

## 08 Formatted Printing and Nested Loops

*Instructor:* Ke Wei (柯韋)

➡ A319    ☎ Ext. 6452    ✉ [wke@ipm.edu.mo](mailto:wke@ipm.edu.mo)

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

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

September 24, 2018



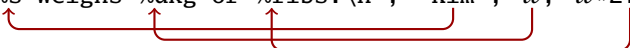
# Outline

- 1 **Formatted Printing**
- 2 **Nested Loops**
- 3 **Practice**
- 4 **Reading Homework**

# C-style Formatted Printing

- Inspired by C's *printf* function (*print formatted*), Java provides the *printf* and *format* methods, which are equivalent to each other, for the *System.out*.

```
int w = 65;
System.out.printf("Mr. %s weighs %dkg or %flbs.\n", "Kim", w, w*2.205);
```



prints: Mr. Kim weighs 65kg or 143.325000lbs.

- The first argument is the *format* string, followed by a list of expressions.
- In the format string, there are format *specifiers* led by (%), such as %s, %d and %f. They are placeholders for the values of the upcoming expressions, respectively.
- A format specifier specifies the type, width and precision of the value to display.

```
System.out.printf("|%6d|_and_|%+012.4f|_and_|%1$+-8d|", 123, Math.PI);
```

prints: | 123| and |+000003.1416| and |+123 |

# Format Specifiers

A format specifier has the form:

$\%[argument\ index\$][flags][width][.precision]conversion$

- The optional *argument index* is a decimal integer indicating the position of the argument in the argument list. The first argument is referenced by “1\$”, the second by “2\$”, etc.
- The optional *flags* is a set of characters that modify the output format. For examples, “+”, “-” and “0”.
- The optional *width* is a non-negative decimal integer indicating the minimum number of characters to be written to the output.
- The optional *precision* is a non-negative decimal integer used to restrict the number of fractional digits.
- The required *conversion* is a character indicating how the argument should be formatted. For examples, “d”, “X”, “f”, “e”, “g” and “s”.

# Nested Loops

- Loops can contain inner loops, called nested loops.
- Although there is no limit to the level to which loops may be nested, one can hardly master too deeply nested loops.
- The following code prints the multiplication table using a nested loop.

---

```
1  int i = 1;
2  while ( i <= 9 ) {
3      int j = 1;
4      while ( j < i ) { System.out.print("_____"); ++j; }
5      while ( j <= 9 ) { System.out.printf("_%dx%d=%2d_", i, j, i*j); ++j; }
6      System.out.println();
7      ++i;
8  }
```

---

# The Multiplication Table

Each row  $i$  of the table consists of two parts:

- ❶ the  $i-1$  segments of leading blanks (may be empty), and
- ❷ the formulas starting from  $i \times i$ .

1x1= 1	1x2= 2	1x3= 3	1x4= 4	1x5= 5	1x6= 6	1x7= 7	1x8= 8	1x9= 9
	2x2= 4	2x3= 6	2x4= 8	2x5=10	2x6=12	2x7=14	2x8=16	2x9=18
		3x3= 9	3x4=12	3x5=15	3x6=18	3x7=21	3x8=24	3x9=27
			4x4=16	4x5=20	4x6=24	4x7=28	4x8=32	4x9=36
				5x5=25	5x6=30	5x7=35	5x8=40	5x9=45
					6x6=36	6x7=42	6x8=48	6x9=54
						7x7=49	7x8=56	7x9=63
							8x8=64	8x9=72
								9x9=81

## Practice 8-1: Fibonacci Numbers

- Fibonacci numbers are the numbers in the following integer sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

By definition, the first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two.

- Write a program *Fibonacci* to input a number  $n$  ( $n \geq 0$ ), and print the  $n^{\text{th}}$  Fibonacci number (starting from the 0<sup>th</sup> number). For example,

Input the Fibonacci index: 8

21

- Hint: you may establish a loop and compute two consecutive Fibonacci numbers in each iteration. Also, Fibonacci numbers can be very large, use variables of type **long** to hold them.

## Practice 8-2: A Right Triangle

- We use asterisks (\*) to print shapes.
- Write a program *RightTriangle* to input the height  $n$  ( $n \geq 0$ ) of a right triangle, and print the triangle line by line. An example is shown on the right.
- Setup an outer loop to loop through the lines.
- Setup two inner loops, one for the leading blanks, one to fill the rest of the line with asterisks.
- Can you modify the program to print only the outline of the triangle?

Input the height: 15

```

      *
     **
    ***
   ****
  *****
 *****
*****
*****
*****
*****
*****
*****
*****
*****
*****

```



### Practice 8-3: A Sine Wave

- This time we use asterisks (\*) to fill out a sine wave.
- Since the range of the sine function is between  $-1$  and  $+1$ , we arrange our  $Y$ -axis horizontally rightward, and  $X$ -axis vertically downward.
- We also need to define the scale, like how many lines per  $\pi$  and how many columns per  $Y$ -unit.
- Write a program *SineWave* to input the number of lines  $n$  ( $n \geq 0$ ) of a sine wave, and print the sine wave line by line. An example is shown on the right.
- Blanks and asterisks are printed similarly, the length of asterisks is determined by the *Math.sin* function.
- You must also handle negative sine values by adding to it an offset.

Input the # of lines: 15

\*  
\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*

```

          ****
        *****
      *
    *****
  *****
*****
*****
  *****
    *****
      *
        *****
          ****

```

## Practice 8-4: Diagonal Counting

- Given a positive integer  $n$ , write a program *Diagonal* to print, row by row, the following triangular array that lists the numbers from 1 to  $n$  diagonally:

1	3	6	10	...		1	3	6	10	15	21
2	5	9				2	5	9	14	20	
4	8					4	8	13	19		
7		$n$			for example $n = 24$ :	7	12	18			
:	..					11	17	24			
*						16	23				
						22					

- You may *not* use arrays. You need to align the numbers to the right.
- Hint: use nested loops, pay attention to the increment steps of each row and column, and the conditions to stop the loops. You also need to compute the number of digits of the maximum number.

## Practice 8-5: Up-Down Counting

- Given two positive integers  $w, h$ , write a program *UpDown* to print, row by row, the numbers from 1 to  $w \times h$  in the following up-down manner (suppose  $w=7, h=6$ ):

1	12	13	24	25	36	37
2	11	14	23	26	35	38
3	10	15	22	27	34	39
4	9	16	21	28	33	40
5	8	17	20	29	32	41
6	7	18	19	30	31	42

1	12	13	24	25	36	37
2	11	14	23	26	35	38
3	10	15	22	27	34	39
4	9	16	21	28	33	40
5	8	17	20	29	32	41
6	7	18	19	30	31	42

- You need to align the numbers to the right.
- Hint: separate each row into even and odd columns, figure out the relation between the number on the column and the number on the first column.

# Reading Homework

## Textbook

- Section 4.6.
- Section 5.8, 5.10.

## Internet

- Operator associativity ([http://en.wikipedia.org/wiki/Operator\\_associativity](http://en.wikipedia.org/wiki/Operator_associativity)).
- Formatting (<http://docs.oracle.com/javase/tutorial/essential/io/formatting.html>).
- printf format string ([http://en.wikipedia.org/wiki/Printf\\_format\\_string](http://en.wikipedia.org/wiki/Printf_format_string)).

## Self-test

- 3.38 – 3.49 (<http://tiger.armstrong.edu/selftest/selftest9e?chapter=3>).
- 4.1 – 4.5 (<http://tiger.armstrong.edu/selftest/selftest9e?chapter=4>).

