# 实验3 函数和程序结构实验

## 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方法；以及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）熟悉多文件编译技术。

## 3.2 实验内容

**3.2.1 源程序改错**

下面是计算s=1!+2!+3!+…+n!的源程序，在这个源程序中存在若干语法和逻辑错误。要求在计算机上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。

1. #include "stdio.h"
2. void main(void)
3. {
4. int k;
5. for(k=1;k<6;k++)
6. printf("k=%d\tthe sum is %ld\n",k,sum\_fac(k));
7. }
8. long sum\_fac(int n)
9. {
10. long s=0;
11. int i;
12. long 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) 第1行的“stdio.h”，正确形式为：

< stdio.h >

2) 在1和2之间加入long sum\_fac(int n)的函数声明：

long sum\_fac(int n);

3) 第12行的fac初始化，正确形式为：

Long fac=1;

4) 第15行的s+=fac;加入for循环体下面，正确形式为：

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

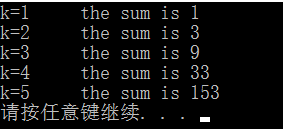
{

fac \*= i;

s += fac;

}

（2）错误修改后运行结果：



**3.2.2 源程序修改替换**

（1）修改第1题中sum\_fac函数，使其计算量最小。

（2）修改第1题中sum\_fac函数，计算C:\Users\10074\AppData\Local\Temp\ksohtml\wpsCAC8.tmp.png。

**解答：**

（1）

#include <stdio.h>

long sum\_fac(int n);

int k;

void main(void)

{

for (k = 1; k<6; k++)

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

}

long s = 0;

long fac = 1;

long sum\_fac(int n)

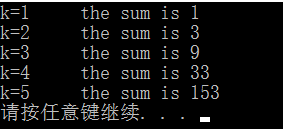
{

fac \*= k;

s += fac;

return s;

}



（2）修改第1题中sum\_fac函数，计算C:\Users\10074\AppData\Local\Temp\ksohtml\wpsCAC8.tmp.png。

#include <stdio.h>

float sum\_fac(int n);

void main(void)

{

int k;

for (k = 1; k<6; k++)

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

}

float sum\_fac(int n)

{

float s = 0;

int i;

float fac = 1;

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

{

fac \*= (float)1 / i;

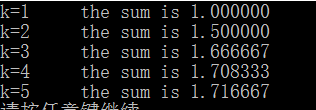
//printf("%lf\n", fac);

s += fac;

}

return s;

}



**3.2.3 跟踪调试题**

计算fabonacci数列前n项和的程序如下：

其中，long sum=0,\*p=∑声明p为长整型指针并用&sum取出sum的地址对p初始化。\*p表示引用p所指的变量（\*p即sum）。

void main(void)

{

int i,k;

long sum=0,\*p=∑

scanf("%d",&k);

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

sum+=fabonacci(i);

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

}

}

long fabonacci(int n)

{

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

return 1;

else

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

}

单步执行程序，观察p,i,sum,n值。

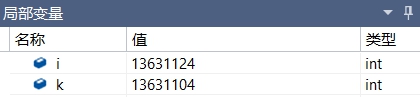
（1）刚执行完scanf("%d",&k);语句，p,i值是多少？

（2）从fabonacci函数返回后光条停留在哪个语句上？

（3）进入fabonacci函数，watch窗口显示的是什么？

（4）当i=3，从调用fabonacci函数到返回，n值如何变化？

（1）



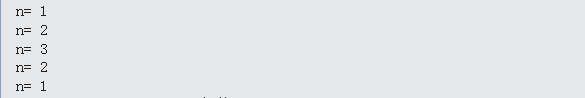
（2）



（3）

选择监视的变量值

（4）



**3.2.4 程序设计**

（1）编程让用户输入两个整数，计算两个数的最大公约数并且输出之（要求用递归函数实现求最大公约数）。同时以单步方式执行该程序，观察递归过程。

**解答：**

1） 算法流程如图3.1所示。



图3-1 编程题1的程序流程图

2）源程序清单

#include<stdio.h>

int MaxComDivisor(int a, int b);

int main()

{

int a;

int b;

scanf("%d", &a);

while (a != 0)

{

scanf("%d", &b);

printf("%d\n", MaxComDivisor(a, b));

scanf("%d", &a);

}

return 0;

}

int MaxComDivisor(int a, int b)

{

if (b == 0)

{

return a;

}

else

{

return MaxComDivisor(b, a%b);

}

}

3）测试

（a） 测试数据：

1 6

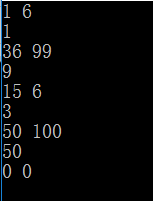
36 99

15 6

50 100

0 0

（b） 对应测试数据的运行结果截图



（2）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。

编写一个程序证明对于在符号常量BEGIN和END之间的偶数这一猜测成立。例如，如果BEGIN为10，END为20，程序的输出应为：

GOLDBACH'S CONJECTURE:

Every even number n>=4 is the sum of two primes.

10=3+7

12=5+7

……

20=3+17

**解答：**

1) 解题思路



图3-2 编程题2的程序流程图

2）程序清单

#include<stdio.h>

#include<math.h>

#include<stdbool.h>

void Goldbach(int k);

bool IsPrime(int k);

bool IsEven(int i);

int main()

{

int begin, end;

scanf("%d", &begin);

while (begin!=0)

{

scanf("%d", &end);

for (int i = begin; i<= end; i++)

{

if (IsEven(i))

{

Goldbach(i);

}

}

printf("\n");

scanf("%d", &begin);

}

return 0;

}

void Goldbach(int k)

{

for (int i = 2; i <= k / 2; i++)

{

if (IsPrime(i) && IsPrime(k - i))

{

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

break;

}

}

}

bool IsPrime(int k)

{

int m = (int)sqrt(k);

for (int j = 2; j <= m; j++)

{

if (k%j == 0)

{

return false;

}

}

return true;

}

bool IsEven(int i)

{

if (i % 2 == 0)

{

return true;

}

else

{

return false;

}

}

3）测试

（a） 测试数据：

6 10

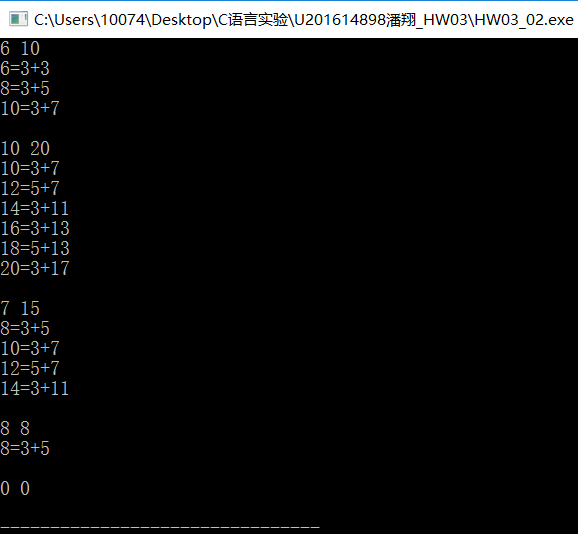
10 20

7 15

8 8

0 0

（b） 对应测试测试用例的运行结果如图所示。



**3.2.5 选做s题**

1、设file1.c如下：

#include <stdio.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 \*/

void main(void)

{

x=10;

y=20;

ch=getchar();

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

func1();

}

file2.c如下：

extern int x,y; /\* 外部变量的引用性说明 \*/

extern char ch; /\* 外部变量的引用性说明 \*/

void func1(void)

{

x++;

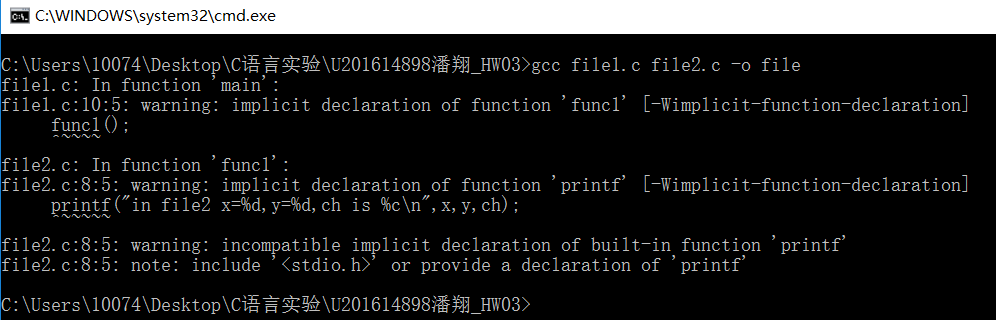
y++;

ch++;

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

}

试用TCC进行多文件编译和链接。然后在DOS环境下运行生成的可执行文件。



## 3.3 自设题

**（1）** 自设实验题目：

描述

在一个给定形状的棋盘（形状可能是不规则的）上面摆放棋子，棋子没有区别。要求摆放时任意的两个棋子不能放在棋盘中的同一行或者同一列，请编程求解对于给定形状和大小的棋盘，摆放k个棋子的所有可行的摆放方案C。

输入

输入含有多组测试数据。  
每组数据的第一行是两个正整数，n k，用一个空格隔开，表示了将在一个n\*n的矩阵内描述棋盘，以及摆放棋子的数目。 n <= 8 , k <= n  
当为-1 -1时表示输入结束。  
随后的n行描述了棋盘的形状：每行有n个字符，其中 # 表示棋盘区域， . 表示空白区域（数据保证不出现多余的空白行或者空白列）。

输出

对于每一组数据，给出一行输出，输出摆放的方案数目C （数据保证C<2^31）。

样例输入

2 1

#.

.#

4 4

...#

..#.

.#..

#...

-1 -1

样例输出

2

1

**（2）** 实验目的：通过设计实验程序，熟练函数和程序结构设计

**（3）**实验程序：

#include<stdio.h>

#include<string.h>

int n,k,ans;

char map[12][12];//棋盘

int vis[12];

int DFS(int i,int cur)

{

if(cur>=k) //

{

ans++; //方案数

return 0;

}

int x,y;

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

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

if(!vis[y] && map[x][y]=='#')

{

vis[y]=1;//标记

DFS(x+1,cur+1);//递归

vis[y]=0;

}

return 0;

}

int main()

{

while(scanf("%d%d",&n,&k)&&n!=-1)

{

ans=0;

memset(map,0,sizeof(map));

memset(vis,0,sizeof(vis));

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

scanf("%s",map[i]);

DFS(0,0);

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

}

return 0;

}

**（4）**实验用例：

2 1

#.

.#

4 4

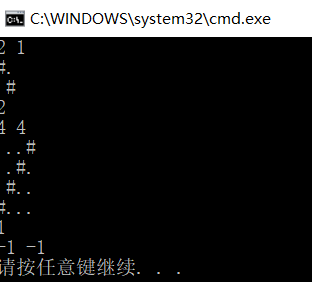
...#

..#.

.#..

#...

-1 -1



**（5）**实验结论：

可以使用函数块设计，使程序清晰

## 1.4 实验小结

**（1）**对于养成结构化设计习惯，自顶向下设计，使结构清晰。

**（2）**对于单步调试DEBUG的使用。

**（3）**学习使用测试用例和有效的测试方法。

**（4）**注重细节，培养习惯才能在工程代码中少出错。