第 3 章　程序控制结构

3.1　流程图及程序控制结构简介


流程图

3.1.1　流程图

关于什么是程序，计算机科学家沃思曾给出如下定义：

程序＝数据结构＋算法

其中，算法是为了解决特定问题而规定的一个有限长的操作序列。

算法可以采用自然语言、流程图、伪代码、计算机语言等表示方法描述。

采用流程图描述算法的优点是直观形象、易于理解。流程图包括传统流程图、结构化流程图等，本书仅介绍传统流程图。

一个传统流程图包括表示相应操作的框；带箭头的流程线；框内外必要的文字说明。

传统流程图的基本要素包括起止框、输入/输出框、处理框、判断框、流程线、连接点等，具体框图如图 3-1 所示。

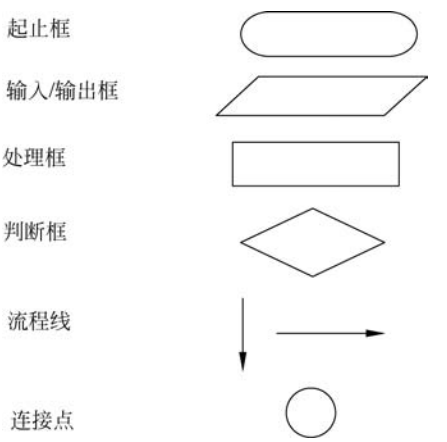

图 3-1　传统流程图基本要素框图

其中，起止框表示算法的开始与结束；输入/输出框表示输入与输出；处理框表示处理；判断框表示条件判断；流程线表示算法的流程；连接点用于连接不同页内的同一算法。

例 3.1.1　“统计闰年”算法的流程图表达

请用传统流程图描述“统计并输出年份区间[s , t]中有几个闰年”的算法。

闰年的条件：闰年是能被 4 整除，但不能被 100 整除的年份或者能被 400 整除的年份。

统计闰年的算法主体部分主要包含输入、处理、输出三部分，具体流程图如图 3-2 所示。其中，虚框标注的是处理部分。

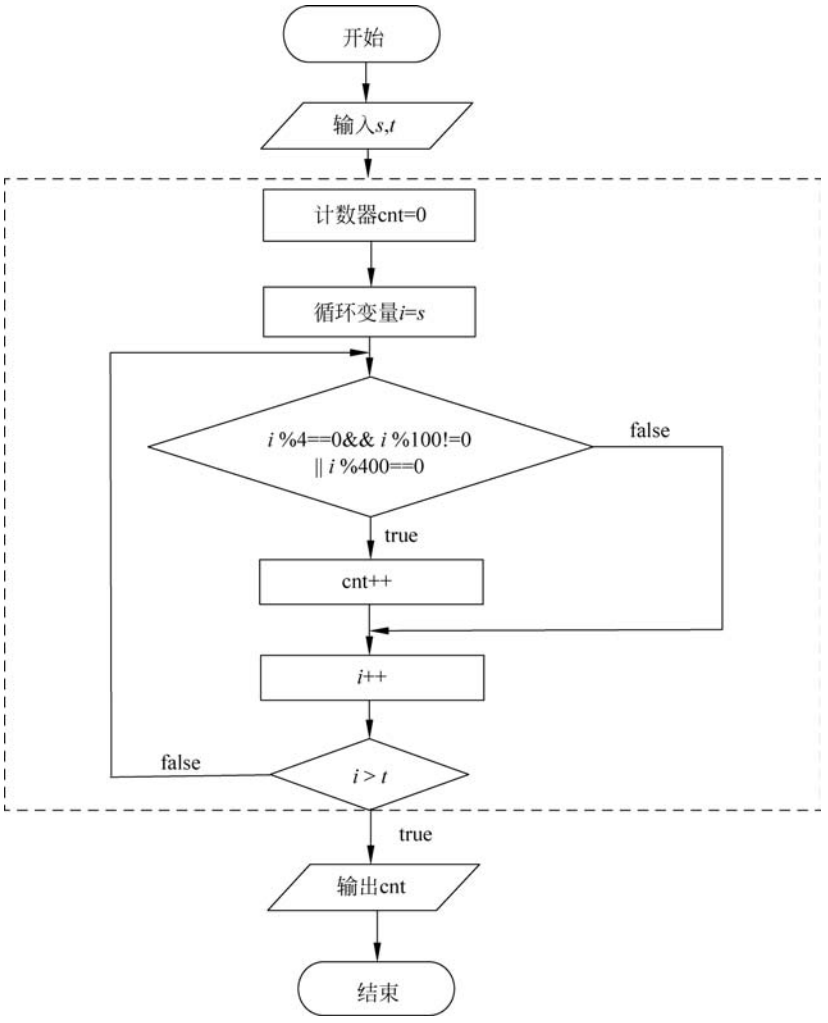

图 3-2 “统计闰年”的传统流程图表达

采用流程图描述算法是非常重要的一项程序设计技能，本书中以此例说明算法的流程图表示方法。作为入门练习，读者可以尝试用传统流程图表达以下问题的算法。

(1) 输入三角形的三边长，用海伦公式求其面积。

(2) 输入三个人的身高，找出其中的高个。

(3) 输入 n 及 n 个人的身高，找出 n 个人中身高的最大值。

此后，读者可以针对以后的例题、习题画出流程图，进一步熟悉和巩固采用传统流程图表达算法的方法。

3.1.2 程序控制结构简介

程序控制结构主要包括顺序结构、选择结构、循环结构。

顺序结构是按语句的书写顺序执行的程序结构。

选择结构是根据特定的条件决定执行哪个语句的程序结构,常用 if 语句和 switch 语句。

循环结构是在满足特定的条件时重复执行某些语句的程序结构,常用 for、while 和 do…while 等语句。

顺序结构的流程图如图 3-3 所示。

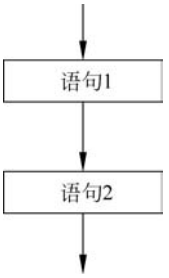

图 3-3 顺序结构流程图

例如,下面的代码先输入 a、b,再输出它们的和。

```
int a, b;                          //定义变量
cin >> a >> b;                     //输入
cout << a + b << endl;             //输出
```

上面代码定义变量、输入及输出三个基本语句按顺序依次执行下来。

在 C/C++源程序中,基本语句以分号";"作为结束标记;仅由单个分号";"构成的语句称为空语句;复合语句则以大括号"{}"括起来,例如:

```
{
    a = a + b;
    cout << a << endl;
}
```

3.2 选择结构

3.2.1 if 语句及其使用

例 3.2.1

例 3.2.1 两者中的大者

输入两个整数 a,b,找出其中的大者并输出。

思路 1:先假设第一个数大,然后与后一个数比较,若后一个数大,则大者为后一个。可用单分支 if 语句实现,具体代码如下。

```
//C++风格,单分支选择结构
#include <iostream>
using namespace std;
int main() {
    int a,b,c;
    cin >> a >> b;
    c = a;                         //假设第一个数大,并放到 c 中
    if (b > c)c = b;               //若第二个数大于假设的最大数 c,则把第二个数放到 c 中
    cout << c << endl;
    return 0;
}
```

```
/*C风格,单分支选择结构*/
#include<stdio.h>
int main() {
    int a,b,c;
    scanf("%d%d",&a,&b);
    c=a;                          /*假设第一个数最大,并放到c中*/
    if (b>c) c=b;                 /*如果第二个数大于假设的最大数c,则把第二个数放到c中*/
    printf("%d\n",c);
    return 0;
}
```

思路2：直接比较两个数，若前一个数大，则把它作为结果输出，否则输出后一个数。可用双分支if语句实现，具体代码如下。

```
//C++风格,双分支选择结构
#include<iostream>
using namespace std;
int main() {
    int a,b;
    cin>>a>>b;
    if (a>=b)                     //若第一个数大于等于第二个数
        cout<<a<<endl;            //则输出第一个数
    else
        cout<<b<<endl;            //否则输出第二个数
    return 0;
}
/*C风格,双分支选择结构*/
#include<stdio.h>
int main() {
    int a,b;
    scanf("%d%d",&a,&b);
    if (a>=b)                     //若第一个数大于等于第二个数
        printf("%d\n",a);         //则输出第一个数
    else
        printf("%d\n",b);         //否则输出第二个数
    return 0;
}
```

本题两种思路的代码中分别使用了单分支、双分支if语句。

1. 基本的if语句

基本if语句格式如下。

if（条件）语句1［else 语句2］

描述语法时，［］表示其中的内容是可选项，即if语句可以是单分支if语句，如下。

if（条件）语句1

此if语句在满足条件时执行语句1，否则不执行任何语句，其流程图如图3-4所示。

或者是双分支if语句，如下。

if（条件）语句1 else 语句2

此if语句在满足条件时执行语句1，否则执行语句2，其流程图如图3-5所示。

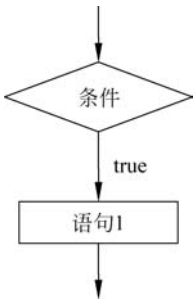

图 3-4　单分支 if 语句流程图

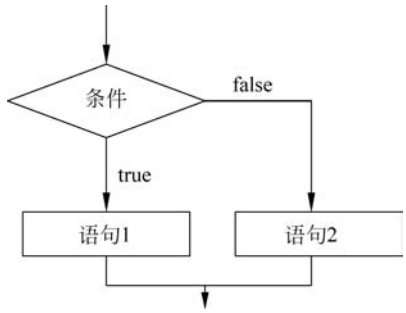

图 3-5　双分支 if 语句流程图

可见,if 语句可带 else 子句(双分支选择结构),也可以不带(单分支选择结构)。

if 语句的条件需用小括号括起来,该条件一般是一个逻辑表达式或关系表达式等值为 true(或 1)或 false(或 0)的表达式,否则一切 0 值转换为 false,一切非 0 值转换为 true。

语句 1 和语句 2 可以是基本语句,也可以是复合语句。例如:

```
if(x)                                  //x 相当于 x!= 0
    printf("x is non - zero\n");       //基本语句
else
    printf("x is zero\n");
if(x == 1) {                           //复合语句
   a = 1;
   b = 2;
}
else {
   a = - 1;
   b = - 2;
}
```

2. 嵌套的 if 语句

嵌套的 if 语句是指在 if 语句中又使用 if 语句。if 语句可以嵌套在 if 子句中,也可以嵌套在 else 子句中。

从第一个 else 开始,else 总与离它最近的且未被匹配的 if 配对(最近匹配原则)。

观察如下代码,思考能否达到“当 n 小于等于 0 时把 z 赋值为 b,否则当 a 大于 b 时,把 z 赋值为 a”的目的。

```
if(n > 0)
    if(a > b) z = a;
else
    z = b;
```

上面的 else 虽然在缩排上与 if(n>0)对齐,但根据最近匹配原则,这个 else 与 if(a>b)匹配,达到的效果是“当 n 大于 0 且 a 小于等于 b 时把 z 赋值为 b”。为避免类似问题,建议在 else 子句中嵌套 if 语句。

思考:如何才能使得 else 与 if(n>0)匹配?

可以把第 2 个 if 语句用{}括起来:

```
if(n>0) {
    if(a>b) z=a;
}
else
    z=b;
```

或者，把 if 语句嵌套到 else 子句中：

```
if(n<=0)
    z=b;
else if(a>b)
    z=a;
```

3. if 选择结构示例

例 3.2.2　三者的最大值


例 3.2.2

输入三个整数，找出其中最大的一个并显示出来。

思路：假设第一个数最大并放到结果 d 中，若后面的数大于 d，则把 d 变为该数。可用单分支语句实现，代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int a, b, c, d;
    cin>>a>>b>>c;
    d=a;                          //假设第一个数最大，把其存放在 d 中
    if (b>d) d=b;                 //若第二个数大于假设的最大数 d，则把 d 改为该数
    if (c>d) d=c;                 //若第三个数大于假设的最大数 d，则把 d 改为该数
    cout<<d<<endl;
    return 0;
}
/*C 风格*/
#include<stdio.h>
int main() {
    int a, b, c, d;
    scanf("%d%d%d",&a,&b,&c);
    d=a;                          /*假设第一个数最大，把其存放在 d 中*/
    if (b>d) d=b;                 /*若第二个数大于假设的最大数 d，则把 d 改为该数*/
    if (c>d) d=c;                 /*若第三个数大于假设的最大数 d，则把 d 改为该数*/
    printf("%d\n",d);
    return 0;
}
```


例 3.2.3

这种思路比较简单，而且容易扩展到多个数的情况（结合循环结构）。读者可以思考本题还有哪些其他实现方法，并自行编写代码实现。

例 3.2.3　三数排序

输入三个整数，然后按从大到小的顺序把它们显示出来。

这个问题该如何实现呢？仔细思考，可以想到多种方法。下面给出两种方法，分别用到选择排序和冒泡排序的思想。

思路1：采用选出当前最大者放到当前最前面位置(选择排序)的思想。实现代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int a,b,c,d;
    cin>>a>>b>>c;
    if (a<b)                          //前两个数比较,若位置不对则交换,使得a>=b
        d=a,a=b,b=d;
    if (a<c)                          //前两个数的大者与c比较,若位置不对则交换,a为最大者
        d=a,a=c,c=d;
    if (b<c)                          //后两个数的位置若不对,则交换
        d=b,b=c,c=d;
    cout<<a<<" "<<b<<" "<<c<<endl;
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int a,b,c,d;
    scanf("%d%d%d",&a,&b,&c);
    if (a<b)                          /*前两个数比较,若位置不对则交换,使得a>=b*/
        d=a,a=b,b=d;
    if (a<c)                          /*前两个数的大者与c比较,若位置不对则交换,a为最大者*/
        d=a,a=c,c=d;
    if (b<c)                          /*后两个数的位置若不对,则交换*/
        d=b,b=c,c=d;
    printf("%d %d %d\n",a,b,c);
    return 0;
}
```

思路2：采用把当前最小者放到当前最后面位置(冒泡排序)的思想。实现代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int a,b,c,d;
    cin>>a>>b>>c;
    if (a<b)                          //前两个数比较,若位置不对则交换,使得a>=b
        d=a,a=b,b=d;
    if (b<c)                          //后两数比较,若位置不对则交换,c为最小者
        d=b,b=c,c=d;
    if (a<b) {                        //前两个数的位置若不对,则交换
        d=a,a=b,b=d;
    cout<<a<<" "<<b<<" "<<c<<endl;
    return 0;
}
/*C风格*/
#include<stdio.h>
```

```
int main() {
    int a,b,c,d;
    scanf("%d%d%d",&a,&b,&c);
    if (a<b)                          /*前两个数比较,若位置不对则交换,使得 a>=b*/
        d=a,a=b,b=d;
    if (b<c)                          /*后两数比较,若位置不对则交换,c 为最小者*/
        d=b,b=c,c=d;
    if (a<b)                          /*前两个数的位置若不对,则交换*/
        d=a,a=b,b=d;
    printf("%d %d %d\n",a,b,c);
    return 0;
}
```

例 3.2.4　成绩转换


例 3.2.4(1)

百分制成绩转换为五级计分制时,90～100 分以上等级为 A,80～89 分等级为 B,70～79 分等级为 C,60～69 分等级为 D,0～59 分等级为 E。输入一个百分制成绩,输出等级。

本例是多分支结构,可以用嵌套 if 语句实现,这里使用 if 嵌套在 else 子句中的方法,实现代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int score;
    char rank;
    cin>>score;
    if (score>=90) rank='A';
    else if (score>=80) rank='B';
    else if (score>=70) rank='C';
    else if (score>=60) rank='D';
    else rank='E';
    cout<<rank<<endl;
    return 0;
}
/*C 风格*/
#include<stdio.h>
int main() {
    int score;
    char rank;
    scanf("%d",&score);
    if (score>=90) rank='A';
    else if (score>=80) rank='B';
    else if (score>=70) rank='C';
    else if (score>=60) rank='D';
    else rank='E';
    printf("%c\n",rank);
    return 0;
}
```

3.2.2 switch 语句及其使用

例 3.2.4 是多分支选择结构,也可以使用 switch 语句实现。具体代码如下。


例 3.2.4(2)

```
//C++风格
#include <iostream>
using namespace std;
int main() {
    int score;
    char rank;
    cin >> score;
    switch(score/10) {              //对百分制成绩作除以 10 的运算,减少常量个数
        case 9:
        case 10:
            rank = 'A'; break;
        case 8:
            rank = 'B'; break;
        case 7:
            rank = 'C'; break;
        case 6:
            rank = 'D'; break;
        default:
            rank = 'E';
    }
    cout << rank << endl;
    return 0;
}
/* C 风格 */
#include <stdio.h>
int main() {
    int score;
    char rank;
    scanf("%d",&score);
    switch(score/10) {              /* 对百分制成绩作除以 10 的运算,减少常量个数 */
        case 9:
        case 10:
            rank = 'A'; break;
        case 8:
            rank = 'B'; break;
        case 7:
            rank = 'C'; break;
        case 6:
            rank = 'D'; break;
        default:
            rank = 'E';
    }
    printf("%c\n",rank);
    return 0;
}
```

上面的程序根据表达式 score/10 的值去匹配 case 后面的常量值，匹配成功则执行其后的语句，执行到 break 时则跳出 switch 语句；若都匹配不上，则执行 default(对应 0、1、2、3、4、5 等常量)后的语句。另外，case 标号后的多条语句不需要用{}括起来；多个 case 标号可以共有一组语句。

switch 语句常用于实现多分支选择结构，其语句格式如下。

switch(表达式){

case 常量表达式 1:语句序列 1

case 常量表达式 2:语句序列 2

…

case 常量表达式 *n*:语句序列 *n*

[default:语句序列 *n*+1]

}

switch 语句中的表达式及常量表达式一般应为整型或字符型。执行时，按表达式的值与 case 后的常量表达式相匹配，若匹配成功则执行相应语句序列，否则执行 default 后的语句序列，流程图如图 3-6 所示。

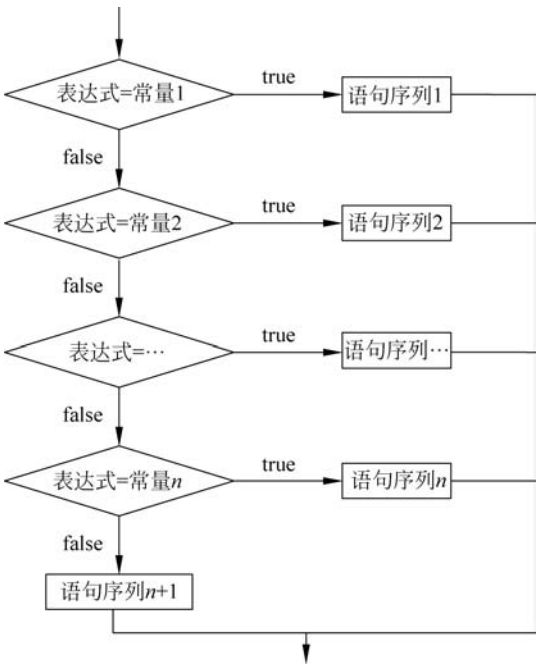

图 3-6　switch 语句流程图

各个 case 块(含 default 块)的顺序可以交换。一般每个语句序列(处于最后面的除外)都是由包含 break 的多条语句构成，但不需要用{}括起来。

如果各 case 块的语句序列中没有用 break 语句跳出 switch 语句，则语句序列将顺序执行下去(不再考虑表达式是否匹配常量值)。例如：

```
int n;
cin >> n;                              //scanf("%d",&n);
```

```
switch(n) {
    case 10:
        cout <<'A'<< endl;          //printf("A\n");
    case 11:
        cout <<'B'<< endl;          //printf("B\n");
    case 12:
        cout <<'C'<< endl;          //printf("C\n");
}
```

若输入 10,则运行结果如下。

```
A
B
C
```

如上面的代码所示,switch 语句可以省略 default 子句。

若各 case 语句序列中含有 break 语句,即:

```
int n;
cin >> n;                           //scanf("%d",&n);
switch(n) {
    case 10:
        cout <<'A'<< endl; break;   //printf("A\n"); break;
    case 11:
        cout <<'B'<< endl; break;   //printf("B\n"); break;
    case 12:
        cout <<'C'<< endl; break;   //printf("A\n"); break;
}
```

若输入 10,则运行结果如下。

```
A
```

例 3.2.5

例 3.2.5　某月的天数

输入年份 year、月份 month,判断该月的天数。

已知大月有 31 天,小月有 30 天,2 月有 28 天或 29 天(闰年)。大月有 1、3、5、7、8、10、12,共 7 个月;小月有 4、6、9、11,共 4 个月。

本题需要根据不同的情况(大月、小月、2 月)确定该月的天数,用 if 或 switch 语句都可以完成。下面给出使用 switch 语句的代码。

```
//C++风格
#include <iostream>
using namespace std;
int main() {
    int year,month,days;
    cin >> year >> month;
    switch(month) {
        case 2:
```

```
            if (year % 4 == 0&&year % 100!= 0||year % 400 == 0)
                days = 29;
            else
                days = 28;
            break;
        case 4:case 6:case 9:case 11:
            days = 30;
            break;
        default:
            days = 31;
            break;
    }
    cout << days << endl;
    return 0;
}
/ * C 风格 * /
# include < stdio.h >
int main() {
    int year,month,days;
    scanf(" % d % d",&year,&month);
    switch(month) {
        case 2:
            if (year % 4 == 0&&year % 100!= 0||year % 400 == 0)
                days = 29;
            else
                days = 28;
            break;
        case 4:case 6:case 9:case 11:
            days = 30;
            break;
        default:
            days = 31;
            break;
    }
    printf(" % d\n",days);
    return 0;
}
```

3.3 循环结构

例 3.3.1

3.3.1 引例与三种循环语句

1. 引例

例 3.3.1 *n* 个整数中的最大值

首先输入一个整数 n，然后输入 n 个整数，请输出这 n 个整数中的最大值。

在例 3.2.2 中，使用两条类似的单分支 if 语句可求得 3 个数的最大值。现在要求 n 个数中最大值，若程序运行前已知 n 的值，则也可以写 $n-1$ 个单分支语句完成，但 n 是一个变量，在程序运行中输入后才能确定其值，因此无法在程序运行前写好 $n-1$ 个 if 语句。由

于 $n-1$ 个 if 语句是重复的,可以写一条 if 语句并使其执行 $n-1$ 次,这可以使用循环结构完成。循环结构中的 for 语句常用于实现次数固定且重复执行的要求。求 n 个整数中的最大值的代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int n,t,max;
    cin>>n;                          //先输入数据个数
    cin>>t;                          //先输入第1个数
    max=t;                           //假设第1个数最大,并放到max中
    for(int i=1; i<n; i++) {         //控制输入n-1个数并做比较
        cin>>t;
        if (t>max)                   //若后面的数大于当前的最大数
            max=t;                   //则把当前的最大数赋值为后面的数
    }
    cout<<max<<endl;
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int i,n,t,max;
    scanf("%d",&n);                  /*先输入数据个数*/
    scanf("%d",&t);                  /*先输入第1个数*/
    max=t;                           /*假设第1个数最大,并放到max中*/
    for(i=1; i<n; i++) {             /*控制输入n-1个数并做比较*/
        scanf("%d",&t);
        if (t>max)                   /*若后面的数大于当前的最大数*/
            max=t;                   /*则把当前的最大数赋值为后面的数*/
    }
    printf("%d\n",max);
    return 0;
}
```

上面的代码中,for 语句中循环变量 i 从 1 到 $n-1$ 进行循环,共执行 $n-1$ 次循环体(输入 t,再比较 max 和 t,把大者保存在 max 中)。

2. 三种循环语句

思考:如何求 $1+2+\cdots+n$ 的累加和(结果保证在 int 类型的范围内)?

一种常见的方法是直接采用循环逐项累加,下面给出使用 for、while 和 do…while 语句的 C 风格代码。

```
/*C风格*/
#include<stdio.h>
int main() {
    int sum1=0,sum2=0,sum3=0,n,i;
    scanf("%d",&n);
    for(i=1; i<=n; i++) sum1 += i;
```

```
    printf("%d\n",sum1);
    i=1;
    while(i<=n) {
        sum2 += i;
        i++;
    }
    printf("%d\n",sum2);
    i=1;
    do {
        sum3 += i;
        i++;
    } while(i<=n);
    printf("%d\n",sum3);
    return 0;
}
```

运行结果：

```
10↵
55
55
55
```

可见，三种循环语句的运行结果相同。实际上，三种循环语句可以相互转换，例 3.3.1 也可以使用另外两种循环语句实现。若循环体有多条语句，则必须以{}括起来构成复合语句。建议：循环体只有一条语句也要用{}括起来。

3.3.2 for 语句及其使用

1. for 语句基本格式

for 循环语句的基本格式如下。

for(表达式 1；表达式 2；表达式 3)

**　　循环体**

例如，在线做题时，对于 T 组测试数据，可用如下代码控制。

```
int i, T;
cin>>T;                              //C 风格：scanf("%d",&T);
for(i=1;i<=T;i++) {
    //一组测试的代码
}
```

for 后()中两个分号必不可少，三个表达式可以根据具体情况省略。其中，表达式 1 通常是循环变量初始化或给循环变量赋值，可以逗号表达形式给多个变量赋值；表达式 2 是循环条件，在满足循环条件时反复执行循环体，否则结束循环；表达式 3 通常是循环变量调整，可以从小往大调整也可以从大往小调整，如果调整多个循环变量则构成逗号表达式。for 循环的流程图如图 3-7 所示。

注意：在标准编译器下，循环变量初始化中的变量仅在该 for 语句中有效。若希望循环变量在 for 语句中及其后都能使用，则把该循环变量定义在 for 语句之前。另外，纯粹 C 语

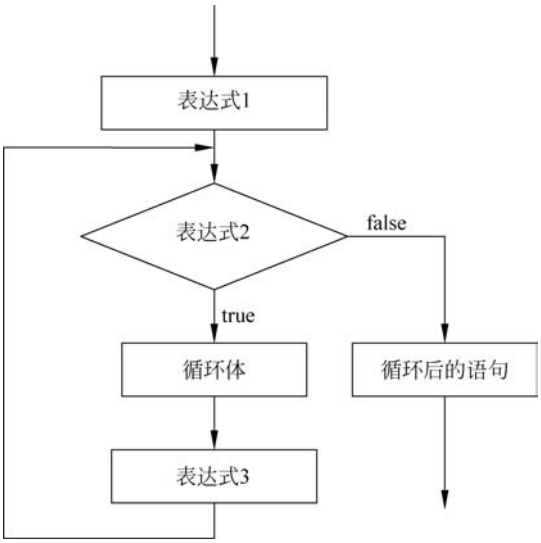

图 3-7　for 循环流程图

言的 for 语句中不能定义循环变量。

最简单的 for 语句形式如下。

for(; ;)

　　循环体

即三个表达式同时省略,此时循环体中一般要有结束循环的语句,例如,跳出循环的 break 语句,否则将成为无限循环(死循环),因为表达式 2 省略时表示循环条件为 true,是永真条件。

2. for 语句的使用

例 3.3.2　数据统计


例 3.3.2

首先输入一个整数 n,然后输入 n 个整数,请统计其中负数、零和正数的个数。

本例可以设置 3 个计数器(统计个数的变量,初值为 0),每输入一个数就判断其是正、负或 0 的哪一种,再把对应计数器加 1。具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int n,d;
    cin>>n;
    int zero=0,positive=0,negative=0; //计数器清 0
    for(int i=0; i<n; i++) {
        cin>>d;
        if (d>0) positive++;
        else if (d<0) negative++;
        else zero++;
    }
    cout<<negative<<" "<<zero<<" "<<positive<<endl;
    return 0;
```

```
}
/*C风格*/
#include<stdio.h>
int main() {
    int d,i,n;
    int zero=0,positive=0,negative=0; /*计数器清0*/
    scanf("%d",&n);
    for(i=0; i<n; i++) {
        scanf("%d",&d);
        if (d>0) positive++;
        else if (d<0) negative++;
        else zero++;
    }
    printf("%d %d %d\n",negative,zero,positive);
    return 0;
}
```

例 3.3.3　亲和数判断


例 3.3.3

古希腊数学家毕达哥拉斯在自然数研究中发现，220 的所有真约数（即不是自身的约数）之和为：1＋2＋4＋5＋10＋11＋20＋22＋44＋55＋110＝284。而 284 的所有真约数为 1、2、4、71、142，加起来恰好为 220。人们称这样的数对为亲和数。也就是说，若两个数中任何一个数都是另一个数的真约数之和，则它们就是亲和数。请判断输入的两个整数是否是亲和数，是则输出“YES”，否则输出“NO”。引号不必输出。

本题根据亲和数的定义把两个整数的真约数之和各自求出，再判断各自是否等于另一个数即可。具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int a,b,i,sumA=0,sumB=0;
    cin>>a>>b;
    for(i=1;i<a;i++) {
        if(a%i==0) sumA+=i;
    }
    for(i=1;i<b;i++) {
        if(b%i==0) sumB+=i;
    }
    if(sumA==b&&sumB==a) cout<<"YES"<<endl;
    else cout<<"NO"<<endl;
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int a,b,i,sumA=0,sumB=0;
    scanf("%d%d",&a,&b);
    for(i=1;i<a;i++) {
        if(a%i==0) sumA+=i;
```

```
    }
    for(i = 1;i < b;i++) {
        if(b % i == 0) sumB += i;
    }
    if(sumA == b&&sumB == a) printf("YES\n");
    else printf("NO\n");
    return 0;
}
```

例 3.3.4　星号三角形


例 3.3.4

输入整数 n,显示星号 * 构成的三角形。例如,$n=6$ 时,显示输出的三角形如下。

```
     *
    ***
   *****
  *******
 *********
***********
```

二维图形的输出,一般在观察图形得到规律后用二重循环实现。对于本题,若输入 n,则输出 n 行,且第 i 行有 $n-i$ 个空格和 $2i-1$ 个 * 。故采用二重循环,第一重循环(外循环)控制行数,第二重循环(内循环)控制每行的空格数和 * 的个数。具体代码如下。

```
//C++风格
#include < iostream >
using namespace std;
int main() {
    int n, j;
    cin >> n;
    for(int i = 1; i < = n; i++) {          //外循环,控制共输出 n 行
        for(j = 1; j < = n - i; j++) {      //内循环,输出每行前面的空格
            cout <<' ';
        }
        for(j = 1; j < = 2 * i - 1; j++) { //内循环,输出每行的 *
            cout <<'*';
        }
        cout << endl;
    }
    return 0;
}
/*C 风格*/
#include < stdio.h >
int main() {
    int i, j, n;
    scanf("%d",&n);
    for(i = 1; i < = n; i++) {              /*外循环,控制共输出 n 行*/
        for(j = 1; j < = n - i; j++) {      /*内循环,输出每行前面的空格*/
            printf(" ");
        }
        for(j = 1; j < = 2 * i - 1; j++) { /*内循环,输出每行的"*" */
```

```
            printf("%c", '*');
        }
        printf("\n");
    }
    return 0;
}
```

多重循环的执行过程：外循环变量每取一个值，内循环完整执行一遍。上面的程序当外循环变量 i 为 1 时，内循环中第一个循环控制输出 $n-1$ 个空格，内循环中第二个循环控制输出 1 个 *；外循环变量 i 为 2 时，内循环中第一个循环控制输出 $n-2$ 个空格，内循环中第二个循环控制输出 3 个 *，……，外循环变量 i 为 n 时，内循环中第一个循环条件不成立，该循环不执行，没有空格输出，内循环中第二个循环控制输出 $2n-1$ 个 *。作为二维图形输出练习，读者可以尝试输入整数 n，再输出类似 $n=5$ 时如例 1.6.2 中所示的漏斗图形。

例 3.3.5　百钱百鸡问题


例 3.3.5

100 元钱买 100 只鸡，公鸡 5 元 1 只，母鸡 3 元 1 只，小鸡 1 元 3 只。问公鸡、母鸡、小鸡三种鸡各多少只(某种鸡可以为 0 只)？

分别设三种鸡的只数为 x、y、z，然后利用总数量、总金额两个条件，可以列出两个方程：

(1) $x+y+z=100$

(2) $5x+3y+\frac{z}{3}=100$

两个方程共有三个未知数，不能直接求出结果。可以使用穷举法(也称枚举法，对待求解问题的所有可能情况逐一检查是否为该问题的解)对 x(0～20)、y(0～33)、z(0～100)的各种取值逐一检查是否满足这两个方程，如果满足，则得出一组结果。

另外，还要注意到一个隐含的条件，就是小鸡的数量是 3 的倍数，即 $z\%3==0$。

根据以上分析，可以用三重循环求解本题，具体实现代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int x,y,z;
    for (x=0; x<=20; x++) {
        for (y=0; y<=33; y++) {
            for (z=0; z<=100; z++) {
                if ( x+y+z==100 && 5*x+3*y+z/3==100 && z%3==0 ) {
                    cout<<x<<" "<<y<<" "<<z<<endl;
                }
            }
        }
    }
    return 0;
}
/*C风格*/
#include<stdio.h>
```

```
int main() {
    int x,y,z;
    for (x = 0; x < = 20; x++) {
        for (y = 0; y < = 33; y++) {
            for (z = 0; z < = 100; z++) {
                if ( x + y + z == 100 && 5 * x + 3 * y + z/3 == 100 && z % 3 == 0 ) {
                    printf(" % d % d % d\n",x,y,z);
                }
            }
        }
    }
    return 0;
}
```

实际上,当公鸡和母鸡的只数分别为 x、y 时,可以直接得到小鸡的只数 $z=100-x-y$,因此小鸡只数不需要从 0 到 100 去穷举,如此只需要用二重循环就能求解本题,具体代码如下。

```
//C++风格
#include < iostream >
using namespace std;
int main() {
    int x,y,z;
    for (x = 0; x < = 20; x++) {
        for (y = 0; y < = 33; y++) {
            z = 100 - x - y;
            if (5 * x + 3 * y + z/3 == 100 && z % 3 == 0) {
                cout << x <<" "<< y <<" "<< z << endl;
            }
        }
    }
    return 0;
}
/ * C 风格 * /
#include < stdio.h >
int main() {
    int x,y,z;
    for (x = 0; x < = 20; x++) {
        for (y = 0; y < = 33; y++) {
            z = 100 - x - y;
            if (5 * x + 3 * y + z/3 == 100 && z % 3 == 0) {
                printf(" % d % d % d\n",x,y,z);
            }
        }
    }
    return 0;
}
```

3.3.3 while 语句及其使用

1. while 语句基本格式

while 循环语句的格式如下。

while(循环条件)

　　循环体

while 语句在满足循环条件时反复执行循环体，否则结束循环，流程图如图 3-8 所示。

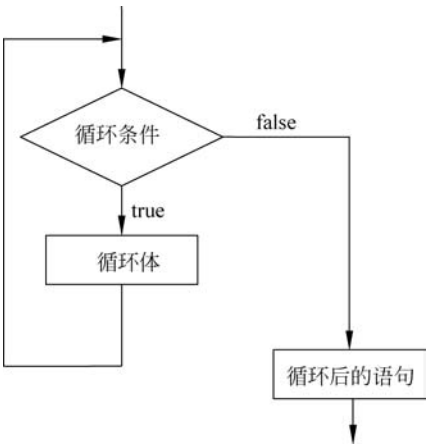

图 3-8　while 循环流程图

循环体中一般需要有改变循环变量使循环趋于结束或跳出循环的语句，以避免死循环。若循环体有多条语句，则必须以{}括起来构成复合语句。

在线做题时，对于 T 组测试的题目，可用如下代码控制。

```
int T;
cin >> T;                              //C 风格: scanf("%d",&T);
while(T--) {
    //一组测试的代码
}
```

其中，while(T--) 当 $T \geqslant 1$ 时，非 0 值转换为逻辑值 true，执行循环体；当 $T == 1$ 时，先取 T 的值 1 转换为 true，最后执行一次循环，之后 T 变为 0，再取 T 的值 0，转换为 false，循环结束。

2. while 语句的使用

例 3.3.6　奇数的和


例 3.3.6

输入一个整数 n，用 while 循环求[1,n]中所有奇数的和。

可从 1 开始，当奇数在 n 的范围内就逐个累加，超出 n 则不再累加，循环结束。具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
```

```
    int n,sum;
    cin >> n;
    sum = 0;                          //累加单元清 0
    int i = 1;                        //循环变量赋初值
    while(i <= n) {                   //满足循环条件则执行循环体
        sum += i;
        i += 2;                       //改变循环变量使循环趋于结束
    }
    cout << sum << endl;
    return 0;
}
/* C 风格 */
#include <stdio.h>
int main() {
    int i,n,sum;
    scanf("%d",&n);
    sum = 0;                          /* 累加单元清 0 */
    i = 1;                            /* 循环变量赋初值 */
    while(i <= n) {                   /* 满足循环条件则执行循环体 */
        sum += i;
        i += 2;                       /* 改变循环变量使循环趋于结束 */
    }
    printf("%d\n",sum);
    return 0;
}
```

这个例子使用了 while 语句来实现循环,当满足 $i<=n$ 条件时,反复执行循环体;当 i 不断增加使得 $i<=n$ 条件不成立,即 $i>n$ 时,结束循环。

例 3.3.7　数位之和

例 3.3.7

输入一个正整数,求其各个数位上的数字之和。例如,输入 12345,输出 15。

本题需要数位分离,即把一个整数的个位、十位、百位等数位分离出来。可以不断地取得个位相加,再把个位去掉,直到该数等于 0 为止。具体代码如下。

```
//C++风格
#include <iostream>
using namespace std;
int main() {
    int n, t, sum;
    cin >> n;
    sum = 0;
    while(n > 0) {                    //当 n > 0 进行循环
        t = n % 10;                   //取得个位
        n = n/10;                     //去掉个位
        sum = sum + t;
    }
    cout << sum << endl;
    return 0;
}
/* C 风格 */
#include <stdio.h>
```

```
int main() {
    int n, t, sum;
    scanf("%d",&n);
    sum = 0;
    while(n > 0) {                    /* 当 n > 0 进行循环 */
        t = n % 10;                   /* 取得个位 */
        n = n/10;                     /* 去掉个位 */
        sum = sum + t;
    }
    printf("%d\n",sum);
    return 0;
}
```

例 3.3.8　数列求和


例 3.3.8

求下面数列的所有大于等于 0.000 001 的数据项之和，显示输出计算的结果(四舍五入保留 6 位小数)。

$\frac{1}{2}$、$\frac{3}{4}$、$\frac{5}{8}$、$\frac{7}{16}$、$\frac{9}{32}$、…

观察上面的数列，发现有什么规律？

规律 1：分子为从 1 开始的奇数、分母为 2 的幂次；即第 i 项的通项公式为$(2i-1)/2^i$。

规律 2：第一项分子为 1，分母为 2，后一项与前一项相比，分子值增加 2，分母值增加 1 倍。

这里采用按规律 2，逐项累加。另外，0.000 001 可以表示为 1e-6。具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    double sum = 0,t;
    int a = 1,b = 2;
    t = (double)a/b;                  //强制转换,避免类似 1/2 结果为 0
    while(t >= 1e-6) {
        sum += t;
        a += 2;
        b *= 2;
        t = a * 1.0/b;                //a * 1.0,向 double 类型自动转换,避免类似 3/4 结果为 0
    }
    printf("%.6lf\n", sum);           //结果保留 6 位小数用 C 语言写法更方便,自动四舍五入
    return 0;
}
/* C 风格 */
#include<stdio.h>
int main() {
    double sum = 0,t;
    int a = 1,b = 2;
    t = (double)a/b;                  /* 强制转换,避免类似 1/2 结果为 0 */
    while(t >= 1e-6) {
        sum += t;
        a += 2;
```

```
        b *= 2;
        t = a * 1.0/b;                    /* a * 1.0,向 double 类型自动转换,避免类似 3/4 结果为 0 */
    }
    printf("%.6lf\n", sum);               /* 结果保留 6 位小数,自动四舍五入 */
    return 0;
}
```

例 3.3.9　计算 sin$x$ 的近似值


例 3.3.9

按下面的计算公式,设计一个程序,输入弧度 x,通过累加所有绝对值大于等于 0.000 001 的项来计算 sin$x$ 的近似值,显示输出计算的结果。结果保留 6 位小数。

$$\sin x=\frac{x}{1}-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\cdots$$

观察计算公式,发现有什么规律?

规律 1:第 n 项的分子为 x 的 $2n-1$ 次幂(与前一项相差 $-x^2$)、分母为$(2n-1)!$。

规律 2:第一项为 x,第 n 项的通项为 $x^{2n-1}/(2n-1)!$,后一项与前一项相比,相差一个因子 $-x^2/((2n-1)(2n-2))$。

根据规律 2,逐项累加。0.000 001 可以表示为 1e-6。代码如下。

```
//C++风格
#include<iostream>
using namespace std;
#include<cmath>
int main() {
    double sum=0,t,x;
    cin>>x;
    t=x;                              //t 表示某一项,首项为 x
    int n=1;
    while(fabs(t)>=1e-6) {            //fabs(t)求实数 t 的绝对值
        sum+=t;
        n++;
        t*=-x*x/(2*n-1)/(2*n-2);
    }
    printf("%.6lf\n",sum);            //结果保留 6 位小数用 C 语言写法更方便
    return 0;
}
/*C 风格*/
#include<math.h>
#include<stdio.h>
int main() {
    double sum=0,t,x;
    int n=1;
    scanf("%lf",&x);
    t=x;                              /*t 表示某一项,首项为 x*/
    while(fabs(t)>=1e-6) {            /*fabs(t)求实数 t 的绝对值*/
        sum+=t;
        n++;
        t*=-x*x/(2*n-1)/(2*n-2);
    }
    printf("%.6lf\n",sum);            /*结果保留 6 位小数*/
```

```
    return 0;
}
```

其中，用到数学函数 fabs(double)求实数绝对值，故须包含头文件 cmath 或 math.h。也可以采用规律 1 实现，具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
#include<cmath>
int main() {
    double sum=0,t,x,r=1;
    cin>>x;
    t=x;
    int n=2;
    while(fabs(t/r)>=1e-6) {
        sum+=t/r;
        t*=-x*x;
        r=1;
        for(int i=1;i<=2*n-1;i++) r*=i;
        n++;
    }
    printf("%.6lf\n",sum);
    return 0;
}
/*C风格*/
#include<math.h>
#include<stdio.h>
int main() {
    double sum=0,t,x,r=1;
    int i,n=2;
    scanf("%lf",&x);
    t=x;
    while(fabs(t/r)>=1e-6) {
        sum+=t/r,t*=-x*x,r=1;
        for(i=1;i<=2*n-1;i++) r*=i;
        n++;
    }
    printf("%.6lf\n",sum);
    return 0;
}
```

此代码在 while 循环中嵌套了 for 循环求阶乘，是二重循环的实现方式。注意，由于阶乘的速度增长很快，为避免溢出，保存阶乘结果的变量 r 定义为 double 类型。另外，本题中阶乘是除数，也可以考虑不计算阶乘而在 for 循环中用“t/=i;”求得各项，具体代码留给读者自行实现。

3.3.4 do…while 语句及其使用

1. do…while 语句基本格式

do…while 循环语句格式如下。

do

　　循环体

while(循环条件);

do…while 循环体中需要有改变循环变量使循环趋于结束或跳出循环的语句,以避免死循环。若循环体有多条语句,则必须以{}括起来构成复合语句。do…while 语句先做一次循环体,再判断循环条件,在满足循环条件时反复执行循环体,否则结束循环,流程图如图 3-9 所示。

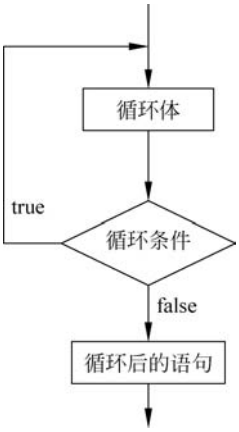

图 3-9　do…while 循环流程图

2. do…while 语句的使用

思考:如何保证输入的数据一定在[1,10]范围内?

显然,当不在[1,10]范围内时需要重新输入,用 do…while 语句可以方便实现。具体代码如下。

```
do {
    cin >> n;                          //C 风格: scanf("%d",&n);
} while(n < 1 || n > 10);              //当满足此条件时执行循环体
```

例 3.3.10

例 3.3.10　偶数之和

输入 n,求[1,n]范围内的所有偶数之和。

本例的循环变量 i 可以从 0 开始,每次增 2,逐个把 i 加到求和单元中,具体代码如下。

```
//C++风格
#include < iostream >
using namespace std;
int main() {
    int n,sum = 0;
    cin >> n;
    int i = 0;                         //循环变量赋初值
    do {                               //循环体
        sum += i;
        i += 2;                        //改变循环变量使循环趋于结束
    }while(i < = n);                   //满足循环条件则执行循环体
    cout << sum << endl;
    return 0;
}
/* C 风格 */
#include < stdio.h >
int main() {
    int i,n,sum = 0;
    scanf("%d",&n);
    i = 0;                             /* 循环变量赋初值 */
    do {                               /* 循环体 */
        sum += i;
        i += 2;                        /* 改变循环变量使循环趋于结束 */
    }while(i < = n);                   /* 满足循环条件则执行循环体 */
    printf("%d\n",sum);
```

```
    return 0;
}
```

do…while 语句与 while 语句的区别在于：若一开始循环条件就不成立时，do…while 语句执行 1 次循环体，while 语句执行 0 次循环体。

3.3.5 continue、break 语句及其使用

在 C/C++语言中，continue 语句用于提前结束本轮循环的执行，即本轮循环不执行 continue 之后的语句，而是继续进行下一次循环的准备与条件判断；break 语句用来跳出其所在的循环，接着执行该循环之后的语句。

对于例 3.3.10，循环变量也可以从 1 开始，依次递增 1，但在循环体中增加判断语句，跳过奇数。具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int n,sum=0;
    cin>>n;
    for(int i=1;i<=n;i++) {
        if (i%2==1)                    //i为奇数
            continue;                  //跳过本次循环continue之后的语句
        sum+=i;
    }
    cout<<sum<<endl;
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int i,n,sum=0;
    scanf("%d",&n);
    for(i=1;i<=n;i++) {
        if (i%2==1)                    /*i为奇数*/
            continue;                  /*跳过本次循环continue之后的语句*/
        sum+=i;
    }
    printf("%d\n",sum);
    return 0;
}
```


例 3.3.11

例 3.3.11　素数判定

输入一个正整数 $m(m>1)$，判断该数是否为素数。如果 m 为素数则输出“yes”；反之输出“no”。

根据素数的定义，除了 1 和本身之外没有其他因子的自然数是素数。因此可以从 2 到该数的前一个数去看有没有因子，只要这个范围内有任意一个因子就可以确定该数不是素数，不必再看是否有其他因子。根据这个思路，实现代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int m;
    cin>>m;
    bool isPrime=true;              //标记变量,一开始假设是素数
    for (int i=2;i<m;i++) {
        if (m%i==0) {               //有其他因子
            isPrime=false;          //更改标记,表示不是素数
            break;                  //跳出其所在的循环
        }
    }
    if(isPrime==true)               //标记变量保留原值,说明没有其他因子
        cout<<"yes"<<endl;
    else
        cout<<"no"<<endl;
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int i,m;
    int isPrime=1;                  /*标记变量,一开始假设是素数*/
    scanf("%d",&m);
    for (i=2;i<m;i++) {
        if (m%i==0) {               /*有其他因子*/
            isPrime=0;              /*更改标记,表示不是素数*/
            break;                  /*跳出其所在的循环*/
        }
    }
    if(isPrime==1)                  /*标记变量保留原值,说明没有其他因子*/
        printf("yes\n");
    else
        printf("no\n");
    return 0;
}
```

上面的代码中,for循环有两个出口,一个是循环条件不成立,即!($i<m$)(实际上 $i==m$ 时结束循环);另一个是break语句,执行该语句时程序流程直接从循环中跳出(此时循环条件 $i<m$ 依然成立)。

注意,一个break语句只能跳出一个循环。如果要用break跳出二重循环,可以在内、外循环中都使用break,也可以设置一个标记变量(初值设为true用&&运算符)添加到两个循环条件中,当需要结束循环时把标记变量改为false,从而构成“短路”表达式。

注意,如果本题没有说明 $m>1$,则需要对1做特殊处理,因为根据上面的代码,输入1将得到“yes”,而实际上1不是素数。

对于2 147 483 647这个素数而言,上面判断是否有因子的循环需要执行2 147 483 646次。显然效率很低,能否改进代码,提高效率呢?在效率提高方面,因为 m 除了本身之外的

最大因子是$m/2$,循环条件可以改为$i<=m/2$,这样对于一个素数的判断循环次数少了约一半,效率得到提高。实际上,效率可以进一步提高,因为若m是合数(不是素数),则可以分解为两个因子(设为a,b,且$a\leqslant b$)之积,即

$m=a\times b$

则 $a^2\leqslant a\times b\leqslant b^2$

即 $a^2\leqslant m\leqslant b^2$

即 $a\leqslant \sqrt{m}\leqslant b$

可见,a这个因子不大于$\sqrt{m}$,因此可以判断到$\sqrt{m}$,因为$\sqrt{m}$之前没有因子的话,$\sqrt{m}$之后也不会有因子。具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
#include<cmath>
int main() {
    int m;
    cin>>m;
    bool isPrime=true;
    int sqm=(int)sqrt(m*1.0);
    for (int i=2;i<=sqm;i++) {
        if (m%i==0) {
            isPrime=false;
            break;
        }
    }
    if (isPrime==true)
        cout<<"yes"<<endl;
    else
        cout<<"no"<<endl;
    return 0;
}
/*C风格*/
#include<math.h>
#include<stdio.h>
int main() {
    int i,m,sqm;
    int isPrime=1;
    scanf("%d",&m);
    sqm=(int)sqrt(m*1.0);
    for (i=2;i<=sqm;i++) {
        if (m%i==0) {
            isPrime=0;
            break;
        }
    }
    if (isPrime==1)
        printf("yes\n");
    else
```

```
        printf("no\n");
    return 0;
}
```


对于 2 147 483 647 而言,上面判断是否有因子的循环需要执行 46 339 次,效率远高于前一种方法。另外,调用 sqrt 函数(头文件 cmath)时,建议实际参数用 double 类型,以避免在线提交时出现编译错误。


例 3.3.12

例 3.3.12　最大公约数

求两个正整数 m,n 的最大公约数(Greatest Common Divisor,GCD)。

本题的一种解法是直接利用数学性质求解,即从 m,n 两个数中的小者到 1 逐个尝试(穷举法),找第一个能同时整除 m,n 的因子(找到则直接用 break 跳出循环),实现代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int m,n,gcd;
    cin>>m>>n;
    int k=m;
    if (n<k) k=n;
    for (int i=k;i>=1;i--) {
        if (m%i==0&&n%i==0) {
            gcd=i;
            break;
        }
    }
    cout<<gcd<<endl;
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int i,k,m,n,gcd;
    scanf("%d%d",&m,&n);
    k=m;
    if (n<k) k=n;
    for (i=k;i>=1;i--) {
        if (m%i==0&&n%i==0) {
            gcd=i;
            break;
        }
    }
    printf("%d\n",gcd);
    return 0;
}
```

另一种方法是利用欧几里得(Euclid)算法。欧几里得算法又称辗转相除法,用于计算两个整数 m,n 的最大公约数。其计算原理依赖于下面的定理:$\gcd(m,n)=\gcd(n,m\%n)$。

例如，求 $m=70$ 和 $n=16$ 的最大公约数，计算过程如表 3-1 所示。

表 3-1　求最大公约数计算过程

轮　　次	m	n	$t=m\%n$
1	70	16	6
2	16	6	4
3	6	4	2
4	4	2	0
5	2	0	

计算时，m、n 不断用新值代替旧值(迭代法)，直到 n 为 0 时，m 为最大公约数。因此，实现代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int m,n,gcd;
    cin>>m>>n;
    while(n>0) {
        int t=m%n;
        m=n;
        n=t;
    }
    gcd=m;
    cout<<gcd<<endl;
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int m,n,t,gcd;
    scanf("%d%d",&m,&n);
    while(n>0) {
        t=m%n;
        m=n;
        n=t;
    }
    gcd=m;
    printf("%d\n",gcd);
    return 0;
}
```

思考：怎么求 m，n 的最小公倍数(Least Common Multiple，LCM)lcm?

一种思路是从 m、n 中的大者出发，逐个检查该数的 1 倍、2 倍、…是否是另一个数的倍数。另一种思路是基于求得的最大公约数 gcd。设原来的 m、n 已经分别保存在 a、b 中，则最小公倍数 lcm$=a*b/$gcd。考虑到 $a*b$ 可能产生溢出，可以先除后乘(这也是程序设计竞赛中要注意的一个小技巧)，即 lcm$=a/$gcd$*b$，因为最大公约数 gcd 肯定能整除 a、b 中

的任何一个数。

思考：如何求 n 个正整数的最小公倍数？

可以在求解两个数的最小公倍数的基础上进行，设最小公倍数 lcm 的初值为 1，在执行 n 次的循环中每输入一个整数 t，就求 lcm 与 t 的最小公倍数并保存在 lcm 中，则最终的 lcm 为结果。具体代码留给读者自行实现。

3.4 OJ 题目求解


例 3.4.1

例 3.4.1 求绝对值(HLOJ 1948)

Problem Description

输入一个实数，求其绝对值。

Input

测试数据有多组，处理到文件尾。每组测试数据输入一个实数。

Output

对于每组测试数据，在一行上输出该实数的绝对值，结果保留两位小数。

Sample Input	Sample Output
-456.01	456.01

取绝对值时，对于非负数，可以直接输出，而对于负数，可用负号－把该数转换为正数再输出。对于处理到文件尾，可用循环 while(cin >> d)的方式。具体代码如下。

```
//C++风格
#include <iostream>
using namespace std;
int main() {
    double d;
    while(cin>>d) {                    //控制处理到文件尾(此时 cin 得到 NULL,循环结束)
        if (d<0) d= -d;
        printf("%.2lf\n",d);
    }
    return 0;
}
```

当然，求实数的绝对值也可以直接调用数学库函数 fabs，例如，fabs(－1.23)的结果等于 1.23。注意 fabs 函数的参数类型为 double，头文件为 cmath 或 math.h。

```
/*C 风格*/
#include<stdio.h>
#include<math.h>
int main() {
    double d;
    while(scanf("%lf",&d)!= EOF)   /*控制处理到文件尾(此时 scanf 得到 EOF,循环结束)*/
        printf("%.2lf\n",fabs(d));
    }
    return 0;
}
```

例 3.4.2　求和(HLOJ 1001)


例 3.4.2

Problem Description

输入一个数 n,求出由 1 加至 n 的总和。测试数据保证结果在 int 范围内。

Input

测试数据有多组,处理到文件尾。每组测试输入一个正整数 n。

Output

对于每组测试,输出 1+2+3…+n 的结果。

Sample Input	Sample Output
4	10

在前面的章节中,曾使用逐个累加的方法求 1+2+3…+n。但在程序设计竞赛中或在线做题时,可能经常出现 n 较大且数据量很大的情况。这种情况下,逐个累加将得到超时反馈。为避免超时,需要考虑更高效的方法。因为 1,2,…,n 等各项构成等差数列,可以直接使用求和公式 $n(n+1)/2$。需要注意的是,比赛时测试数据一般都比较完善,若 n 较大,例如 50 000,则 50 000×50 001=2 500 050 000 超出 int 能表示的最大值 2 147 483 647,即产生溢出,解决办法:一是使用更长的类型,例如 long long int;二是判断 n 的奇偶性,先用偶数除以 2 再乘另一个数。具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int n;
    while(cin>>n) {                    //控制处理到文件尾(此时 cin 得到 NULL,循环结束)
        int sum;
        if(n%2==0) sum=n/2*(n+1);
        else sum=(n+1)/2*n;
        cout<<sum<<endl;
    }
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int n,sum;
    while(scanf("%d",&n)!=EOF) {  /*scanf 得不到数据时返回 EOF(-1)*/
        if(n%2==0) sum=n/2*(n+1);
        else sum=(n+1)/2*n;
        printf("%d\n",sum);
    }
    return 0;
}
```

例 3.4.3　求 $n!$（HLOJ 1909）

Problem Description

$$n!=\begin{cases}1, & n=0,1\\ 1\times 2\times\cdots\times n, & n\geqslant 2\end{cases}$$

例 3.4.3

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。每组测试数据输入一个正整数 n($n\leqslant 12$)。

Output

对于每组测试,输出整数 n 的阶乘。

Sample Input	Sample Output
1 5	120

实现思想：外循环控制测试组数,内循环从 1 连乘到 n。连乘单元初值置为 1。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int T;
    cin>>T;
    for(int i=0;i<T;i++) {
        int n;
        cin>>n;
        int res=1;
        for(int j=2;j<=n;j++) res*=j;
        cout<<res<<endl;
    }
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int T,i,j,n,res;
    scanf("%d",&T);
    for(i=0;i<T;i++) {
        scanf("%d",&n);
        res=1;
        for(j=2;j<=n;j++) res*=j;
        printf("%d\n",res);
    }
    return 0;
}
```

需要注意的是阶乘的增长速度很快,13! 已经超出 int 范围,若要求稍大的 n 的阶乘,

可以用 long long int 类型，对于更大的 n，则需要考虑使用数组或字符串处理。

例 3.4.4　闰年判断（HLOJ 1910）


例 3.4.4

Problem Description

闰年是能被 4 整除但不能被 100 整除或者能被 400 整除的年份。请判断给定年份是否闰年。

Input

首先输入一个正整数 T，表示测试数据的组数，然后是 T 组测试数据。每组测试数据输入一个年份 y。

Output

对于每组测试，若 y 是闰年输出“YES”，否则输出“NO”。引号不必输出。

Sample Input	Sample Output
2 2008 1900	YES NO

实现思想：在循环中逐个判断闰年条件是否成立，具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int n;
    cin>>n;
    for(int i=0; i<n; i++) {
        int y;
        cin>>y;
        if (y%4==0&&y%100!=0||y%400==0) cout<<"YES"<<endl;
        else cout<<"NO"<<endl;
    }
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int i,n,y;
    scanf("%d",&n);
    for(i=0; i<n; i++) {
        scanf("%d",&y);
        if (y%4==0&&y%100!=0||y%400==0) printf("YES\n");
        else printf("NO\n");
    }
    return 0;
}
```

例 3.4.5

例 3.4.5　统计数字(HLOJ 1950)

Problem Description

输入一个字符串,统计其中数字字符的个数。

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。每组测试输入一个仅由字母和数字组成的字符串(长度不超过 80)。

Output

对于每组测试,在一行上输出该字符串中数字字符的个数。

Sample Input	Sample Output
2 ac520ac520 a1c2m3sdf	6 3

实现思想:外循环控制测试组数,内循环中输入字符串后逐个扫描字符串的每个字符,若是数字字符,则计数器增 1。具体代码如下。

```
//C++风格
#include <iostream>
#include <string>
using namespace std;
int main() {
    int n;
    cin >> n;
    for(int j = 0; j < n; j++) {
        string s;
        cin >> s;
        int cnt = 0;
        for(int i = 0;i < s.size();i++) {
            if (s[i]>= '0'&&s[i]<= '9') cnt++;
        }
        cout << cnt << endl;
    }
    return 0;
}
```

本题使用 STL 之 string。处理字符串时,使用 string 比使用 char 类型的数组要简单、方便。但对于大量字符串的输入的比赛题目,用 cin 输入到 string 类型变量可能导致超时,此时可以使用字符数组作为缓冲避免超时。具体代码如下。

```
//C++风格
#include <iostream>
#include <string>
using namespace std;
int main() {
    int n;
    scanf("%d",&n);
```

```
    for(int j = 0; j < n; j++) {
        string s;
        char ts[81];                          //定义一个字符数组,长度为 81
        scanf("%s",ts);                       //缓冲输入,若输入的字符串包含空格,则用 gets(ts);
        s = ts;                               //可以把 char 数组赋值给 string 类型变量
        int cnt = 0;
        for(int i = 0;i < s.size();i++) {
            if (s[i] >= '0'&&s[i] <= '9') cnt++;
        }
        cout << cnt << endl;
    }
    return 0;
}
```

上面的代码使用了字符数组,用 scanf 输入时格式字符是 s。下面再给出 C 风格代码,也使用字符数组,读者可以先行了解相关知识,或暂时跳过此内容,待学完字符数组的知识后再回头掌握。

本题的 C 风格代码具体如下。

```
/*C 风格*/
#include <stdio.h>
#include <string.h>                           /*char 数组处理函数的头文件*/
int main() {
    int i,j,n,cnt;
    char s[81];                               /*定义一个字符数组 s,长度为 81*/
    scanf("%d",&n);
    for(j = 0; j < n; j++) {
        scanf("%s",s);                        /*输入,格式字符为 s*/
        cnt = 0;
        for(i = 0;i < strlen(s);i++) {        /*strlen(s)求 s 的长度*/
            if (s[i] >= '0'&&s[i] <= '9') cnt++;
        }
        printf("%d\n",cnt);
    }
    return 0;
}
```

例 3.4.6　查找最大元素(HDOJ 2025)

例 3.4.6

Problem Description

对于输入的每个字符串(由大小写英文字母构成),查找其中的最大字母,在该字母后面插入字符串“(max)”。

Input

测试数据有多组,处理到文件尾。每组测试输入一个字符串(仅由大小写字母构成,且长度不超过 100)。

Output

对于每组测试,输出插入字符串“(max)”后的结果,如果存在多个最大的字母,就在每一个最大字母后面都插入“(max)”。

Sample Input	Sample Output
abcgfcba zzzzz	abcg(max)fcba z(max)z(max)z(max)z(max)z(max)

OJ 通常按测试数据比对来判断程序对错,故本题可以采用直接输出的方法。本题的实现思想:外循环控制到文件尾,内循环先找到最大字符,然后逐个输出字符,若该字符是最大字符,则再输出字符串"(max)"。

```
//C++风格
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s;
    while(cin >> s) {                  //控制处理到文件尾(此时 cin 得到 NULL,循环结束)
        int i,n = s.size();
        char maxCh = s[0];             //假设第一个字符最大
        for(i = 1; i < n; i++) {       //找最大字符
            if (maxCh < s[i]) maxCh = s[i];
        }
        for(i = 0;i < n;i++) {         //逐个输出字符,若是最大字符,则再输出(max)
            cout << s[i];
            if(maxCh == s[i]) cout <<"(max)";
        }
        cout << endl;
    }
    return 0;
}
```

相应的 C 风格代码如下。

```
/* C 风格 */
#include <stdio.h>
#include <string.h>
int main() {
    char s[101],maxCh;
    int i,n;
    while(scanf("%s",s)!= EOF) {   /* 控制处理到文件尾 */
        n = strlen(s);
        maxCh = s[0];              /* 假设第一个字符最大 */
        for(i = 1;i < n;i++) {     /* 找最大字符 */
            if(maxCh < s[i]) maxCh = s[i];
        }
        for(i = 0;i < n;i++) {     /* 逐个输出字符,若是最大字符,则再输出(max) */
            printf("%c",s[i]);
            if(maxCh == s[i]) printf("(max)");
        }
        printf("\n");
    }
    return 0;
}
```

若本题确实要把(max)插入到字符串中,则可以最大字符位置为界分别取得 string 类型字符串的左子串(含最大字符)和右子串,然后用连接符+把"(max)"连接到左、右子串之间。另外需要注意插入(max)之后的串长度发生了变化,下次比较的起始位置也要偏移"(max)"的长度,即 5。具体代码如下。

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s;
    while(cin >> s) {                       //控制处理到文件尾
        int i,n = s.size();
        char maxCh = s[0];                  //假设第一个字符最大
        for(i = 1;i < n;i++) {              //找最大字符
            if(maxCh < s[i]) maxCh = s[i];
        }
        for(i = 0;i < s.size();i++) {       //因为 s 不断变长,串长应为 s.size()
            if(maxCh == s[i]) {             //若是最大字符,则在其后插入(max)
                s = s.substr(0,i + 1) + "(max)" + s.substr(i + 1);
                i += 5;                     //跳过刚插入的(max)
            }
        }
        cout << s << endl;
    }
    return 0;
}
```

其中,s.substr(0,i+1)取 s 从 0 开始长度为 i+1 的子串,s.substr(i+1)取 s 从 i+1 开始的剩余子串。

使用 char 数组时,插入(max)需要频繁移动数组元素,处理起来没有 string 类型变量方便,读者可以在掌握了 char 数组的相关知识后再自行尝试。

例 3.4.7　单词首字母大写(HDOJ 2026)


例 3.4.7

Problem Description

输入一个英文句子,要求将每个单词的首字母改成大写字母。

Input

测试数据有多组,处理到文件尾。每组测试输入一行,包含一个长度不超过 100 的英文句子(仅包含大小写英文字母和空格),单词之间以一个空格分隔。

Output

对于每组测试,输出按照要求改写后的英文句子。

Sample Input	Sample Output
i like acm	I Like Acm
i want to get accepted	I Want To Get Accepted

实现思想:根据空格取单词,每次把空格之前的子串作为一个单词从句子中截取下来,存放到临时串 t 中,使用 cnt 统计单词个数。方便处理起见,在输入串的最后添加一个空格。具体代码如下。

```
//C++风格
#include<iostream>
#include<string>
using namespace std;
int main() {
    string s;
    while(getline(cin,s)) {
        s=s+" ";                    //按空格取单词,方便处理起见,在最后加一个空格
        int i,n=s.size(),cnt=0;
        string t="";                //临时保存一个单词
        for(i=0;i<n;i++) {
            if (s[i]==' ') {        //遇到空格,表示其前面的一个单词结束
                if (t[0]>='a'&&t[0]<='z') t[0]+='A'-'a';
                cnt++;              //单词数增1
                if(cnt>1) cout<<" ";
                cout<<t;
                t="";
                continue;
            }
            t=t+s[i];               //把字符连接到临时串t中
        }
        cout<<endl;
    }
    return 0;
}
```

上面的代码能逐个单词取出放到临时字符串 t 中,可以对单词做更多的处理,例如,单词逆置、求最长单词等。实际上,本题还有其他一些求解方法,例如,用字符串的成员函数 find 找空格,再用成员函数 substr 取子串来完成,具体留给读者自行实现。本程序对于单词之间有多个空格的情况也能正确处理,请读者自行分析。

由于 C 语言使用字符数组处理字符串时不如 string 类型变量方便,推荐使用 C++的 string 处理字符串。为方便读者体会字符数组的使用,给出本题的 C 风格代码如下。

```
#include<stdio.h>
#include<string.h>
int main() {
    char s[101];                    /*定义字符数组*/
    int i,n;
    while(gets(s)) {                /*有空格,故用gets函数,遇文件尾返回NULL*/
        n=strlen(s);                /*求串长*/
        if (s[0]>='a'&&s[0]<='z') /*处理首字符*/
            putchar(s[0]+('A'-'a'));
        for(i=1;i<n;i++) {
            if (s[i-1]==' ')        /*遇到空格,表示一个单词开始,把其后字母大写*/
                if (s[i]>='a'&&s[i]<='z') putchar(s[i]+('A'-'a'));
            else putchar(s[i]);
        }
        putchar('\n');
    }
```

```
    return 0;
}
```

其中，gets 函数用于输入包含空格的字符串，得不到数据时返回空指针 NULL。由于使用字符串操作函数，需包含头文件 string.h。这里使用 putchar 函数输出一个字符。需要注意的是，此代码与前面 C++截取单词的思路有所不同。本程序把每个空格之后的字符作为一个新单词的开始，若前一个字符是空格则判断其后字符是否大写，否则改为大写。

例 3.4.8　列出完数(HLOJ 1911)


例 3.4.8

Problem Description

输入一个整数 n，要求输出[1,n]范围内的所有完数。完数是一个正整数，该数恰好等于其所有不同真因子之和。例如，6、28 是完数，因为 $6=1+2+3$，$28=1+2+4+7+14$；而 24 不是完数，因为 $24\neq1+2+3+4+6+8+12=36$。

Input

测试数据有多组，处理到文件尾。每组测试数据输入一个整数 n($1\leqslant n\leqslant10\ 000$)。

Output

对于每组测试，首先输出 n 和一个冒号":"；然后输出所有不大于 n 的完数(每个数据之前留一个空格)；若[1,n]范围内不存在完数，则输出"NULL"。引号不必输出。具体输出格式参考 Sample Output。

Sample Input	Sample Output
100	100: 6 28
5000	5000: 6 28 496
5	5: NULL

对于本题，一个很自然的思路是从 1 到 n 逐个检查数据，判断一个数的所有真因子之和是否等于该数，是则输出。具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int n;
    while(cin>>n) {
        cout<<n<<":";
        int cnt=0;                  //计数器清 0
        for(int k=1;k<=n;k++) {
            int sum=0;              //存放 k 的真因子之和
            for(int i=1;i<=k/2;i++) {
                if(k%i==0) {
                    sum=sum+i;
                }
            }
            if(sum==k) {            //k 是完数
                cout<<" "<<k;       //输出完数
                cnt++;              //计数器加 1
            }
```

```
        }
        if(cnt == 0)                    //若计数器等于初值 0,则表示没有完数
            cout <<" NULL";
        cout << endl;
    }
    return 0;
}
/ * C 风格 * /
# include < stdio.h >
int main() {
    int i,k,n,cnt,sum;
    while(scanf(" % d",&n)!= EOF) {
        printf(" % d:",n);
        cnt = 0;                        / * 计数器清 0 * /
        for(k = 1;k < = n;k++) {
            sum = 0;                    / * 存放 k 的真因子之和 * /
            for(i = 1;i < = k/2;i++) {
                if(k % i == 0) {
                    sum = sum + i;
                }
            }
            if(sum == k) {              / * k 是完数 * /
                printf(" % d",k);
                cnt++;                  / * 计数器加 1 * /
            }
        }
        if(cnt == 0)                    / * 若计数器等于初值 0,则表示没有完数 * /
            printf(" NULL");
        printf("\n");
    }
    return 0;
}
```

运行结果:

```
10000↵
10000: 6 28 496 8128
5↙
5: NULL
```

上面的代码用计数器控制没有完数时输出 NULL,这是一种常用的方法,希望读者熟练掌握。当然,也可以用标记变量(设为 flag)的方法,flag 初值设为 false,若有完数时则把 flag 的值改为 true,最后检查 flag 的值,若其值为 false 则输出 NULL。另外,若已知 6 是最小的完数,则可以在 n 小于 6 时直接输出 NULL。

上面的代码在本地(自己使用的计算机)运行无误。但细心的读者会注意到在输入 10 000 时,程序运行后需要稍作等待才能得到结果,即程序运行耗时较多。若在线题目的测试数据接近 10 000 的较多,则提交代码后将得到超时反馈。此时,可以用空间换时间的方法避免超时,即把完数先保存起来,输入数据后再从保存的结果中把答案取出来。从上面代码的运行结果可见,10 000 以内的完数仅有 4 个,即 6、28、496、8128,如此相当于结果已经保存,则在输入 n 时,可以判断 n 与这 4 个完数的大小关系输出相应的完数。具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int n;
    while(cin>>n) {
        cout<<n<<":";
        if(n<6) cout<<" NULL";
        else if (n<28) cout<<" 6";
        else if (n<496) cout<<" 6 28";
        else if (n<8128) cout<<" 6 28 496";
        else if (n<=10000) cout<<" 6 28 496 8128";
        cout<<endl;
    }
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int n;
    while(scanf("%d",&n)!=EOF) {
        printf("%d:",n);
        if(n<6) printf(" NULL");
        else if (n<28) printf(" 6");
        else if (n<496) printf(" 6 28");
        else if (n<8128) printf(" 6 28 496");
        else if (n<=10000) printf(" 6 28 496 8128");
        printf("\n");
    }
    return 0;
}
```

实际上,空间换时间的方法经常需要使用数组。即一次性先把所有结果保存在数组中,输入数据时再从数组中把相应结果取出来并输出(可能在输出前要稍做处理)。

例 3.4.9　组合数(HLOJ 1953)


例 3.4.9

Problem Description

输入两个正整数 n、m,要求输出组合数 C_n^m。

例如,当 $n=5$、$m=2$ 时,组合数 $C_5^2=(5\times4)/(2\times1)=10$。

Input

测试数据有多组,处理到文件尾。每组测试输入两个整数 n,m($0<m\leqslant n\leqslant20$)。

Output

对于每组测试,输出组合数 C_n^m。

Sample Input	Sample Output
5 3 20 12	10 125970

本题如果直接使用组合数的公式,分别把分子、分母求出来再相除则需要用 unsigned int 或 long long int 类型,否则会产生数据溢出问题。若仅使用 int 类型,则可以采用程序设计竞赛可能用到的一个小技巧:循环变量 i 从 1 到 m 进行循环,每次先乘以一个数$(n-i+1)$再除以一个数(i),具体代码如下。

```
//C++风格
#include<iostream>
using namespace std;
int main() {
    int m,n;
    while(cin>>n>>m) {
        int c=1;
        for(int i=1;i<=m;i++) {
            c=c*(n-i+1)/i;
        }
        cout<<c<<endl;
    }
    return 0;
}
/*C风格*/
#include<stdio.h>
int main() {
    int c,i,m,n;
    while(scanf("%d%d",&n,&m)!=EOF) {
        c=1;
        for(i=1;i<=m;i++) {
            c=c*(n-i+1)/i;
        }
        printf("%d\n",c);
    }
    return 0;
}
```

例 3.4.10

例 3.4.10* 平均值(HLOJ 1912)

Problem Description

在一行上输入若干整数,每个整数以一个空格分开,求这些整数的平均值。

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。每组测试输入一个字符串(仅包含数字字符和空格)。

Output

对于每组测试,输出以空格分隔的所有整数的平均值,结果保留一位小数。

Sample Input	Sample Output
1 1 2 3 4 5 6 7 8 9 10	5.5

由于本题未明确一行上输入几个整数，可以直接用 getline 输入一个字符串，再根据空格把各个整数字符串分离出来。一种方法是类似于单词分离（参看例 3.4.7）的方法取得各整数串再把其转换为整数，这种方法的代码由读者自行实现。另一种方法是使用 C++STL 中的串流 stringstream（需要包含头文件 sstream），用这种方法可以很容易得到各个整数，具体代码如下。

```
//C++风格
#include<iostream>
#include<string>
#include<sstream>                    //串流的头文件
using namespace std;
int main() {
    int T;
    cin>>T;
    cin.get();                       //吸收测试组数后的换行符
    while(T--) {
        string s;
        getline(cin,s);              //输入包含空格的串
        stringstream ss;             //定义串流 ss
        ss<<s;                       //把串插入到串流，此处 ss 类似 cout
        int t;
        int sum=0, cnt=0;
        while(ss>>t) {               //从串流中提取以空格分隔的数据，此处 ss 类似 cin
            sum+=t;
            cnt++;
        }
        printf("%.1lf\n",sum*1.0/cnt);
    }
    return 0;
}
```

可见，使用 stringstream 可以方便地取得输入的字符串中以空格分隔的各个数据，方法是先把输入的数据“输出”到串流中，再从串流中“输入”到变量中。另外，可以发现，代码中没有做把数字串转换为整数的工作，因为 ss>>t 中的 t 是整型变量，所以得到的结果就是整数。如果要把整数转换为数字串，也可以用串流完成，例如，以下代码把整数 123456 转换为数字串"123456"。

```
stringstream ss;
ss<<123456;
string s;
ss>>s;
```

在线做题时或程序设计竞赛中使用串流有些情况下可以简化编程；但需要注意的是，串流的执行效率较低。本题也能用 C 语言处理，有兴趣的读者可以自行完成。

本书前三章的大部分代码同时提供了 C 风格和 C++风格的写法，其目的在于使读者体会在过程化程序设计中 C++风格和 C 风格代码的区别不大，并为读者在需要把后续章节中的 C++风格代码转换为 C 风格代码时提供参考。除了动态内存分配、字符数组等个别章节，本书后续章节不再特意提供 C 风格的写法，需要的读者可以参考前三章把后续章节的

C++代码转换为C风格的写法。在不同的C语言标准下,C风格的代码可能有所不同。若读者拟编写纯粹的C风格代码,为避免在线提交时选择C编译器而导致编译错,建议如下。

(1) 函数中的所有变量都定义在可执行语句之前。

(2) for语句的()中不定义循环变量。

(3) 不使用"//"注释符。

习　　题

一、选择题

1. C/C++过程化程序设计的三种基本程序控制结构是(　　)。

A. 顺序结构、选择结构、循环结构

B. 输入、处理、输出

C. for、while、do…while

D. 复合语句、基本语句、空语句

2. 在C/C++语言中,if语句中的else子句总是与(　　)尚无else匹配的if相配对。

A. 缩排位置相同　　B. 其之前最近　　C. 其之后最近　　D. 同一行上

3. 以下代码段的输出结果是(　　)。

```
int i = 1, j = 2;
if ( -- i && -- j)cout << i <<" ";
else cout << j <<" ";
if (++i || ++j) cout << i << endl;
else cout << j << endl;
```

A. 1 1　　B. 1 2　　C. 2 3　　D. 2 1

4. 执行以下代码段后,变量i的值为(　　)。

```
int i = 1;
switch (i) {
    case 1: i += 10;
    case 2: i += 20;
    case 3: i++; break;
    default: i++; break;
}
```

A. 11　　B. 31　　C. 33　　D. 32

5. 下面有关for循环的正确描述是(　　)。

A. for循环只能用于循环次数确定的情况

B. for循环先执行循环体语句,后判断循环条件

C. 在for循环中,不能用break语句跳出循环体

D. for循环的循环体可以包含多条语句,但多条语句必须构成复合语句

6. 执行语句"for(s=0,k=0;s<=20||k<=10;k+=2) s+=k;"后,k、s的值为(　　)。

A. 30、12　　B. 12、30　　C. 20、10　　D. 10、20

7. k、s的当前值为5、0,执行语句"while(--k)s+=k;"后,k、s值分别为(　　)。

A. 15、0　　B. 0、15　　C. 10、0　　D. 0、10

8. 关于 do…while 循环，以下说法中正确的是(　　)。

A. 循环体语句只能有一条基本语句

B. 在 while(循环条件)后面不能写分号

C. 当 while 后面循环条件的值为 false 时结束循环

D. 根据情况可以省略 while

9. 以下不是无限循环的语句为(　　)。

A. for(y=10,x=1;x<++y;x++);

B. for(; ;);

C. while(1){x++;}

D. for(i=10;true ;i--) sum+=i;

10. 下列代码段中循环体执行的次数为(　　)。

```
int k = 10;
while (k = 1) k = k - 1;
```

A. 循环体语句一次都不执行　　B. 循环体语句执行无数次

C. 循环体语句执行一次　　D. 循环体语句执行 9 次

二、OJ 编程题

1. 输入输出练习(1)(HLOJ 2096)

Problem Description

共有 T 组测试数据，每组测试求 n 个整数之和。

Input

首先输入一个正整数 T，表示测试数据的组数，然后是 T 组测试数据。每组测试先输入数据个数 n，然后再输入 n 个整数，数据之间以一个空格分隔。

Output

对于每组测试，在一行上输出 n 个整数之和。

```
Sample Input
2
4 1 2 3 4
5 1 8 3 4 5
```

```
Sample Output
10
21
```

2. 输入输出练习(2)(HLOJ 2097)

Problem Description

测试数据有多组，处理到文件尾。每组测试求 n 个整数之和。

Input

测试数据有多组，处理到文件尾。每组测试先输入数据个数 n，然后再输入 n 个整数，数据之间以一个空格分隔。

Output

对于每组测试，在一行上输出 n 个整数之和。

```
Sample Input
5 1 8 3 4 5
```

```
Sample Output
21
```

3. 输入输出练习(3)(HLOJ 2098)

Problem Description

测试数据有多组,每组测试求 n 个整数之和,处理到输入的 n 为 0 为止。

Input

测试数据有多组。每组测试先输入数据个数 n,然后再输入 n 个整数,数据之间以一个空格间隔,当 n 为 0 时,输入结束。

Output

对于每组测试,在一行上输出 n 个整数之和。

```
Sample Input
5 1 8 3 4 5
0
```

```
Sample Output
21
```

4. 输入输出练习(4)(HLOJ 2099)

Problem Description

求 n 个整数之和。T 组测试,且要求每两组输出之间空一行。

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。每组测试先输入数据个数 n,然后再输入 n 个整数,数据之间以一个空格分隔。

Output

对于每组测试,在一行上输出 n 个整数之和,每两组输出结果之间留一个空行。

```
Sample Input
2
4 1 2 3 4
5 1 8 3 4 5
```

```
Sample Output
10

21
```

5. 电费(HLOJ 2092)

Problem Description

某电价规定:月用电量在 150kW·h 及以下部分按每千瓦时 0.4463 元收费,月用电量在 151~400kW·h 的部分按每千瓦时 0.4663 元收费,月用电量在 401kW·h 及以上部分按每千瓦时 0.5663 元收费。

请编写一个程序,根据输入的月用电量(单位 kW·h),按该电价规定计算出应缴的电费(单位以元计)。

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。对于每组测试,输入一个整数 n($0 \leqslant n \leqslant 10\ 000$),表示月用电量。

Output

对于每组测试，输出一行，包含一个实数，表示应缴的电费。结果保留两位小数。

Sample Input	Sample Output
1 267	121.50

6. 小游戏（HLOJ 2011）

Problem Description

有一个小游戏，6 个人上台去算手中扑克牌点数之和是否 5 的倍数，据说是小学生玩的。这里稍微修改一下玩法，n 个人上台，算手中数字之和是否同时是 5，7，3 的倍数。

Input

首先输入一个正整数 T，表示测试数据的组数，然后是 T 组测试数据。每组测试先输入 1 个整数 $n(1\leqslant n\leqslant 15)$，再输入 n 个整数，每个都小于 1000。

Output

对于每组测试，若 n 个整数之和同时是 5，7，3 的倍数则输出“YES”，否则输出“NO”。引号不必输出。

Sample Input	Sample Output
2 3 123 27 60 3 23 27 6	YES NO

7. 购物（HLOJ 2012）

Problem Description

小明购物之后搞不清最贵的物品价格和所有物品的平均价格，请帮他编写一个程序实现。

Input

测试数据有多组，处理到文件尾。每组测试先输入 1 个整数 $n(1\leqslant n\leqslant 100)$，接下来的 n 行中每行输入 1 个英文字母表示的物品名及该物品的价格。测试数据保证最贵的物品只有 1 个。

Output

对于每组测试，在一行上输出最贵的物品名和所有物品的平均价格，两者之间留一个空格，平均价格保留 1 位小数。

Sample Input	Sample Output
3 a 1.8 b 2.5 c 1.5	b 1.9

8. 等边三角形面积(HLOJ 2013)

Problem Description

数学基础对于程序设计能力而言很重要。对于等边三角形面积,请选择合适的方法计算之。

Input

测试数据有多组,处理到文件尾。每组测试输入1个实数表示等边三角形的边长。

Output

对于每组测试,在一行上输出等边三角形的面积,结果保留两位小数。

Sample Input	Sample Output
1.0 2.0	0.43 1.73

9. 三七二十一(HLOJ 2017)

Problem Description

某天,诺诺看到三七二十一(3721)数,觉得很神奇,这种数除以3余2,而除以7则余1。例如,8是一个3721数,因为8除以3余2,8除以7余1。现在给出两个整数a、b,求区间$[a,b]$中的所有3721数,若区间内不存在3721数则输出"none"。

Input

首先输入一个正整数T,表示测试数据的组数,然后是T组测试数据。每组测试输入两个整数a,b($1\leqslant a<b<2000$)。

Output

对于每组测试,在一行上输出区间$[a,b]$中所有的3721数,每两个数据之间留一个空格。如果给定区间不存在3721数,则输出"none"(引号不必输出)。

Sample Input	Sample Output
2 1 7 1 100	none 8 29 50 71 92

10. 胜者(HLOJ 2014)

Problem Description

Sg和Gs进行乒乓球比赛,进行若干局之后,想确定最后是谁胜(赢的局数多者胜)。

Input

测试数据有多组,处理到文件尾。每组测试先输入一个整数n,接下来的n行中每行输入两个整数a,b($0\leqslant a,b\leqslant 20$),表示Sg与Gs的比分是$a$比$b$。

Output

对于每组测试数据,若还不能确定胜负则输出"CONTINUE",否则在一行上输出胜者"Sg"或"Gs"。引号不必输出。

```
Sample Input
3
3  11
13 11
11 9
```

```
Sample Output
Sg
```

11. 加密(HLOJ 2015)

Problem Description

信息安全很重要,特别是密码。给定一个 5 位的正整数 n 和一个长度为 5 的字母构成的字符串 s,加密规则很简单,字符串 s 的每个字符变为它后面的第 k 个字符,其中,k 是 n 的每一个数位上的数字。第一个字符对应 n 的万位上的数字,最后一个字符对应 n 的个位上的数字。简单起见,s 中的每个字符为 ABCDE 中的一个。

Input

测试数据有多组,处理到文件尾。每组测试数据在一行上输入非负的整数 n 和字符串 s。

Output

对于每组测试数据,在一行上输出加密后的字符串。

```
Sample Input
12345 ABCDE
```

```
Sample Output
BDFHJ
```

12. 比例(HLOJ 2016)

Problem Description

某班同学在操场上排好队,请确定男、女同学的比例。

Input

测试数据有多组,处理到文件尾。每组测试数据输入一个以“.”结束的字符串,串中每个字符可能是“MmFf”中的一个,“m”或“M”表示男生,“f”或“F”表示女生。

Output

对于每组测试数据,在一行上输出男、女生的百分比,结果四舍五入到 1 位小数。输出形式参照 Sample Output。

```
Sample Input
FFfm.
MfF.
```

```
Sample Output
25.0 75.0
33.3 66.7
```

13. 某校几人(HLOJ 2018)

Problem Description

某学校教职工人数不足 n 人,在操场排队,7 个一排剩 5 人,5 个一排剩 3 人,3 个一排剩 2 人;请问该校人数有多少种可能?最多可能有几人?

Input

测试数据有多组,处理到文件尾。每组测试输入一个整数 n($1 \leqslant n \leqslant 10\ 000$)。

Output

对于每组测试,输出一行,包含两个以一个空格间隔的整数,分别表示该校教职工人数的可能种数和最多可能的人数。

Sample Input	Sample Output
1000	9 908

14. 昨天(HLOJ 2019)

Problem Description

小明喜欢上了日期的计算。这次他要做的是日期的减 1 天操作,即求在输入日期的基础上减去 1 天后的结果日期。

例如,日期为 2019-10-01,减去 1 天,则结果日期为 2019-09-30。

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。每组测试输入 1 个日期,日期形式为"yyyy-mm-dd"。保证输入的日期合法,而且输入的日期和计算结果都在[1000-01-01,9999-12-31]范围内。

Output

对于每组测试,在一行上以"yyyy-mm-dd"的形式输出结果。

Sample Input	Sample Output
1 2019-10-01	2019-09-30

15. 直角三角形面积(HLOJ 2020)

Problem Description

已知直角三角形的三边长,求该直角三角形的面积。

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。每组数据输入 3 个整数 a,b,c,代表直角三角形的三边长。

Output

对于每组测试输出一行,包含一个整数,表示直角三角形面积。

Sample Input	Sample Output
2 3 4 5 3 5 4	6 6

16. 转向三角形(HLOJ 2021)

Problem Description

输入一个整数 n,要求用数字 1～n 排列出一个转向三角形。例如,$n=5$ 时,转向三角形如 Sample Output 所示。

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。每组测试数据输入 1 个整数 $n(1\leqslant n\leqslant 9)$。

Output

对于每组测试数据,输出一个有 $2n-1$ 行的,由数字 $1\cdots n\cdots 1$ 组成的转向三角形(参看 Sample Output)。

```
Sample Input
1
5
```

```
Sample Output
1
22
333
4444
55555
4444
333
22
1
```

17. 求累加和(HLOJ 2024)

Problem Description

输入两个整数 n 和 a,求累加和 $S=a+aa+aaa+\cdots+aa\cdots a$($n$ 个 a)之值。

例如,当 $n=5$,$a=2$ 时,$S=2+22+222+2222+22\ 222=24\ 690$。

Input

测试数据有多组,处理到文件尾。每组测试输入两个整数 n 和 $a(1\leqslant n,a<10)$。

Output

对于每组测试,输出 $a+aa+aaa+\cdots+aa\cdots a$($n$ 个 a)之值。

```
Sample Input
5 3
8 6
```

```
Sample Output
37035
74074068
```

18. 字符梯形(HLOJ 2022)

Problem Description

用从 m 到 n 的数字字符排列出一个字符梯形。

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。每组测试数据输入两个整数 m、$n(1\leqslant m\leqslant n\leqslant 9)$。

Output

对于每组测试数据,输出一个有 $n-m+1$ 行的,由数字 $m\cdots n$ 排列而成的梯形,每行的长度依次为:m,$m+1$,$m+2$,…,n,每行的数字依次是 m,$m+1$,$m+2$,…,n。

```
Sample Input
1
3 6
```

```
Sample Output
333
4444
55555
666666
```

19. 菱形(HLOJ 2023)

Problem Description

输入一个整数 n,输出 $2n-1$ 行构成的菱形,例如,$n=5$ 时的菱形如 Sample Output 所示。

Input

测试数据有多组,处理到文件尾。每组测试输入一个整数 $n(3\leqslant n\leqslant 20)$。

Output

对于每组测试数据,输出一个共 $2n-1$ 行的菱形,具体参看 Sample Output。

```
Sample Input
5
```

```
Sample Output
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

20. 水仙花数(HLOJ 2025)

Problem Description

输入两个 3 位的正整数 m,n,输出[m,n]区间内所有的"水仙花数"。所谓"水仙花数"是指一个 3 位数,其各位数字的立方和等于该数本身。

例如,153 是一水仙花数,因为 $153=1\times1\times1+5\times5\times5+3\times3\times3$。

Input

测试数据有多组,处理到文件尾。每组测试输入两个 3 位的正整数 m,$n(100\leqslant m<n\leqslant 999)$。

Output

对于每组测试,若[m,n]区间内没有水仙花数则输出"none"(引号不必输出),否则逐行输出区间内所有的水仙花数,每行输出的格式为:$n=a*a*a+b*b*b+c*c*c$,其中,n 是水仙花数,a、b、c 分别是 n 的百、十、个位上的数字,具体参看 Sample Output。

```
Sample Input
100 150
100 200
```

```
Sample Output
none
153 = 1 * 1 * 1 + 5 * 5 * 5 + 3 * 3 * 3
```

21. 猴子吃桃(HLOJ 2026)

Problem Description

猴子第一天摘下若干个桃子,当即吃了 2/3,还不过瘾,又多吃了一个,第二天早上又将剩下的桃子吃掉 2/3,又多吃了一个。以后每天早上都吃了前一天剩下的 2/3 再多吃一个。到第 n 天早上想再吃时,发现只剩下 k 个桃子了。求第一天共摘了多少桃子。

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。每组数据输入两个正整数 n,k($1 \leqslant n$,$k \leqslant 15$)。

Output

对于每组测试数据,在一行上输出第一天共摘了多少个桃子。

```
Sample Input
2
2 1
4 2
```

```
Sample Output
6
93
```

22. 最小回文数(HLOJ 2027)

Problem Description

若一个数正向看和反向看等价,则称作回文数。例如,6,2552,12321 均是回文数。给出一个正整数 n,求比 n 大的最小的回文数。(n 和运算结果均不会超出 int 类型范围。)

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。每组测试输入 1 个正整数 n。

Output

对于每组测试数据,输出比 n 大的最小回文数。

```
Sample Input
2
12
123456
```

```
Sample Output
22
124421
```

23. 分解素因子(HLOJ 2028)

Problem Description

假设 n 是一个正整数,它的值不超过 1 000 000,请编写一个程序,将 n 分解为若干个素数的乘积。

Input

首先输入一个正整数 T,表示测试数据的组数,然后是 T 组测试数据。每组测试数据输入一个正整数 n($1 < n \leqslant 1\ 000\ 000$)。

Output

每组测试对应一行输出,输出 n 的素数乘积表示式,式中的素数从小到大排列,两个素数之间用一个“*”表示乘法。若输入的是素数,则直接输出该数。

```
Sample Input
2
9828
88883
```

```
Sample Output
2 * 2 * 3 * 3 * 3 * 7 * 13
88883
```

24. Fibonacci 分数序列(HLOJ 2029)

Problem Description

求 Fibonacci 分数序列的前 n 项之和。Fibonacci 分数序列的首项为 2/1,后面依次是 3/2,5/3,8/5,13/8,21/13…

Input

测试数据有多组,处理到文件尾。每组测试输入一个正整数 n($2 \leqslant n \leqslant 20$)。

Output

对于每组测试,输出 Fibonacci 分数序列的前 n 项之和。结果保留 6 位小数。

```
Sample Input
3
8
15
```

```
Sample Output
5.166667
13.243746
24.570091
```

25. n 马 n 担问题(HLOJ 2030)

Problem Description

有 n 匹马,驮 n 担货,大马驮 3 担,中马驮 2 担,两匹小马驮 1 担,问有大、中、小马各多少匹?(某种马的数量可以为 0。)

Input

测试数据有多组,处理到文件尾。每组测试输入一个正整数 n($8 \leqslant n \leqslant 1000$)。

Output

对于每组测试,逐行输出所有符合要求的大、中、小马的匹数。要求按大马数从小到大的顺序输出,每两个数字之间留一个空格。

```
Sample Input
20
```

```
Sample Output
1 5 14
4 0 16
```