

**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：**

**指导教师：**

**报告日期：**

**软件学院**

**目 录**

[实验3 函数与程序结构实验 2](#_Toc181993089)

[3.1实验目的 2](#_Toc181993090)

[3.2实验内容 2](#_Toc181993091)

[3.3实验小结 16](#_Toc181993092)

# 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. 错误修改
   1. long类型大小不足以存储要求的数字，应将变量s, fac的类型改为long long，并将fac初始化为1，改为：  
      long long s = 0, fac = 1;

int i;

* 1. 同时，将第9行中sum\_fac函数的返回值类型改为long long，即：  
     long long sum\_fac(int n)
  2. 将第6行中printf的sum\_fac(k)对应的类型改为%lld，即：  
     printf("k=%d\tthe sum is %lld\n",k,sum\_fac(k));
  3. 使用sum\_fac前未对其进行声明，应在第1行后添加函数原型：  
     long long sum\_fac(int n);
  4. 计算出的s实际上仅为n!，为使求和生效，应将for循环改为：  
     for(i=1;i<=n;i++) {

fac\*=i;

s+=fac;

}

1. 修改后完整代码

#include <stdio.h>

long long sum\_fac(int n);

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, fac = 1;

int i;

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

fac\*=i;

s+=fac;

}

return s;

}

1. 文本

   描述已自动生成修改后效果

图3-1 实验3-1的运行结果图

**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：**

修改后的sum\_fac函数：

long long sum\_fac(int n)

{

if (n == 1) return 1;

long long s = sum\_fac(n-1), fac = 1;

int i;

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

fac\*=i;

s+=fac;

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。修改该程序使之计算量最小，即减少程序中乘法和加法的运算次数，不是指减少代码的行数。

/\*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：**

修改后的代码：

#include<stdio.h>

double sum(double x,int n)

{

int i;

double z=1.0;

double last=1.0;

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

{

last \*= x / i;

z+=last;

}

return z;

}

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;

}

此方法精简了计算次数，同时使可计算的x, n取值范围变得更大。

**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. sum=0, k=5, i=1; 返回后光条停留在printf("i=%d\tthe sum is %ld\n",i,sum);
3. 跳到if(n==1 || n==2)
4. 递归结束后，光条先在最后一行的}处停留多次，最后回到printf处。
5. n先从n-1减少至1，再从n-2减少到1。  
   k, sum是main函数内部的局部变量，fabonacci函数不在其作用域内。

**4．程序设计**

以下（1）至（3）题对应Educoder教学平台 “C语言实验”课程，实验3，第7关实验3-1、第8关实验3-2，以及第9关实验3-3。

（1）编程验证歌德巴赫猜想：任何一个大于等于4的偶数都是两个素数之和。要求设计一个函数，接受形参n，以“n=n1+n2”的形式输出结果，若有多种分解情况，取n1最小的一个输出。

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：**

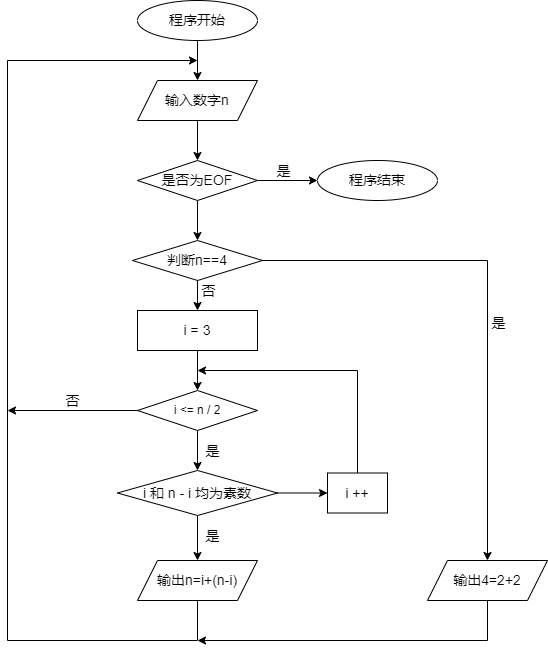
1. 算法流程

图3-2 编程题1的程序流程图

1. 代码清单

#include<stdio.h>

int prime(int n) {

for (int i = 2; i \* i <= n; i++) {

if (n % i == 0) {

return 0;

}

}

return 1;

}

int main() {

int n;

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

if (n == 4) {

printf("4=2+2\n");

} else if (n > 4 && n % 2 == 0) {

for (int i = 3; i <= n / 2; i += 2) {

if (prime(i) && prime(n - i)) {

printf("%d=%d+%d\n", n, i, n - i);

break;

}

}

}

}

}

1. 测试数据

表3-1 编程题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 48  90 | 48=5+43  90=7+83 | 48=5+43  90=7+83 |
| 用例2 | 58  244 | 58=5+53  244=3+241 | 58=5+53  244=3+241 |

1. 手机屏幕的截图

   描述已自动生成手机屏幕的截图

   描述已自动生成运行效果

图3-3 编程题1的运行结果图

（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）之和恰好等于它本身。例如6=1+2+3，28=1+2+4+7+14等。

编程寻找10000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，返回1，否则返回0。在main函数中调用该函数求10000以内的所有完全数，并以完全数的真因子之和的形式输出结果，例如“6=1+2+3”。程序输出中，每个完全数单独占一行。

**解答：**

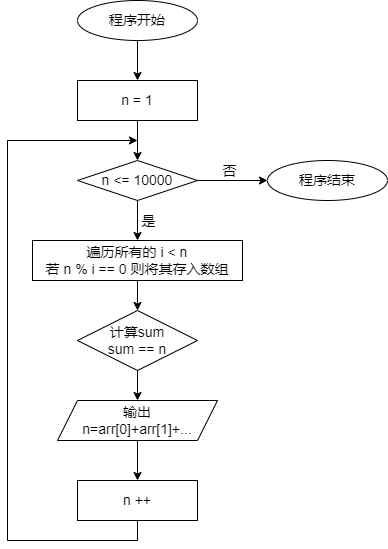
1. 算法流程

图3-4 编程题2的程序流程图

1. 代码清单

#include<stdio.h>

int main() {

for (int n = 1; n <= 10000; n++) {

int sum = 0;

int arr[100];

int index = 0;

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

if (n % i == 0) {

sum += i;

arr[index] = i;

index++;

}

}

if (sum == n) {

printf("%d=", n);

for (int i = 0; i < index; i++) {

if (i == index - 1) {

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

} else {

printf("%d+", arr[i]);

}

}

}

}

}

1. 测试数据

表3-2 编程题2的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | / | 6=1+2+3  28=1+2+4+7+14  496=1+2+4+8+16+31+62+124+248  8128=1+2+4+8+16+32+64+127+254+508+1016+2032+4064 | 6=1+2+3  28=1+2+4+7+14  496=1+2+4+8+16+31+62+124+248  8128=1+2+4+8+16+32+64+127+254+508+1016+2032+4064 |

1. 文本

   描述已自动生成运行效果

图3-5 编程题2的运行结果图

（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，返回1；否则，返回0。要求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：**

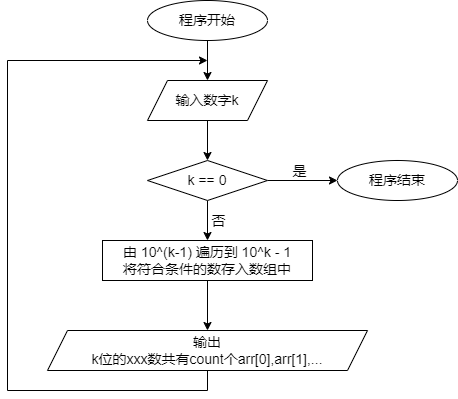
1. 算法流程

图3-6 编程题3的程序流程图

1. 代码清单

#include<stdio.h>

#include<math.h>

char prefix[9][50] = {

"", "", "", "3位的水仙花数", "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"

};

int main() {

int k;

while (scanf("%d", &k)) {

if (k == 0) {

break;

}

int arr[1000];

int index = 0;

int start = pow(10, k - 1);

int end = pow(10, k) - 1;

for (int i = start; i <= end; i++) {

int sum = 0;

int num = i;

while (num > 0) {

int digit = num % 10;

sum += pow(digit, k);

num /= 10;

}

if (sum == i) {

arr[index] = i;

index++;

}

}

printf("%s共有%d个", prefix[k], index);

for (int i = 0; i < index; i++) {

printf("%d", arr[i]);

if (i < index - 1) {

printf(",");

} else {

printf("\n");

}

}

}

}

1. 测试数据

表3-3 编程题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程序输入 | 理 论 结 果 | 运 行 结 果 |
| 用例1 | 5  6  7  0 | 5位的五角星数共有3个54748,92727,93084  6位的六合数共有1个548834  7位的北斗星数共有4个1741725,4210818,9800817,9926315 | 5位的五角星数共有3个54748,92727,93084  6位的六合数共有1个548834  7位的北斗星数共有4个1741725,4210818,9800817,9926315 |
| 用例2 | 5  3  4  0 | 5位的五角星数共有3个54748,92727,93084  3位的水仙花数共有4个153,370,371,407  4位的四叶玫瑰数共有3个1634,8208,9474 | 5位的五角星数共有3个54748,92727,93084  3位的水仙花数共有4个153,370,371,407  4位的四叶玫瑰数共有3个1634,8208,9474 |

1. 文本

   描述已自动生成文本

   描述已自动生成运行效果

图3-7 编程题3的运行结果图

3.3 实验小结

1. 在本次实验中，我学会并练习了函数的应用，学到了如何与函数进行数据传输。
2. 在本次实验中，我更加熟练的使用了debug功能对程序进行调试，使问题定位更加快速便捷。