,math)==0){

head->math = new;

flag1 = 0;

}

else if(strcmp(m,physic)==0){

head->physic = new;

flag1 = 0;

}

else if(strcmp(m,cp)==0){

head->cp = new;

flag1 = 0;

}

else

flag = 1;

break;

}

else

head = head->next;

}

if(flag==1||flag1==1)

printf("error\n");

}

void op(score \*head)

{

printf("number\tname\tall\taverage\n");

while(head)

{

printf("%s\t%s\t%d\t%.2f\n", head->number, head->name, head->cp + head->math + head->physic, (head->cp + head->math + head->physic) / 3.0);

head = head->next;

}

}

1. 测试

测试数据：

126879 ejc 23 45 76

276453 fwf 32 87 98

766123 uwe 23 76 32

部分运行结果：

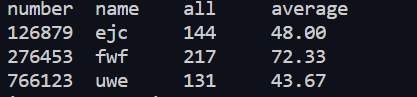


图7-5 程序运行结果

**4. 选做题**

（1）对编程设计题第（2）题的程序，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。

（2）对选做题第（1）题，进一步写出用交换结点指针域的方法升序排序的函数。

（3）采用双向链表重做编程设计题第（2）题。

解答：

1. 思路和流程图

对于交换数据域，只需遍历链表即可



图7-6 交换数据域流程图

对于交换指针域，用指针变换即可



图7-7 交换指针域流程图

1. 源代码

（1）

void sort(score \*head)

{

score \*p1=head, \*p2;

int len = 0, i, j, t;

char s[10];

while(p1){

len++;

p1 = p1->next;

}

for (i = 0,p1 = head; i < len - 1;i++,p1=p1->next)

for (j = i + 1, p2 = p1->next; j < len;j++,p2=p2->next)

if((p1->cp+p1->physic+p1->math)>(p2->math+p2->cp+p2->physic))

{

t = p1->physic;p1->physic = p2->physic;p2->physic = t;

t = p1->cp;p1->cp = p2->cp;p2->cp = t;

t = p1->math;p1->math = p2->math;p2->math = t;

strcpy(s, p1->number);strcpy(p1->number, p2->number);strcpy(p2->number, s);

strcpy(s, p1->name);strcpy(p1->name, p2->name);strcpy(p2->name, s);

}

}

（2）

void sort(score \*\*head)

{

score \*pr1, \*pr2, \*p1, \*p2, \*t;

int i = 0;

p1 = (score \*)malloc(sizeof(score));

p1->next = \*head;

(\*head) = pr1 = p1;

for (p1 = pr1->next; p1->next != NULL;pr1=p1,p1=p1->next)

for (p2 = p1->next, pr2 = p1; p2 != NULL;pr2=p2,p2=p2->next)

if((p1->cp+p1->physic+p1->math)>(p2->math+p2->cp+p2->physic)){

t = p2->next;

pr1->next = p2;

pr2->next = p1;

p2->next = p1->next;

p1->next = t;

t = p1;

p1 = p2;

p2 = t;

}

p1 = \*head;

(\*head) = (\*head)->next;

free(p1);

}

（3）

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

typedef struct Score

{

char number[10];

char name[10]; /\* data \*/

int math;

int physic;

int cp;

struct Score \*next;

struct Score \*prior;

} score;

void input(score \*\*head);

void output(score \*head);

void change(score \*head);

void average(score \*head, double a[]);

void sort(score \*\*head);

void op(score \*head);

1. 运行结果

（1）（2）

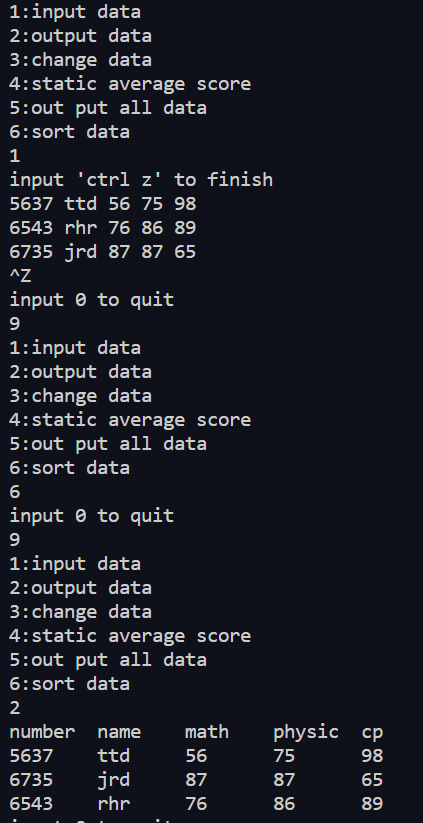


图7-8 程序运行结果

（3）

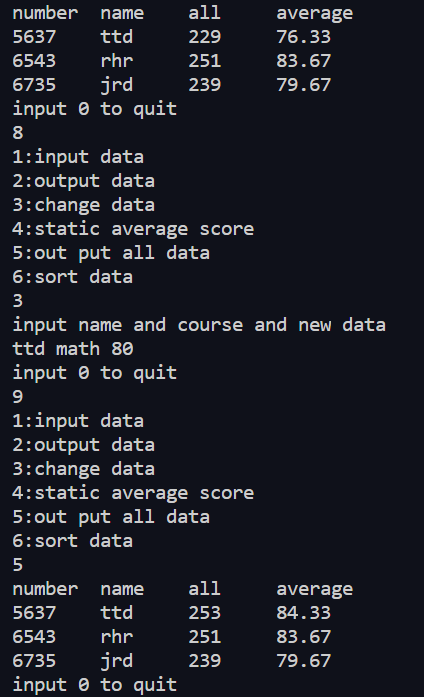


图7-9 程序运行结果

**7.3实验小结**

通过结构体实验，我掌握了结构体的说明，引用，以及和其他数据结构的嵌套使用，结构体使得存储数据更加灵活。使用结构体可以解决不同数据类型的数据存储的问题。然后，链表作为自引用结构，通过结构指针实现动态存储。相比数组在查找，替换，删除节点方面具有显著的优越性。

**实验8 文件操作实验**

**8.1实验目的**

（1）熟悉文本文件和二进制文件在磁盘中的存储方式；

（2）熟练掌握流式文件的读写方法。

**8.2实验题目及要求**

**8.2.1．文件类型的程序验证题**

设有程序：

#include <stdio.h>

int main(void)

{

short a=0x253f,b=0x7b7d;

char ch;

FILE \*fp1,\*fp2;

fp1=fopen("d:\\abc1.bin","wb+");

fp2=fopen("d:\\abc2.txt","w+");

fwrite(&a,sizeof(short),1,fp1);

fwrite(&b,sizeof(short),1,fp1);

fprintf(fp2,"%hx %hx",a,b);

rewind(fp1); rewind(fp2);

while((ch = fgetc(fp1)) != EOF)

putchar(ch);

putchar('\n');

while((ch = fgetc(fp2)) != EOF)

putchar(ch);

putchar('\n');

fclose(fp1);

fclose(fp2);

return 0;

}

1. 请思考程序的输出结果，然后通过上机运行来加以验证。
2. 将两处sizeof(short)均改为sizeof(char)结果有什么不同，为什么？

（3）将fprintf(fp2,"%hx %hx",a,b) 改为 fprintf(fp2,"%d %d",a,b)结果有什么不同。

**解答：**

1. 输出结果是：

?%}{

253f 7b7d

验证如图：



图1-1 程序运行结果

1. 结果是：

？}

因为char占一个字节，因此只写入一个字节

1. 结果是：

9535 31613

以十进制写入

**8.2.2．源程序修改替换题**

将指定的文本文件内容在屏幕上显示出来，命令行的格式为：

type filename

1. 源程序中存在什么样的逻辑错误（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。

#include<stdio.h>

#include<stdlib.h>

int main(int argc, char\* argv[])

{

char ch;

FILE \*fp;

if(argc!=2){

printf("Arguments error!\n");

exit(-1);

}

if((fp=fopen(argv[1],"r"))==NULL){ /\* fp 指向 filename \*/

printf("Can't open %s file!\n",argv[1]);

exit(-1);

}

while(ch=fgetc(fp)!=EOF) /\* 从filename中读字符 \*/

putchar(ch); /\* 向显示器中写字符 \*/

fclose(fp); /\* 关闭filename \*/

return 0;

}

解答：

错误：while(ch=fgetc(fp)!=EOF)有错

改为：while((ch=fgetc(fp))!=EOF)

运行结果：



图8-2 程序运行结果

（2）用输入输出重定向freopen改写main函数。

解答：

修改后的程序：

if((freopen(argv[1],"r",stdin))==NULL){ /\* fp 指向 filename \*/

printf("Can't open %s file!\n",argv[1]);

exit(-1);

}

while((ch=fgetc(stdin))!=EOF) /\* 从filename中读字符 \*/

putchar(ch);

fclose(stdin); /\* 向显示器中写字符 \*/ /\* 关闭filename \*/

return 0;

}

运行结果：



图8-3 程序运行结果

**3．程序设计题**

（1）编写一个程序replace，采用命令行方式，用给定的字符串替换指定文件中的目标字符串，并显示输出替换的个数。例如，命令行：

replace filename.txt you they

解答：

1. 流程图



图8-4 程序流程图

1. 源代码

#include <stdio.h>

#include <string.h>

int replace(char str[], char key[], char swap[]);

int replace(char str[], char key[], char swap[]) //定义替换函数

{

int l1, l2, l3, i, j, flag, ans = 0;

char tmp[1000]; //临时存放字符串

l1 = strlen(str);

l2 = strlen(key);

l3 = strlen(swap);

for (i = 0; i < l1 - l2; i++)

{

flag = 1;

for (j = 0; j < l2; j++) //开始查找字符串

{

if (str[i + j] != key[j])

{

flag = 0;

break;

}

}

if (flag) //找到字符串

{

strcpy(tmp, str); //将源串赋值到临时数组tmp中

strcpy(&tmp[i], swap); //用swap中的字符串替换tmp中的i之后的部分

strcpy(&tmp[i + l3], &str[i + l2]); //将str除去key后的后半部分赋值给tmp

strcpy(str, tmp); //再将tmp赋值到str中

i += l3 - 1;

l1 = strlen(str);

ans++;

}

}

return ans;

}

1. 运行结果

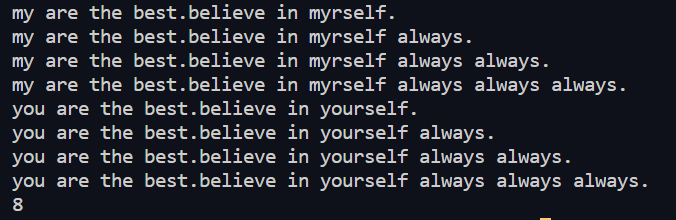


图8-5 程序运行结果

（2）从键盘输入10个单精度浮点数，以二进制形式存入文件float.dat中。再从文件中读出这10个单精度浮点数显示在屏幕上。之后要求将float.dat中的单精度浮点数按字节读出来，观察写入文件的浮点数字节数据是不是和计算机内存中表示的浮点数字节数据一致。

解答:

1. 流程图



图8-6 程序流程图

1. 源代码

#include<stdio.h>

int main()

{

FILE \*fp;

float a[10], b[10];

unsigned char c[4];

fp=fopen("d:\\float.dat","wb+");

for (int i = 0; i < 10;i++)

scanf("%f", &a[i]);

for (int i = 0; i < 10;i++)

fwrite(&a[i],sizeof(float),1,fp);

rewind(fp);

for (int i = 0; i < 10;i++)

fread(&b[i], sizeof(float), 1, fp);

for (int i = 0; i < 10;i++)

printf("%f ", b[i]);

printf("\n");

rewind(fp);

for (int i = 0; i < 4;i++)

fread(&c[i], sizeof(char), 1, fp);

3）运行结果

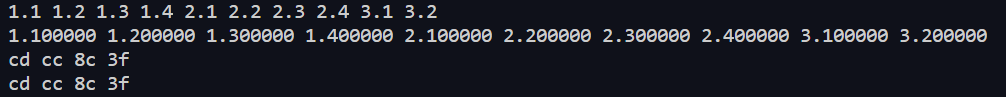


图8-7 程序运行结果

**8.3实验小结**

通过实验，我了解到了文件的相关操作，掌握了文件的读写操作。

参考文献

[1] 曹计昌,卢萍,李开. C语言程序设计,北京： 科学出版社,2013

[2] 李开,卢萍,曹计昌. C语言实验与课程设计, 北京：科学出版社,2011