10 More on Loops

Instructor: Ke Wei（柯韋）

A319 Ext. 6452 wke@ipm.edu.mo

http://brouwer.ipm.edu.mo/COMP112/18/

Bachelor of Science in Computing, School of Public Administration, Macao Polytechnic Institute

October 5, 2018

Outline

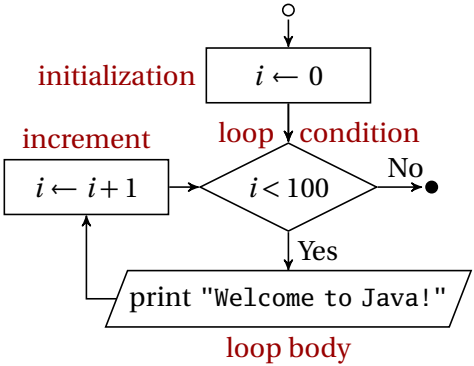
A Typical Counter-Controlled Loop

To print `Welcome to Java!` 100 times, we write a loop like this:

```
// initialization
int i = 0;

while ( i < 100 ) { // loop condition
    // loop body
    System.out.println("Welcome to Java!");

    // increment
    i++;
}
```

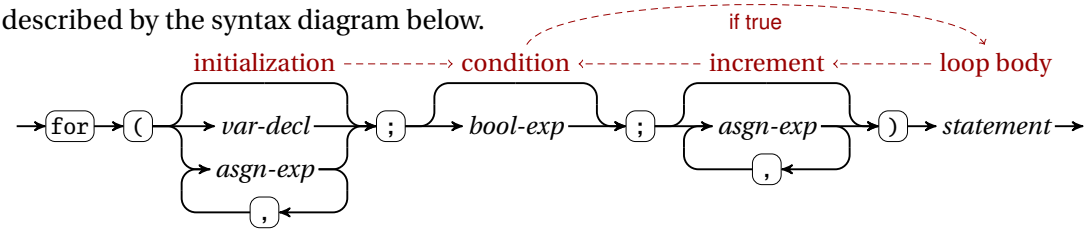


The loop can be formulated in a more concise way using the `for` statement.

```
for ( int i = 0; i < 100; i++ ) System.out.println("Welcome to Java!");
```

for Statement

- The `for` statement provides a more convenient and clearer way to combine the initialization, the condition test and the increment step into one structure, whose form is described by the syntax diagram below.



- The initialization is performed first and only once; next, the condition is evaluated and checked; if `true`, the loop body is executed and the increment is performed after that. The execution comes back to the evaluation and checking of the condition.
- Any of the three parts can be omitted, if the condition is omitted, it is assumed `true`.
- The variables declared in the initialization part are not available outside the loop.

Examples of for Statement

- This loop prints some multiples of 3:

```
for ( int i = 0; i < 10; i++ ) System.out.println(3+"x"+i+" is "+3*i);
// i cannot be used here outside the loop.
```

- This loop computes the series $1+3+5+\cdots+99$:

```
int i, sum;
for ( sum = 0, i = 1; i <= 99; sum += i, i += 2 )
    ;
System.out.println(sum);  // sum can be used here, since it is declared outside the loop.
```

- This loop computes the Fibonacci number of index 80:

```
long a, b;  int n;
for ( a = 0, b = 1, n = 80; n >= 2; n -= 2, a += b, b += a );
```

- The `for` loops are very expressive, especially used with the assignment expression list. Sometimes all the actions are performed in those three parts, with an empty loop body.

Assignment Expressions in `for` Loops

- We may put multiple assignments in the initialization and the increment.

$$e^x = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

```
int i;
double e, f, x = 2.0;

for ( i = 0, e = 0.0, f = 1.0;
      i < 100;
      e += f, ++i, f *= x/i )
    ;

System.out.println("exp("+x+") is "+e);
```

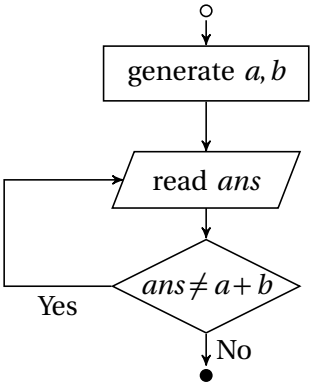
- Although the above style is not unusual, it is a good practice to only put those counters used to control the loop in the three parts of a `for` statement.

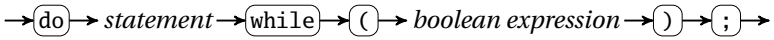
do-while Statement

- Sometimes we need to perform some actions before we can test the loop condition.
- For example, in a quiz, we want to allow the user to try multiple times until the answer is correct.

```
int a = (int)(System.currentTimeMillis()%100);
int b = (int)(System.currentTimeMillis()/7%10);
int ans;

do {
    System.out.print(a+"+"+b+"=? ");
    ans = scanner.nextInt();
} while ( ans != a + b );
```





Which Loop to Use?

- The three forms of loop statements, `while`, `do-while`, and `for`, are expressively equivalent; that is, you can write a loop in any of these three forms.
- Use the one that is most intuitive and comfortable for you.
- In general, a `for` loop may be used if the number of repetitions is known, as, for example, when you need to print a message 100 times.
- A `while` loop may be used if the number of repetitions is not known, as in the case of reading the numbers until the input is 0.
- A `do-while` loop can be used to replace a while loop if the loop body has to be executed before testing the continuation condition.

break and continue in Loops

- To exit the enclosing loop: `break;`
- To continue to the next iteration of the enclosing loop: `continue;`

```
while ( cond1 )
{
    ...
    if ( cond2 )
        break;
    ...
    if ( cond3 )
        continue;
    ...
    last statement
}
```

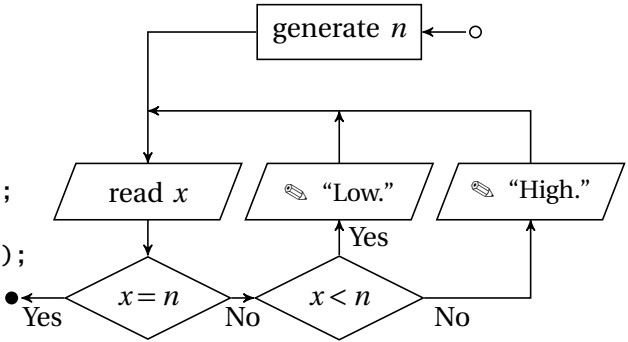
```
do
{
    ...
    if ( cond2 )
        break;
    ...
    if ( cond3 )
        continue;
    ...
    last statement
} while ( cond1 )
```

```
for ( ini1; cond1; inc1 )
{
    ...
    if ( cond2 )
        break;
    ...
    if ( cond3 )
        continue;
    ...
    last statement
}
```

Guessing Numbers

- The program randomly generates an integer between 0 and 100, inclusive.
- The program prompts the user to enter a number continuously until the number matches the generated number.
- For each user input, the program tells the user whether the input is too low or too high.

```
int n = (int)(Math.random()*101);
for ( ; ; ) {
    int x = scanner.nextInt();
    if ( x == n ) break;
    if ( x < n )
        System.out.println("Too low.");
    else
        System.out.println("Too high.");
}
System.out.println("Bingo!");
```



Factorization

```
> 2
2(1)
> 12
2(2)3(1)
> 123
3(1)41(1)
> 123456789
3(2)3607(1)3803(1)
> 400000000
2(10)5(8)
> 23456789
23456789(1)
```

```
1  int n = scanner.nextInt();
2  int p = 2, c = 0;
3  for ( ; ; ) {
4      if ( n % p == 0 ) {
5          n /= p;
6          ++c;
7          continue;
8      }
9      if ( c != 0 ) {
10         System.out.print(p+"("+c+")");
11         c = 0;
12     }
13     if ( ++p > n ) break;
14 }
15 System.out.println();
```

Reading Homework

Textbook

- Section 5.3–5.7, 5.9, 5.11.

Internet

- Control flow (`http://en.wikipedia.org/wiki/Control_flow`).
- goto (`http://en.wikipedia.org/wiki/GOTO`).
- Approximations of π (`http://en.wikipedia.org/wiki/Approximations_of_%CF%80`).

Self-test

- 4.6 – 4.33 (`http://tiger.armstrong.edu/selftest/selftest9e?chapter=4`).