

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 网络空间安全学院2003班**

**学 号： U202012043**

**姓 名： 范启航**

**指导教师： 张云鹤**

**报告日期： 2020年10月31日**

**网络空间安全学院**

**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

**3.2.1．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!\*/

#include <stdio.h>

int main(void)

{

int k;

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

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

return 0;

}

long sum\_fac(int n)

{

long s=0;

int i,fac;

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

fac\*=i;

s+=fac;

return s;

}

**解答：**

1. 错误修改：

1）第2行未申明函数，正确形式为：

long long sum\_fac(int);

2）第6行输出格式错误，正确形式为：

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

3）函数返回类型，fac,s应用long long型整数

4）fac循环后未初始化；

5）第15行累加操作应在循环内部进行，正确形式为：

{

fac\*=i;

s+=fac;

}

1. 错误修改后运行结果：



图3-2-1运行结果示意图

**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**：

代码如下：

#include <stdio.h>

long long sum\_fac(int);

int main(void)

{

int k;

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

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

return 0;

}

long long sum\_fac(int n)

{

long long s=0;

long long fac;

int i;

fac = 1;

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

{

fac \*=i;

}

if(n == 1)

s = 1;

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

return s;

}

**运行结果图：**

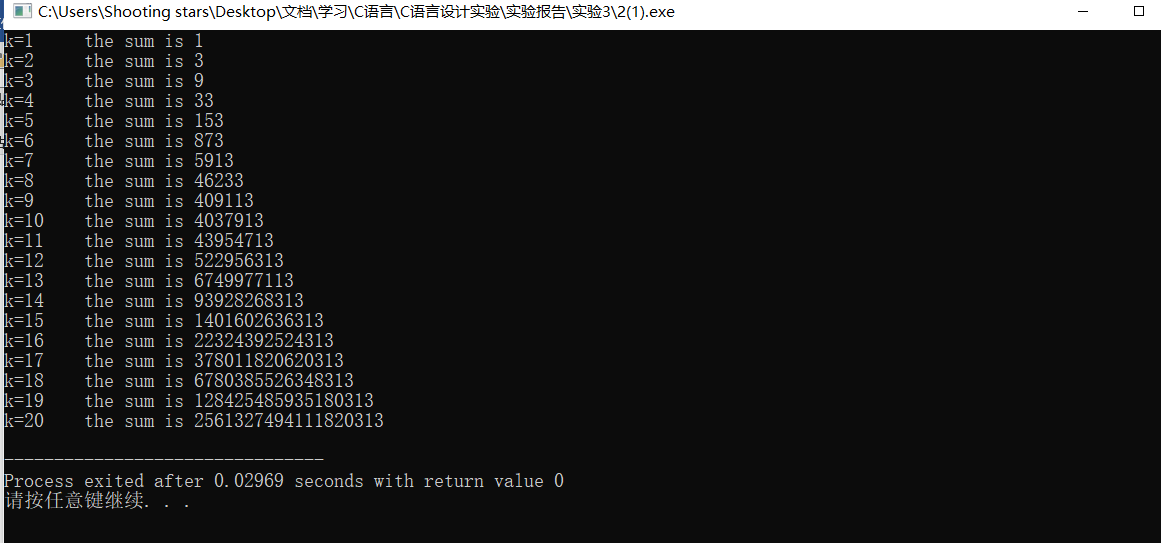


图3-2-2运行结果示意图

（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\*/

#include<stdio.h>

double mulx(double x,int n);

long fac(int n);

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：**

代码如下：

#include <stdio.h>

double sum(double , int);

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;

}

double sum(double x,int n )

{

double z = 1.0;

double last\_x = 1.0;

long long last\_i= 1;

int i;

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

{

z = z+last\_x\*x/(last\_i\*i);

last\_x \*=x;

last\_i \*=i;

}

return z;

}

**运行结果图：**

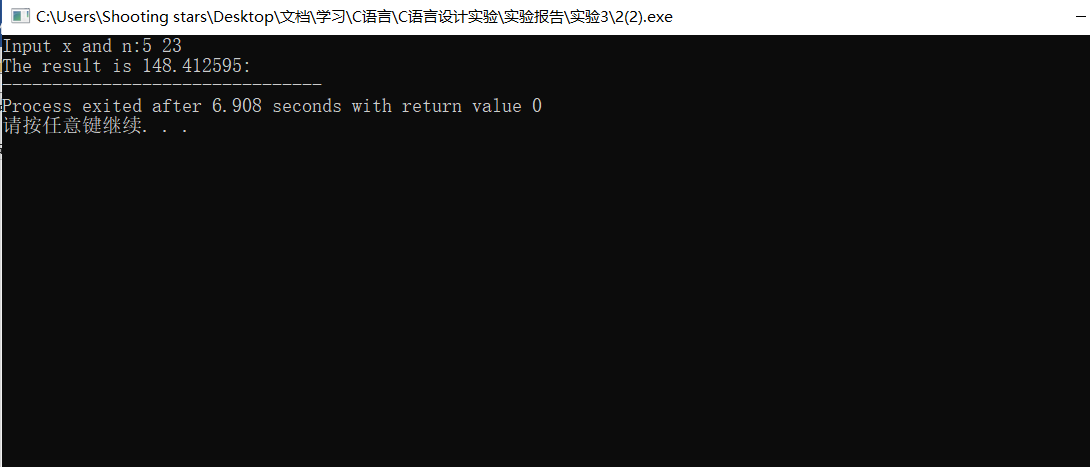


图3-2-3运行结果示意图

**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 = 0; k= 5;

(2)i=1;sum = 0; k =5

(3)调到了printf语句

(4)当n = 3时先进行if判断，后进入如fabonacci(1)返回1，在进入fabonacci(2)返回1，最后回到fabonacci(3)返回1+1=2；

(5)n先由大变小后由小变大，k,sum为main函数中的变量，在函数fabonacci中无法访问

**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。

（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。

（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。

**3.解答：**

（1）

程序流程图：



图3-4-1 程序设计流程图

代码如下：

#include <stdio.h>

int k = 0;

int gd(int);

int a[1000];

void isprime(int);

int main()

{

int n;

scanf("%d", &n);

if(n >=4 && n%2 == 0)

{

isprime(n);

gd(n);

}

return 0;

}

int gd(int n)

{

int j,i;

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

{

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

{

if(a[i]+a[j]== n)

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

}

}

}

void isprime(int n)

{

int i,j,flag;

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

{

flag = 1;

for(j = 2 ; j<=i/2;j ++)

{

if( i%j == 0)

{

flag = 0;

break;

}

}

if(flag)

{

a[k] = i;

k++;

}

}

}

**运行结果图：**

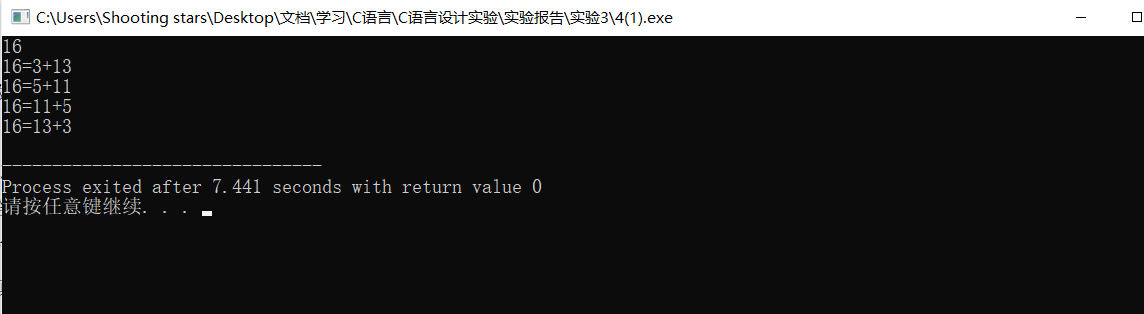


图3-4-2程序设计运行结果图1

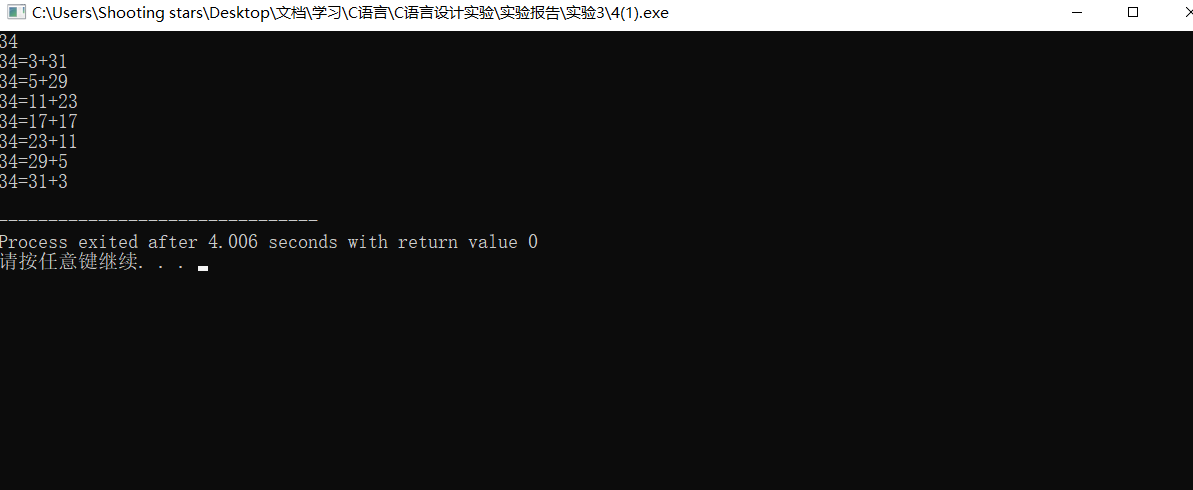


图3-4-3程序设计运行结果图2

**（2）**

**代码如下：**

#include <stdio.h>

void pfn(int );

int main()

{

int n;

printf("请输入一个数：");

scanf("%d", &n);

pfn(n);

return 0;

}

void pfn(int n)

{

int i,j,s = 0;

j = 0;

int a[100000]= {0};

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

{

if( n%i == 0)

{

a[j] = i;

j++;

}

}

j = 0;

while(a[j] != 0)

{

s += a[j];

j++;

}

j = 0;

if(s == n)

{

printf("%d=", n);

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

j++;

while(a[j] != 0)

{

printf("+%d", a[j]);

j++;

}

}

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

}

**运行结果图：**

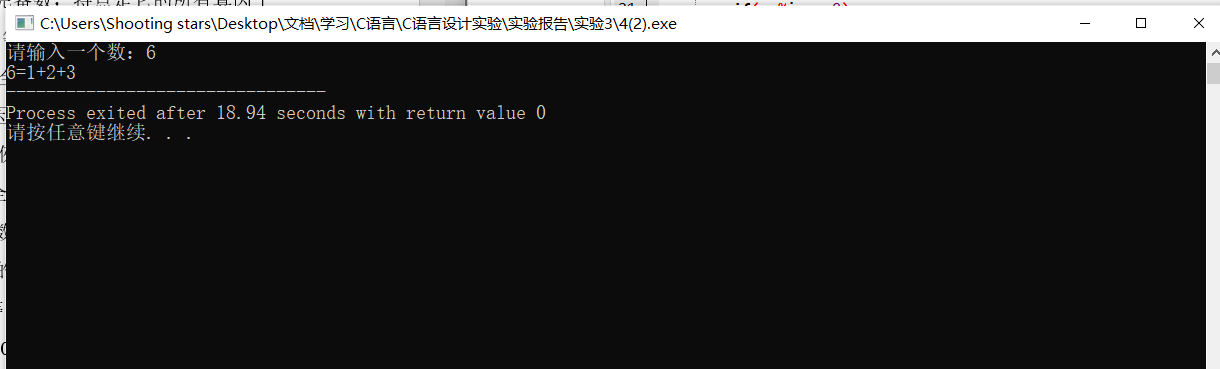


图3-4-4运行结果示意图

Plus版本：  
寻找10^8以内的完全数：

代码：

#include <stdio.h>

#include <math.h>

void pfn(long long);

int isprime(long long);

int main()

{

int p;

for(p = 2;p<15;p++)

{

if(isprime(p))

{

long long x,y;

x = pow(2,p)-1;

if(isprime(x))

{

long long z;

y = pow(2,p-1);

z = x\*y;

pfn(z);

}

}

}

return 0;

}

int isprime(long long n)

{

int i,flag;

flag = 1;

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

{

if(n%i == 0)

{

flag = 0;

break;

}

}

return flag;

}

void pfn(long long n)

{

int i,j,s;

s = 0;

j = 0;

int a[10000]= {0};

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

{

if( n%i == 0)

{

a[j] = i;

j++;

}

}

j = 0;

printf("%d=", n);

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

j++;

while(a[j] != 0)

{

printf("+%d", a[j]);

j++;

}

printf("\n");

}

**运行结果图：**



图3-4-5运行结果示意图2

（3）

代码如下：

#include <stdio.h>

#include <math.h>

void zimi(int );

int main()

{

int k;

do

{

scanf("%d", &k);

if( k== 0)

break;

else zimi(k);

}while(1);

return 0;

}

void zimi (int n)

{

int i,j,k,s,x,temp;

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

{

s = 0;

temp = i;

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

{

x = temp%10;

s += pow( x, n);

temp /=10;

}

if( s == i)

printf("%d,", s);

}

**运行结果图：**

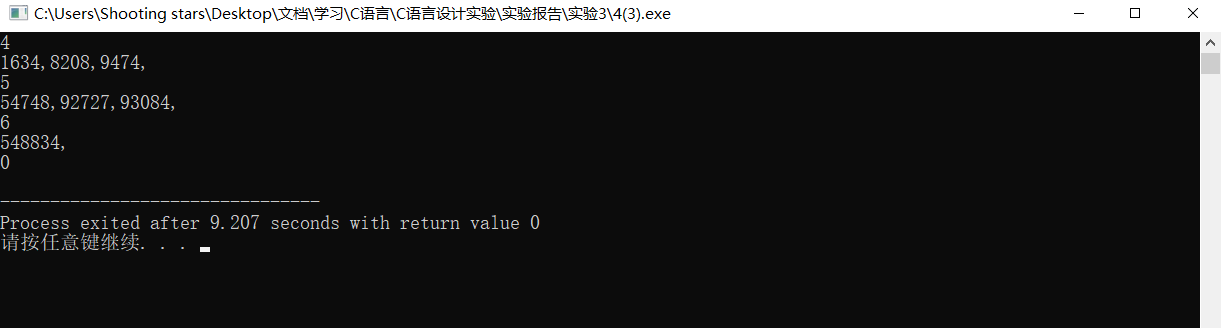


图3-4-6运行结果示意图

## 3.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

实验过程中遇到了代码量过大，导致运行效率偏低的情况，通过简化算法，使用数论等方法对代码运行速率进行了提示，同时增加了代码的可读性。

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