

学号	2021144301	姓名	王大殿	实验日期	
实验名称	函数递归				
实验目的：					
1. 掌握主调函数和被调函数之间的参数传递方式；					
2. 理解函数递归使用，使用递归解决实际问题。					
实验任务、步骤与结果：					
任务 1：求圆的周长与面积					
题目描述					
任务编制一个计算圆的面积与周长的程序，要求从键盘输入圆的半径，打印输出圆的周长与面积。					
输入					
1 行：					
圆的半径					
输出					
2 行：					
第 1 行，圆的周长（精确到小数点后 2 位）					
第 2 行，圆的面积（精确到小数点后 2 位）					
代码：					
#include<stdio.h>					
#define PI 3.1415926					
float circumference(float);					
float area(float);					
int main(){					
float radius;					
printf("Input the radius:\n");					
scanf("%f",&radius);					
printf("Circumference is %.2f\n",circumference(radius));					
printf("Area is %.2f\n",area(radius));					
return 0;					
}					
float circumference(float radius){					
return 2*PI*radius;					
}					
float area(float radius){					
return PI*radius*radius;					
}					
运行结果					

```
Input the radius:
2
Circumference is 12.57
Area is 12.57
```

任务 2：求 $1\sim n$ 的阶乘和

题目描述

使用递归方法求 n 阶乘， $1\leq n\leq 100$ ， $0! = 1$ ， $1! = 1$ 。

计算 $S=1\sim n$ 的阶乘和： $1! + 2! + 3! + \dots + n!$

输入

1 行：

n , $1\leq n\leq 100$

输出

1 行：

S 的值

代码：

```
#include<stdio.h>
int factorial(int);
int main(){
    int n,sum=0;
    printf("Input a number:\n");
    scanf("%d",&n);
    for (int i = 1; i <= n; i++)
    {
        sum+=factorial(i);
    }
    printf("%d\n",sum);
    return 0;
}
int factorial(int n){
    if(n==0 || n==1){
        return 1;
    }else{
        return factorial(n-1)*n;
    }
}
```

运行结果：

```
(base) daniel@DESKTOP-PG8AU51:/mnt/c/Users/Dadian/Documents/WPS Cloud Files/248970588/
Input a number:
1
1
(base) daniel@DESKTOP-PG8AU51:/mnt/c/Users/Dadian/Documents/WPS Cloud Files/248970588/
Input a number:
2
3
(base) daniel@DESKTOP-PG8AU51:/mnt/c/Users/Dadian/Documents/WPS Cloud Files/248970588/
Input a number:
5
153
```

任务 3：汉诺塔 1

题目描述

汉诺塔（又称河内塔）问题是印度的一个古老的传说。开天辟地的神勃拉玛在一个庙里留下了三根金刚石的棒，第一根上面套着 64 个圆的金片，最大的一个在底下，其余一个比一个小，依次叠上去，庙里的众僧不倦地把它们一个个地从这根棒搬到另一根棒上，规定可利用中间的一根棒作为帮助，但每次只能搬一个，而且大的不能放在小的上面。面对庞大的数字(移动圆片的次数)18446744073709551615，看来，众僧们耗尽毕生精力也不可能完成金片的移动。

后来，这个传说就演变为汉诺塔游戏：

- 1.有三根杆子 A,B,C。A 杆上有若干碟子
- 2.每次移动一块碟子,小的只能叠在大的上面
- 3.把所有碟子从 A 杆全部移到 C 杆上

经过研究发现，汉诺塔的破解很简单，就是按照移动规则向一个方向移动金片：

如 3 阶汉诺塔的移动：A→C,A→B,C→B,A→C,B→A,B→C,A→C

此外，汉诺塔问题也是程序设计中的经典递归问题。

算法思路：

- 1.如果只有一个金片，则把该金片从源移动到目标棒，结束。
- 2.如果有 n 个金片，则把前 n-1 个金片移动到辅助的棒，然后把自己移动到目标棒，最后再把前 n-1 个移动到目标棒。

输入

1 行：

一个整数 N，表示 A 柱上有 N 个碟子。

输出

若干行：

移动的最少步骤

```
#include<stdio.h>
```

```
int Hanoi(int,char,char,char);
```

```
int main(){
```

```

    int n;
    char origin='A',middle='B',destination='C';
    printf("Input a number:\n");
    scanf("%d",&n);
    Hanoi(n,origin,middle,destination);
    return 0;
}
int Hanoi(int n, char origin, char middle,char destination){
    if(n==1){
        printf("%c To %c\n",origin,destination);
    }else{
        Hanoi(n-1,origin,destination,middle);
        printf("%c To %c\n",origin,destination);
        Hanoi(n-1,middle,origin,destination);
    }
    return 0;
}

```

运行结果:

```

Input a number:
3
A To C
A To B
C To B
A To C
B To A
B To C
A To C

```

任务 4: 汉诺塔 2

题目描述

模拟汉诺塔的移动方案，每次输出第 n 个盘子的移动方法，如 “ $a-2-c$ ” 表示从柱子 “ a ” 将编号为 “2” 的盘子移动到柱子 “ c ”。

输入

1 行:

一个整数 N ，表示 A 柱上有 N 个碟子。

输出

若干行:

每次的移动方案

最后一行，移动的总次数

```

#include<stdio.h>
int NumOfMoves=0;
int Hanoi(int,char,char,char);
int main(){
    int n;
    char origin='A',middle='B',destination='C';
    printf("Input a number:\n");

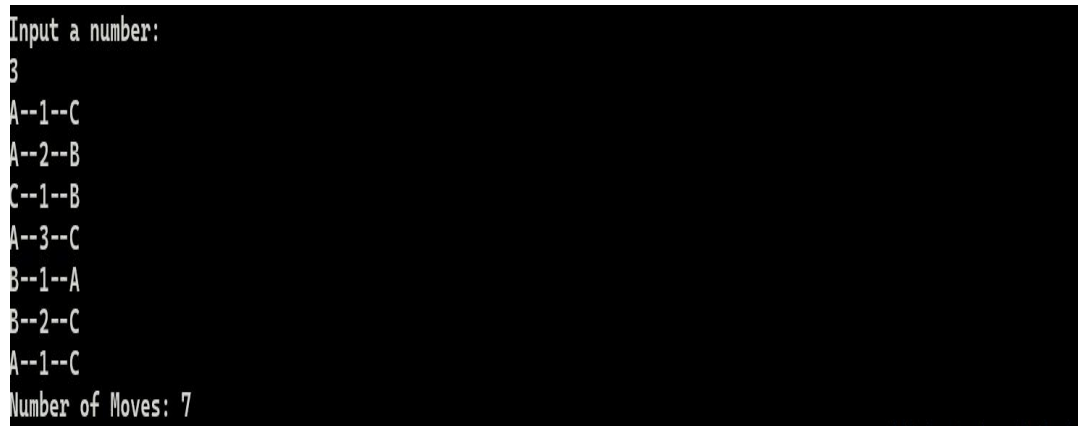
```

```

scanf("%d",&n);
Hanoi(n,origin,middle,destination);
printf("Number of Moves: %d\n",NumOfMoves);
return 0;
}
int Hanoi(int n, char origin, char middle,char destination){
    if(n==1){
        NumOfMoves++;
        printf("%c--%d--%c\n",origin,n,destination);
    }else{
        Hanoi(n-1,origin,destination,middle);
        NumOfMoves++;
        printf("%c--%d--%c\n",origin,n,destination);
        Hanoi(n-1,middle,origin,destination);
    }
    return 0;
}

```

运行结果:



```

Input a number:
3
A--1--C
A--2--B
C--1--B
A--3--C
B--1--A
B--2--C
A--1--C
Number of Moves: 7

```

任务 5：找出超过平均身高的人

题目描述

小明班上有 n 个同学，编程找出谁的身高超过全班的平均身高(整数)。

编写函数 `float inputdata(int a[],int n)`，输入 n 个的身高，数值存放在主函数的数组 a 中 (a 的元素个数最多为 100)，并返回平均身高；

编写函数 `int findhigh(int a[],int n,float ave)`，依次打印出每个超出平均身高 ave 的序号和身高，最后输出超出平均身高的人数。

输入

2 行:

第一行有一个整数 n ($1 < n < 50$)。第二行是 n 个整数，用空格隔开。

输出

3 行:

第一行为全家的平均身高（保留一位小数）；

第二行有若干个数，为超过平均身高的人的序号和身高厘米数(每项之前都有一个空格。)；

第三行为超过平均身高的同学人数。

```
#include<stdio.h>
float inputdata(int a[],int n);
int findhigh(int a[], int n, float ave);
int main(){
    int n, count;
    int a[50]={0};
    float average=0;
    printf("Input the number of classmates:\n");
    scanf("%d",&n);
    printf("Enter the height of classmates one by one:\n");
    average=inputdata(a,n);
    printf("AV=%.1f\n",average);
    count=findhigh(a,n,average);
    printf("\nTotal:%d\n",count);
    return 0;
}
float inputdata(int a[],int n){
    float sum=0;
    for (int i = 0; i < n; i++){
        scanf("%d",&a[i]);
        sum+=a[i];
    }
    return sum/n;
}
int findhigh(int a[], int n, float ave){
    int count=0;
    for(int i=0; i<n; i++){
        if(a[i]>ave){
            count++;
            printf("%d:%d\t",i+1,a[i]);
        }
    }
    return count;
}
```

运行结果:

```
Input the number of classmates:
5
Enter the height of classmates one by one:
177 188 193 175 190
AV=184.6
2:188 3:193 5:190
Total:3
```

任务 6: 分数化简

题目描述

编写函数 `reduction(int m,int n)`，调用最公约数函数 `gcd(int a,int b)`，实现分数化简，如果是真分数，写成 a/b 形式，如果是假分数，写成 $n+a/b$ 形式。

输入一个分数，输出该分数的最简分数。

输入

1 行:

两个整数，中间用 “/” 隔开。

输出

1 行:

约分后的最简分数，整数与分数部分使用 “+” 连接，分数中间 “/” 隔开。

```
#include<stdio.h>
int reduction(int,int);
int gcd(int,int);
int main(){
    int m,n;
    printf("Enter a fractional number:\n");
    scanf("%d/%d",&m,&n);
    reduction(m,n);
    return 0;
}
int reduction(int m, int n){
    if(m==n){
        printf("1\n");
    }else if (m>n)
    {
        int num;
        num=m/n;
        m=m-n;
        printf("%d+",num);
        return reduction(m,n);
    }else{
        int divisor=gcd(m,n);
        m=m/divisor;
        n=n/divisor;
```

```

        printf("%d/%d\n",m,n);
    }

}

int gcd(int m, int n){
    int temp;
    if(n>m){
        temp=m;
        m=n;
        n=temp;
    }
    if(m%n==0){
        return n;
    }else{
        return gcd(n,m%n);
    }
}

```

运行结果:

```

(base) daniel@DESKTOP-PG8AU51:/mnt/c/Users/Dadian/Documents/
Enter a fractional number:
20/16
1+1/4
(base) daniel@DESKTOP-PG8AU51:/mnt/c/Users/Dadian/Documents/
Enter a fractional number:
16/12
1+1/3

```

发现问题与分析：在分数化简函数中，这里原来写成了 m 和 n 的值直接调用了 gcd() 来算，没有加一个中间变量。m=m/gcd(m,n);n=n/gcd(m,n); 导致出错。因为第二次调用 gcd() 时，m 值已经变了。后来重新定义了一个 divisor 变量，用于固定值。

备注：

说明：

- 1、各项记录应完备详尽。
- 2、各栏目如果长度不够可以自行调整大小，但应注意排版的美观。
- 3、可将实现过程的代码、运行效果等内容以贴图方式放在相应栏目。