



清华大学

Tsinghua University

计算机程序设计基础

第5讲 循环结构程序设计

沈瑜 (010-62782951)

shenyu@tsinghua.edu.cn

清华大学电机系

2021.10.12





常见错误1

```
char cInputA, cInputB;  
printf(“分别输入甲乙的手势”);  
scanf(“%c%c”, &cInputA, &cInputB) ;  
If (cInputA== ‘R’ || ‘r’ )  
{  
    ... // 代码段1  
}  
else  
{  
    ... // 代码段2  
}
```



主要内容

- 循环结构C程序设计举例
- 循环语句
 - while
 - do-while
 - for
- 跳转语句
 - break
 - continue
 - goto
- 循环结构小结

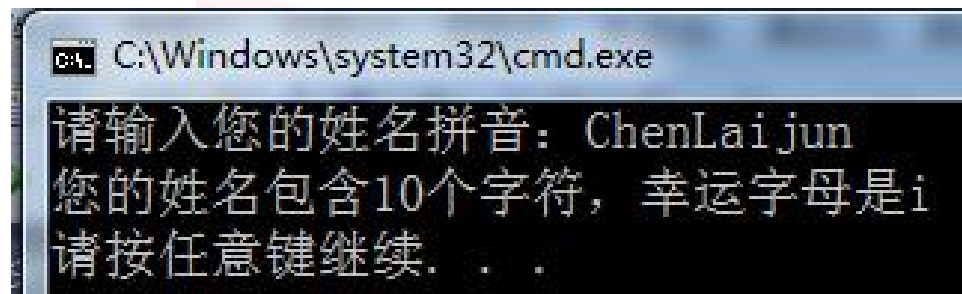




5.1 循环结构C程序设计举例

● 例1: 幸运字母

```
1 #include <stdio.h>
2 int main()
3 {
4     int sum = 0;
5     char ch;
6     int num = 0; //姓名拼音字符数
7
8     printf("请输入您的姓名拼音:");
9
10    while(1)
11    {
12        scanf("%c", &ch);
13        if(ch>='A' && ch<='Z')
14            ch = ch - 'A' + 'a';
15        else if(ch>='a' && ch<='z');
16        else
17            break;
18        num++;
19        sum += ch;
20    }
21    printf("您的姓名包含%d个字符, 幸运字母是%c\n", num, sum/num);
22    return 0;
23 }
```

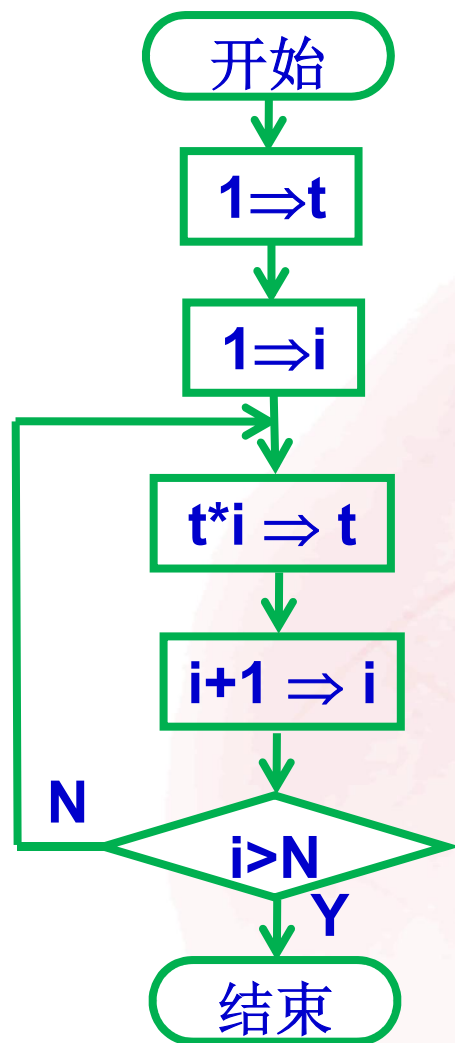


循环

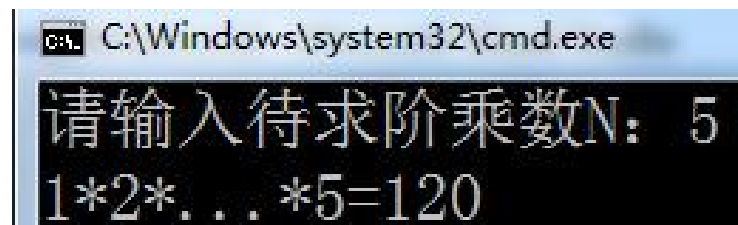


●例2:求阶乘 ($N!$) 参考教材P17, 例2.1; P23, 例2.6

流程图



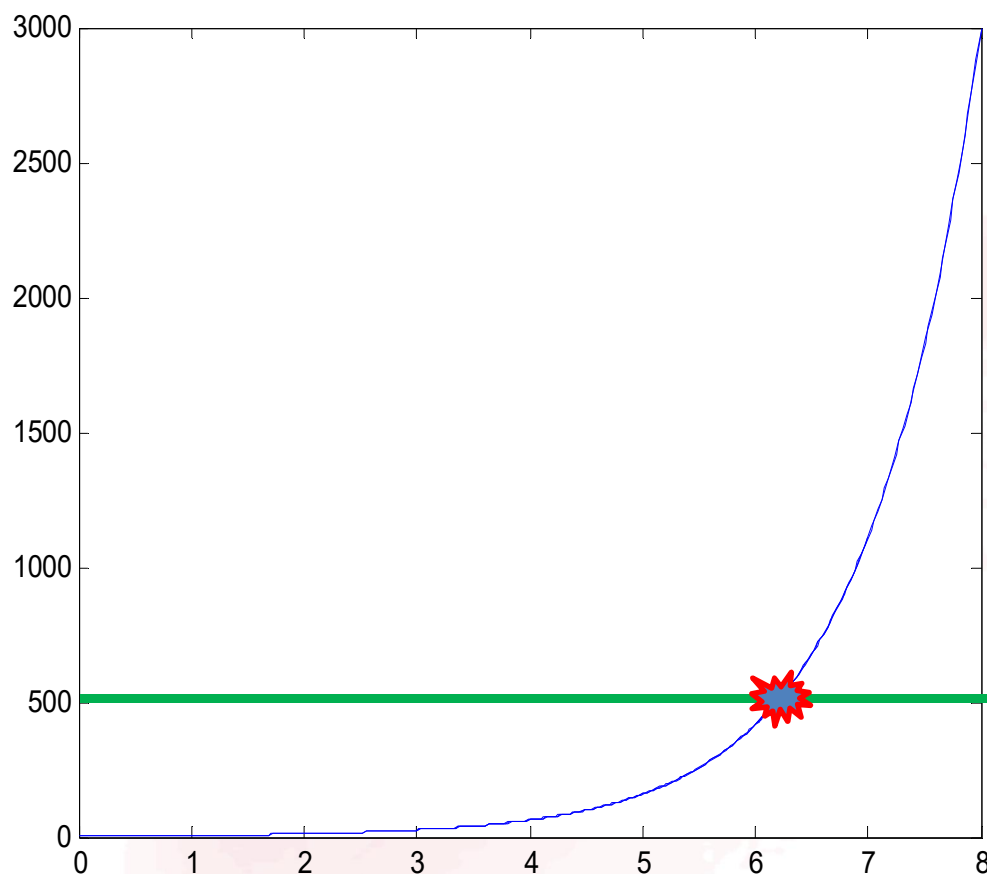
```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i, N, fac;
6     printf("请输入待求阶乘数N: ");
7     scanf("%d",&N);
8     fac = 1;
9     for(i=1;i<=N;i++)
10    {
11        fac = fac * i;
12    }
13    printf("1*2*...*%d=%d\n",N,fac);
14    return 0;
15 }
```



循环



● 例3:超越方程求解: $e^x + 2x = 500$, 要求误差小于 10^{-6}



方程特点:
左侧单调递增
右侧 (常数)

问题关键:
找交点

求解思路: 借鉴幸运52猜价格游戏,
不断缩小价格 (解) 区间





● 例3:超越方程求解: $e^x + 2x = 500$, 要求误差小于 10^{-6}

```
1 #include <stdio.h>
2 #include <math.h>
3
4 int main(){
5     double x, low, high, feval;
6     low=0.0;
7     high=10.0;
8
9     for(x=5.0; high-low>1.0e-6; )
10     {
11         feval = exp(x) + 2*x - 500;
12         if(feval<0)
13             low=x;
14         else
15             high=x;
16         x=(high+low)/2;
17     }
18
19     printf("方程exp(x)+2*x=500的近似解为 : %.6f\n", x);
20     printf("该解所在区间 : [%.6f, %.6f]\n", low, high);
21     printf("该解的精度小于 %.8f\n", high-low);
22     printf("exp(%.6f) + 2*%.6f = %lf\n", x, x, exp(x) + 2*x);
23     return 0;
24 }
```

需要用到数学函数库

```
C:\Windows\system32\cmd.exe
方程exp(x)+2*x=500的近似解为: 6.189539
该解所在区间: [6.189538, 6.189539]
该解的精度小于0.00000060
exp(6.189539) + 2*6.189539 = 500.000113
```

循环

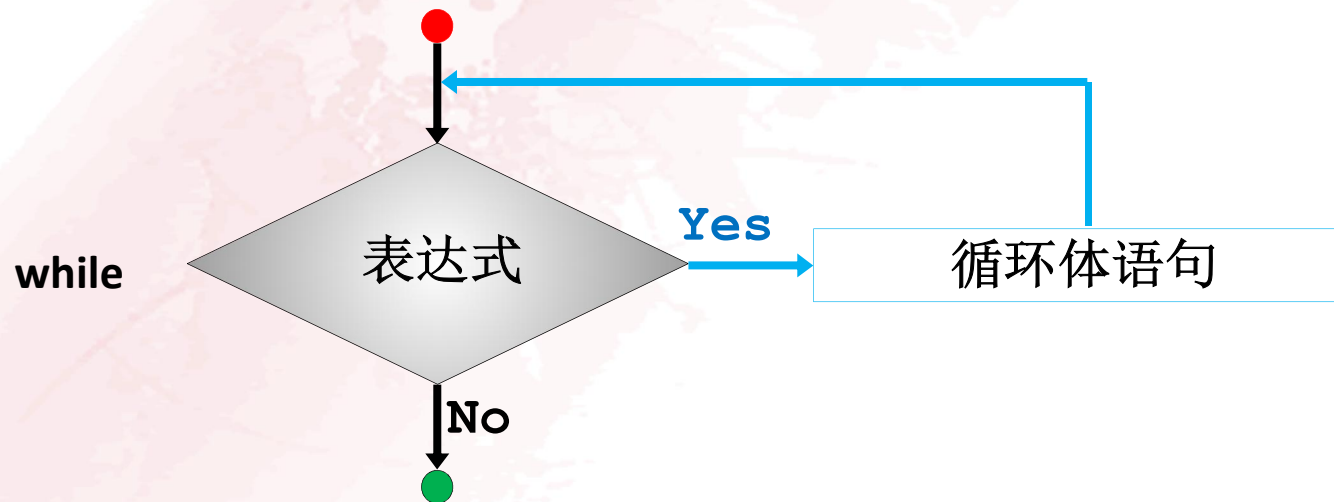


5.2 循环语句

(1) while 循环

语法

while(表达式) 循环体语句



特点

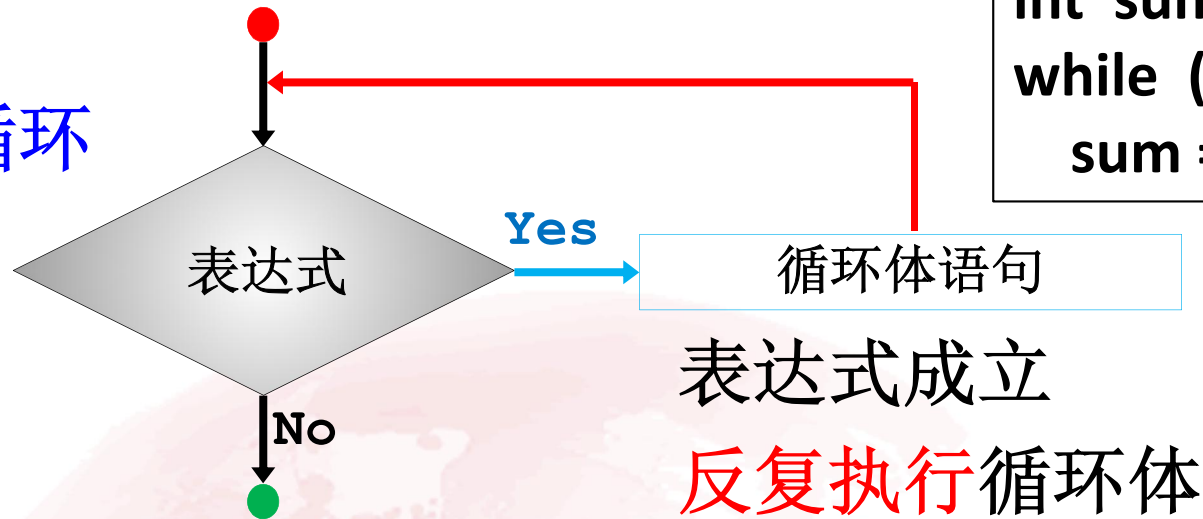
表达式成立时，反复执行循环体语句



while循环与if语句对比

while循环

while



```
int sum = 0;
while (sum<=100)
    sum = sum+2;
```

if语句

if



```
int sum = 0;
if (sum<=100)
    sum = sum+2;
```

while循环示例（1-100求和）

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i,sum;
6     i=0;
7     sum=0;
8
9     while (i<=100)
10    {
11        sum=sum+i;
12        i=i+1;
13    }
14    printf("sum = %d\n",sum);
15
16    return 0;
17 }
18
```

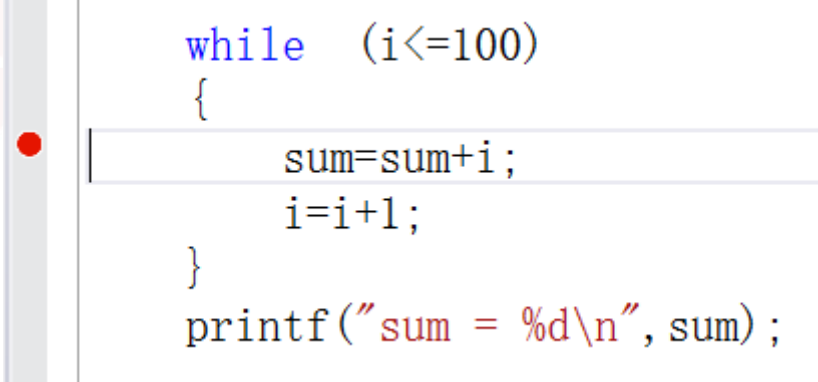
C:\Windows\system32\cmd.exe

sum = 5050



Debug手段

- 如何调试程序
 - 以前我们用printf打印中间结果的方法观察程序是否正确
 - 现在使用断点、跟踪、监视窗口等调试工具
 - VS中，在代码左侧点击即可添加断点（F9）
 - 再次点击即可删除
 - 右键单击断点可看到禁用、触发条件等设置



```
while (i<=100)
{
    sum=sum+i;
    i=i+1;
}
printf("sum = %d\n", sum);
```

The screenshot shows a code editor with a C program. A red dot, representing a breakpoint, is placed on the left margin next to the first line of the while loop, `while (i<=100)`. The code is as follows:

Debug手段

• 如何调试程序

- 按F5键即可进入调试模式
- 程序运行到第一个断点处暂停
- 此时VS窗口上方有若干按钮：



- **继续** (F5)：运行至下一个断点处
- **暂停**：陷入死循环时可用
- **中止** (Shift+F5)：结束当前程序
- **重新开始** (Shift+Ctrl+F5)：重新开始当前程序
- **逐语句** (F11)：执行下一条语句，如果**遇到函数调用则进入**
- **逐过程** (F10)：执行当前行语句，如果**遇到函数调用不进入**
- **跳出** (Shift+F11)：执行当前函数或复合语句块的剩余语句，然后返回上一级 (**不小心按了F11时推荐使用**)
- “**逐语句**” (F11) 进入子函数内部，而“**逐过程**” (F10) 不进入

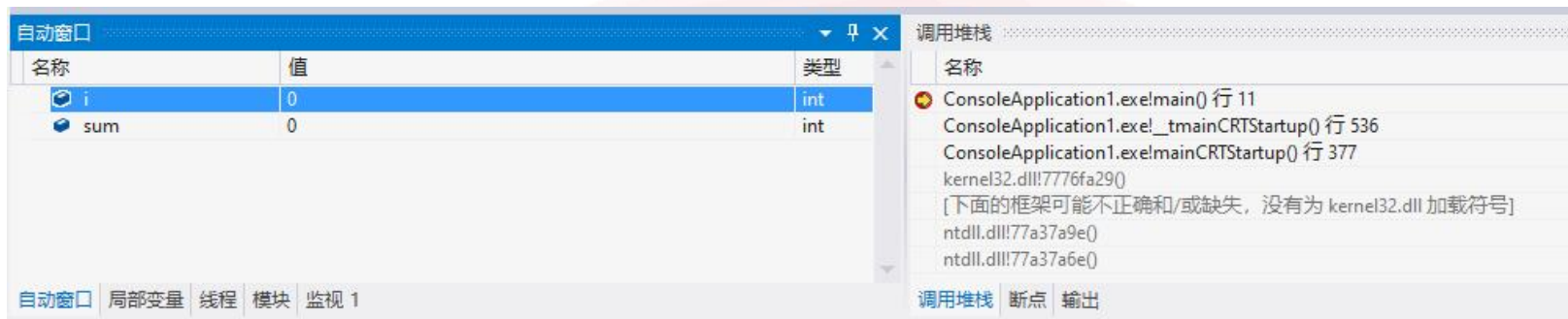


Debug手段

• 如何调试程序

– 跟踪查看变量值

- 进入调试模式后，下方窗口中将显示程序中的变量值



- 进入调试模式后，下方窗口中将显示程序中的变量值

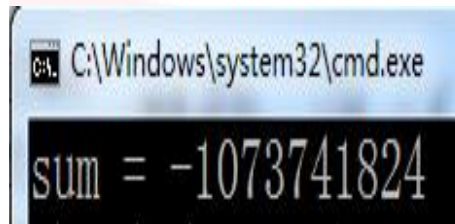
while循环示例：小修改，大变化（1）

参考教材P144-45

猜猜看，会出现什么现象？

是死循环吗？

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i, sum;
6     i=0;
7     sum=0;
8
9     while (i<=100)
10    {
11        sum=sum+i;
12        i=i-1;
13    }
14    printf("sum = %d\n", sum);
15
16    return 0;
17 }
18
```



不是死循环！

知识点回顾：

32位计算机可以表示的整数范围：

最大正数（2147483647）

$0x7FFF\ FFFF = 2^{31} - 1$

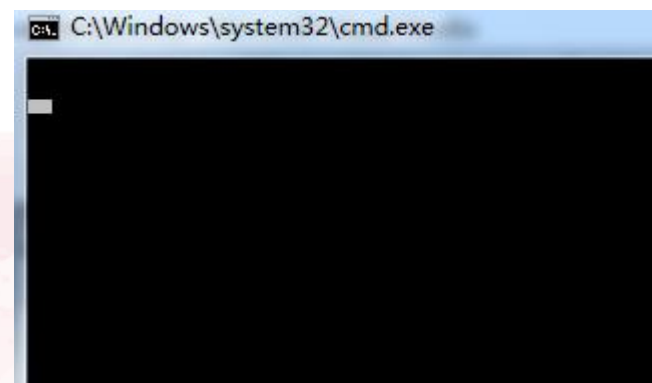
最小负数（-2147483648）

$0x8000\ 0000 = -2^{31}$

while循环示例：小修改，大变化（2）

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i,sum;
6     i=0;
7     sum=0;
8
9     while (i<=100);
10    {
11        sum=sum+i;
12        i=i+1;
13    }
14    printf("sum = %d\n",sum);
15
16    return 0;
17 }
18
```

猜猜看，会出现什么现象？



循环体为空语句，条件一直满足，进入死循环！

编程小提示

- 在if语句的圆括号后放置分号会导致if语句失效

```
if (d == 0); /** WRONG **/  
    printf( "Error: Division by zero\n" );
```

– 无论d值是否为0，总会执行printf语句。

- 在while语句的圆括号后放置分号可能导致死循环

```
i = 10;  
while (i>0); {  
    //...  
}
```

– 花括号中内容执行不到



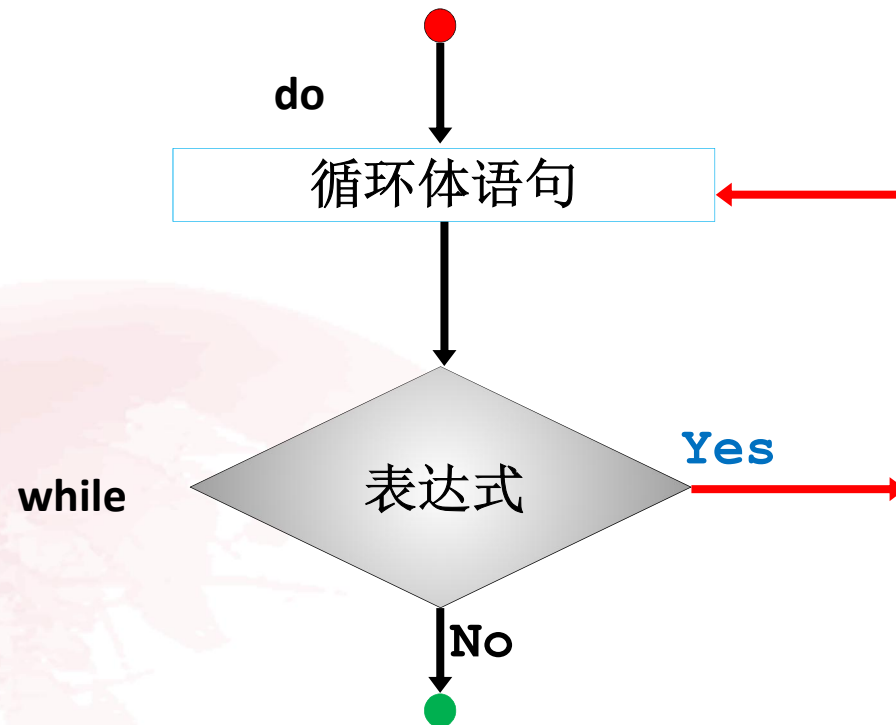
(2) do-while 循环

语法

```
do  
    循环体语句  
while(表达式);
```

特点

先执行一次循环体
表达式成立时，反复执行循环体



do-while 与while的比较及其等价写法

do

循环体语句

while(表达式);

循环体语句

while(表达式)

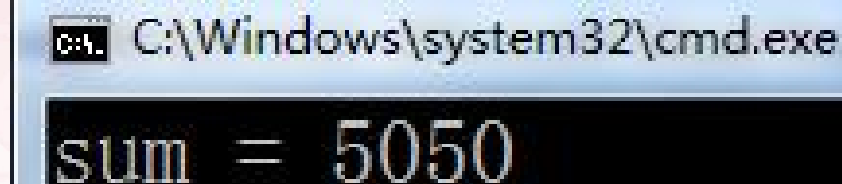
循环体语句

do-while循环
先执行循环体语句再判断表达式



do-while循环示例（还是1-100求和）

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i,sum;
6     i=0;
7     sum=0;
8
9     do
10    {
11        sum=sum+i;
12        i=i+1;
13    }
14    while (i<=100);
15
16    printf("sum = %d\n",sum);
17
18    return 0;
19 }
```



C:\Windows\system32\cmd.exe
sum = 5050

标准语法，不是死循环！

(3) for 循环

语法

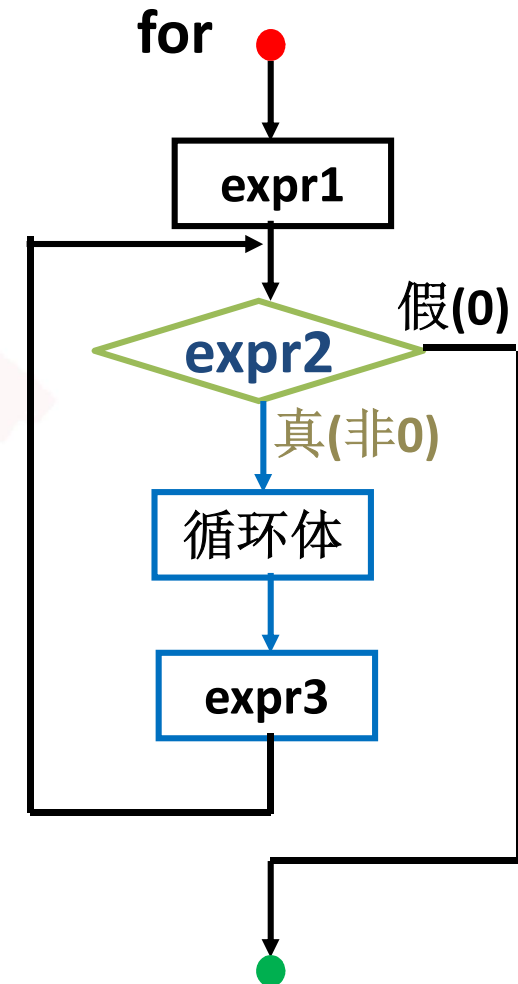
```
for([expr1] ;[ expr2] ;[ expr3])  
    循环体语句
```

特点

更简洁：一行语句完成多个功能

更灵活：三个表达式均可省略

更实用：三种循环中最常用



for与while的比较及其等价写法

for 循环

```
for([expr1] ;[ expr2] ;[ expr3])  
    循环体语句
```

while 循环

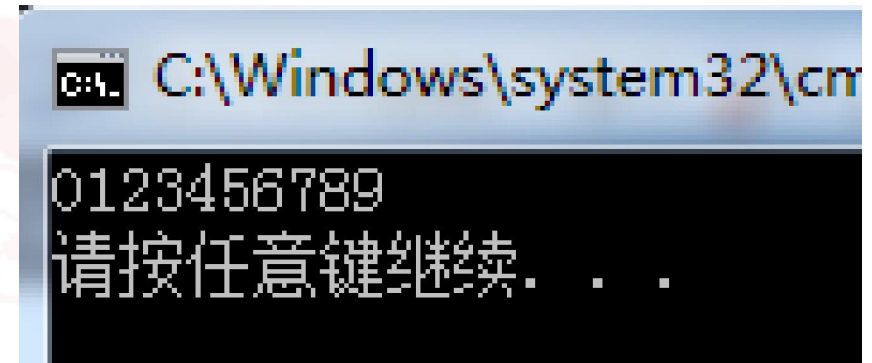
```
expr1;  
while(expr2) {  
    循环体语句  
    expr3;  
}
```

猜猜看: **expr2** 省略时会怎样?

for循环示例

```
1 #include <stdio.h>
2
3 int main(){
4     int i=0;
5     for(i=0;i<10;i++)
6         putchar('0'+i);
7     printf("\n");
8
9     return 0;
10 }
11
```

程序执行结束后，
输出是多少？



将数值0 - 9转换为字符'0' - '9'并输出

知识点回顾：整数0、字符'0'以及 '\0'

编程小提示

- 在for语句的圆括号后放置分号会导致for语句程序体只执行一遍

```
for ( i =10; i>0; i--)  /*** WRONG ***/  
    printf( "Error\n" );
```

– 程序体仅执行一遍，输出结果为：

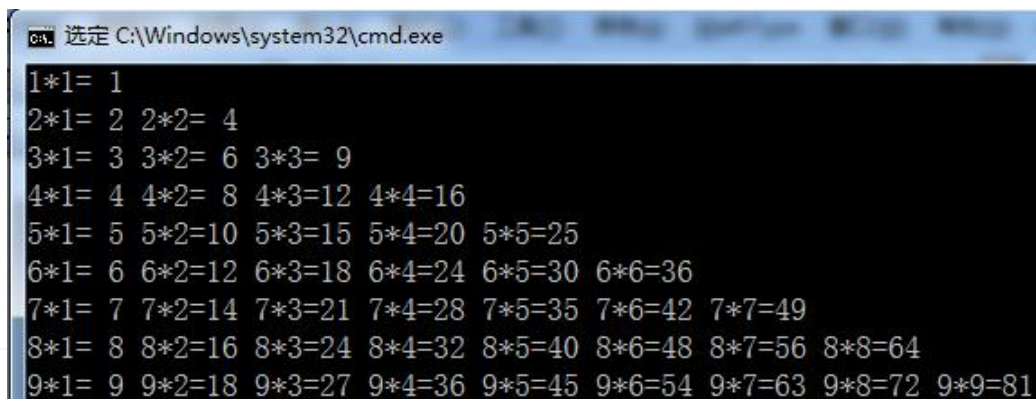
Error



嵌套循环

示例：乘法口诀表

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i ,j;
6     for(i=1;i<10;i++)
7     {
8         for(j=1;j<=i;j++)
9         {
10             printf("%d*%d=%2d ",i,j,i*j);
11         }
12         printf("\n");
13     }
14
15     return 0;
16 }
```



```
选定 C:\Windows\system32\cmd.exe
1*1= 1
2*1= 2 2*2= 4
3*1= 3 3*2= 6 3*3= 9
4*1= 4 4*2= 8 4*3=12 4*4=16
5*1= 5 5*2=10 5*3=15 5*4=20 5*5=25
6*1= 6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
7*1= 7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
8*1= 8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
9*1= 9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
```

小技巧1: 代码的格式化（自动对齐）

调整前

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i ,j;
6     for(i=1;i<10;i++)
7     {
8         for(j=1;j<=i;j++)
9         {
10            printf("%d*%d=%2d ",i,j,i*j);
11        }
12        printf("\n");
13    }
14
15    return 0;
16 }
```

调整后

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i ,j;
6     for(i=1;i<10;i++)
7     {
8         for(j=1;j<=i;j++)
9         {
10            printf("%d*%d=%2d ",i,j,i*j);
11        }
12        printf("\n");
13    }
14
15    return 0;
16 }
```

调整步骤
CTRL + K, F

- 1、选中部分或全部代码（**CTRL+A**）
- 2、按住**CTRL**键，依次按下**K**和**F**

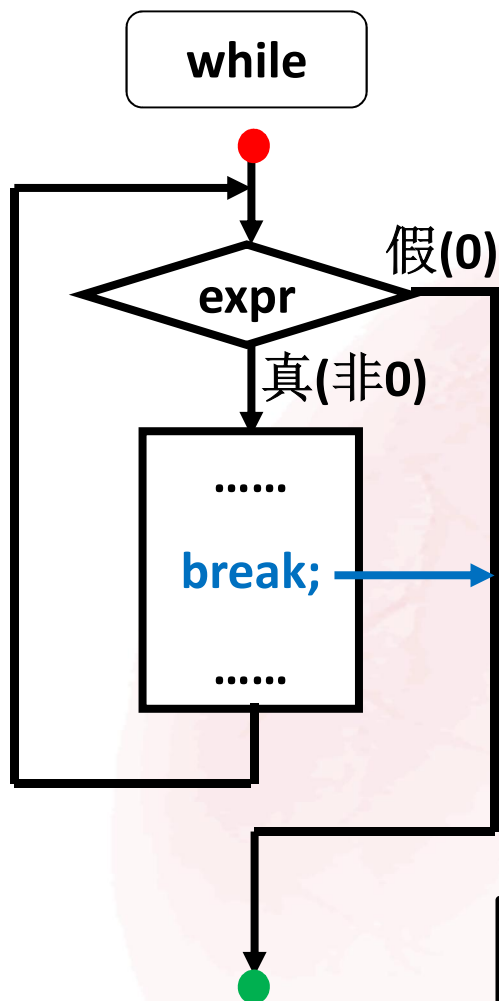
小技巧2：匹配括号（小括号`()`，大括号`{}`）

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i ,j;
6     for(i=1;i<10;i++)
7     {
8         for(j=1;j<=i;j++)
9         {
10             printf("%d*%d=%2d ",i,j,i*j);
11         }
12         printf("\n");
13     }
14
15     return 0;
16 }
```

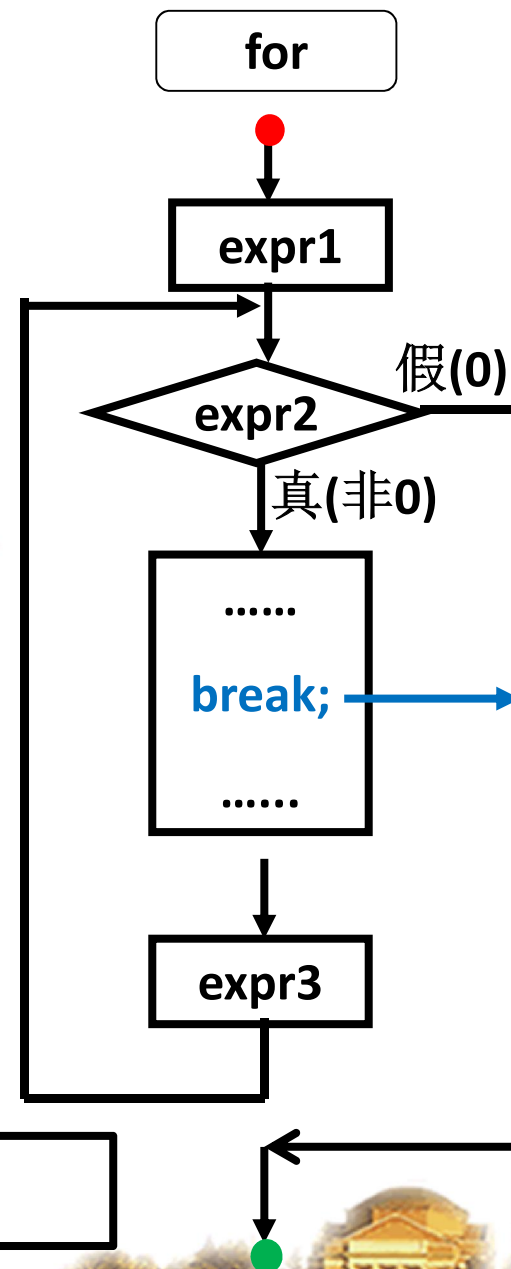
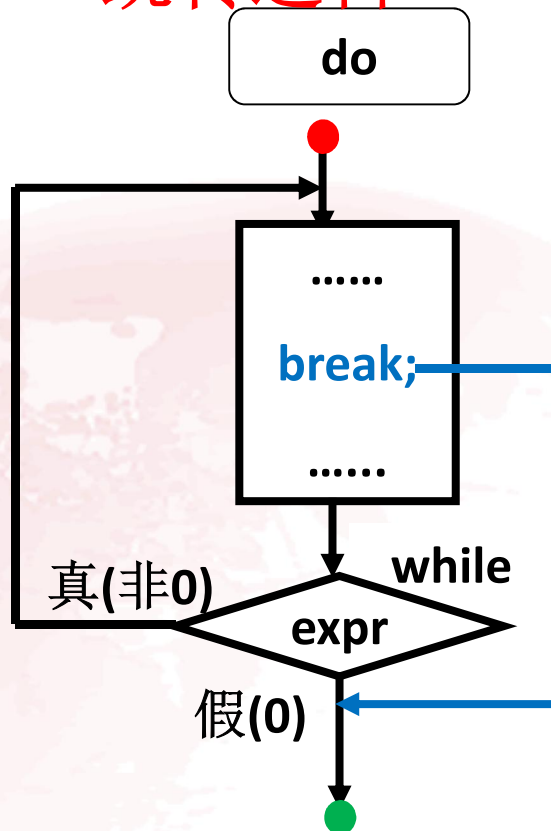
匹配快捷键：CTRL +]

5.3 跳转语句

(1) break



跳转逻辑



作用：跳出整个循环

break用法示例（例1：幸运字母）

```
1 #include <stdio.h>
2 int main()
3 {
4     int sum = 0;
5     char ch;
6     int num = 0; //姓名拼音字符数
7
8     printf("请输入您的姓名拼音：");
9
10    while(1)
11    {
12        scanf("%c", &ch);
13        if(ch>='A' && ch<='Z')
14            ch = ch - 'A' + 'a';
15        ? else if(ch>='a' && ch<='z');
16        else
17            break;
18        num++;
19        sum += ch;
20    }
21    printf("您的姓名包含%d个字符，幸运字母是%c\n", num, sum/num);
22    return 0;
23 }
```

问题1：执行**break**语句后，
跳转到哪条语句？

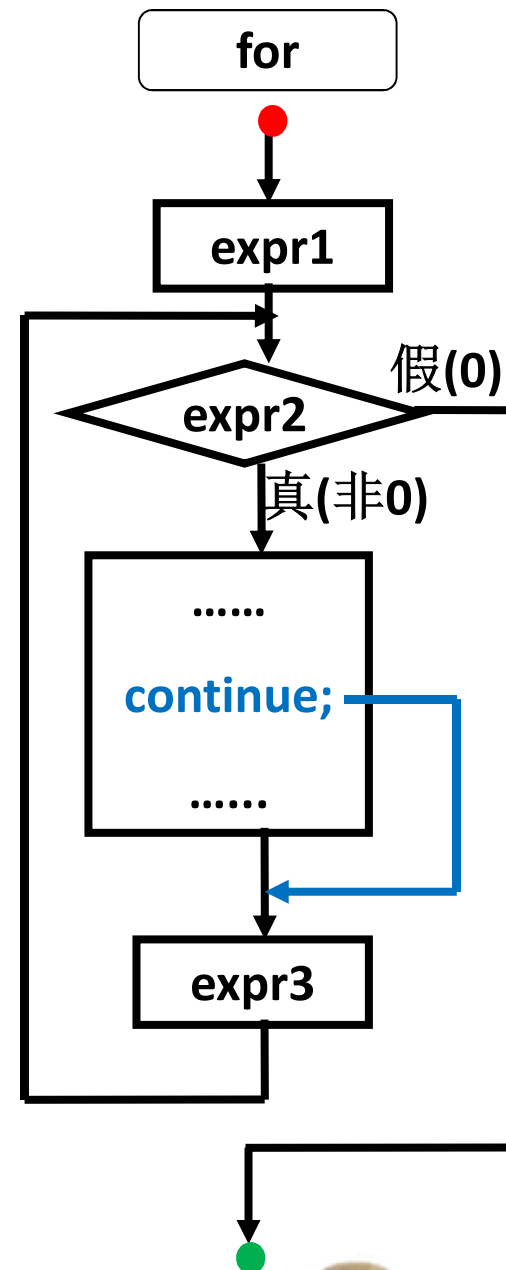
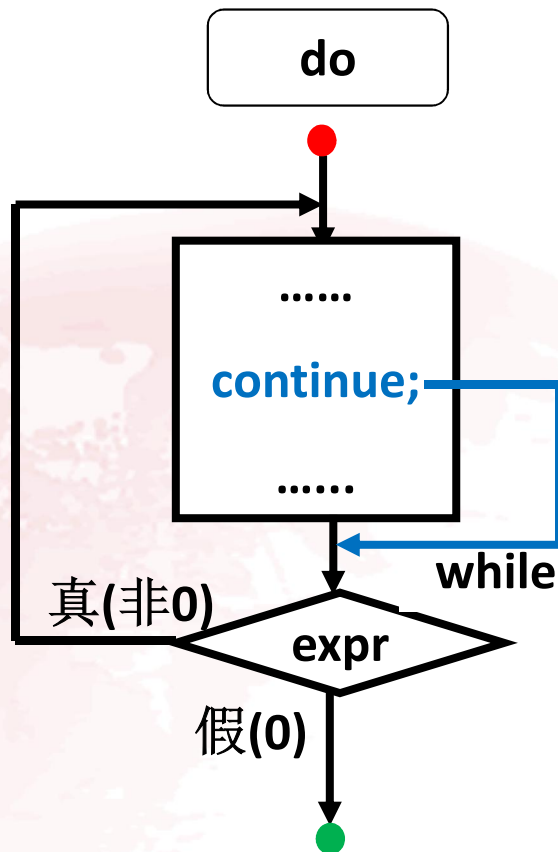
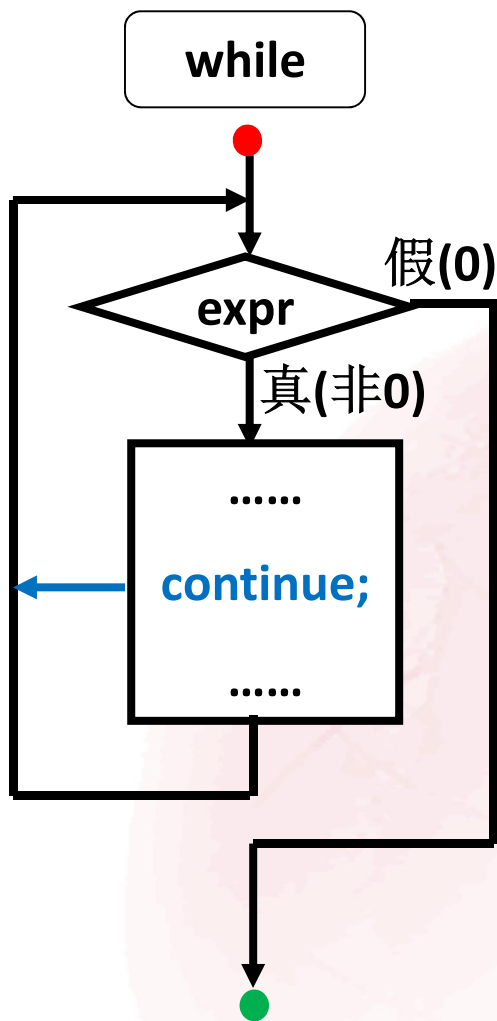
问题2：没有**break**语句会怎样？（死循环！）

break的作用：跳出整个循环



(2) continue

跳转逻辑



作用：跳出本次循环（结束执行循环体）

continue用法示例

任务：输出1—100之间可以被15整除的数

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i;
6     for(i=1;i<=100;i++)
7     {
8         if (i%15!=0)
9         {
10             ? continue;
11         }
12         printf("%d\n",i);
13     }
14     return 0;
15 }
```

问题1：执行continue语句后，跳转到哪条语句？

问题2：换成break语句会怎样？

循环提前结束！

continue作用：跳出本次循环（结束执行循环体）

编程小提示

- 无限循环(编程习惯)
 - 编程中常用的无限循环形式

```
while (1) 或 for( ; ; )  
{  
    if 语句1  
        break;  
    if 语句2  
        continue;  
}
```

- C程序员喜欢采用这种构造无限循环+中断的方式进行编程



(3) goto

语法

goto 语句标号;

执行完**goto**语句后，
程序下一个执行语句是？



goto 用法示例（1-100的求和）

```
1 #include <stdio.h>
2
3 int main(){
4     int i=1,sum=0;
5     loop:
6     if(i<=100) {
7         sum+=i;
8         i++;
9         goto loop;
10    }
11    printf("%d\n",sum);
12
13    return 0;
14 }
```

仅仅只是语句标号！



建议尽量少地使用goto语句



使用goto语句的一种情况

● 从包含switch语句的循环中退出

由于break语句在此只能跳出switch，无法跳出外层循环。

举例

账簿余额维护程序，程序将为用户提供选择菜单，清空账户余额，往账户上存钱，从账户上取钱，显示当前余额，退出程序。



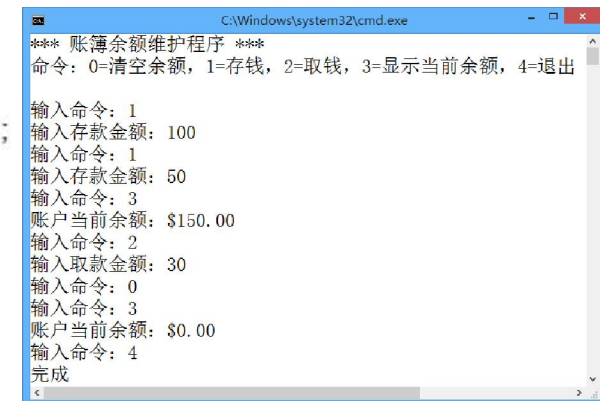
●示例：从包含switch语句的循环中退出

```
#include <stdio.h>
int main()
{
    int cmd = 0;
    float balance = 0.0f, credit, debit;
    printf("*** 账簿余额维护程序 ***\n");
    printf("命令: 0=清空余额, 1=存钱, 2=取钱, ");
    printf("3=显示当前余额, 4=退出\n\n");

    while(1)
    {
        printf("输入命令: ");
        scanf("%d", &cmd);
        switch(cmd)
        {
            case 0:
                balance = 0.0f;
                break;
            case 1:
                printf("输入存款金额: ");
                scanf("%f", &credit);
                balance += credit;
                break;
```

```
            case 2:
                printf("输入取款金额: ");
                scanf("%f", &debit);
                balance -= debit;
                break;
            case 3:
                printf("账户当前余额: $%.2f\n", balance);
                break;
            case 4:
                goto loop_done; //退出大循环
            default:
                printf("命令: 0=清空余额, 1=存钱, 2=取钱, ");
                printf("3=显示当前余额, 4=退出\n\n");
                break;
        }
    }
    loop_done:printf("完成\n");
```

输入4
执行goto命令
退出程序



```
C:\Windows\system32\cmd.exe
*** 账簿余额维护程序 ***
命令: 0=清空余额, 1=存钱, 2=取钱, 3=显示当前余额, 4=退出
输入命令: 1
输入存款金额: 100
输入命令: 1
输入存款金额: 50
输入命令: 3
账户当前余额: $150.00
输入命令: 2
输入取款金额: 30
输入命令: 0
输入命令: 3
账户当前余额: $0.00
输入命令: 4
完成
```

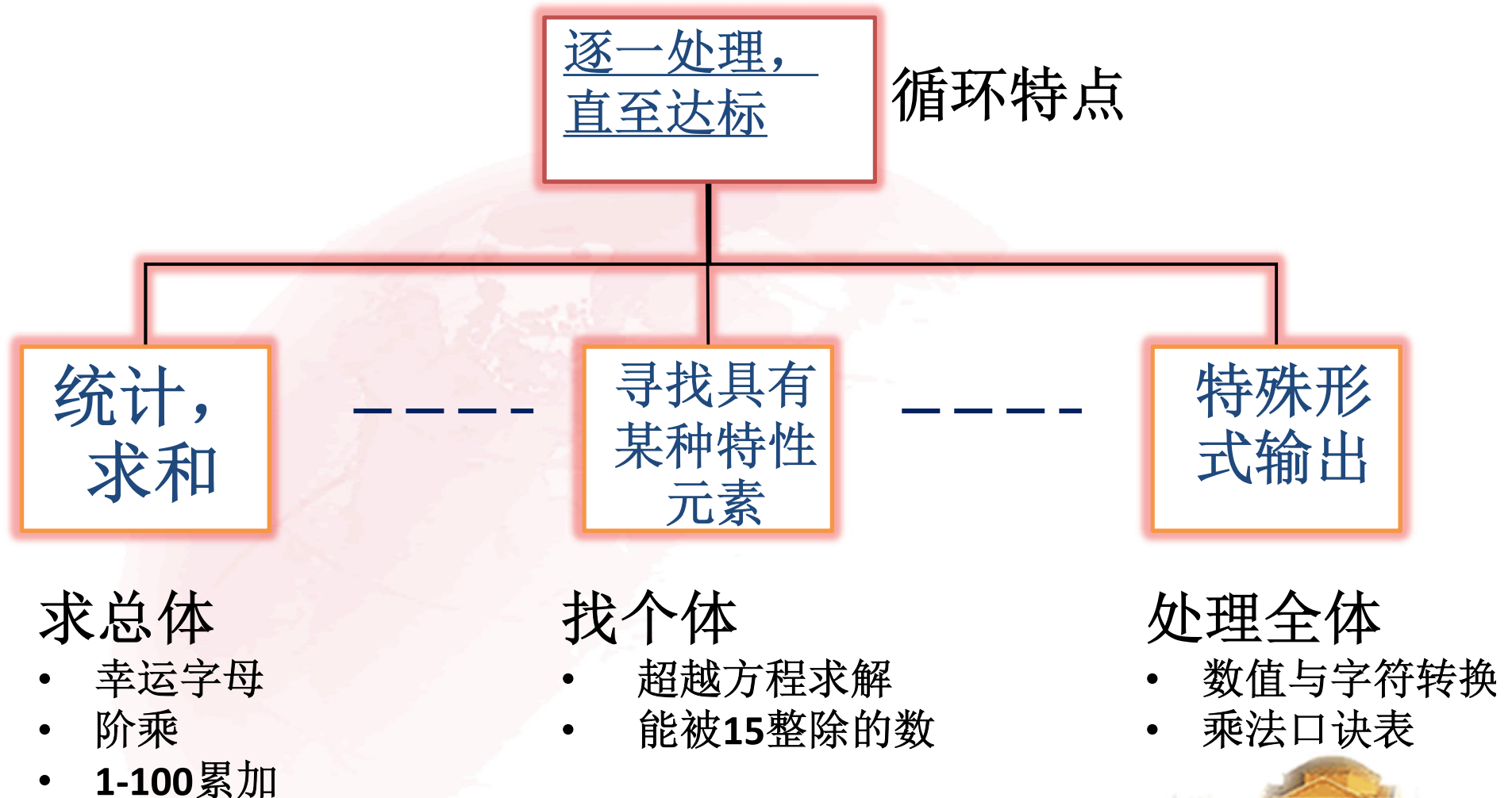

尽量少用goto语句

- goto是条件跳转指令，可以使程序的执行跳转到另一个位置，因此它会破坏其它的控制流机制（如FOR, IF SWITCH)所提供的有用结构，从而造成难以阅读的“垃圾代码”。由于goto语句既可以往前跳又可以往后跳，所以使得程序难以阅读。
- 理论上，goto语句是可以通过顺序、选择和循环结构来避免的。

——《A Case against the GOTO Statement filetype》



5.4 循环结构小结



综合示例：寻找30以内的质数

- 算法思考

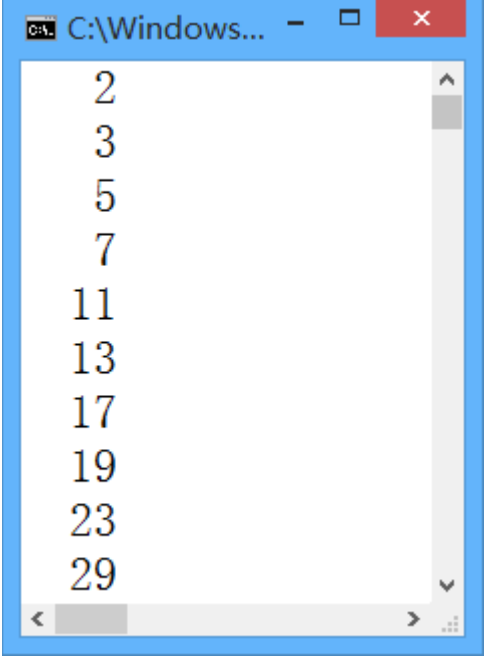
- **质数定义：**只能被1或者自身整除的自然数（不包括1），称为质数。
- 利用它的可以循环判断该数除以比它小的每个自然数（大于1），如果有能被它整除的，则它就不是质数。



综合示例：寻找30以内的质数

```
#include <stdio.h>

int main(){
    int num,i,count = 0;
    int N=30;
    for (num = 2; num <= N; num++) {
        for (i = 2; i < num; i++)
            if (num % i == 0)
                break;
        if (i == num)
            printf("%4d\n",num);
    }
    return 0;
}
```



A screenshot of a Windows command prompt window titled "C:\Windows...". The window displays the output of the program, which lists prime numbers from 2 to 29, each on a new line and right-aligned. The numbers are: 2, 3, 5, 7, 11, 13, 17, 19, 23, and 29. The window has a standard Windows title bar with minimize, maximize, and close buttons.

*算法的艺术

寻找30以内的质数

- 其他算法思考

- **定理：**如果一个数是合数，那么它的最小质因数肯定小于等于他的平方根。
 - 例如：50，最小质因数是2， $2 < 50$ 的开根号
- 即：如果一个数能被它的最小质因数整除的话，那它肯定是合数，即不是质数。所以判断一个数是否是质数，只需判断它是否能被小于它开根后的所有数整除，这样做的运算就会少了很多，因此效率也高了很多。

其他方法：筛法求质数（有兴趣自行学习）



应用1：求总体

上机练习题1

- 将英语**26**个字母由**A**到**Z**分别编上**1**到**26**的分数
 - 你的知识(**KNOWLEDGE**)只能得到
 $11+14+15+23+12+5+4+7+5=96$ 分；
 - 你努力工作(**HARDWORK**)也只能得到
 $8+1+18+4+23+15+18+11=98$ 分；
 - 只有你的态度(**ATTITUDE**)才是左右你生命全部的
 $1+20+20+9+20+21+4+5=100$ 分。
- 试编程验证



应用2：找个体

上机练习题2

- 编写程序，提示输入一个整数N，然后显示出1~N的所有偶数平方数。如输入100，则应输出：

4

16

36

64

100

- 请画出该程序的流程图，手绘/不凌乱即可。
不需要提交。

如果上机验证发现有误，请重新绘制流程图之后提交。

应用3：处理全体

上机练习题3

- 编写程序，显示本月的月历。首先提示本月的天数和本月1号是周几？

提示统一为：

请输入本月的天数？ 31

请输入1号是周几？ 2

排出这个月历：

	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

- 提示：最重要的是使用变量*i*从1计数到N的for语句（N为本月天数），显示*i*，并判断*i*是不是一个星期的最后一天（周日），若是则输出一个换行符。
- 提醒：日期位数有1位也有2位，注意输出对齐，参考乘法口诀表

