程序控制结构-循环结构

车万翔

哈尔滨工业大学

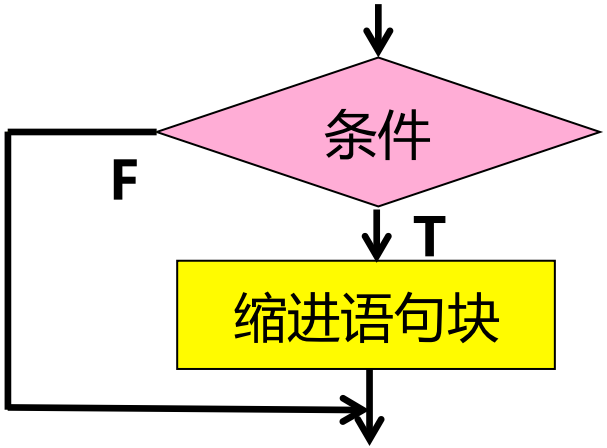
多次求解一元二次方程

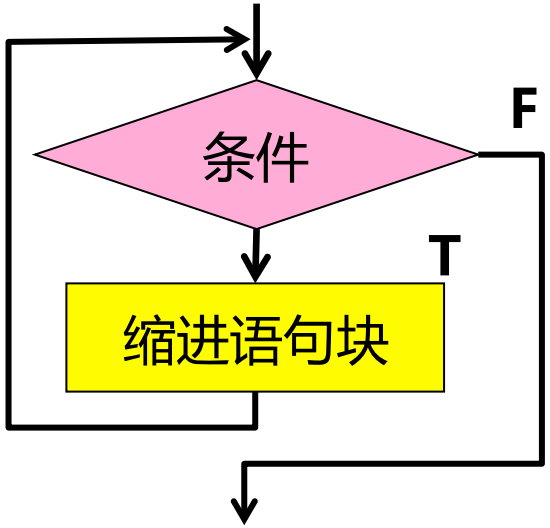
- a=0, b=1, c=1
- a=1, b=2, c=1
- a=1, b=3, c=1
- a=1, b=1, c=1
- a=-1, b=-1, c=-1

- 程序可以多次计算（输入字符 'q' 退出程序，输入其它字符则继续执行）。

循环结构

while 循环结构





while 循环继续条件：

　　缩进语句块（循环体）

其余语句

while 循环结构的分析策略

- 循环体外设定循环可执行的初始条件
- 书写需重复执行的代码（循环体）
- 设定循环条件并在循环体内设定条件改变语句

- 打印字符串 5 次

```
count = 0

while count < 5:
    print 'Programming is fun!'
    count += 1
```

循环执行过程

生成 count 变量，值为 0

```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值为 0，小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

打印字符串

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值加 1，结果为 1

循环执行过程

```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值为 1，小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

打印字符串

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值加 1，结果为 2

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值为 2，小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

打印字符串

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值加 1，结果为 3

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值为 3，小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

打印字符串

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值加 1，结果为 4

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值为 4，小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

打印字符串

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

count 值加 1，结果为 5

循环执行过程


```
 8 count = 0
 9
10 while count < 5:
11     print 'Programming is fun!'
12     count += 1
13
```

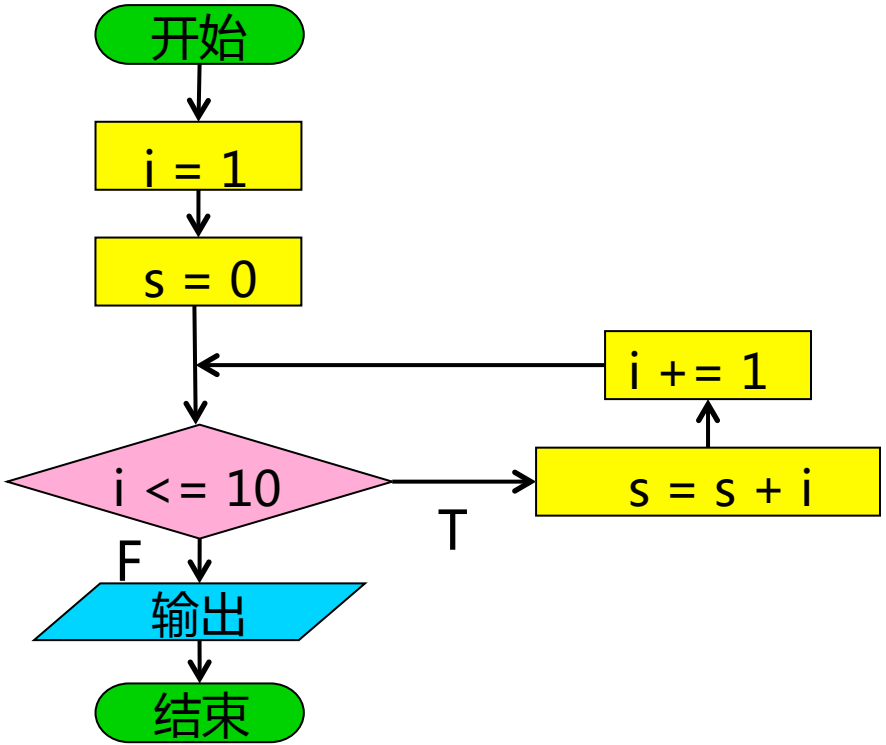
count 值为 5，不小于 5

循环执行过程


```
8  count = 0
9
10 while count < 5:
11     print 'Programming i
12     count += 1
13
```

继续执行 while 后面的语句

循环示例：1 + 2 + … + 10




```
8  s = 0
9  i = 1
10
11 while i < 10:
12     s += i
13     i += 1
14
15 print 'sum is ', s
16
```

错误代码示例


```
8  count = 0
9
10 while count < 10:
11     print count
12
```

```
8  count = 0
9
10 while count < 10:
11     print count
12     count -= 1
13
```

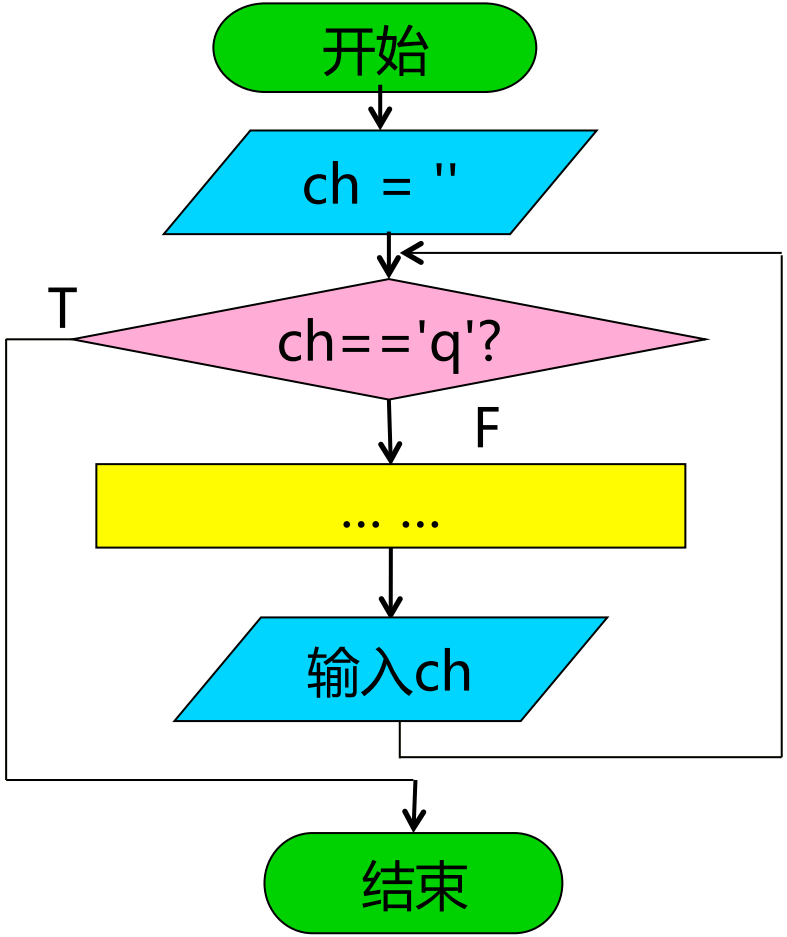
```
8  count = 0
9
10 while count < 10:
11     print count
12 count += 1
13
```

```
8  count = 0
9
10 while count < 10:
11     if count % 2 == 0:
12         print count
13
```

```
8  count = 0
9
10 while count < 10:
11     if count % 2 == 0:
12         print count
13         count += 1
```

```
8  count = 0
9
10 while count < 10:
11     if count % 2 == 0:
12         print count
13     count += 1
14
```

多次求解一元二次方程



代码


```
7  import math
8  ch = 'a'
9  while ch != 'q':
10     a = float(raw_input('Enter coefficient a: '))
11     b = float(raw_input('Enter coefficient b: '))
12     c = float(raw_input('Enter coefficient c: '))
13     if a==0:
14         print 'The equation is linear, not quadratic'
15     else:
16         delta = b ** 2 - 4 * a * c
17         if delta < 0:
18             print 'Without real roots'
19         elif delta == 0: # elif delta <1e-17:
20             print 'Only one root is ',(-b/2.0/a)
21         else:
22             root = math.sqrt(delta)
23             s1 = (-b + root) / (2 * a)
24             s2 = (-b - root) / (2 * a)
25             print 'Two distinct solutions are:', s1, s2
26     ch = raw_input('Please input \'q\' to end or any keys to continue\n')
27
```

break 和 continue 语句


```
 8 count = 0
 9
10 while count < 5:
11     if count > 2:
12         break
13     print 'Programming is fun!'
14     count += 1
15
```

break 结束当前循环体

```
 8 count = 0
 9
10 while count < 5:
11     count += 1
12     if count > 2:
13         continue
14     print 'Programming is fun!'
15
```

continue 结束当次循环

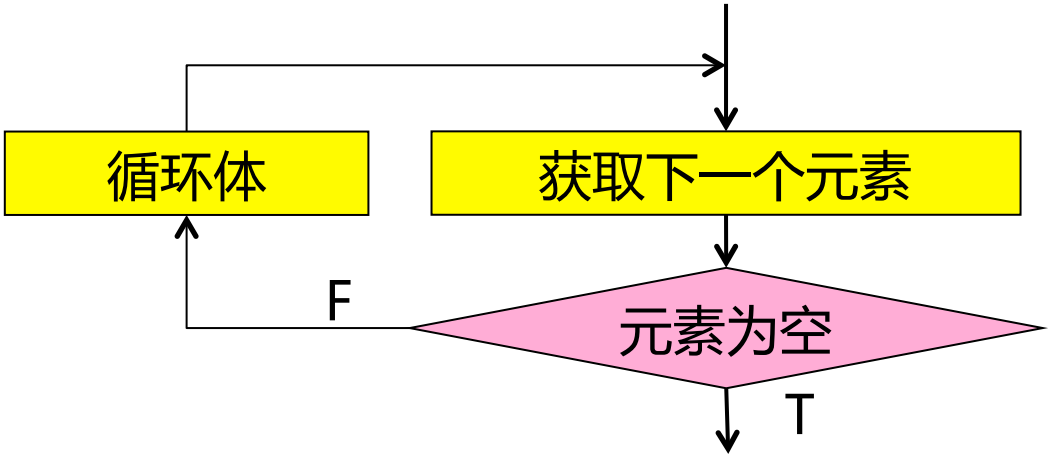
示例


```
7  import math
8  while True:
9      a = float(raw_input('Enter coefficient a: '))
10     b = float(raw_input('Enter coefficient b: '))
11     c = float(raw_input('Enter coefficient c: '))
12     if a==0:
13         print 'The equation is linear, not quadratic'
14     else:
15         delta = b ** 2 - 4 * a * c
16         if delta < 0:
17             print 'Without real roots'
18         elif delta == 0: # elif delta <1e-17:
19             print 'Only one root is ',(-b/2.0/a)
20         else:
21             root = math.sqrt(delta)
22             s1 = (-b + root) / (2 * a)
23             s2 = (-b - root) / (2 * a)
24             print 'Two distinct solutions are:', s1, s2
25     ch = raw_input('Please input \'q\' to end or any keys to continue\n')
26     if ch == 'q':
27         break
28
```

for 循环语句

```
for anElement in object:
    # 缩进语句块（循环体）
```

- 依次遍历对象（object）中的每个元素，并赋值给anElement，然后执行循环体内语句



for 循环语句示例

- 计算1+2+3+...+10的值

range 函数生成 0, 1, ..., 10 序列

```
 8 s = 0
 9
10 for i in range(11):
11     s += i
12
13 print 'sum is:', s
14
```

```
 8 s = 0
 9 i = 1
10
11 while i <= 10:
12     s += i
13     i += 1
14
15 print 'sum is:', s
16
```

range 函数

range

Definition : range(stop)

Type : Function of __builtin__ module

range(stop) -> list of integers range(start, stop[, step]) -> list of integers

Return a list containing an arithmetic progression of integers. range(i, j) returns [i, i+1, i+2, ..., j-1]; start (!) defaults to 0. When step is given, it specifies the increment (or decrement). For example, range(4) returns [0, 1, 2, 3]. The end point is omitted! These are exactly the valid indices for a list of 4 elements.

- range(2, 10) → [2, 3, 4, 5, 6, 7, 8, 9]
- range(2, 10, 3) → [2, 5, 8]
- range(10, 2, -1) → [10, 9, 8, 7, 6, 5, 4, 3]

求常数 e

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \ldots + \frac{1}{i!}$$

```
8  import math
9
10 e = 1
11
12 for i in range(1, 100):
13     e += 1.0 / math.factorial(i)
14
15 print 'e is', e
16
```

```
8  e = 1
9  factorial = 1
10
11 for i in range(1, 100):
12     factorial *= i
13     e += 1.0 / factorial
14
15 print 'e is', e
16
```

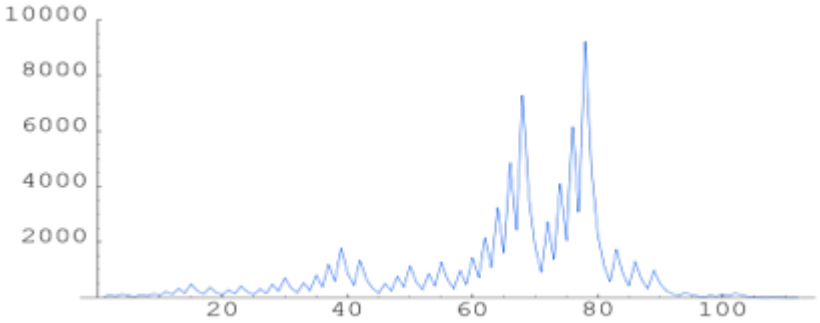
求常数 π

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \ldots + \frac{(-1)^{i+1}}{2i-1}\right)$$

```
 8 pi = 0
 9 sign = 1
10 divisor = 1
11
12 for i in range(1, 1000000):
13     pi += 4.0 * sign / divisor
14     sign *= -1
15     divisor += 2
16
17 print 'pi is', pi
18
```

冰雹猜想（序列）

- 考拉兹猜想（英语：Collatz conjecture），又称奇偶归一猜想，3n + 1 猜想、冰雹猜想、角谷猜想、哈塞猜想、乌拉姆猜想或叙拉古猜想
  - 对于每一个正整数，如果它是奇数，则对它乘 3 再加 1，如果它是偶数，则对它除以 2，如此循环，最终都能够得到 1
  - 如 n = 6，得出序列6, 3, 10, 5, 16, 8, 4, 2, 1



n = 27时的序列分布

嵌套循环

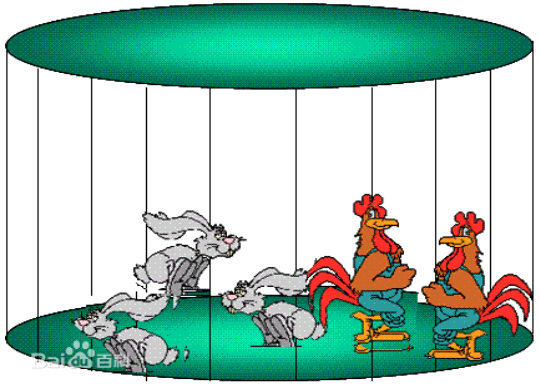
打印乘法表

```
1  2  3  4  5  6  7  8  9
-  -  -  -  -  -  -  -  -
1  2  3  4  5  6  7  8  9
2  4  6  8  10 12 14 16 18
3  6  9  12 15 18 21 24 27
4  8  12 16 20 24 28 32 36
5  10 15 20 25 30 35 40 45
6  12 18 24 30 36 42 48 54
7  14 21 28 35 42 49 56 63
8  16 24 32 40 48 56 64 72
9  18 27 36 45 54 63 72 81
```

```
8 for i in range(1, 10):
9     for j in range(1, 10):
10        print format(i * j, '3'),
11    print
12
```

鸡兔同笼问题

- 鸡兔同笼是中国古代的数学名题之一。大约在1500年前，《孙子算经》中就记载了这个有趣的问题。书中是这样叙述的：“今有雉兔同笼，上有三十五头，下有九十四足，问雉兔各几何？”



- 穷举法

```
for chickens in range(35 + 1):
    for rabbits in range(35 + 1):
        if 2 * chickens + 4 * rabbits == 94 and chickens + rabbits == 35:
            print 'The number of chickens is:', chickens
            print 'The number of rabbits is:', rabbits
            print
```

while vs. for 循环

- **while 循环更通用**
  - 任何 for 循环写的程序都能用 while 循环实现
- **适用场景**
  - for 循环
    - 已知循环的范围（range），即起止值和步长
  - while 循环
    - 其它情况，如：不确定循环何时终止