哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

规格严格 功夫到家

第6章 循环控制结构

哈尔滨工业大学

计算机科学与技术学院

苏小红

sxh@hit.edu.cn

本章学习内容

- 计数控制的循环
- 条件控制的循环
- for语句，while语句，do-while语句
- continue语句，break语句
- 嵌套循环
- 结构化程序设计的基本思想
- 程序调试与排错

问题的提出

Example:

Draw a flowchart for the following problem:
读入5个整数，计算并显示它们的和.

Input : 5 个整数n1, n2, n3, n4, n5

Output: n1, n2, n3, n4, n5的和

Input example: 2 3 4 5 6

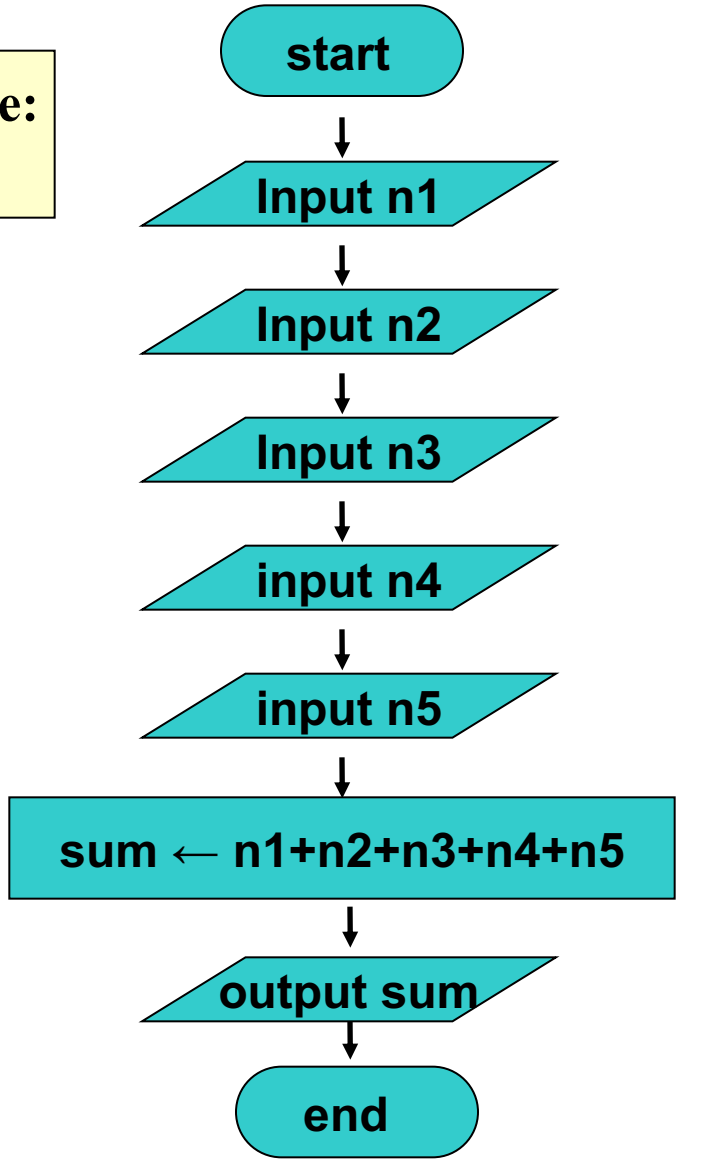
Output example: 20

如何确定程序的输入和输出呢？

问题的提出

Assume input example:
2 3 4 5 6

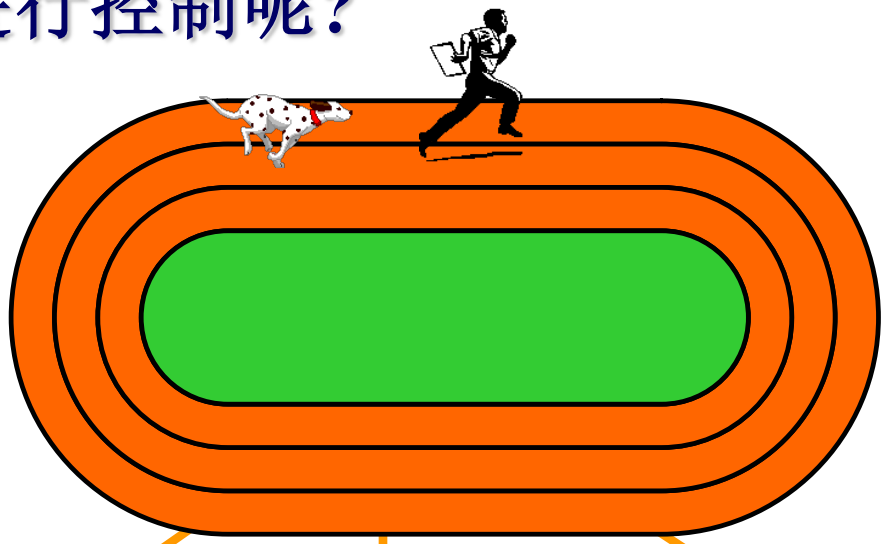
使用了6个不同的变量



n1	2
n2	3
n3	4
n4	5
n5	6
sum	20

6.1 循环控制结构与循环语句

如何对循环进行控制呢？



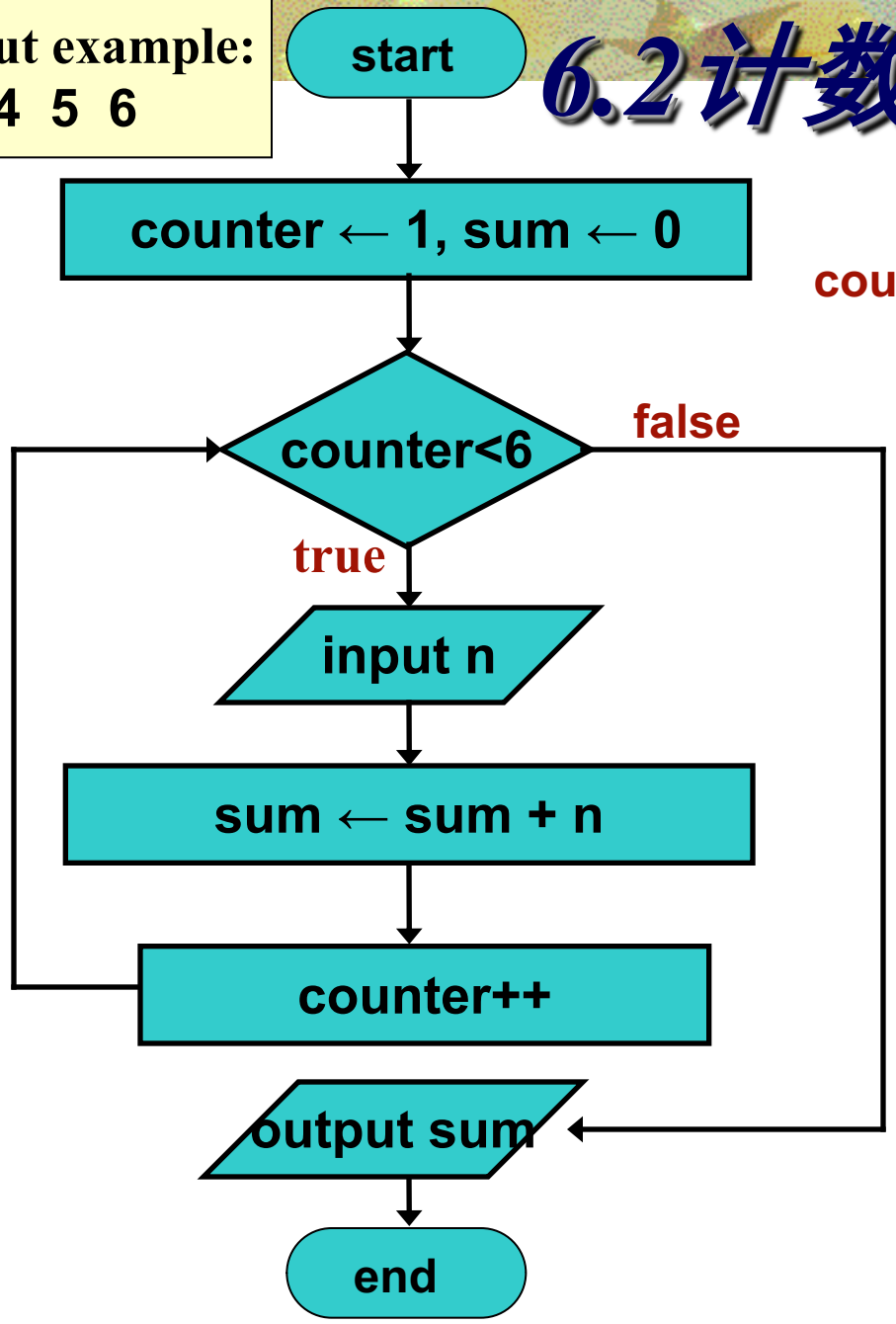
条件控制
Condition Controlled

计数控制
Counter Controlled
- 1, 2, 3, 4, …
- …, 4, 3, 2, 1

标记控制
Sentinel Controlled

6.2计数控制的循环

Assume input example:
2 3 4 5 6



counter 6

sum 20

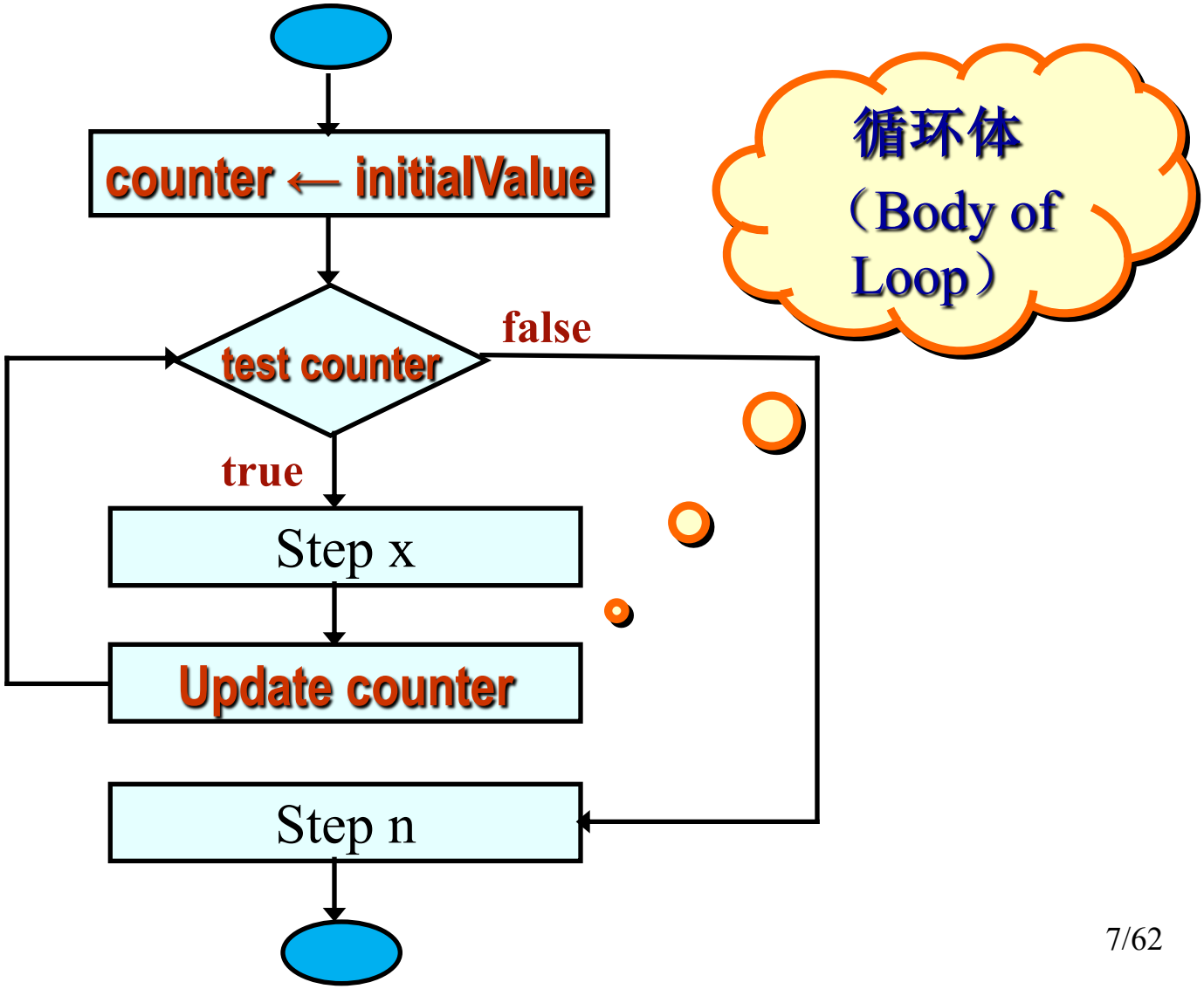
n 6

counter-controlled

计数器每次增1

使用了3个变量

6.2 计数控制的循环


counter ← initialValue
test counter
false
true
Step x
Update counter
Step n
循环体
（Body of Loop）

for循环语句

- 当型循环——Condition is tested first
- 计数控制loop is controlled by a count
- Syntax

循环起始条件　循环结束条件　循环变量增值

```
for (initial value ;  condition;  update counter)
        statement;
```

Or

```
for (initial value ;  condition;  update counter)
{
        statement;
        statement;
}
```

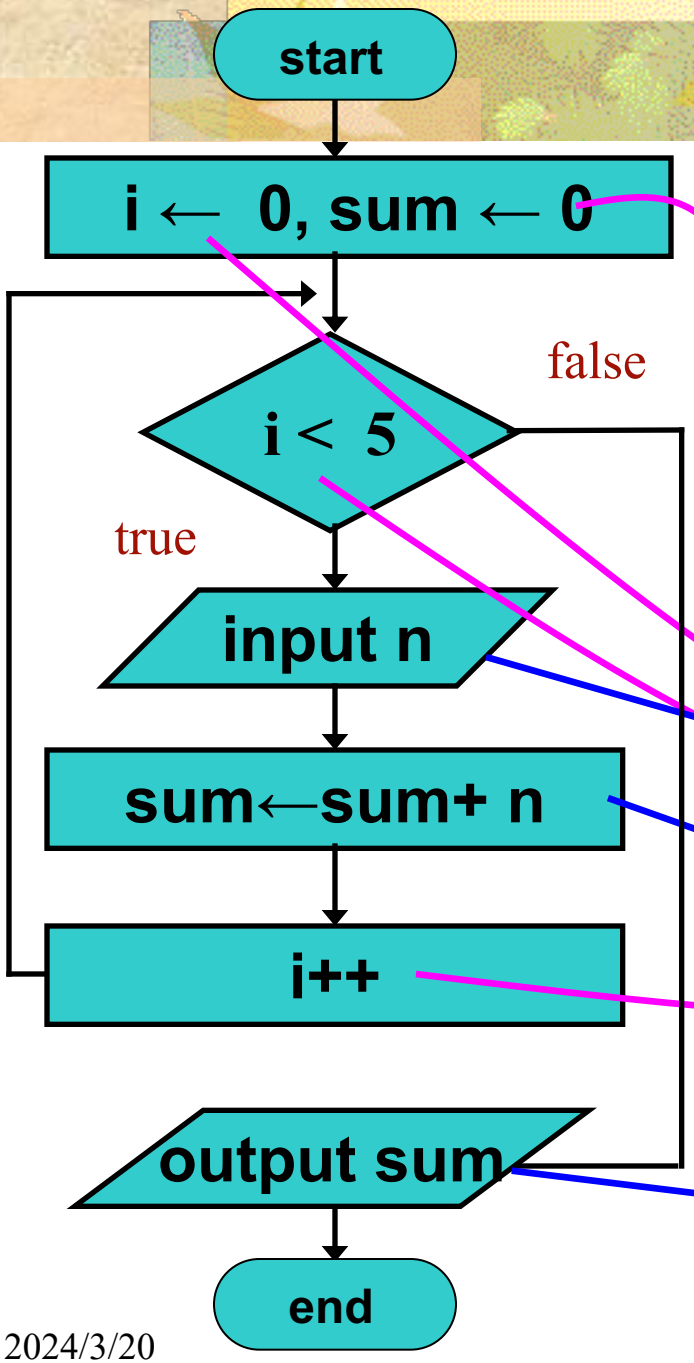
循环变量控制循环次数，不要在循环体内改变这个变量的值

复合语句

compound statement

被当作一条语句看待

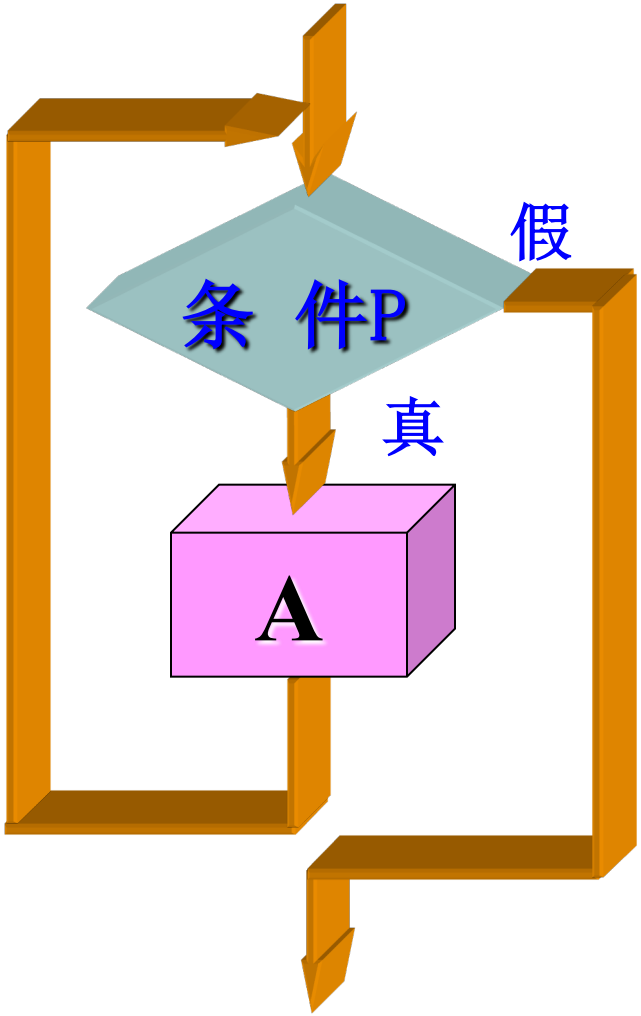
for循环语句



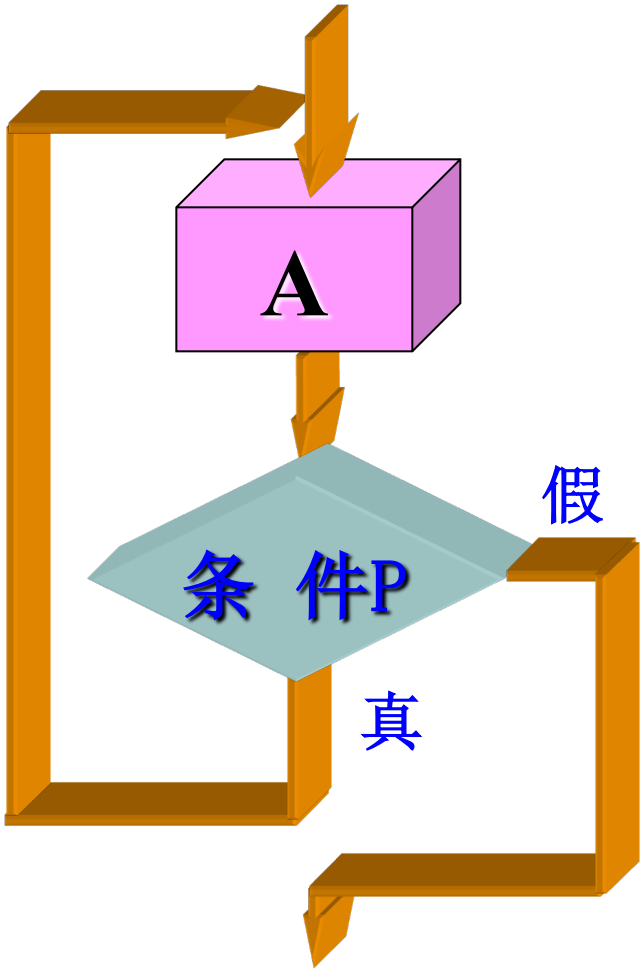

```
int i, sum, n;
sum = 0;
for (i = 0; i < 5; i++)
{
    scanf("%d", &n);
    sum = sum + n;
}
printf("%d", sum);
```

6.3 条件控制的循环

当型循环



直到型循环



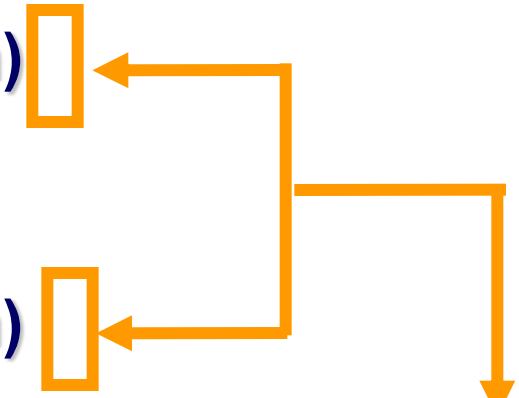
while循环语句

- 当型循环——Condition is tested first
- 条件或计数控制
  ——Loop is controlled by condition or a counter
- 语法

```
while (condition)
        statement;
```

Or

```
while (condition)
{
        statement;
        statement;
}
```



No semicolon!!

do-while循环语句

- 直到型循环——Statements in the loop are executed first (at least once), and condition is tested last
- 条件或计数控制

  —— Loop is controlled by condition or a counter
- 语法

```
do{
   statement;
   statement;
} while (condition);
statement;
```

Don't forget the semicolon!!

【例6.1】计算并输出1+2+3+...+n的值

- 循环次数已知，计数控制的循环

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

```
Input n: 100 ↙
sum = 5050
```

【例6.1】计算并输出1+2+3+...+n的值

- 循环次数已知，计数控制的循环

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

sum = 0的作用?

```
Input n: 10 ↙
i=1,sum=1
i=2,sum=3
i=3,sum=6
i=4,sum=10
i=5,sum=15
i=6,sum=21
i=7,sum=28
i=8,sum=36
i=9,sum=45
i=10,sum=55
sum = 55
```

【例6.1】计算并输出1+2+3+...+n的值

- 循环次数已知，计数控制的循环

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

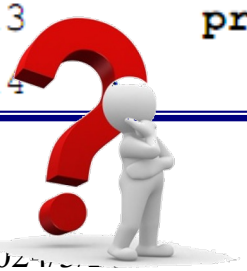
```
Input n: 10 ↙
i=1,sum=1
i=2,sum=3
i=3,sum=6
i=4,sum=10
i=5,sum=15
i=6,sum=21
i=7,sum=28
i=8,sum=36
i=9,sum=45
i=10,sum=55
sum = 55
```

```
Input n: 10 ↙
i=1,sum=4199399
i=2,sum=4199401
i=3,sum=4199404
i=4,sum=4199408
i=5,sum=4199413
i=6,sum=4199419
i=7,sum=4199426
i=8,sum=4199434
i=9,sum=4199443
i=10,sum=4199453
sum = 4199453
```

【例6.1】计算并输出1+2+3……+n的值

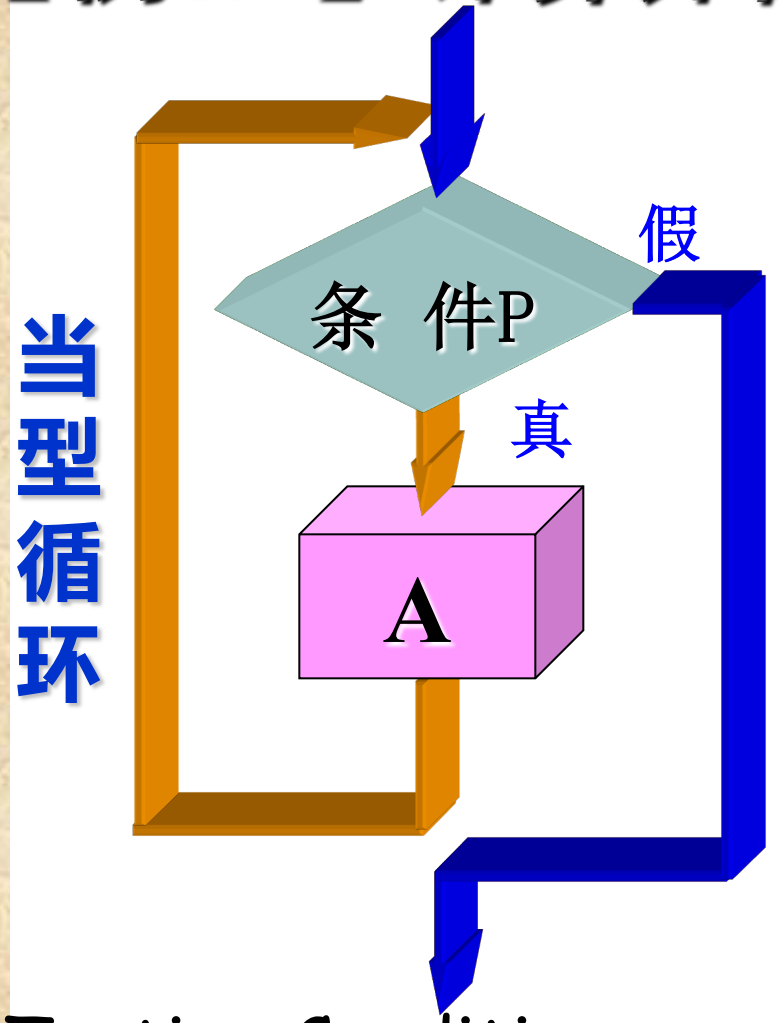
```
1    #include <stdio.h>
2    main()
3    {
4        int  i, n, sum;
5        printf("Input n:");
6        scanf("%d", &n);
7        sum = 0;
8        i = 1;
9        do{
10           sum = sum + i;
11           i++;
12       }while (i <= n);
13       printf("sum = %d\n", sum);
14   }
```

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



循环条件第一次就为假（如输入-1）时会怎样？

【例6.1】计算并输出1+2+3+...+n的值

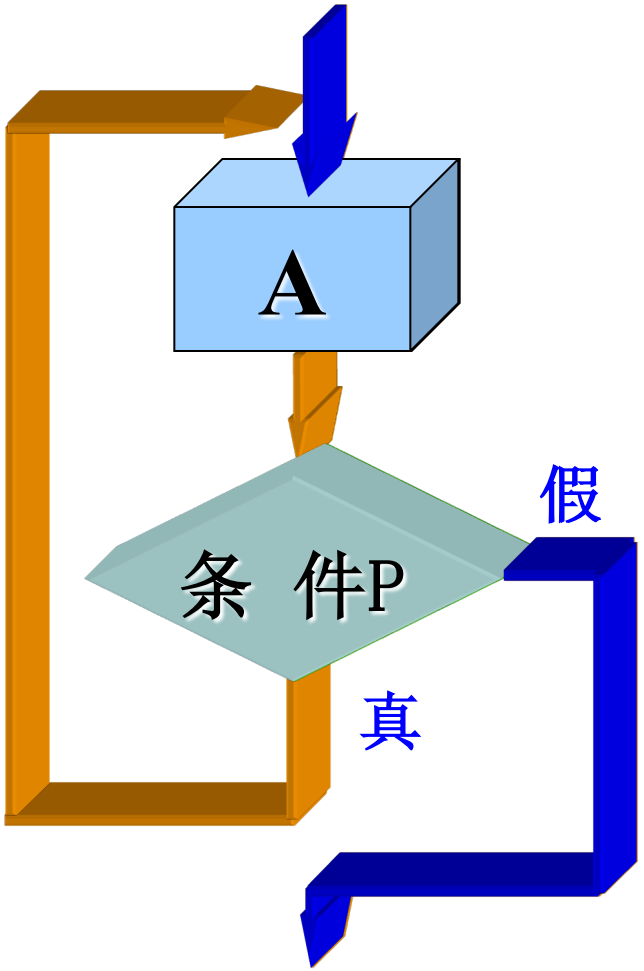



```
1    #include <stdio.h>
2    main()
3    {
4        int  i, n, sum;
5        printf("Input n:");
6        scanf("%d", &n);
7        sum = 0;
8        i = 1;
9        while (i <= n)
10       {
11           sum = sum + i;
12           i++;
13       }
14       printf("sum = %d\n", sum);
15   }
```

Testing Condition First

【例6.1】计算并输出1+2+3+...+n的值

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



直到型循环

Testing condition last

注意

- 在for和while语句之后一般没有分号
- 有分号表示循环体就是分号之前的内容
  - 空语句——表示循环体内什么都不做

```
while (i < 100);
    i++;
```

  - 死循环

```
for (i = 0; i < 100; i++);
    printf("%d", i);
```

  - 用于延时

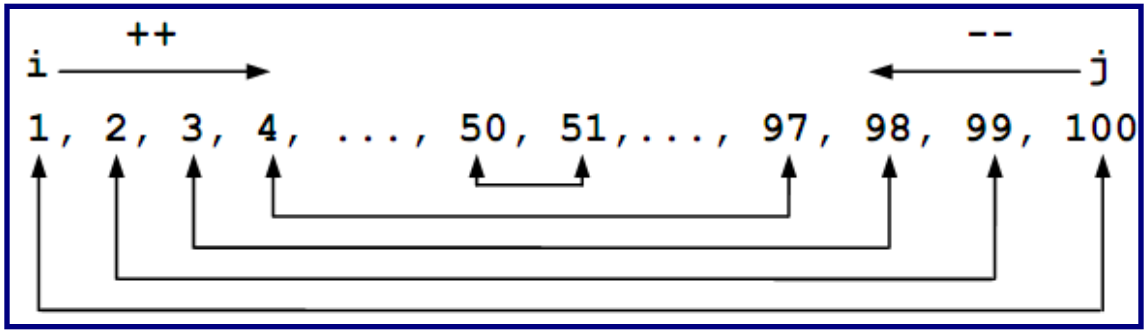
【例6.1】计算并输出1+2+3+...+n的值

- 如何减少循环的次数？

```
1     #include <stdio.h>
2     main()
3     {
4           int  i, j, n, sum = 0;
5           printf("Input n:");
6           scanf("%d", &n);
7           for (i=1, j=n; i<=j; i++, j--)
8           {
9               sum = sum + i + j;
10          }
11          printf("sum = %d\n", sum);
12    }
```

【例6.1】计算并输出1+2+3+...+n的值

```
1    #include <stdio.h>
2    main()
3    {
4        int  i, j, n, sum = 0;
5        printf("Input n:");
6        scanf("%d", &n);
7        for (i=1, j=n; i<=j; i++, j--)
8        {
9            sum = sum + i + j;
10       }
11       printf("sum = %d\n", sum);
12   }
```



逗号运算符（Comma Operator）

表达式1，表达式2，…，表达式n

- 多数情况下，并不使用整个逗号表达式的值，更常见的情况是要分别得到各表达式的值
- 主要用在循环语句中，同时对多个变量赋初值等

```
for (i = 1 , j = 100; i < j; i++, j--)
```

循环起始条件　　循环结束条件　　循环变量增值

【例6.2】计算并输出

$$n! = 1 \times 2 \times 3 \times \ldots \times n$$

```
1     #include <stdio.h>
2     main()
3     {
4         int   i, n;
5         long  p = 1;
6         printf("Please enter n:");
7         scanf("%d", &n);
8         for (i=1; i<=n; i++)
9         {
10            p = p * i;
11        }
12        printf("%d! = %ld\n", n, p);
13    }
```

```
Please enter n:10↙
10! = 3628800
```

【例6.3】计算并输出 1!, 2!, 3!, … , n!

```
1     #include <stdio.h>
2     main()
3     {
4         int   i, n;
5         long  p = 1;
6         printf("Please enter n:");
7         scanf("%d", &n);
8         for (i=1; i<=n; i++)
9         {
10            p = p * i;
11            printf("%d! = %ld\n", i, p);
12        }
13    }
```

```
Please enter n:10↙
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

【例6.4】输入n值，计算并输出 $1!+2!+3!+\ldots+n!$

```
1   #include <stdio.h>
2   main()
3   {
4       int i, n;
5       long sum = 0, term = 1;
6       printf("Input n:");
7       scanf("%d", &n);
8       for (i=1; i<=n; i++)
9       {
10          term = term * i;
11          sum = sum + term;
12      }
13      printf("1!+2!+…+%d! = %ld\n", n, sum);
14  }
```


```
Input n: 10↙
1!+2!+…+10! = 4037913
```

【例6.4】输入n值，计算并输出 1!+2!+3!+…+n!

```
1  #include <stdio.h>
2  main()
3  {
4      int i, j, n;
5      long term, sum = 0;
6      printf("Input n:");
7      scanf("%d", &n);
8      for (i=1; i<=n; i++)
9      {
10         term = 1;
11         for (j=1; j<=i; j++)
12         {
13             term = term * j;
14         }
15         sum = sum + term;
16     }
17     printf("1!+2!+…+%d! = %ld\n", n, sum);
18 }
```

每次单独计算累加项

```
Input n: 10↙
1!+2!+…+10! = 4037913
```

6.3嵌套循环

- 使用嵌套循环的注意事项
- 使用复合语句，以保证逻辑上的正确性
  - 即用一对花括号将各层循环体语句括起来
- 内层和外层循环控制变量不能同名，以免造成混乱
- 采用右缩进格式书写，以保证层次的清晰性

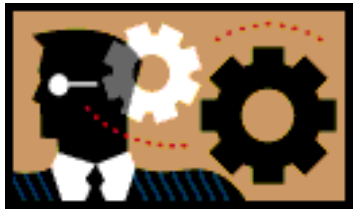
选择三种循环的一般原则

- 如果循环次数已知，计数控制的循环
  - 用**for**
- 如果循环次数未知，条件控制的循环
  - 用**while**
- 如果循环体至少要执行一次
  - 用**do-while**
- 这只是“一般”原则，不是“原则”

6.4 条件控制的循环
——例6.6：猜数游戏

想一个1~100之间的数

猜对: right!
猜错:wrong!并提示大小



循序渐进式编程：猜数游戏

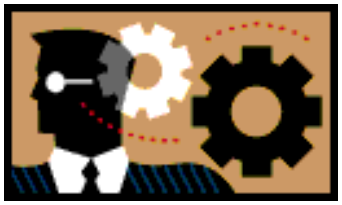
——例6.6,例6.7,例6.8 ,例6.9 ,例6.10

猜多个数
10次猜不对
就猜下一个数

最多猜10次

直到猜对为止

只猜1次



猜数游戏用到的库函数

- 怎样模拟计算机“想”一个数呢?
- 随机函数rand()
- 产生[0,RAND_MAX] 之间的随机数
  - magic = rand();
- #include <stdlib.h>
  - RAND_MAX在stdlib.h中定义，不大于双字节整数的最大值32767
- 产生[0,b-1] 之间的随机数
  - magic = rand()%b;
- 产生[a,a+b-1] 之间的随机数
  - magic = rand()%b + a;

只猜1次

例6.6

```
#include  <stdlib.h>
#include  <stdio.h>
main()
{
   int  magic;          /*计算机"想"的数*/
   int  guess;          /*人猜的数*/

   magic = rand()%100 + 1;    /*"想"一个[1，100]之间的数magic*/

   printf("Please guess a magic number:");
   scanf("%d", &guess);

   if (guess > magic)
   {
        printf("Wrong! Too high!\n");
   }
   else if (guess < magic)
   {
        printf("Wrong! Too low!\n");
   }
   else
   {
       printf("Right! \n");
       printf("The number is:%d \n", magic);
   }
}
```

直到猜对为止

例6.7

```
#include  <stdlib.h>
#include  <stdio.h>
main()
{
   int  magic;
   int  guess;
   int  counter;                  /*记录人猜次数的计数器变量*/

   magic = rand() % 100 + 1;

   counter = 0;                   /*计数器变量count初始化为0*/
   do{
       printf("Please guess a magic number:");
       scanf("%d", &guess);
       counter ++;                          /*计数器变量count加1*/
       if (guess > magic)
           printf("Wrong! Too high!\n");
       else if (guess < magic)
           printf("Wrong! Too low!\n");
       else
           printf("Right!\n");
   }while (guess != magic);

   printf("counter = %d \n", counter);
}
```

猜数游戏用到的库函数

- 每次运行程序时计算机所“想”的数都是一样的，这是什么原因呢?
  - 函数rand()产生的只是伪随机数
- 随机函数srand
  - 为函数rand()设置随机数种子来实现对函数rand所产生的伪随机数的“随机化”
- 通过输入随机数种子，产生[0,100]之间的随机数

```
scanf("%u", &seed);
srand(seed);
magic = rand() % 100 + 1;
```

直到猜对为止

例6.7

```
#include  <stdlib.h>
#include  <stdio.h>
main()
{
   int  magic;
   int  guess;
   int  counter;                    /*记录人猜次数的计数器变量*/
   unsigned int seed;
   printf("Please enter seed:");
   scanf("%u", &seed);
   srand(seed);
   magic = rand() % 100 + 1;
   counter = 0;                     /*计数器变量count初始化为0*/
   do{
       printf("Please guess a magic number:");
       scanf("%d", &guess);
       counter ++;                          /*计数器变量count加1*/
       if (guess > magic)
           printf("Wrong! Too high!\n");
       else if (guess < magic)
           printf("Wrong! Too low!\n");
       else
           printf("Right!\n");
   }while (guess != magic);
   printf("counter = %d \n", counter);
}
```

猜数游戏用到的库函数

- 使用计算机读取其时钟值并把该值自动设置为随机数种子，产生[0,100]之间的随机数
- 函数time()返回以秒计算的当前时间值，该值被转换为无符号整数并用作随机数发生器的种子

```
#include  <time.h>

srand(time(NULL));
magic = rand() % 100 + 1;
```

- 函数time()能为程序员提供代表时间的字符串，使用NULL作为函数参数，使其不具备此功能

直到猜对为止 例6.7

```c
#include  <stdlib.h>
#include  <stdio.h>
#include  <time.h>
main()
{
   int  magic;
   int  guess;
   int  counter;                    /*记录人猜次数的计数器变量*/

   srand(time(NULL));
   magic = rand() % 100 + 1;

   counter = 0;                     /*计数器变量count初始化为0*/
   do{
       printf("Please guess a magic number:");
       scanf("%d", &guess);
       counter ++;                          /*计数器变量count加1*/
       if (guess > magic)
           printf("Wrong! Too high!\n");
       else if (guess < magic)
           printf("Wrong! Too low!\n");
       else
           printf("Right!\n");
   }while (guess != magic);
   printf("counter = %d \n", counter);
}
```

最多猜10次

例6.8

```
#include  <stdlib.h>
#include  <stdio.h>
#include  <time.h>
main()
{
   int  magic;
   int  guess;
   int  counter;                  /*记录人猜次数的计数器变量*/

   srand(time(NULL));
   magic = rand() % 100 + 1;

   counter = 0;                   /*计数器变量count初始化为0*/
   do{
       printf("Please guess a magic number:");
       scanf("%d", &guess);
       counter ++;                          /*计数器变量count加1*/
       if (guess > magic)
           printf("Wrong! Too high!\n");
       else if (guess < magic)
           printf("Wrong! Too low!\n");
       else
           printf("Right!\n");
   }while (guess != magic && counter < 10);
   printf("counter = %d \n", counter);
}
```

防止非法字符输入

例6.9

```c
#include  <stdlib.h>
#include  <stdio.h>
#include  <time.h>
main()
{
   int  magic;
   int  guess;
   int  counter;

   srand(time(NULL));
   magic = rand() % 100 + 1

   counter = 0;
   do{
       printf("Please guess a magic number:");
       scanf("%d", &guess);
       counter ++;                          /*计数器变量count加1*/
       if (guess > magic)
           printf("Wrong! Too high!\n");
       else if (guess < magic)
           printf("Wrong! Too low!\n");
       else
           printf("Right!\n");
   }while (guess != magic && counter < 10);
   printf("counter = %d \n", counter);
}
```

```c
ret = scanf("%d", &guess);
while (ret != 1)
{
    while (getchar() != '\n');
    printf("Please guess a magic number:");
    ret = scanf("%d", &guess);
}
counter ++;
```

猜多个数
10次猜不对就猜下一个数
直到用户选择结束为止

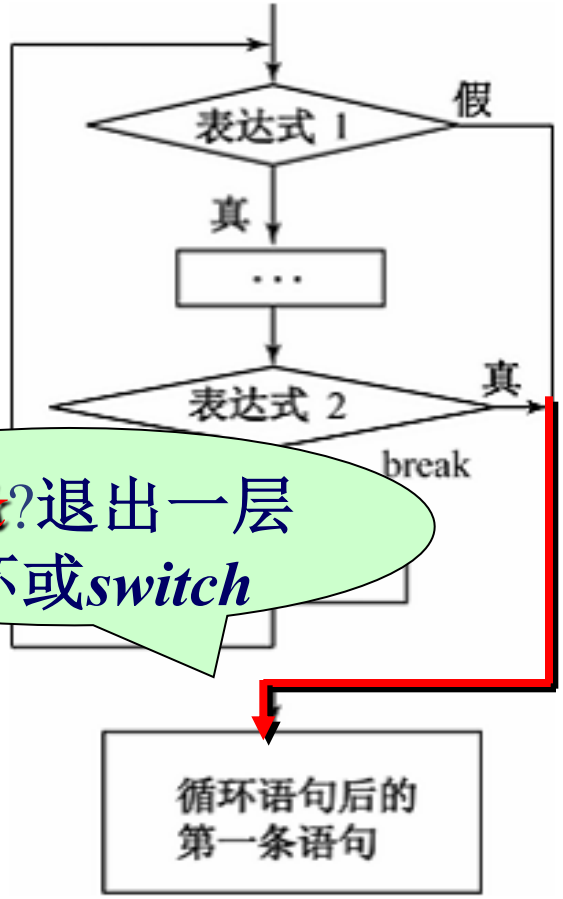
```
do{
        magic = rand() % 100 + 1;
        counter = 0;
        do{
                printf("Please guess a magic number:");
                scanf("%d", &guess);
                counter ++;
                if (guess > magic)
                    printf("Wrong!Too high!\n");
                else if (guess < magic)
                    printf("Wrong!Too low!\n");
                else
                    printf("Right!\n");
        }while (guess != magic && counter < 10);

        printf("counter = %d\n", counter);
        printf("Do you want to continue(Y/N or y/n)?");
        scanf(" %c", &reply);
    }while ((reply == 'Y') || (reply == 'y'));
```

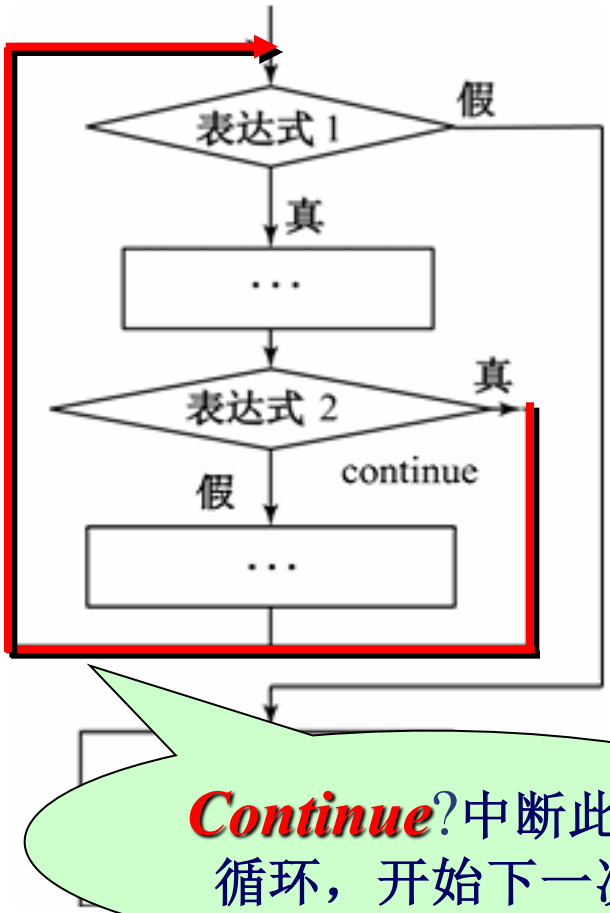
例6.10

6.5流程的转移控制

- break语句 和 continue语句
  - 对for、while、do-while循环进行内部手术



Break?退出一层循环或*switch*



Continue?中断此次循环，开始下一次

【例6.11】演示break与continue

```
#include <stdio.h>
main()
{
  int  i, n;
  for (i=1; i<=5; i++)
  {
     printf("Please enter n:");
     scanf("%d", &n);
     if (n < 0)
          break;
     printf("n = %d\n", n);
  }
  printf("Program is over!\n");
}
```

n

-10

```
Please enter n:10↙
n = 10
Please enter n: -10↙
Program is over!
```

【例6.12】演示break与continue

```
#include <stdio.h>
main()
{
  int  i, n;
  for (i=1; i<=5; i++)
  {
     printf("Please enter n:");
     scanf("%d", &n);
     if (n < 0)
          continue;
     printf("n = %d\n", n);
  }
  printf("Program is over!\n");
}
```

n

30

```
Please enter n:10↙
n = 10
Please enter n: -10↙
Please enter n:20↙
n = 20
Please enter n: -20↙
Please enter n:30↙
n = 30
Program is over!
```

goto语句与语句标号

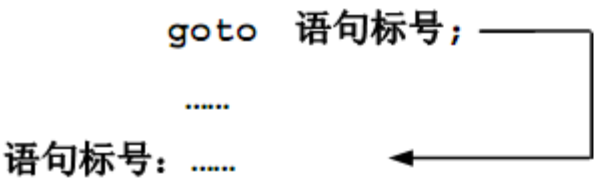
- 语句标号（Label）举例

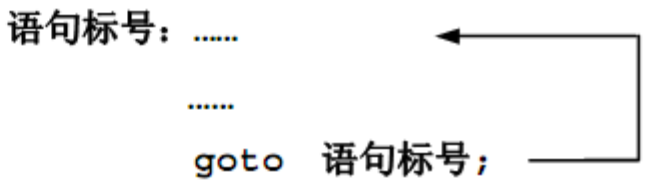
```
error:
```

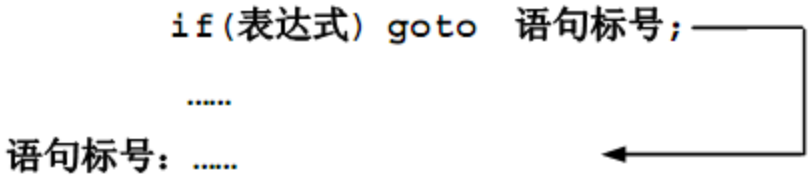
- goto语句举例

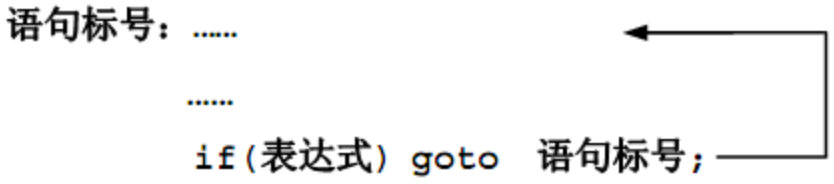
```
goto error;
```

- 一般形式









【例6.13】韩信点兵

- 韩信有一队兵，他想知道有多少人，便让士兵排队报数。按从1至5报数，最末一个士兵报的数为1；按从1至6报数，最末一个士兵报的数为5；按从1至7报数，最末一个士兵报的数为4；最后再按从1至11报数，最末一个士兵报的数为10。你知道韩信至少有多少兵吗?
  - 设兵数为x，则x应满足：

```
x%5==1 && x%6==5 && x%7==4 &&
  x%11==10
```

  - 穷举法，对x从1开始试验

【例6.13】韩信点兵

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

   for (x=1; x < 5000 ;x++)
   {
       if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
       {
             printf("x = %d\n", x);
       }
   }
}
```

【例6.13】韩信点兵

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

   for (x=1; ;x++)
   {
       if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
       {
             printf("x = %d\n", x);
       }
   }
}
```

【例6.13】韩信点兵——goto

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

   for (x=1; ;x++)
   {
       if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
       {
             printf("x = %d\n", x);
             goto END;
       }
   }
    END:   ;
}
```

【例6.13】韩信点兵——break

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

   for (x=1; ;x++)
   {
       if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
       {
             printf("x = %d\n", x);
             break;
       }
   }
}
```

【例6.13】韩信点兵——break

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

   for (x=1; ;x++)
   {
       if (x%5==1 && x%6==5 && x%7==4 &&
       {
             printf("x = %d\n", x);
             exit(0);
       }
   }
}
```

标准库函数，作用是终止整个程序的执行，强制返回操作系统

【例6.13】韩信点兵——标志变量

```
#include <stdio.h>
main()
{
   int  x;
   int  find = 0;          /*置找到标志为假*/

   for (x=1; !find ;x++)
   {
       if (x%5==1 && x%6==5 && x%7==4 && x%11==10)
       {
             printf("x = %d\n", x);
             find = 1; /*置找到标志为真*/
       }
   }
}
```

【例6.13】韩信点兵

```
#include <stdio.h>
main()
{
   int  x = 0;
   int  find = 0;          /*置找到标志变量为假*/
   do{
       x++;
       find = (x%5==1 && x%6==5 && x%7==4 && x%11==10);
   }while (!find);
   printf("x = %d\n", x);
}
```

【例6.13】韩信点兵

```
#include <stdio.h>
main()
{
   int  x = 0;
   do{
       x++;
   }while (!(x%5==1 && x%6==5 && x%7==4 && x%11==10));

   printf("x = %d\n", x);
}
```

6.6本章扩充内容

- 结构化程序设计（Structured Programming）
- 1965年，最早由E.W.Dijkstra在一次国际会议上提出
- 1966年，C.Bohm和G.Jacopini首先证明了：
  - 只用顺序、选择、循环三种基本的控制结构就能够实现任何单入口、单出口的程序
  - 给结构化程序设计奠定了基础
- 1971年，IBM公司的Mills提出：
  - 程序应该只有一个入口和一个出口
  - 进一步补充了结构化程序的规则

6.6本章扩充内容

- 目前，还没有一个严格的定义
  - 1974年，D.Gries教授将已有的对结构化程序设计的不同解释归纳为13种。
- 一个比较流行的定义是：
  - 结构化程序设计是一种进行程序设计的原则和方法
  - 它避免使用goto语句
  - 采用“自顶向下、逐步求精”方法进行程序设计
  - 按照这种原则和方法设计出的程序的特点为：
    - 结构清晰　容易阅读　容易修改　容易验证

```
START_LOOP:
if (fStatusOk)
{
    if (fDataAvaiable)
    {
        i = 10;
        goto MID_LOOP;
    }
    else
    {
        goto END_LOOP;
    }
}
else
{
    for (i = 0; i < 100; i++)
    {
MID_LOOP:
        // lots of code here
    }
    goto START_LOOP;
}
END_LOOP:
```

糟糕的goto语句

用goto语句跳向共同的出口位置

```
void Init(void)
{
   char *p1 = NULL;
   char *p2 = NULL;
   char *p3 = NULL;

   p1 = (char*)malloc(256);
   if (p1 == NULL)
        goto Exit;

   p2 = (char*)malloc(256);
   if (p2 == NULL)
        goto Exit;

   p3 = (char*)malloc(256);
   if (p3 == NULL)
        goto Exit;

   /*正常处理的代码*/

Exit:
   if (p1 != NULL)
        free(p1);
   if (p2 != NULL)
        free(p2);
   if (p3 != NULL)
        free(p3);

   return;
}
```

Evil goto's ? Maybe Not...
凌波微步，未必摔跤

- 现代观点认为：
  - 混乱根源不在goto语句，而在语句标号
  - 任何程序都可以不用goto语句就实现其功能
  - 但在某些情况下，使用goto语句可使程序更清晰
- 两种适合使用goto语句的情况
  - 跳向共同的出口位置，进行退出前的处理工作
  - 跳出多重循环的一条捷径
  - {…

```
        {…
            {…
                goto error;
            }
        }
}
```

结构化程序设计关注的焦点

- 不能简单地认为
  - 避免使用goto语句的程序设计方法就是结构化程序设计方法
- 结构化程序设计关注的焦点
  - 程序结构的好坏
  - 有无goto语句，并不是程序结构好坏的标志
  - 限制和避免使用goto语句，只是得到结构化程序的一种手段，而不是目的

结构化程序设计的核心思想

- 采用顺序、选择和循环三种基本结构作为程序设计的基本单元
  - 只有一个入口
  - 只有一个出口
  - 无死语句，即不存在永远都执行不到的语句
  - 无死循环，即不存在永远都执行不完的循环
- 采用“自顶向下、逐步求精”和模块化的方法进行结构化程序设计
- **Top-down, Stepwise refinement**
  - **1971年，wirth提出**
  - 先全局后局部，先整体后细节，先抽象后具体

使用goto语句的原则

- 主张少用、慎用，而不是禁用
- 保证使用之后，程序仍然是单入口，单出口
- 不要使用一个以上的标号
- 不要用goto语句往回跳，要向下跳
- 不要让goto语句制造出永远不会被执行的代码

Questions and answers