# 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!\*/

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. }

**3.2.2 源程序修改替换**

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：**

按照要求修改后的sum\_fac()函数如下：

1. long long sum\_fac(int n)
2. {
3. if (n==1)
4. return 1 ;
5. long long fac=1, sum=0 ;
6. for (int count = 1;count<=n;count++)
7. fac \*= count ;
8. sum = fac + sum\_fac(n-1) ;
9. return sum ;
10. }
11. 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\*/

1. #include<stdio.h>
2. double mulx(double x,int n);
3. long fac(int n);
4. double sum(double x,int n)
5. {
6. int i;
7. double z=1.0;
8. for(i=1;i<=n;i++)
9. {
10. z=z+mulx(x,i)/fac(i);
11. }
12. return z;
13. }
14. double mulx(double x,int n)
15. {
16. int i;
17. double z=1.0;
18. for(i=0;i<n;i++)
19. {
20. z=z\*x;
21. }
22. return z;
23. }
24. long fac(int n)
25. {
26. int i;
27. long h=1;
28. for(i=2;i<=n;i++)
29. {
30. h=h\*i;
31. }
32. return h;
33. }
34. int main()
35. {
36. double x;
37. int n;
38. printf("Input x and n:");
39. scanf("%lf%d",&x,&n);
40. printf("The result is %lf:",sum(x,n));
41. return 0;
42. }

**3.2.3 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);

}

**3.2.4 程序设计题**

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104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，则返回1，否则返回0。在main函数中调用该函数求104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”；。
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。