ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΠΛΗΡΟΦΟΡΙΚΗ ΙΙ

Ενότητα 3: Έλεγχος ροής προγράμματος

Μιχάλης Δρακόπουλος

Σχολή Θετικών επιστημών

Τμήμα Μαθηματικών

ΠΛΗΡΟΦΟΡΙΚΗ ΙΙ (Java)
Ενότητα 3

ΕΛΕΓΧΟΣ ΡΟΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

I. Ελεγκτές συνθηκών ή περιπτώσεων:
- i) `if/else`
- ii) `switch`

II. Επαναληπτικές διαδικασίες:
- i) `for`
- ii) `while` (με έλεγχο συνθήκης)
- iii) `do/while` (με έλεγχο συνθήκης)

- **if/else:**

Με επιλογές μιας μόνο εντολής:

```
if (<λογική_έκφραση>)
   <εντολή_1>;  // if true
else
   <εντολή_2>;  // if false
```

Με επιλογές πολλαπλών εντολών:

```
if (<λογική_έκφραση>)
{   <εντολή_1>;
  <εντολή_2>;
            ...
}
else
{
        <εντολή_Α>;
  <εντολή_Β>;
       ...
}
```

- `λογική_έκφραση` → `boolean`, δηλαδή `true` ή `false`

π.χ.

```
int a=5, b=7, c;
if (a>=b)
   c = a+b;
else
   c = a-b;
System.out.println(c);   // → -2
```

Άλλα παραδείγματα:

- Μέγιστος μεταξύ των x και y:

```
if (x>y)                                    max = x;
   max = x;                                 if (y>x)
else                          ή                max = y;
   max = y;
```

- Απόλυτη τιμή του x:

```
if (x>=0)
   abs = x;                                 abs = x;
else                          ή             if (x<0)
   abs = -x;                                   abs = -x;
```

- *Δηλαδή, μπορεί να υπάρχει if χωρίς else:*

```
if (<λογική_συνθήκη>)                       if (<λογική_συνθήκη>)
   <εντολή>;                          ή          {
                                                 <εντολές>;
                                                    ...
                                                 }
```

Δηλαδή, π.χ., αν `test` είναι μια `boolean` μεταβλητή:

```
if (test)
  <εντολή1>;
<εντολή2>;
<εντολή3>;
```

Σε αυτό το παράδειγμα δεν υπάρχει `else`. Επομένως μόνο η `<εντολή1>` «ανήκει» στο `if` και εκτελείται μόνο εάν η `test` είναι `true`. Η `<εντολή2>` και η `<εντολή3>` εκτελούνται ούτως ή άλλως, αφού βρίσκονται *μετά* το `if`.

Άρα: - Αν η `test` είναι `true`: `<εντολή1>, <εντολή2>, <εντολή3>`
- Αν η `test` είναι `false`: `<εντολή2>, <εντολή3>`

Όμως:

```
if (test)
  <εντολή1>;
else
  <εντολή2>;
<εντολή3>;
```

Σε αυτό το παράδειγμα η `<εντολή2>` ανήκει στην "`else`" περίπτωση του `if`, άρα εκτελείται *μόνο* εάν η `test` είναι `false`. Η `<εντολή3>` εκτελείται ούτως ή άλλως, αφού βρίσκεται μετά το `if`.

Άρα: - Αν η `test` είναι `true`: `<εντολή1>, <εντολή3>`
- Αν η `test` είναι `false`: `<εντολή2>, <εντολή3>`

- **Πρώτη διευκρίνιση:** Η στοίχιση των εντολών μπορεί πολλές να είναι παραπλανητική. Η Java δεν λαμβάνει υπ' όψιν της τη στοίχιση που κάνει ο προγραμματιστής στον κώδικά του και απλά διαχωρίζει τις εντολές της (π.χ. πού τελειώνει ένα `if` ή ένα `else`) σύμφωνα με τους κανόνες της. Π.χ., στο δεύτερο παράδειγμα παραπάνω, δεν θα άλλαζε τίποτα εάν κατά λάθος η <εντολή3> είχε γραφεί ακριβώς κάτω από την <εντολή2> ως εξής:

```
if (test)
   <εντολή1>;
else
   <εντολή2>;
   <εντολή3>;
```

Η στοίχιση αυτή είναι παραπλανητική, γιατί δεν αλλάζει τίποτα σε σχέση με πριν: στο `else` «ανήκει» μόνο η <εντολή2>, γιατί δεν υπάρχουν `{ }` άρα στο `else` ανήκει μόνο μία εντολή. Άρα η <εντολή3> παρόλο που φαίνεται να «ανήκει» στο `else`, είναι στην ουσία εκτός `if-else`, όπως ακριβώς πριν, και εκτελείται ούτως ή άλλως, ανεξαρτήτως της τιμής της `test`.

- **Δεύτερη διευκρίνιση:** Προφανώς η «λογική έκφραση» ή η «συνθήκη» που ελέγχεται σε ένα `if` μπορεί να είναι και απλά μια `boolean` μεταβλητή, αφού και αυτή, όπως το αποτέλεσμα μιας λογικής έκφρασης, παίρνει τιμές `true` ή `false`. Άρα, στο προηγούμενο παράδειγμα, για την `boolean` μεταβλητή `test`, αντί να γράψουμε:

```
if (test == true)
```

είναι το ίδιο να γράψουμε:

```
if (test)
```

αφού και στις δύο περιπτώσεις το αποτέλεσμα είναι true αν η `test` έχει την τιμή `true` και `false` αν η `test` έχει την τιμή `false`.
Αντίστοιχα, για τον έλεγχο:

```
if (test == false)
```

γράφουμε απλά:

```
if (!test).
```

Ένθετο if (nested)

a)

```
if (...)
{
   ...
}
else if (...)
{
   ...
}
else if (...)
{
   ...
}
else
{
   ...
}
```

b)

```
if (...)
{
   ...
   if (...)
   {
      ...
   }
   else
   {
      ...
   }
}
else
{
   ...
}
```

Παράδειγμα με if/else if και user input:

```
import java.util.*;
public class Grades
{
   public static void main(String [] args)
   {
      int score;   // η βαθμολογία στην κλίμακα 0-100
      char grade;  // ο βαθμός σαν A, B, C, D ή F

      Scanner input = new Scanner(System.in);
      System.out.println("Δώσε τη βαθμολογία σου (0-100):");
      score = input.nextInt();

      if (score > 90)
         grade = 'A';
      else if (score > 80)
         grade = 'B';
      else if (score > 70)
         grade = 'C';
      else if (score > 60)
         grade = 'D';
      else
         grade = 'F';

      System.out.println("Με βαθμολογία " + score +
                         " στα 100, ο βαθμός σου είναι: " + grade);

   }  // end of main
}   // end of class
```

● switch:

```
switch (varName)
{
  case value1:
     εντολές;
     break;
  case value2:
     εντολές;
     break;
  case value3:
     εντολές;
     break;
  default:
     εντολές;
}
```

→ Η μεταβλητή `VarName` είναι είτε ακέραια, είτε τύπου `boolean`, είτε τύπου `char`.

Το `switch` «στέλνει» τη ροή του κώδικα στο ισχύον case. Αν δεν υπάρχει κάποιο `break` στο τέλος ενός `case`, τότε η ροή του κώδικα συνεχίζει στο επόμενο `case`! Δηλαδή, κάνει έλεγχο των `case` μέχρι να βρει αυτό που ισχύει και από εκεί και πέρα δεν ελέγχει εάν τα υπόλοιπα ισχύουν ή όχι, αλλά απλά εκτελεί τον κώδικα μέχρι να βρει κάποιο `break` ώστε να βγει από το `switch`.

Παράδειγμα με switch:

```
char grade;
  .
  .  // απόδοση τιμής στη μεταβλητή grade
switch (grade)
{
  case 'A':
     System.out.println("Πολύ καλά!");
  case 'B':
     System.out.println("Μπράβο!");
     break;
  case 'C':
      System.out.println("Έτσι κι έτσι...");
      break;
  .
  .
}
```

Αν το grade είναι 'A' θα εκτυπώσει:

```
Πολύ καλά!
Μπράβο!
```

(γιατί δεν υπάρχει `break` στην 1η περίπτωση)

Αν το grade είναι 'B' θα εκτυπώσει:

```
Μπράβο!
```

Αν το grade είναι 'C' θα εκτυπώσει:

```
Έτσι κι έτσι...
```

- **for:** Επανάληψη μέρους του κώδικα για συγκεκριμένο αριθμό επαναλήψεων

Γενική σύνταξη:

```
for (αρχικοποίηση; συνθήκη; ανανέωση)
  εντολή;
```

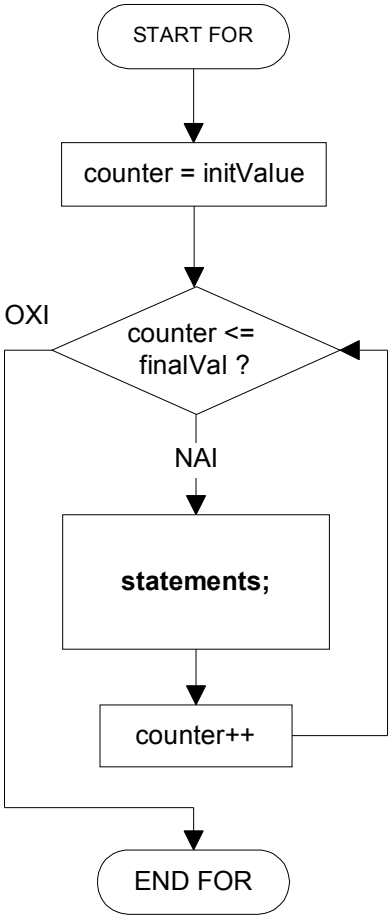
συνήθως:

```
for (int counter=initialValue; counter<=finalValue; counter++)
{
  εντολές;
}
```

Σειρά εκτέλεσης εντολών:

```
i)   int counter = initialVal;
ii)  έλεγχος counter<=finalValue ?
iii) εντολές;
iv)  counter++
v)   έλεγχος, κτλ.
```

Διάγραμμα Ροής του for-loop



→ Συνήθως ο μετρητής (counter) ορίζεται τοπικά, στο πρώτο μέρος της παρένθεσης του for-loop, δηλαδή είναι μια τοπική μεταβλητή. Παράδειγμα:

```
for (int i=1; i<=5; i++)
{
   System.out.println(i);
}
```

Τυπώνει:
```
1
2
3
4
5
```

Σε αυτό το παράδειγμα, εάν υπήρχε μια εντολή `System.out.println(i);` έξω από το for-loop **δεν** θα εκτύπωνε την τιμή 6 αλλά θα εμφάνιζε σφάλμα κατά το compilation του προγράμματος. Η μεταβλητή i που έχει δημιουργηθεί μέσα στο for είναι τοπική μεταβλητή του for και άρα "υπάρχει" μόνο μέσα στο μπλοκ του for (όπως αυτό ορίζεται από τα άγκιστρα { ... }).
Φυσικά ο μετρητής του for μπορεί να έχει ορισθεί πιο πάνω στο πρόγραμμα. Στην περίπτωση αυτή δεν θα έπρεπε να ξαναορισθεί μέσα στο for αλλά απλά να χρησιμοποιηθεί:

```
int i;
...
for (i=1; i<=5; i++)
  ...
```

και τότε θα είχε νόημα ένα `System.out.println(i)` εκτός του for.

→ Για να αυξάνεται ο μετρητής ενός for κατά 2 (αντί για 1 που είναι συνήθως το βήμα):

```
for (int i=0; i<10; i+=2)
```

και ομοίως για οποιοδήποτε άλλο βήμα.

- **- while:**

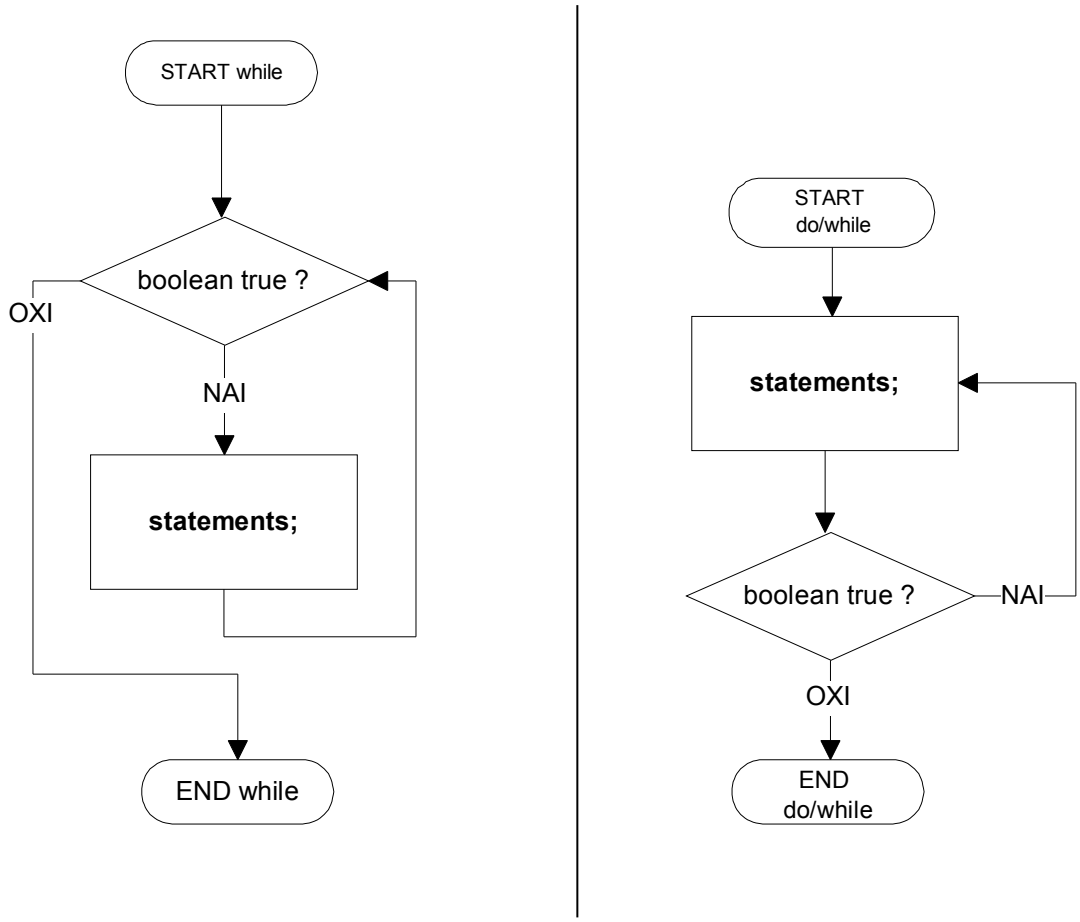
```
while(συνθήκη)
{
  εντολές;
}
```

- do/while:

```
do
{
  εντολές;
}
while(συνθήκη);
```

Η συνθήκη είναι μία `boolean` έκφραση (άρα: `true` ή `false`).
Και το `while` και το `do/while` κάνουν επανάληψη των εντολών καθ' όσον η συνθήκη είναι αληθής. Η διαφορά μεταξύ τους είναι ότι το `do/while` θα εκτελέσει τις εντολές *τουλάχιστον μία φορά*, ανεξάρτητα με το αν είναι `true` η συνθήκη, αφού την ελέγχει στο τέλος του loop, ενώ η `while` δεν μπαίνει καθόλου στο loop αν αρχικά η συνθήκη είναι `false`.

<u>Διαγράμματα Ροής του `while` και του `do/while`</u>



Παραδείγματα while:

```
int n=0;
while (n<5)
{
 S.O.P.(n++);
}
```

Θα τυπώσουν:

0
1
2
3
4

```
int n=0;
while (n<5)
{
 S.O.P.(++n);
}
```

1
2
3
4
5

```
int n=0;
while (n<5)
{
 S.O.P.(n--);
}
```

0
-1
-2
-3
-4
...

(ατέρμονες επαναλήψεις)

```
int n=5;
while (n<5)
{
 S.O.P.(n--);
}
```

-

Άλλο παράδειγμα while:

- Άθροισμα ψηφίων ακεραίου:

Αλγόριθμος: i) εύρεση τελευταίου ψηφίου: n%10 (π.χ., 1975%10 → 5)
ii) αποκοπή τελευταίου ψηφίου: n/10 (π.χ., 1975/10 → 197)

```
.
.
int n = input.nextInt();
int dsum = 0;
while (n>0)
{
   dsum += n%10;
   n /= 10;
}
System.out.println(dsum);
```

Παράδειγμα με switch:

Να γραφεί πρόγραμμα που να ζητάει από το χρήστη τον αριθμό του μήνα (1-12) και να εκτυπώνει στην οθόνη το πλήθος των ημερών του μήνα αυτού, υποθέτοντας ότι το έτος δεν είναι δίσεκτο.

```
import java.util.*;
public class Ex1
{
   public static void main(String [] args)
   {
      Scanner input = new Scanner(System.in);
      System.out.println("Δώσε τον αριθμό του μήνα (1-12)");
      int month = input.nextInt();
```

```
      switch (month)
      {
         case 4:
         case 6:
         case 9:
         case 11:
         {
            System.out.println("Ο μήνας έχει 30 ημέρες.");
            break;
         }
         case 2:
         {
            System.out.println("Ο μήνας έχει 28 ημέρες.");
            break;
         }
         default:
            System.out.println("Ο μήνας έχει 31 ημέρες.");
      } // end switch
   } // end main
} // end class
```

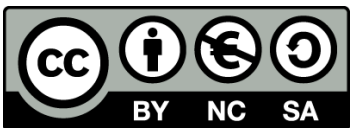
Σημειώματα

Σημείωμα Αναφοράς

Copyright Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών, Μιχάλης Δρακόπουλος, 2014.
Μιχάλης Δρακόπουλος. «Πληροφορική ΙΙ. Ενότητα 3: Έλεγχος ροής προγράμματος». Έκδοση: 1.0. Αθήνα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: http://opencourses.uoa.gr/courses/MATH106/.

Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».



[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

Διατήρηση Σημειωμάτων

- Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.