

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级：计算机类（二）2007班**

**学 号：U202017180**

**姓 名：吴彬杰**

**指导教师：万琳**

**报告日期：2020年11月 4日**

**软件学院**

**目 录**

[**1 表达式和标准输入输出实验 1**](#_Toc404837920)

[1.1 实验目的 1](#_Toc404837921)

[1.2 实验内容 1](#_Toc404837922)

[1.3 实验小结 1](#_Toc404837923)

[**2 流程控制实验 2**](#_Toc404837924)

[2.1 实验目的 1](#_Toc404837921)

[2.2 实验内容 1](#_Toc404837922)

[2.3 实验小结 1](#_Toc404837923)

[**3 函数与程序结构实验 3**](#_Toc404837929)

[3.1 实验目的 1](#_Toc404837921)

[3.2 实验内容 1](#_Toc404837922)

[3.3 实验小结 1](#_Toc404837923)

[**4 编译预处理实验 4**](#_Toc404837934)

[4.1 实验目的 1](#_Toc404837921)

[4.2 实验内容 1](#_Toc404837922)

[4.3 实验小结 1](#_Toc404837923)

[**5 数组实验 5**](#_Toc404837938)

[5.1 实验目的 1](#_Toc404837921)

[5.2 实验内容 1](#_Toc404837922)

[5.3 实验小结 1](#_Toc404837923)

[**6 指针实验 6**](#_Toc404837943)

[6.1 实验目的 1](#_Toc404837921)

[6.2 实验内容 1](#_Toc404837922)

[6.3 实验小结 1](#_Toc404837923)

[**7 结构与联合实验 7**](#_Toc404837948)

[7.1 实验目的 1](#_Toc404837921)

[7.2 实验内容 1](#_Toc404837922)

[7.3 实验小结 1](#_Toc404837923)

[**8 文件实验 8**](#_Toc404837953)

[8.1 实验目的 1](#_Toc404837921)

[8.2 实验内容 1](#_Toc404837922)

[8.3 实验小结 1](#_Toc404837923)

[**参考文献 9**](#_Toc404837957)

# 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

01 /\*实验3-1改错题程序：计算s=1!+2!+3!+…+n!\*/

02 #include <stdio.h>

03 int main(void)

04 {

05     int k;

06     for (k = 1; k <= 20; k++)

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

08     return 0;

09 }

10 long sum\_fac(int n)

11 {

12     long s = 0;

13     int i, fac;

14     for (i = 1; i <= n; i++)

15         fac \*= i;

16     s += fac;

17     return s;

18 }

**解答：**

（1）错误修改：

1) 第3行前应插入sun\_fac函数的声明，正确形式为：

在第2行后插入：long long sum\_fac(int);

2) 第10行改为long long sum\_fac(int n)

3) 第12行改为long long s=0, fac = 1

4) 第16行的赋值表达式应包含在前面的for循环中。

/\*实验3-1改错题程序：计算s=1!+2!+3!+…+n!\*/

#include <stdio.h>

long long sum\_fac(int n);

int main(void)

{

    int k;

    long long sum;

    for (k = 1; k <= 20; k++)

    {

        sum = sum\_fac(k);

        printf("k=%d\tthe sum is %lld\n", k, sum);

    }

    return 0;

}

long long sum\_fac(int n)

{

    long long s = 0 , fac = 1;

    int i;

    for (i = 1; i <= n; i++)

    {

        fac \*= i;

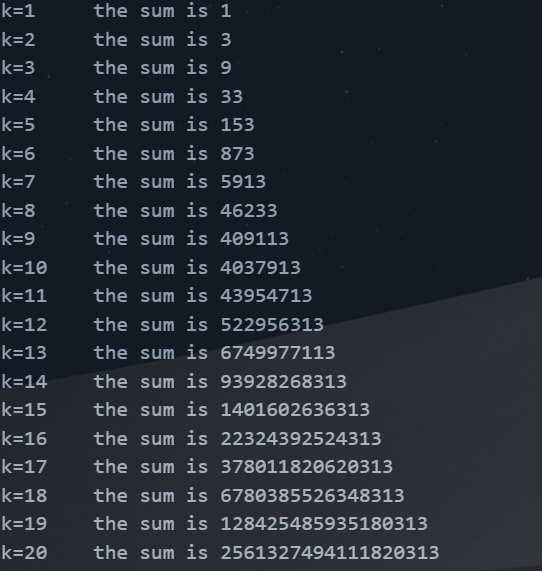
        s += fac;

    }

    return s;

}

（2）错误修改后运行结果：



**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：

/\*实验3-1改错题程序：计算s=1!+2!+3!+…+n!\*/

#include <stdio.h>

long long sum\_fac(int n);

long long fact(int m);

int main(void)

{

    int k;

    long long sum;

    for (k = 1; k <= 20; k++)

    {

        sum = sum\_fac(k);

        printf("k=%d\tthe sum is %lld\n", k, sum);

    }

    return 0;

}

long long sum\_fac(int n)

{

    long long s = 0, fac = 1;

    if (n == 1)

        return 1;

    else

        return sum\_fac(n - 1) + fact(n);

    return s;

}

long long fact(int m)

{

    long long s = 1;

    for (int i = 1; i <= m; i++)

        s \*= i;

    return s;

}

（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。

01 /\*实验3-2程序修改替换第(2)题程序：根据公式计算 s\*/

02 #include<stdio.h>

03 double mulx(double x,int n);

04 long fac(int n);

05 double sum(double x,int n)

06 {

07 int i;

08 double z=1.0;

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

10 {

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

12 }

13 return z;

14 }

15 double mulx(double x,int n)

16 {

17 int i;

18 double z=1.0;

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

20 {

21 z=z\*x;

22 }

23 return z;

24 }

25 long fac(int n)

26 {

27 int i;

28 long h=1;

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

30 {

31 h=h\*i;

32 }

33 return h;

34 }

35 int main()

36 {

37 double x;

38 int n;

39 printf("Input x and n:");

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

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

42 return 0;

43 }

解答:

//Unfinished;

#include <stdio.h>

int factx[32768];

double xlev[32768];

int fac(int n);

double sum(double x, int n);

int main()

{

    double x;

    int n;

    printf("Input x and n:");

    scanf("%lf%d", &x, &n);

    factx[1] = 1;

    xlev[1] = x;

    for (int i = 2; i <= n; i++)

        factx[i] = n \* factx[i - 1];

    for (int i = 2; i <= n; i++)

        xlev[i] = xlev[i - 1] \* x;

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

    return 0;

}

int fac(int n)

{

    return factx[n];

}

double sum(double x, int n)

{

    double sumy = 1;

    for (int i = 1; i <= n; i++)

        sumy += (xlev[i] / factx[i]);

    return sumy;

}

**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) sum、k分别为0、5

(2) i、k、sum分别为1、5、0

(3) 跳到了fibonacci函数的第一条语句

(4) 返回到了第11行

(5) k、sum是在main函数里声明的，具有块作用域。

**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：

流程图：



程序清单：

#include <stdio.h>

#include <math.h>

int n, n1, n2;

int nxBreak(int n);

int isPrime(int a);

int main(void)

{

    printf("Enter the number:\n");

    while (scanf("%d", &n) != EOF)

    {

        if (n < 4 || (n % 2 == 1))

        {

            printf("Error. Enter the number again:\n");

            continue;

        }

        else

        {

            nxBreak(n);

            printf("%d=%d+%d\n", n, n1, n2);

        }

    }

    return 0;

}

int isPrime(int a)

{

    if (a == 1)

        return 0;

    if (a == 2)

        return 114514;

    for (int i = 2, k = floor(sqrt(a)); i <= k; i++)

        if ((a % i) == 0)

            return 0;

    return 114514;

}

int nxBreak(int n)

{

    for (int i = 1; i <= n; i++)

        if (isPrime(i) && isPrime(n - i))

        {

            n1 = i;

            n2 = n - i;

        }

    return 0;

}

测试数据：

|  |  |  |
| --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 |
|
| 用例1 | 18 | 5+13 |
| 用例2 | 23 | Good bye! |
| 用例3 | 214748 | 5+214743 |

（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：

流程图：



程序清单：

#include <stdio.h>

int isPerfect(int n);

int main(void)

{

    for (int j = 1; j <= 10000; j++)

        isPerfect(j);

    return 0;

}

int isPerfect(int n)

{

    int sum = 0;

    for (int i = 1; i < n; i++)

        if (n % i == 0)

            sum += i;

    if (sum == n)

    {

        printf("%d=1",n);

        for (int i = 2; i < n; i++)

        {

            if (n % i == 0)

                printf("+%d", i);

        }

        printf("\n");

    }

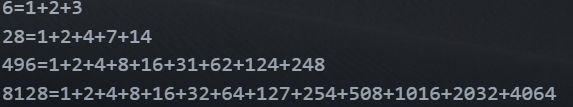
    else

        printf("%d is not a perfect number.\n",n);

    return 0;

}

运行：



（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：

流程图：



程序清单：

#include <stdio.h>

#include <math.h>

int isSelfPower(int n, int le);

int main(void)

{

    int l, count = 0;

    scanf("%d", &l);

    for (int i = pow(10, l - 1); i < pow(10, l); i++)

        if (isSelfPower(i, l))

            count++;

    printf("%d",count);

    return 0;

}

int isSelfPower(int n, int le)

{

    int sum = 0;

    int tmp = n;

    for (int i = 1; i <= le; i++)

    {

        sum += pow(tmp % 10, le);

        tmp /= 10;

    }

    if (sum == n)

        return 114514;

    return 0;

}

测试：



## 3.3 实验小结

采用函数结构，自顶向下设计原则，提内聚解耦合，提高代码可读性。