

段江涛

# 计算机导论与程序设计 [CS006001038,X07]

机试练习参考程序代码

2021 年 10 月 24 日



# 目录

<b>1</b>	<b>第 1 次机试练习: 熟悉 DEV-C++ 开发平台, 基本输入输出语句练习</b>	<b>5</b>
1.1	计算球体重量	5
1.2	温度转化	6
1.3	整数简单运算	6
1.4	A+B+C	7
1.5	字符输入输出	8
1.6	数字字符	8
1.7	实数运算	9
<b>2</b>	<b>第 2 次机试练习: 选择与循环语句练习</b>	<b>11</b>
2.1	四则运算	12
2.2	数位输出	13
2.3	阶梯电价计费	17
2.4	计算某月天数	19
2.5	计算整数各位数字之和	20
2.6	完数	22
<b>3</b>	<b>第 3 次机试练习: 继续分支与循环练习</b>	<b>27</b>
3.1	最大公约数	27
3.2	角谷定理	32
3.3	整数分析	33
3.4	冰箱温度预测	34
3.5	除法计算器	35
3.6	自然数分解	36
3.7	选号程序	37



## Chapter 1

### 第 1 次机试练习: 熟悉 DEV-C++ 开发平台, 基本输入输出语句练习

#### 1.1 计算球体重量

已知铁的比重是 7.86(克/立方厘米), 金的比重是 19.3(克/立方厘米)。写一个程序, 分别计算出给定直径的铁球与金球的质量, 假定  $PI=3.1415926$

输入说明:

输入两个整数, 分别表示铁球与金球的直径 (单位为毫米)

输出说明:

输出两个浮点数, 分别表示铁球与金球的质量 (单位为克), 小数点后保留 3 位小数, 两个浮点数之间用空格分隔

输入样例:

100 100

输出样例:

4115.486 10105.456

提示:

用scanf输入, 用printf输出, 保留 3 位小数的格式控制字符为%.3f

```
#include<stdio.h>
#include<math.h>    // 数学库函数
#define PI 3.1415926
int main()
{
    int a,b;
    scanf("%d%d",&a,&b);
    float v1= 4.0/3.0*pow(a/2.0/10,3)*PI;
    float v2= 4.0/3.0*pow(b/2.0/10,3)*PI;
    printf("%.3f□%.3f\n",7.86*v1,19.3*v2);
    return 0;
}
```

Note 1.1 (要点).

1. 整数除以整数, 结果为整数。

4.0/3.0 结果是浮点数, 4/3 结果是整数

2. 化简公式会引起精度问题, 不要随意化简公式。

3. pow 函数原型: `double pow(double x, double y)`

当形参数是整数时, 由于精度问题, 不要使用此函数计算  $x^y$ . 推荐使用循环语句, 易计算  $x^y$ 。如果必要, 可自定义函数: `int mypow(int x, int y)`。见课件。

## 1.2 温度转化

已知华氏温度到摄氏温度的转换公式为: 摄氏温度 = (华氏温度 - 32) × 5/9, 写程序将给定的华氏温度转换为摄氏温度输出。

输入说明:

只有一个整数, 表示输入的华氏温度

输出说明:

输出一个表示摄氏温度的实数, 小数点后保留 2 位有效数字, 多余部分四舍五入

输入样例:

50

输出样例:

10.00

提示:

用 scanf 输入, 用 printf 输出, 保留 2 位小数的格式控制字符为

```
#include <stdio.h>

int main()
{
    int f;
    float c;
    scanf("%d",&f);
    c = (f-32)*5.0/9;    // (1)
    //c = (f-32)*5/9;    // (2)
    printf("%.2f\n",c);
    return 0;
}
```

*Note 1.2 (思考).* 为何语句 (1),(2) 计算结果不一致, 哪一条语句正确?

## 1.3 整数简单运算

编写程序, 计算用户输入的两个整数的和、差、乘积 (\*) 和商 (/)。

输入格式：输入两个整数，整数之间用空格分隔。

输出格式：输出四个整数结果，分别表示和、差、积和商，每输出一个结果换行。

输入样例：

3 4

输出样例：

7

-1

12

0

```
#include<stdio.h>
int main()
{
    int a,b;
    scanf("%d%d",&a,&b);
    printf("%d\n%d\n%d\n%d\n",a+b,a-b,a*b,a/b);
    return 0;
}
```

*Note 1.3 (思考).* b=0 时如何处理?

## 1.4 A+B+C

通过键盘输入三个整数 a, b, c, 求 3 个整数之和。

输入说明：

三整形数据通过键盘输入，输入的数据介于-100000 和 100000 之间，整数之间以空格、跳格或换行分隔。

输出说明：

输出 3 个数的和。

输入样例：

-6 0 39

输出样例：

33

```
#include<stdio.h>
int main()
{
    int a,b,c;
    scanf("%d%d%d",&a,&b,&c);
    printf("%d\n",a+b+c);
    return 0;
}
```

## 1.5 字符输入输出

通过键盘输入 5 个大写字母, 输出其对应的小写字母, 并在末尾加上 “!”。

输入说明:

5 个大写字母通过键盘输入, 字母之间以竖线 “|” 分隔。

输出说明:

输出 5 个大写字母对应的小写字母, 之间无分隔, 并在末尾加上 “!”。

输入样例:

H|E|L|L|O

输出样例:

hello!

```
#include <stdio.h>
int main()
{
    char c1, c2, c3, c4, c5;
    scanf("%c| %c| %c| %c| %c", &c1, &c2, &c3, &c4, &c5);
    c1 += 32; c2 += 32; c3 += 32; c4 += 32; c5 += 32;
    printf("%c%c%c%c%c!", c1, c2, c3, c4, c5);
    return 0;
}
```

Note 1.4 (要点). scanf(“原样输入”,...);

Note 1.5. (大小写字符转化关系) 小写字符 ASCII 码 = 大写字符 ASCII 码 + 32

## 1.6 数字字符

通过键盘输入 1 个整数  $a(0 \leq a \leq 4)$ , 1 个数字字符  $b('0' \leq b \leq '5')$  求  $a+b$ 。

输入说明:

整形数据、数字字符通过键盘输入, 输入的整形数据介于 0 和 4 之间, 输入的数字字符介于 ‘0’ 和 ‘5’ 之间, 二个输入数之间用 “,” 分隔。

输出说明:

分别以整数形式及字符形式输出  $a+b$ , 输出的二个数之间用 “,” 分隔。

输入样例:

3,5

输出样例:

56,8



```
#include<stdio.h>
int main()
{
    int a;
    char b;
    scanf("%d,%c",&a,&b);
    printf("%d,%c",a+b,a+b);
    return 0;
}
```

*Note 1.6.* (scanf 函数) scanf("原样输入",...);

*Note 1.7.* (整型数值与字符混合运算) 字符对应的 ASCII 编码参与整数运算, 其结果也是整数。注意 '0' 与 0 不同, 本例中输入 0,0, 则 a=0, b='0', 变量 a 的值是整数 0, 变量 b 的值是字符 '0' 对应的 ASCII 编码, 即整数 48。

如果本题改为计算整数 a+(字符 b 对应的数字), 则, printf("%d",a+b-'0');

## 1.7 实数运算

通过键盘输入长方体的长、宽、高, 求长方体的体积 V(单精度)。

输入说明:

十进制形式输入长、宽、高, 输入数据间用空格分隔。

输出说明:

单精度形式输出长方体体积 V, 保留小数点后 3 位, 左对齐。

输入样例:

15 8.12 6.66

输出样例:

811.188

```
#include<stdio.h>
int main()
{
    float a,b,c;
    scanf("%f%f%f",&a,&b,&c);
    printf("%.3f",a*b*c);
    return 0;
}
```

*Note 1.8.* (精度问题) 32 位编译器: a\*b\*c 与 a\*c\*b 结果一致。但是在 64 位编译器中, 二者不一致。

因此, 浮点数运算会存在精度问题, 不要随意改变运算顺序。



## Chapter 2

### 第 2 次机试练习：选择与循环语句练习

#### — 特别提示 —

*Note 2.1* (不该再次发生的常见错误, 输入输出格式转换符不对应, 导致的严重错误).

```
int a; float b; double c; char d;
scanf("%d",a); // 遗忘变量前的取地址符&
scanf("%d\n",&a); // 多余'\n', 导致不能正常输入
scanf("%d%f%lf%c",&a,&b,&c,%d); // 正确对应关系
scanf("%d%c%f",&a,&d,&b); // 正确对应关系
printf("%d,%f,%lf,%c",a,b,c,d); // 正确对应关系
```

*Note 2.2* (不该再次发生的常见错误, 由 ‘;’ 引发的悲剧).

```
if ();
{
    ...
}

while ();
{
    ...
}

for (;;);
{
    ...
}
```

Note 2.3 (用 C 语言关系表达式准确表达数学含义).

```
int a;
if(110<=a<=210) // 错误
{ }
if(110<=a && a<=210) // 正确
{ }
if(a>=110 && a<=210) // 正确
{ }
```

Note 2.4 (学习体会编程技巧).

- 使用 printf() 语句, 追踪程序执行细节, 查找出错原因。
- 对于条件结构, 循环结构, 首先书写整体结构, 再添加细节, 避免低级错误。
- 提倡一题多解, 举一反三, 体会编程技巧。

## 2.1 四则运算

输入两个整数和一个四则运算符, 根据运算符计算并输出其运算结果 (和、差、积、商、余之一)。注意做整除及求余运算时, 除数不能为零。

输入说明:

使用 scanf() 函数输入两个整数和一个运算符, 格式见输入样例。

输出说明:

输出使用 printf() 函数, 格式见输出样例。

输入样例:

5%2

输出样例:

5%2=1

```
#include <stdio.h>
int main()
{
    int a,b;
    char op;
    scanf("%d%c%d",&a,&op,&b);
    switch(op)
    {
        case '+': printf("%d%c%d=%d\n",a,op,b,a+b); break;
        case '-': printf("%d%c%d=%d\n",a,op,b,a-b); break;
        case '*': printf("%d%c%d=%d\n",a,op,b,a*b); break;
        // 注意分母为0时, 不会正确运算/,%
        case '/': if (b!=0) printf("%d%c%d=%d\n",a,op,b,a/b); break;
```

```
        case '%': if (b!=0) printf("%d%c%d=%d\n",a,op,b,a%b); break;
    }
    return 0;
}
```

*Note 2.5* (*printf* 双引号中的 % 输出, %% 表示输出 %).

```
int a,b;
char op;
printf("%d%%d=%d\n",a,b,a%b);
// 或当 op='%'时
printf("%d%c%d=%d\n",a,op,b,a%b);
```

## 2.2 数位输出

输入一个 5 位整数, 求出其各数位数值, 并按照从高位到低位的顺序输出, 如: 输入 12345, 输出为 1 2 3 4 5。

输入说明:

输入一个五位正整数。

输出说明:

按数位从高到低依次输出, 各数位之间以一个空格相分隔。

输入样例:

96237

输出样例:

9 6 2 3 7

```

#include<stdio.h>

/*****
5位整数已知. 首先用10000除以整数a(分子), 得到分子最高位。
改变分子分母, 循环迭代, 依次获得分子的最高位。
*****/

int main1()
{
    int a,b=10000,i=5; // i记录整数a的初始位数
    scanf("%d",&a);
    while(i>=1)
    {
        if (i==1) printf("%d\n",a/b); // 输出当前a的最高位
        else printf("%d□",a/b);
        a = a-a/b*b; // 去除当前a的最高位, 准备下轮迭代的分子a
        b/=10; // b=b/10, 准备下轮迭代的分母b
        i--;
    }
    return 0;
}

/*****
假设不知整数a的位数。
除10取余, 迭代循环, 可方便获取整数a的个位, 十位, 百位, 千位, ...
利用数组存储个位, 十位, 百位, 千位, ... 最后反序输出即是所求。
*****/

int main2()
{
    int a, tmp[100]; // tmp数组存储100(估计的最大值)个整数, 用tmp[0],tmp
[1],tmp[2],...读写各个整数。
    int i=0, j; // i: 记录整数a的位数
    scanf("%d",&a);
    if(a==0) // 考虑整数0的特殊情况, 直接输出即可。
    {
        printf("%d\n",a);
    }
    else // 因为循环语句判断a是否为0, 因此要有上述判断才能考虑到所有可能情
况的发生
    {
        while(a!=0) // 迭代逆序求出整数a的各位数字
        {
            tmp[i]=a%10; // 存储本轮循环a的末位数

```

```

        //printf("调式查看tmp[i]=%d\n",tmp[i]); // 提交时，别忘了注释或删除调试语句
        a=a/10;          // 改变分子，准备下轮循环
        i++;             // 位数递增
    }
    //printf("调式查看i=%d\n",i);
    // 逆序输出tmp，此时的i是整数a的位数，注意tmp的下标从i-1开始到下标0结束。
    for(j=i-1;j>=0;j--)
    {
        printf("%d□",tmp[j]);
    }
}
return 0;
}

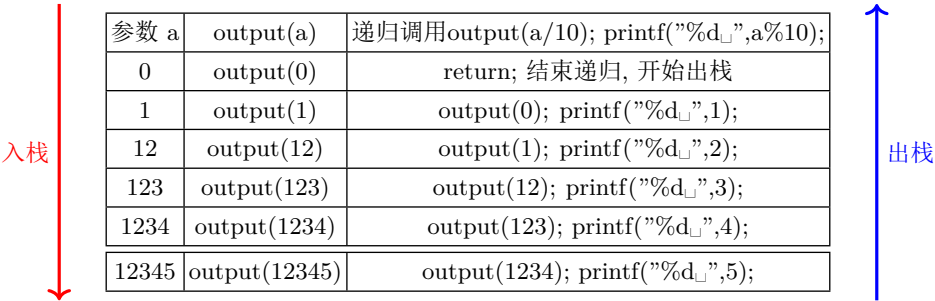
/*****
假设不知整数a的位数。
利用递归函数求解，a==0的情况在函数外处理输出较方便。
因此该函数仅考虑a!=0的情况。
*****/
void output(int a)
{
    if(a!=0) // 如果考虑a==0的情况，不好判断是初始a=0还是迭代后a=0的情况。
    这里考虑后者。前者的处理留给调用它的程序。
    {
        // '栈'是一种'先进后出'的数据结构
        output(a/10); // 递归调用，函数参数会自动存储在系统维护的'栈'中。
        printf("%d□",a%10); // 从内部存储'栈'中，依次弹出各位数，输出之。
    }
    else // a==0 ,可省略else语句，隐含结束递归调用
    {
        return; // 函数结束，注意本函数无返回值，因此return后无表达式。
    }
}

int main()
{
    int a;
    scanf("%d",&a);
    if(a==0) // 考虑整数0的特殊情况，直接输出即可。

```

```
{
    printf("%d\n",a);
}
else
{
    output(a); // 函数调用，完成逆序输出。
}
return 0;
}
```

图 2.1: 递归函数void output(int a)中系统内部维护的‘栈’结构示意图



Note 2.6 (知识点).

- 1. 体会除 10 取余, 迭代循环的整数分解技巧;
- 2. 第一种解法的 b=1000 初值是可计算的, 这样就可扩充此解法为任意位的整数 a。

```
// 因为a要在main1()函数的while循环中使用。
// 因此，定义临时变量，存储a的值，用于计算b的初值。
int tmp;
b=1; tmp=a;
while (tmp!=0)
{
    b=b*10;
    tmp=tmp/10;
}
```

- 3. 预习数组使用技巧;
- 4. 预习函数定义及调用;
- 5. 预习递归函数的定义, 体会系统维护的内部存储‘栈’的数据存储特点。



## 2.3 阶梯电价计费

电价分三个档次，[0,110] 度电，每度电 0.5 元；(110,210] 度电，超出 110 部分每度电 0.55 元，超过 210 度电，超出 210 部分每度电 0.70 元，给出一个家庭一月用电量，请计算出应缴的电费 (四舍五入，保留小数点后两位小数)。

输入说明：

输入数据为一个正实数，表示一月用电量

输出说明：

输出应缴电费，四舍五入保留 2 位小数。

输入样例：

输入样例 1

100

输入样例 2

200

输入样例 3

329.75

输出样例：

输出样例 1

50.00

输出样例 2

104.50

输出样例 3

193.83

```
#include <stdio.h>
int main()
{
    float sum,u1=0.5,u2=0.55,u3=0.70; // 用电量,每度电单价
    float fee = 0; // 应缴电费

    scanf("%f",&sum);

    if (sum > 210)
    {
        fee = (sum-210)*u3;
        sum = 210;
    }
    if (sum > 110)
    {
        fee += (sum-110)*u2; // fee=fee+(sum-110)*u2;
        sum = 110;
    }
}
```

```
    fee += sum*u1;

    printf("%.2f\n", fee);
    return 0;
}

int main2() // 另解
{
    float sum, u1=0.5, u2=0.55, u3=0.70; // 用电量, 每度电单价
    float fee = 0; // 应缴电费
    scanf("%f", &sum);

    if (sum >= 210)
        fee = 110*u1 + (210-110)*u2 + (sum-210)*u3;
    else if (sum >= 110)
        fee = 110*u1 + (sum-110)*u2;
    else
        fee = sum*u1;

    printf("%.2f\n", fee);
    return 0;
}

int main3() // 另解
{
    float sum, u1=0.5, u2=0.55, u3=0.70; // 用电量, 每度电单价
    float fee = 0; // 应缴电费

    scanf("%f", &sum);

    if (sum <= 110) fee = sum*u1;
    else if (sum <= 210)
    {
        fee = 110*u1;
        sum -= 110; // sum=sum-110;
        fee += sum*u2;
    }
    else // sum > 210
    {
        fee = 110*u1;
        fee += (210-110)*u2; // fee = fee+(210-110)*u2
    }
}
```

```

        sum -= 210;    // sum=sum-210;
        fee += sum*u3; // fee = fee+ sum*u3;
    }

    printf("%.2f\n", fee);
    return 0;
}

```

*Note 2.7* (四舍五入问题). 不同的编译系统, 处理结果可能不一致, `printf("%.2f\n", fee);` 默认输出即可。

*Note 2.8.* 练习 if 语句的不同组合形式, 杜绝出现 `if(110<=sum<=210)` 的错误形式。

## 2.4 计算某月天数

每年的 1, 3, 5, 7, 8, 10, 12 月有 31 天, 4, 6, 9, 11 月有 30 天, 闰年 2 月 29 天, 其他年份 2 月 28 天, 给定年份和月份求该月的天数

输入说明:

输入由两个正整数 a 和 b 构成, a 表示年份, b 表示月份, a 和 b 之间用空格分隔

输出说明:

根据年份和月份计算该月天数并输出

输入样例

输入样例 1

2000 3

输入样例 2

2001 2

输出样例

输出样例 1

31

输出样例 2

28

```

#include <stdio.h>
int main()
{
    int a, b, t = 0;
    scanf("%d%d", &a, &b);
    if((a%4 == 0 && a%100 != 0) || (a%100 == 0 && a%400 == 0))
    {
        if(b == 2) t = 29;
    }
    else if (b == 2) t = 28;
}

```

```
    if(b == 1 || b == 3 || b == 5 || b == 7 || b == 8 || b == 10 || b ==  
12) t = 31;  
    else if(b == 4 || b == 6 || b == 9 || b == 11) t = 30;  
  
    printf("%d\n",t);  
    return 0;  
}
```

*Note 2.9.* (逻辑运算符) &&, ||, !, 练习符合逻辑的各种组合形式。

*Note 2.10.* (闰年判断) ppt 中已详细说明, 还有同学写错。

## 2.5 计算整数各位数字之和

假设  $n$  是一个由最多 9 位数字 ( $d_9, \dots, d_1$ ) 组成的正整数。编写一个程序计算  $n$  的每一位数字之和。

输入说明:

输入数据为一个正整数  $n$

输出说明:

对整数  $n$  输出它的各位数字之和后换行

输入样例:

3704

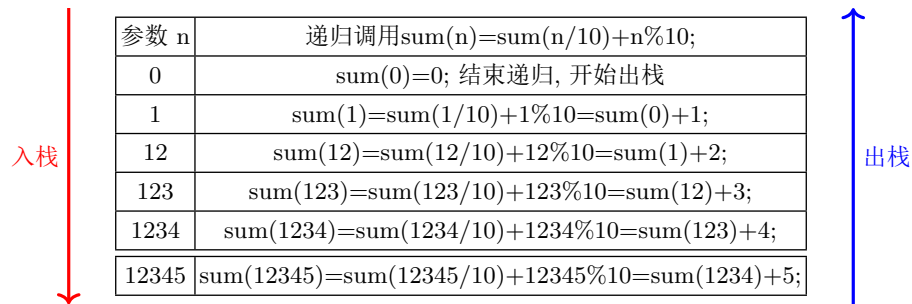
输出样例:

14

```
#include <stdio.h>
// 体会除10取余，迭代循环的整数分解技巧；
int main1()
{
    int n,sum = 0; // 注意初始化sum
    scanf("%d",&n);
    while(n) // 等效于n!=0
    {
        sum += n%10; // 累加本轮循环的末位数
        n /= 10;     // 准备下轮循环的分子
    }
    printf("%d",sum);
    return 0;
}

// 另解：定义递归函数，返回整数n的各位数之和
int sum(int n)
{
    if(n!=0)
    {
        // 递归调用，累加本轮循环的末位数
        return (sum(n/10)+n%10);
    }
    else // n==0时，结束递归调用
    {
        return 0; // 函数结束，返回整数0
    }
}

int main()
{
    int n;
    scanf("%d",&n);
    printf("%d\n",sum(n)); // 函数调用。
    return 0;
}
```

图 2.2: 递归函数 `int sum(int n)` 中系统内部维护的‘栈’结构示意图

Note 2.11 (知识点).

1. 体会除 10 取余, 迭代循环的整数分解技巧;
2. 预习递归函数定义及调用。

## 2.6 完数

请写一个程序, 给出指定整数范围  $[a, b]$  内的所有完数,  $0 < a < b < 10000$ 。一个数如果恰好等于除它本身外的所有因子之和, 这个数就称为“完数”。例如 6 是完数, 因为  $6 = 1 + 2 + 3$

输入说明

输入为两个整数  $a$  和  $b$ ,  $a$  和  $b$  之间用空格分隔

输出说明

输出  $[a, b]$  内的所有完数, 每个数字占一行

输入样例

1 10

输出样例

6

```
#include <stdio.h>

/*****
采用两层循环方案
(1) 外层循环使整数 i 递增, 完成区间 [n1, n2] 区间的完数计算
(2) 内层循环, 累加整数 i 的各因子
(3) 判断整数 i 是否是完数, 如果是, 输出之
*****/

int main1()
{
    int i, j, n1, n2, sum = 0;
    scanf("%d%d", &n1, &n2);
    for (i = n1; i <= n2; i++) // 外层循环使整数 i 递增, 完成区间 [n1, n2] 区间的完数计算
    {
```

```

        if(i == 1) continue; // 避免输出1, 1不是完数
        // i不等于1, 计算各因子
        sum = 1; // 不要忘记, 内层循环前sum的初始化。1总是一个整数的合法因子
        for(j = 2; j < i; j++) // 累加整数i的所有因子
        {
            if(i%j == 0) sum += j; // 如果j是i的因子, 累加之。
        }
        if(sum == i) printf("%d\n", i); // 如果i是完数, 输出之。
    }

    return 0;
}

/*****
采用一重循环 + 调用函数方案
(1) 一重循环使整数i递增, 函数compute调用, 完成区间[n1,n2]区间的完数计算
(2) 定义函数compute, 判断整数参数是否是完数, 如果是, 返回它, 否则返回-1
*****/

// 定义函数compute, 判断整数参数a是否是完数, 如果是, 返回a, 否则返回-1
int compute(int a)
{
    int i, s=1; // s用于存储a的各因子累加值, 1总是一个整数的合法因子
    if(a == 1)
    {
        return -1; // 1不是完数
    }
    // a不为1, 计算各因子
    for(i = 2; i < a; i++) // 累加整数a的所有因子
    {
        if(a%i == 0) s += i; // 如果i是a的因子, 累加之。
    }
    if(s == a)
    {
        return a; // 如果a是完数, 返回之。
    }
    // 如果程序执行到此处必然不是完数
    return -1;
}

```

```
// 另一种方式定义函数compute, 判断整数参数a是否是完数, 如果是, 返回a, 否则返回-1
// 一条return函数返回语句
int compute1(int a)
{
    int i, s=1; // s用于存储a的各因子累加值, 1总是一个整数的合法因子
    int ret=-1; // 用于返回值, 默认为-1

    for(i = 2; i < a; i++) // 累加整数a的所有因子
    {
        if(a%i == 0) s += i; // 如果i是a的因子, 累加之。
    }
    if(s == a && a!=1) // 如果a是完数, 返回值是本身。1不是完数
    {
        ret = a;
    }
    else // a不是完数
    {
        ret = -1;
    }
    return ret;
}

int main()
{
    int i, n1, n2;
    scanf("%d%d", &n1, &n2);
    for(i = n1; i <= n2; i++) // 调用函数compute, 完成区间[n1, n2]区间的完数计算
    {

        if(compute(i) != -1) printf("%d\n", i); // 如果i是完数, 输出之。
        // 测试函数compute1的调用
        // if(compute1(i) != -1) printf("%d\n", i); // 如果i是完数, 输出之。
    }

    return 0;
}
```

*Note 2.12* (特别注意). 且记: 进入内层循环前, 相关变量的初始化问题。

*Note 2.13* (函数定义和调用).



- 函数定义: 返回类型 函数名(参数列表) { 函数体 }
- `int fun1(float a, float b) { return a/b; // 返回整数部分 }`
- `void fun2(float a, float b) { printf(a/b); // 输出整数部分 }`
- 函数调用

```
float m,n;  
int ret;  
ret = fun1(m,n); // 调用函数fun1, 其返回值赋值给变量ret;  
fun2(m,n); // 调用函数fun2, 无返回值可用;
```



## Chapter 3

### 第 3 次机试练习：继续分支与循环练习

#### — 特别提示 —

- 定义变量名, 要有含义, 如: sum, select。否则, 程序不易读。
- 用尽可能少的变量, 完成题设, 好排错。
- 用尽可能简单的循环迭代结构, 好排错。不要将简单问题复杂化, 不好排错。
- 注意进入内层循环前的相关变量初始化问题。
- 变量的默认值必须是实际存在的值。例如, 选号程序中, 假设的 qq 号就是读取的第一个 qq 号。
- 提倡一题多解, 认真消化, 体会参考代码, 积累编程技巧。

#### 3.1 最大公约数

最大公约数 (GCD) 指某几个整数共有因子中最大的一个, 最大公约数具有如下性质,

$\gcd(a, 0) = a$

$\gcd(a, 1) = 1$

因此当两个数中有一个为 0 时, gcd 是不为 0 的那个整数, 当两个整数互质时最大公约数为 1。

输入两个整数 a 和 b, 求最大公约数

输入说明:

输入为两个正整数 a 和 b ( $0 \leq a, b < 10000$ ), a 和 b 之间用空格分隔,

输出说明:

输出其最大公约数

输入样例:

样例 1 输入

2 4

样例 2 输入:

12 6

样例 3 输入:

3 5

输出样例:

样例 1 输出

2

样例 2 输出

6

样例 3 输出

1

```
#include <stdio.h>
// 递归函数
int gcd(int a,int b)
{
    if(b==0) return a;    // 公约数就是a
    return gcd(b,a%b);    // 递归调用
}

int main() // 调用递归函数
{
    int a,b,t;
    scanf("%d%d",&a,&b);
    if(a<b) { t=a; a=b; b=t; } // 交换a,b
    printf("%d\n",gcd(a,b));    // 函数调用
    return 0;
}

int main1() // 暴力循环求解，效率低。
{
    int a,b,t=-1,i; //t给初值是好习惯，否则下面程序逻辑有可能使t得到随机值。
    scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句，除非题目要求
    if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
    if(b==0)
    {
        t=a; // 考虑分母为0的情况，比如：5,0的最大公约数为5
    }
    else
    {
        for(i=b;i>0;i--)
        {
            if(a%i==0 && b%i==0)
            {
                t=i; break; // 求得最大公约数，a,b互质，必然t=1
            }
        }
    }
    printf("%d\n",t);
    return 0;
}

int main2() // 利用欧几里得定理循环求解，效率高。
```

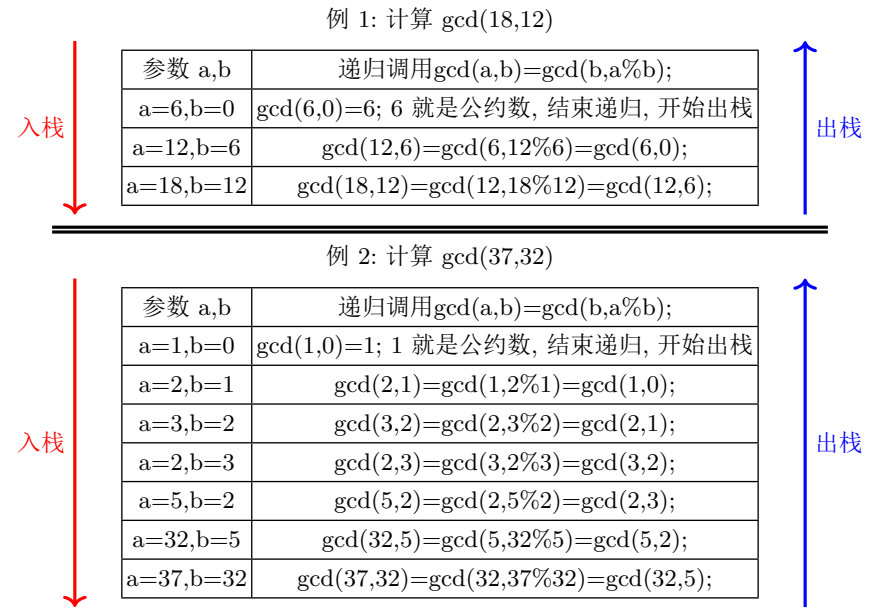
```
{
    int a,b,r,t;
    scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
    if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
    while(1)
    {
        if(b==0) { t=a; break; } // 分母为0时, a就是最大公约数
        r = a%b;
        if(r==0) {t=b; break;} // b就是最大公约数
        a=b; b=r; // 准备下一轮迭代
    }
    printf("%d\n",t); // 输出最大公约数
    return 0;
}

int main3() // 利用欧几里得定理循环求解, 效率高。
{
    int a,b,r,t;
    scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
    if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
    if (b==0) // 考虑分母为0的情况, 比如: 5,0的最大公约数为5
    {
        printf("%d\n",a);
    }
    else
    {
        // 排除了分母为0时不能求余数的情况
        while((r=a%b)!=0) // a/b的余数赋值给r,r不等于0时执行循环体
        {
            a=b;
            b=r;
        }
        printf("%d\n",b);
    }
    return 0; // 主函数结束
}

int main4() // 体会函数结束语句return的使用
{
    int a,b,r,t;
    scanf("%d%d",&a,&b); // 机试系统不要想当然给提示语句, 除非题目要求
```

```
if(a<b) { t=a; a=b; b=t; } // 交换a,b,使a是较大者
if (b==0) // 考虑分母为0的情况，比如：5,0的最大公约数为5
{
    printf("%d\n",a);
    return 0; // 主函数结束
}
// 排除了分母为0时不能求余数的情况
while((r=a%b)!=0) // a/b的余数赋值给r,r不等于0时执行循环体
{
    a=b; b=r; // 准备下一轮迭代
}
printf("%d\n",b);
return 0; // 主函数结束
}
```

图 3.1: 递归函数int gcd(int a,int b)中系统内部维护的‘栈’结构示意图



Note 3.1 (欧几里得定理).

a(大),b(小)的最大公约数：因为： $a=mb+r$ ， $m=a/b$ ； $r=a\%b$ ， $\Rightarrow a,b$ 的公约数能整除  $b$  和  $r$ 。  
 $r=a\%b$ ,  $r$  为 0，则  $b$  就是最大公约数。否则迭代循环， $a=b$ ， $b=r$ ，直到余数为零，则分母就是最大公约数。

Note 3.2. 预习题及递归函数的使用。

### 3.2 角谷定理

角谷定理定义如下: 对于一个大于 1 的整数  $n$ , 如果  $n$  是偶数, 则  $n = n / 2$ 。如果  $n$  是奇数, 则  $n = 3 * n + 1$ , 反复操作后,  $n$  一定为 1。

例如输入 22 的变化过程: 22 -> 11 -> 34 -> 17 -> 52 -> 26 -> 13 -> 40 -> 20 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1, 数据变化次数为 15。

输入一个大于 1 的整数, 求经过多少次变化可得到自然数 1。

输入说明

输入为一个整数  $n$ ,  $1 < n < 100000$ 。

输出说明

输出变为 1 需要的次数

输入样例

样例 1 输入

22

样例 2 输入

33

输出样例

样例 1 输出

15

样例 2 输出

26

```
#include <stdio.h>
int main()
{
    int n, i=0; // 变量i用于计数的辅助变量
    scanf("%d",&n);
    // 因为题目输入假设n>1, 因此不必考虑n=1时的情况
    while(n!=1) // n不等于1时执行循环体中的语句
    {
        if(n%2==0) n=n/2;
        else n=3*n+1;
        i++;
    }
    printf("%d\n", i);
    return 0;
}

// 含程序调试语句, 不吝惜写一些printf语句, 观察程序的执行过程。
int main()
{
    int n=22, i=0; // 变量i用于计数的辅助变量
```



```

//scanf("%d",&n); // 调试时可以注释掉输入语句，改变变量n的值，观察执行过程

printf("%d->",n);
while(n!=1) // n不等于1时执行循环体中的语句
{
    if(n%2==0)
    {
        n=n/2;
    }
    else
    {
        n=3*n+1;
    }
    printf("%d->",n);
    i++;
}
printf("\n总共变化次数%d\n",i);
return 0;
}

```

*Note 3.3.* 试着用do{ }while(); for (;) 改写此程序，执行相同功能。

### 3.3 整数分析

给出一个整数  $n$  ( $0 \leq n \leq 100000000$ )。求出该整数的位数，以及组成该整数的所有数字中的最大数字和最小数字。

输入说明

输入一个整数  $n$  ( $0 \leq n \leq 100000000$ )

输出说明

在一行上依次输出整数  $n$  的位数，以及组成该整数的所有数字中的最大数字和最小数字，各个数字之间用空格分隔。

输入样例

217

输出样例

3 7 1

```

#include <stdio.h>
// 循环除10取余是整数分解的基本技巧
int main()
{

```

```

int i = 0, n, bit, max, min;
scanf("%d",&n);
while(n) // 等效于 while(n!=0)
{
    bit = n%10; // 获取n的最低为
    // 切记: 初始化时, 假设的max和min必须是实际存在的数。
    if(i == 0) // 初始化: 原始n的最低位设为最大和最小数字
    {
        max = min = bit;
    }
    else
    {
        if(bit > max) max = bit;
        if(bit < min) min = bit;
    }
    n /= 10; // 去除最低位
    i++;
}
// (i == 0 ? 1 : i)是条件表达式, 表达式的值是:
// 如果i==0,则表达式的值为1否则表达式的值是i
printf("%d_%d_%d\n", (i == 0 ? 1 : i), max, min); // 考虑原始n==0的情况
return 0;
}

```

Note 3.4 (知识点).

1. 整数数位分解是基本编程练习之一。
2. 切记: 初始化时, 假设的 max 和 min 必须是实际存在的数。比如不能想当然假设 max=1000, min=0.
3. 注意审题: “输入一个整数  $n$ , ( $0 \leq n \leq 100000000$ )”, 因此, 0 也是一个合法输入。

### 3.4 冰箱温度预测

编写一个程序, 用于预测冰箱断电后经过时间  $t$  (以小时为单位) 后的温度  $T$ 。已知计算公式如下所示

$$T = \frac{4t^2}{t+2} - 20$$

输入说明

输入两个整数  $h$  和  $m$  表示冰箱断电后经过的时间,  $h$  表示小时,  $m$  表示分钟

输出说明

输出冰箱断电后经过时间  $t$  (以小时为单位) 后的温度  $T$ , 保留两位小数

输入样例

2 0

输出样例

-16.00

```
#include <stdio.h>
int main()
{
    int h,m;
    float t,T;
    scanf("%d%d",&h,&m);
    t = h + m/60.0;    // 必须是60.0, why?
    T = 4*t*t/(t+2)-20; // 优先级保证了计算的正确性, why?
    printf("%.2f\n",T);

    return 0;
}
```

*Note 3.5 (知识点).* 整数/整数, 表达式的值是整数部分, 自动舍去小数部分。

### 3.5 除法计算器

小明的弟弟刚开始学习除法, 为了检查弟弟的计算结果是否正确, 小明决定设计一个简单计算器程序来验算。

输入说明

输入数据由四个整数  $m$ ,  $n$ ,  $q$ ,  $r$  构成,  $m$  为被除数,  $n$  为除数,  $q$  和  $r$  为小明的弟弟计算出的商和余数。整数之间用空格分隔, 所有整数取值范围在  $(-100000 \sim 100000)$ ,  $n$  不为 0。

输出说明

如果验算结果正确, 输出 yes, 否则输出正确的商和余数

输入样例:

样例 1:

10 3 3 1

样例 2:

10 3 3 2

输出样例

样例 1 输出:

yes

样例 2 输出:

3 1

```
#include <stdio.h>
int main()
{
```

```

int m,n,q,r;
scanf("%d%d%d%d",&m,&n,&q,&r);
if(m==q*n+r && q==m/n && r==m%n) printf("yes\n");
else printf("%d□%d\n",m/n,m%n);
return 0;
}

```

Note 3.6. 改变题设条件, 修改此程序, 进行各种表达式计算练习, 分析优先级。如果  $n=0$  时, 如何处理。

### 3.6 自然数分解

任何一个自然数  $m$  的立方均可写成  $m$  个连续奇数之和。例如:

$$1^3 = 1$$

$$2^3 = 3 + 5$$

$$3^3 = 7 + 9 + 11$$

$$4^3 = 13 + 15 + 17 + 19$$

编程实现: 输入一自然数  $n$ , 求组成  $n^3$  的  $n$  个连续奇数。

输入说明

一个正整数  $n$ ,  $0 < n < 30$ 。

输出说明

输出  $n$  个连续奇数, 数据之间用空格隔开, 并换行

输入样例

4

输出样例

13 15 17 19

```

#include <stdio.h>
// 从估计的第一个奇数开始, 循环迭代求解。
int main()
{
    int n,i,j,sum,first;
    scanf("%d",&n);

    // 第一个可能的奇数:
    if(n%2) first = n;    // n是奇数
    else first = n + 1;   // n是偶数

    while(1)
    {

```

```

sum = 0; // 每趟内层循环前，必须置0
// 从first开始，n个连续奇数，i：表示连续奇数，j：计数。
for(i = first ,j = 1; j <= n; i += 2,j++)
{
    sum += i; // 连续奇数累加
    if(sum == n*n*n)
    {
        // 输出
        for(i = first ,j = 1; j <= n; i += 2,j++)
        {
            if (j == n)printf("%d\n",i);
            else printf("%d_",i);
        }
        return 0; // 函数结束
    }
}
first += 2;
}
return 0;
}

```

*Note 3.7 (要点).* 再次强调进入内层循环前，相关变量的初始化; 以及标志变量 (如本例 first) 的使用技巧。

### 3.7 选号程序

小明决定申请一个新的 QQ 号码，系统随机生成了若干个号码供他选择。小明的选号原则是：

1. 选择所有号码中各位数字之和最大的号码。
2. 如果有多个号码各位数字之和相同则选择数值最大的号码。

请你写一个程序帮助小明选择一个 QQ 号码。

输入说明

输入数据由两行构成，第一行为一个整数  $n$  表示有  $n$  个待选号码 ( $0 < n < 100$ )，第二行有  $n$  个正整数，表示各个待选的号码，每个号码长度不超过 9 位数。每个号码之间用空格分隔，且每个号码都不相同。

输出说明

输出根据小明的选号原则选出的号码。

输入样例

5

10000 11111 22222 333 1234

输出样例

22222

```

#include <stdio.h>
// 在循环语句中, 读取备选qq号, 计算各位之和, 依据筛选条件选取qq号
int main()
{
    // 关键变量含义说明:
    // select_qq, select_sum表示备选qq及其各位之和
    // qq, sum表示当前读取的qq及其各位和
    int i, n, select_qq, select_sum, qq, sum, tmp;
    scanf("%d", &n);
    for (i = 0; i < n; i++) // 注意条件表达式, 表明i的最大值是n-1, 因为i是0开始的,
        因此共执行n次循环
    {
        scanf("%d", &qq); // 读取当前备选qq号
        tmp = qq; // 保存到临时变量中, 因为下面的循环语句要更改。
        sum = 0; // 当前读取qq号的各位之和。 注意: 一定要初始化, 否则上一个
        备选号的sum值会带入本轮循环中。
        while (tmp) // 计算各位之和
        {
            sum += tmp % 10;
            tmp /= 10;
        }
        // 第1轮迭代(i==0), 当前读取的qq就是所选, 其它根据题设条件选号
        // 因为三个表达式为||运算, 从左到右依次计算各表达式的值, 如果为真,
        则不会计算后边表达式。
        // 因此, 当i==0时不会计算其它两个表达式的值, if条件为真。
        if (i == 0 || sum > select_sum || (sum == select_sum && qq > select_qq))
        {
            select_qq = qq;
            select_sum = sum; // i==0时, select_sum初值为第一个号码各位之和。
        }
    }
    printf("%d", select_qq);
    return 0;
}

// 解法2: 用二维数组存储所有qq号及其各位和
#define N 100 // 估计最大数组长度
int main1()
{
    // 二维数组No, 第一列表示qq号, 第二列表示该qq号的各位数字之和。
    int i, n, No[N][2], tmp, sum, max = 0, largest = 0, select;

```

```
scanf("%d",&n);  
// 筛选条件2  
for(i=0;i<n;i++)  
{  
    scanf("%d",&No[i][0]);  
    tmp=No[i][0];  
    sum=0; // 一定初始化  
    while(tmp)  
    {  
        sum+=tmp%10;  
        tmp/=10;  
    }  
    No[i][1]=sum;  
    if(sum>=max) max=sum;  
}  
// 筛选条件1  
for(i=0;i<n;i++)  
{  
    if(No[i][1]==max) // 备选号码  
    {  
        if(No[i][0]>=largest)  
        {  
            select=No[i][0];  
            largest=No[i][0];  
        }  
    }  
}  
printf("%d",select);  
return 0;  
}
```

Note 3.8 (要点).

1. || 和 && 运算从左到右执行, 取得结果, 则不执行后面的表达式。  
取得结果的含义是:  
if (条件 1|| 条件 2|| 条件 3) 运算中, 只要有一个条件表达式为真 (非 0), 即整个条件 () 结果即为真。  
if (条件 1 && 条件 2 && 条件 3) 运算中, 只要有一个条件表达式为假 (0), 即整个条件 () 结果即为假。
2. 比较两种解法的优缺点。
3. 本例是循环迭代的范例, 应反复演练, 领会迭代程序的编程技巧。
4. 试着定义函数, 改写此程序。
5. 本题不必使用排序算法, 使程序复杂化。