C2110 *UNIX and programming*

Lesson 8 / Module 1

PS / 2020 Distance form of teaching: Rev2

Petr Kulhanek

kulhanek@chemi.muni.cz

National Center for Biomolecular Research, Faculty of Science
Masaryk University, Kamenice 5, CZ-62500 Brno

Loop

for Loop

A loop is a control structure that repeatedly executes a sequence of commands. Repetition and end of the loop is controlled by a condition.

performed before starting the loop (counter initialization)

if the condition is met, the commands command etc. are executed

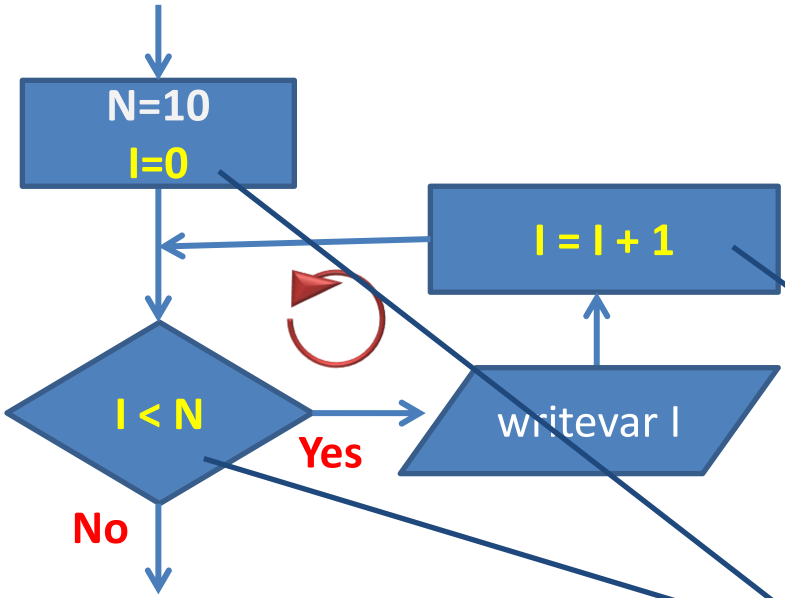
```
for(initialization;condition;change))
      do
      command1
      ...
      done
```

counter is updated after executing commands

Compact notation:

```
for((initialization;condition;change)); do
      command1
      ...
done
```

For Loop and Flowchart



🟨 counter (variable)

```
N=10
for((I = 0; I < N; I = I + 1)); do
     echo $I
done
```

for versus while loop

```
for((I=1;I <= 10;I++)); do
     echo $I
done
```

if the condition is met, the commands in block do/done are executed

performed before starting the loop (counter initialization)

```
I=1
while [[ I -le 10 ]]; do
     echo $I
     (( I = I + 1 ))
done
```

In addition to changing the counter to end of loop, it is possible to make other changes in the body of the loop. However, this is NOT RECOMMENDED because it reduces the readability of the code.

updating the counter after executing commands

It is possible to change the counter anywhere in the body of the loop (even in multiple places).

for Loop, Usage

Lists numbers 1 to 10

```
for((I=1;I <= 10;I++)); do
     echo $I
done
```

Variable **I** has a role of a **counter**.

Initialization follows free rules because the expression is specified in the (()) block.

Change:
You can use any expression that can be interpreted in the (( )) block, e.g.:
- **++** increases value of variable by one
- **–** decreases value of variable by one
- more ...

Lists numbers 10 to 1

```
for((I=10;I >= 1;I--)); do
     echo $I
done
```

Condition:
Following comparison operators can be used:

!=	does not equal
==	equals
<	smaller
<=	less than or equal to
>	larger
>=	greater than or equal to

Can only be applied to integers in (( )).

for Loop, Counter Change

If a variable can be interpreted as an integer, the following arithmetic operators can be used:

++ increases value of variable by one

```
I++
```

-- decreases the value of the variable by one

```
I--
```

+ adds two values

```
A = 5 + 6
A = A + 1
```

- subtracts two values

```
A = 5 - 6
A = A - 1
```

***** multiplies two values

```
A = 5 * 6
A = A * 1
```

/ divides two values (integer division)

```
A = 5 / 6
A = A / 1
```

```
A=A+3
```

+= adds a value to variable

```
A += 3
A += B
```

-= subtracts value from variable

```
A -= 3
A -= B
```

***=** multiplies variable by value

```
A *= 3
A *= B
```

/= divides variable by value

```
A /= 3
A /= B
```

Nesting Loops

loop control groups can be freely nested.

external loop

internal loop

```
for((I=1;I <= 10;I++)); do
     for((J=1;J <= 10;J++)); do
          echo "$I $J"
     done
done
```

the outer loop counter can affect the behavior of the inner loop

```
for((I=1;I <= 10;I++)); do
     for((J=1;J <= I;J++)); do
          echo "$I $J"
     done
done
```

The number of nestings is not limited. It can be combined with other loops (while, until, for in) or conditions.

Exercise 1

1. Write bash scripts solving Tasks 1, 2 and 3. Replace while loop with for loop. Make user enter the dimension of rendered shape as the first argument of the script. The script tests to see if the correct number of arguments is specified and whether the first argument is an integer greater than zero.
2. Modify the solution of Task 1 so that a rectangle is drawn. The dimensions of the rectangle will be entered by the user interactively after running the script.
3. Write scripts in bash solving Tasks 4 a 5.

Task 1

Print a square of characters in the terminal **X**. The length of the side of the square is entered by the user.

```
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
X X X X X X X X X X
```

Ignore the fact that it is not visually a square. However, number of **X** characters for a line and the number of lines must be the same. Possibly use "X " - X and a space.

Task 2

Print a right triangle with **X** characters in the terminal, so that one perpendicular is located at the top and the other on the left. The length of the perpendicular is entered by the user.

```
X X X X X X X X X X
X X X X X X X X X
X X X X X X X X
X X X X X X X
X X X X X X
X X X X X
X X X X
X X X
X X
X
```

Task 3

Print a right triangle with **X** characters in the terminal, so that one perpendicular is located at the bottom and the other on the left. The length of the perpendicular is entered by the user.

```
X
X X
X X X
X X X X
X X X X X
X X X X X X
X X X X X X X
X X X X X X X X
X X X X X X X X X
X X X X X X X X X X
```

Task 4

In the terminal, print the outline of square with **X** characters. The length of the side of the square is entered by the user.

```
X X X X X X X X X X
X                 X
X                 X
X                 X
X                 X
X                 X
X                 X
X                 X
X                 X
X X X X X X X X X X
```

A suitable solution is to use the script from Task 1 and check the printing of characters using appropriately selected condition.

Task 5

In the terminal , print the outline of the square and its diagonals with **X** characters. The length of the side of the square is entered by the user.

```
X X X X X X X X X X
X X             X X
X   X         X   X
X     X     X     X
X       X X       X
X       X X       X
X     X     X     X
X   X         X   X
X X             X X
X X X X X X X X X X
```

A suitable solution is to use the script from Task 1 and check the printing of characters using appropriately selected condition.