Προχωρημένος Προγραμματισμός

Υπολογισμός

ΕΛΕΥΘΕΡΙΟΣ ΚΟΣΜΑΣ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2022-2023 | ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Περίληψη

Σήμερα ...

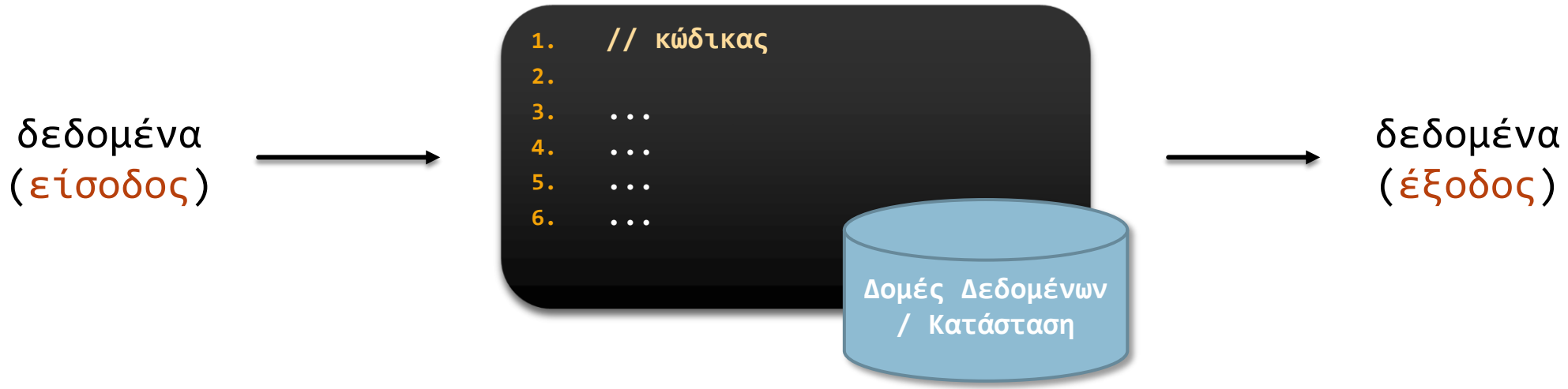
- θα συζητήσουμε τα βασικά στοιχεία του υπολογισμού
  - εκφράσεις
  - τρόπος επανάληψης σε ένα σύνολο τιμών (iteration)
  - επιλογή μεταξύ δύο εναλλακτικών ενεργειών (selection)
- θα δούμε πώς ένας συγκεκριμένος υπο-υπολογισμός μπορεί να ονομαστεί και να οριστεί ξεχωριστά ως συνάρτηση (function)

Γνωρίζετε ήδη πως να ...

- κάνετε αριθμητικούς υπολογισμούς
  - d = a+b*c
- επιλέγετε
  - "if this is true, do that; otherwise do something else "
- επαναλαμβάνετε έναν υπολογισμό
  - "do this until you are finished"
  - "do that 100 times"
- γράφετε συναρτήσεις
  - "go ask Joe and bring back the answer"
  - "hey Joe, calculate this for me and send me the answer"

- αυτά που θα συζητήσουμε σήμερα είναι ως επί το πλείστον λεξιλόγιο και συντακτικό για αυτά που ήδη γνωρίζετε

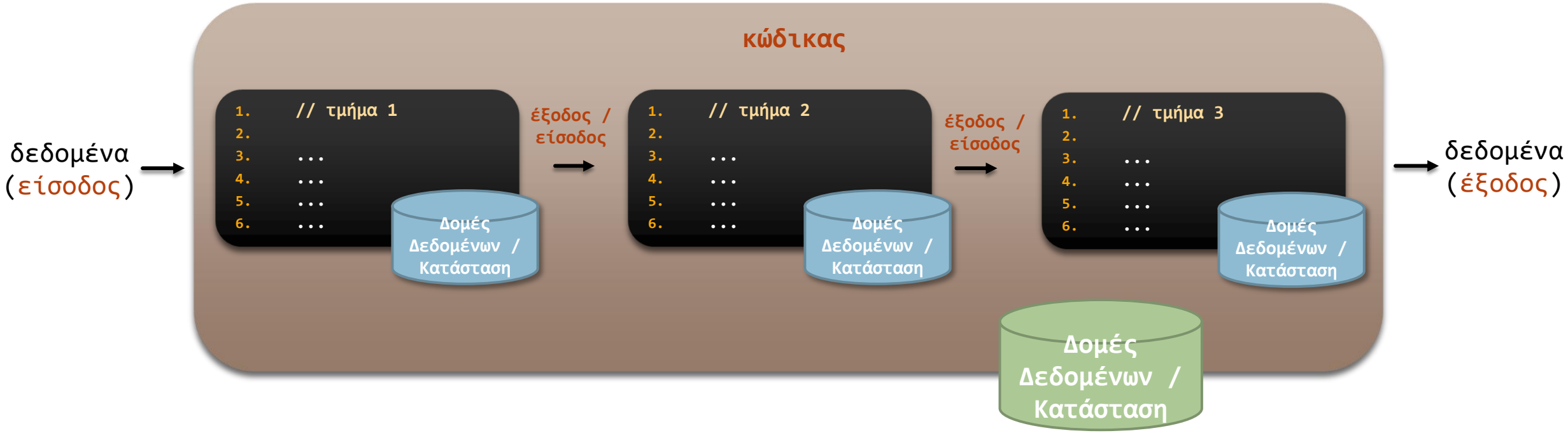
Υπολογισμός



- είσοδος:
  - πληκτρολόγιο
  - αρχεία
  - άλλες συσκευές εισόδου
  - άλλα προγράμματα
  - άλλα μέρη ενός προγράμματος
- υπολογισμός: τι θα κάνει το πρόγραμμά μας με την είσοδο για την παραγωγή της εξόδου
- για να ασχοληθεί με την είσοδο ένα πρόγραμμα περιέχει κάποια δεδομένα → δομές δεδομένων ή κατάσταση
- έξοδος:
  - οθόνη
  - αρχεία
  - άλλες συσκευές εξόδου
  - άλλα προγράμματα
  - άλλα μέρη ενός προγράμματος

Υπολογισμός

Είσοδος από / έξοδος σε άλλα μέρη ενός προγράμματος



Υπολογισμός II

- η δουλειά μας είναι να εκφράσουμε υπολογισμούς σωστά, απλά, αποτελεσματικά
- δομικά εργαλεία
  - διαίρει και βασίλευε (divide and conquer)
    - σπάμε μεγάλους υπολογισμούς σε πολλούς μικρότερους και συνδυάζουμε τα υπο-αποτελέσματα για να προκύψει το τελικό αποτέλεσμα
  - αφαίρεση (abstraction)
    - παρέχουμε μια έννοια υψηλότερου επιπέδου όπου κρύβουμε λεπτομέρειες της υλοποίησης
- η οργάνωση δεδομένων είναι συχνά το κλειδί για τον καλό κώδικα
  - μορφή εισόδου / εξόδου
  - πρωτόκολλα
  - δομές δεδομένων

❖ σημειώστε την έμφαση στη δομή και την οργάνωση

- δεν παίρνετε καλό κώδικα απλά γράφοντας πολλές εντολές (statements)

Χαρακτηριστικά γλώσσας

- κάθε χαρακτηριστικό μιας γλώσσας προγραμματισμού υπάρχει για να εκφράσει μία θεμελιώδη ιδέα

- για παράδειγμα:
  - `+` : πρόσθεση
  - `*` : πολλαπλασιασμός
  - `if (expression) statement else statement ;` : επιλογή
  - `while (expression) statement ;` : επανάληψη
  - `f(x);` : συνάρτηση / λειτουργία
  - ...

- συνδυάζουμε χαρακτηριστικά μιας γλώσσας για να αναπτύξουμε προγράμματα

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) – Παραδείγματα

```
1.  // Υπολογισμός εμβαδού και μέσου όρου
2.  class ComputeAverage
3.  {
4.     public static void main(String args[])
5.     {
6.        int length = 20;                    // η πιο απλή έκφραση, μία σταθερά (20)
7.        int width = 40;
8.        int area = length * width;          // ένας πολλαπλασιασμός
9.        int average = (length + width) / 2; // μία πρόσθεση και μία διαίρεση
10.    }
11. }
```

- ισχύουν οι συνηθισμένοι κανονισμοί προτεραιότητας πράξεων
  - π.χ. `a*b+c/d = (a*b)+(c/d)` (και όχι `a*(b+c)/d`)
  - εάν αμφιβάλλετε ή εάν είναι αρκετά πολύπλοκο → χρησιμοποιήστε παρενθέσεις!
- μη γράφετε παράλογα περίπλοκες εκφράσεις
  - π.χ. `a*b+c/d*(e-f/g)/h+7` ← πολύπλοκο!
- επιλέγετε ονόματα που έχουν νόημα!

Χαρακτηριστικά γλώσσας
Εκφράσεις (expressions)

- οι εκφράσεις αποτελούνται από τελεστές και τελεσταίους
  - οι τελεστές καθορίζουν τι πρέπει να γίνει
  - οι τελεσταίοι ορίζουν τα δεδομένα στα οποία εφαρμόζονται οι τελεστές

  π.χ. `length * width` → πολλαπλασιασμός ακεραίων `length` και `width`

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) - Σταθερή έκφραση

έκφραση με τιμή που αποτελείται αποκλειστικά από σταθερές

- τα προγράμματα χρησιμοποιούν πολλές σταθερές, π.χ. **π:3.14159**
- αποφυγή μαγικών σταθερών → χρήση μεταβλητών
  - ☞ καλύτερη διατηρησιμότητα προγράμματος, π.χ. αύξηση ακρίβειας του **π** σε **3.14159265359**
- δεν πρέπει να αλλάζουμε κατά λάθος αυτές τις σταθερές → χρήση σταθερών εκφράσεων (**final**)

```
1.  final double pi = 3.14159;      // σταθερή έκφραση
2.  pi = 7;                         // σφάλμα: εκχώρηση στο final
3.  v = 2*pi/r;                     // σωστό: μόλις διάβασες το pi, μην προσπαθήσεις να το αλλάξεις
```

```
1.  final int max = 17;             // σταθερή έκφραση
2.  max + 2;                        // σταθερή έκφραση
3.  int val = 19;
4.  val + 2;                        // μη σταθερή έκφραση (χρησιμοποιεί μια μεταβλητή)
```

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) - Τελεστές

- δεν εφαρμόζονται όλοι οι τελεστές σε όλους τους τύπους (τελεσταίων)
  - ο τελεστής `*` δεν εφαρμόζεται σε `String`
- ένα συγκεκριμένος τελεστής έχει διαφορετικό νόημα για διαφορετικούς τύπους
  - `a += b` → πρόσθεση ακεραίων
  - `a += b` → συνένωση string
- ? τι σημαίνει το `a<b<c` ;
  - `(a<b)<c` → `true<c` ή `false<c`
  - × δε σημαίνει ότι το `b` είναι μεταξύ των `a` και `c`!
  - ☞ μη χρήσιμη έκφραση!

τελεστής	όνομα/περιγραφή	σχόλιο
`==`	**ίσο**	αποτέλεσμα `boolean`
`!=`	**όχι ίσο**	
`a < b`	**μικρότερο από**	
`a <= b`	**μικρότερο από ή ίσο**	
`a > b`	**μεγαλύτερο από**	
`a >= b`	**μεγαλύτερο από ή ίσο**	
`&&`	**λογικό και**	
`||`	**λογικό ή**	
`!`	**άρνηση (not)**	
`+ - * / %`	αριθμητική πράξη	
`=`	**εκχώρηση**	δεν πρέπει να συγχέεται με το `==`
`*=`	**σύνθετη εκχώρηση**	επίσης για τα: `+ - / %`
`++a`	**προ-αύξηση**	αυξάνει κατά 1, χρησιμοποιεί την αυξημένη τιμή
`a++`	**αύξηση**	χρησιμοποιεί την (παλιά) τιμή, αυξάνει κατά 1

Χαρακτηριστικά γλώσσας
Εκφράσεις (expressions) - Λογικοί τελεστές

- λογικοί τελεστές για λογικές εκφράσεις
  - άρνηση: `!B`
  - ΚΑΙ: `(A && B)`
  - Ή: `(A || B)`
- έλεγχος για βασικούς τύπους `A`, `B`:
  - ισότητα: `(A == B)`
  - ανισότητα: `(A != B)` ή `(!(A == B))`
  - μεγαλύτερο/μικρότερο ή ίσο: `(A <= B)`, `(A >= B)`
- έλεγχος για μεταβλητές (αντικείμενα) οποιουδήποτε άλλου τύπου γίνεται με την μέθοδο `equals` (πρέπει να έχει οριστεί):
  - ισότητα: `(A.equals(B))`
  - ανισότητα: `(!A.equals(B))`
- λογικές σταθερές: `true` (αληθές) , `false` (ψευδές)

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) - Λογικοί τελεστές - Έλεγχος ισότητας για αντικείμενα - π.χ. Strings

- αν έχουμε δύο μεταβλητές τύπου `String` για να ελέγξουμε αν έχουν την ίδια τιμή πρέπει να χρησιμοποιήσουμε την μέθοδο `equals`

```
1.  String firstString = "abc";
2.  String secondString = "ABC";
3.  boolean test1 = firstString.equals(secondString);   // false, το 1ο String δεν είναι ίσο με το 2ο
4.  boolean test2 = firstString.equals("abc");          // true, το 1ο String είναι ίσο με το "abc"
```

- η παρακάτω εντολή δεν είναι σωστή

```
5.  boolean test3 = (firstString == secondString);    // ελέγχει εάν οι αναφορές στα αντικείμενα είναι ίδιες
```

  - περνάει από τον compiler
  - σε κάποιες περιπτώσεις θα δουλέψει
  - αλλά δεν κάνει αυτό που θέλουμε

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) – «Συνοπτικοί» τελεστές

- για πολλούς δυαδικούς τελεστές, υπάρχουν ισοδύναμοι πιο «συνοπτικοί » τελεστές
  - `a += c` ⟷ `a = a+c`
  - `a *= scale` ⟷ `a = a*scale`
  - `++a` ⟷ `a += 1` ⟷ `a = a+1`

- γενικά, για ένα δυαδικό τελεστή op ισχύει: `a op= b` ⟷ `a = a op b`
  - ☞ `a+=b`, `a-=b`, `a*=b`, `a/=b`, `a%=b`

- προτιμούμε κατά κανόνα να χρησιμοποιούμε «συνοπτικούς» τελεστές → είναι πιο σαφής, εκφράζουν πιο άμεσα μία ιδέα

Χαρακτηριστικά γλώσσας

Εκφράσεις (expressions) – Μετατροπές

```
1.  5/2                                  // 2, όχι 2.5
2.  5.0/2                                // 2.5, διότι σημαίνει 5.0/(double)2
3.
4.  double d = 2.5;
5.  int i = 2;
6.  double d2 = d/i;                     // d2 == 1.25
7.  int i2 = d/i;                        // i2 == 1
8.
9.  // μετατροπή βαθμών κελσίου σε φαρενάιτ
10. double dc = 10;
11. double df = 9/5 * dc + 32;           // df == 42
12. double df2 = 9.0/5 * dc + 32;        // df2 == 50
```

- μπορούμε να συνδυάσουμε διαφορετικούς τύπους σε εκφράσεις, όμως πρέπει να δίνουμε μεγάλη προσοχή στις μετατροπές τύπων → ώστε να γράφουμε σωστές εκφράσεις

Χαρακτηριστικά γλώσσας
Εντολή (statement)

1. μια έκφραση που ακολουθείται από ένα ερωτηματικό
   - `a = b;`
2. ένας ορισμός
   - `double d2 = 2.5;`
   - `int average = (length * width) / 2;`
3. μία "εντολή ελέγχου" που καθορίζει τη ροή του ελέγχου
   - `if (x == 2) y = 4;`
   - `while (input.equals("Yes")) ...    // do something`
   - `return x;`

❖ εάν δεν καταλαβαίνετε κάποια από τα παραπάνω, σύντομα θα τα καταλάβετε ...

Χαρακτηριστικά γλώσσας
Εντολή (statement) II

- από μία εντολή ζητάμε να έχει κάποιο αποτέλεσμα

```
1.   // παράδειγμα εντολών χωρίς αποτέλεσμα
2.
3.   1 + 2;        // εκτέλεσε μία πρόσθεση, αλλά μη χρησιμοποιήσεις το άθροισμα
4.   a * b;        // εκτέλεσε έναν πολλαπλασιασμό, αλλά μη χρησιμοποιήσεις το γινόμενο
```

  - εντολές χωρίς αποτέλεσμα είναι συνήθως λογικά σφάλματα και οι μεταγλωττιστές συχνά μας προειδοποιούν

- κενή εντολή

```
1.   // παράδειγμα (πιθανής) λανθασμένης χρήσης της κενής εντολής
2.
3.   int x = 0, y = 0;
4.   Scanner input = new Scanner(System.in);
5.   x = input.nextDouble();
6.   if (x == 5);
7.   { y = 3; }            // το y παίρνει την τιμή 3 ανεξαρτήτως της τιμής του x
```

  - πρέπει να προσέχουμε κατά τη χρήση (ή αποφυγή) των κενών εντολών

Χαρακτηριστικά γλώσσας
Εντολή (statement) - Επιλογή

μερικές φορές πρέπει να επιλέξουμε μεταξύ κάποιων εναλλακτικών λύσεων

- στη Java αυτό μπορεί να γίνει με χρήση
  - της εντολής **`if`** ή
  - της εντολής **`switch`**

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - if

```
1.  // εύρεση μέγιστου αριθμού
2.
3.  if (a < b)      // συνθήκη
4.     max = b;
5.  else
6.     max = a;
```

```
1.  // φανάρι κυκλοφορίας
2.
3.  if (traffic_light == green) go();
4.  if (traffic_light == red) wait();
```

- συντακτικό εντολής **if**:

```
if (condition)
      statement-1     // if the condition is true, do statement-1
else
      statement-2     // if not, do statement-2
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - if - 1ο Παράδειγμα

```
1.  // αναγνώριση θετικών αριθμών
2.
3.  import java.util.Scanner;
4.
5.  class IfTest1
6.  {
7.     public static void main(String args[])
8.     {
9.        Scanner input = new Scanner(System.in);
10.       int value = input.nextInt();
11.
12.       if (value > 0)
13.           System.out.println(value + " is positive");
14.    }
15. }
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - if - 1ο Παράδειγμα II

```
1.  // αναγνώριση θετικών αριθμών
2.
3.  import java.util.Scanner;
4.
5.  class IfTest1
6.  {
7.     public static void main(String args[])
8.     {
9.        Scanner input = new Scanner(System.in);
10.       int value = input.nextInt();
11.
12.       boolean valueIsPositive = (value > 0);
13.       if (valueIsPositive == true)
14.           System.out.println(value + " is positive");
15. }
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - if - 1ο Παράδειγμα III

```
1.  // αναγνώριση θετικών αριθμών
2.
3.  import java.util.Scanner;
4.
5.  class IfTest1
6.  {
7.     public static void main(String args[])
8.     {
9.        Scanner input = new Scanner(System.in);
10.       int value = input.nextInt();
11.
12.       boolean valueIsPositive = (value > 0);
13.       if (valueIsPositive)
14.           System.out.println(value + " is positive");
15. }
```

ακόμη και εάν δεν το προσδιορίσουμε ελέγχει ισότητα με την τιμή true

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - if - 2ο Παράδειγμα

```
1.  // αναγνώριση θετικών αριθμών και μη θετικών αριθμών
2.
3.  import java.util.Scanner;
4.
5.  class IfTest1
6.  {
7.     public static void main(String args[])
8.     {
9.        Scanner input = new Scanner(System.in);
10.       int value = input.nextInt();
11.
12.       if (value > 0)
13.           System.out.println(value + " is positive");
14.       else
15.           System.out.println(value + " not positive");          // αρνητικός ή μηδέν
16.    }
17. }
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - if - 2ο Παράδειγμα II

```
1.  // αναγνώριση θετικών και αρνητικών αριθμών και του μηδενός
2.
3.  import java.util.Scanner;
4.
5.  class IfTest1
6.  {
7.     public static void main(String args[])
8.     {
9.        Scanner input = new Scanner(System.in);
10.       int value = input.nextInt();
11.
12.       if (value > 0)
13.           System.out.println(value + " is positive");
14.       else if (value < 0)
15.           System.out.println(value + " is negative");
16.       else
17.           System.out.println(value + " is zero");
18.    }
19. }
```

Στυλ Προγραμματισμού
Λογικές μεταβλητές

- συνηθίζεται όταν ορίζουμε λογικές μεταβλητές το όνομα τους να είναι αυτό που εκφράζει την περίπτωση που στη μεταβλητή αποθηκεύεται η τιμή `true`

```
int x = 10;
boolean isPositive = (x > 0);
boolean isNegative = (x < 0);
boolean isNotPositive = !isPositive;
```

- αυτό βολεύει για την εύκολη ανάγνωση του προγράμματος όταν χρησιμοποιούμε τη μεταβλητή

```
if (isPositive){
   System.out.println("Variable is positive");
}
```

- το ίδιο ισχύει και όταν αργότερα θα ορίζουμε μεθόδους που επιστρέφουν λογικές τιμές
  - π.χ. για κάποιο `String` υπάρχουν:
    - η μέθοδος `equals` που επιστρέφει την τιμή `true` σε περίπτωση ισότητας (με άλλο `String`) και
    - η μέθοδος `isEmpty` που επιστρέφει την τιμή `true` σε περίπτωση που το `String` είναι άδειο

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - if - ΠΡΟΣΟΧΗ

ΛΑΘΟΣ!

```
if( i == j )
    if ( j == k )
        System.out.print("i equals k");
else
    System.out.print("i is not equal to j");
```

- το (πράσινο) **else** μοιάζει σαν να πηγαίνει με το (μπλε) **if**
- αλλά ταιριάζεται με το τελευταίο (πράσινο) **if**

ΣΩΣΤΟ!

```
if( i == j ){
    if ( j == k ) {
        System.out.print("i equals k");
    }
}
else {
    System.out.print("i is not equal to j”);
}
```

- ένα **if** μπορεί να είναι φωλιασμένο (nested) κάτω από κάποιο άλλο **if**
  - προσοχή: ένα **else** ταιριάζεται με το τελευταίο **if** του ίδιου σώματος (block) κώδικα (ακόμη κι αν η στοίχιση του κώδικα υπονοεί διαφορετικά)
- πάντα να βάζετε **{ }** στο σώμα των **if**, **else if**, **else** εντολών
- πάντα να στοιχίζετε σωστά τον κώδικα

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - if - 3ο Παράδειγμα

```
1.  // Μετατροπή από ίντσες σε εκατοστά, ή το αντίστροφο
2.
3.  import java.util.Scanner;
4.
5.  class InchToCmConverter
6.  {
7.     public static void main(String args[])
8.     {
9.        final double cmPerInch = 2.54;                    // πλήθος cm ανά inch
10.       System.out.println("Please give a distance value and its unit (0 for inch, 1 for cm): ");
11.       Scanner input = new Scanner(System.in);
12.       double dist = input.nextDouble();
13.       int unit = input.nextInt();
14.
15.       if (unit == 0)
16.           System.out.println(dist + "in = " + dist * cmPerInch + "cm");
17.       else
18.           System.out.println(dist + "cm = " + dist / cmPerInch + "in");
19.    }
20. }
```

? παρατηρείτε κάποιο σφάλμα στο παραπάνω πρόγραμμα;

☞ δεν ελέγχεται η περίπτωση λάθος εισόδου (π.χ. `100 5`)

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - if - 3ο Παράδειγμα II

```
1.  // Μετατροπή από ίντσες σε εκατοστά, ή το αντίστροφο
2.
3.  import java.util.Scanner;
4.
5.  class InchToCmConverter
6.  {
7.     public static void main(String args[])
8.     {
9.        final double cmPerInch = 2.54;                    // πλήθος cm ανά inch
10.       System.out.println("Please give a distance value and its unit (0 for inch, 1 for cm): ");
11.       Scanner input = new Scanner(System.in);
12.       double dist = input.nextDouble();
13.       int unit = input.nextInt();
14.
15.       if (unit == 0)
16.           System.out.println(dist + "in = " + dist * cmPerInch + "cm");
17.       else if (unit == 1)
18.           System.out.println(dist + "cm = " + dist / cmPerInch + "in");
19.       else
20.           System.out.println("Sorry, unit code " + unit + " is unknown!");
21.    }
22. }
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - if II

- συντακτικό εντολής **if**:

```
if (condition-1)
    statement-1          // if the condition-1 is true, do statement-1
else if (condition-2)
    statement-2          // if the condition-2 is true, do statement-2
else if ...
...
else
    statement-3          // otherwise, do statement-3
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - switch - 1ο Παράδειγμα

```
1.  // Μετατροπή από ίντσες σε εκατοστά, ή το αντίστροφο
2.
3.  import java.util.Scanner;
4.
5.  class InchToCmConverter
6.  {
7.     public static void main(String args[])
8.     {
9.        final double cmPerInch = 2.54;                    // πλήθος cm ανά inch
10.       System.out.println("Please give a distance value and its unit (0 for inch, 1 for cm): ");
11.       Scanner input = new Scanner(System.in);
12.       double dist = input.nextDouble();
13.       int unit = input.nextInt();
14.
15.       switch (unit) {
16.           case 0:
17.               System.out.println(dist + "in = " + dist * cmPerInch + "cm");
18.               break;
19.           case 1:
20.               System.out.println(dist + "cm = " + dist / cmPerInch + "in");
21.               break;
22.           default:
23.               System.out.println("Sorry, unit code " + unit + " is unknown!");
24.               break;
25.       }
26.    }
27. }
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - switch

- συντακτικό εντολής **switch**:

```
switch (value) {
      case val-1:
         statement-1          // if value == val-1, do statement-1
         break;
      case val-2:
         statement-2          // if value == val-2, do statement-2
         break;
      case ...
      ...
      default:
         statement
         break;
}
```

Χαρακτηριστικά γλώσσας
Εντολή (statement) - Επιλογή - switch - Τεχνικές λεπτομέρειες

- η τιμή την οποία πρέπει να επιλέξουμε πρέπει να είναι τύπου:
  - **`byte`**, **`short`**, **`char`**, **`int`**,
  - απαρίθμηση (**`enum`**)
  - **`string`**
- οι τιμές στο `case` πρέπει να είναι σταθερές εκφράσεις
- δε μπορούμε να χρησιμοποιήσουμε την ίδια τιμή για δύο `case`
- μπορούμε να αντιστοιχούμε πολλαπλά `case` σε ένα σύνολο εντολών
- ο μεταγλωττιστής δε θα μας προειδοποιήσει εάν ξεχάσουμε κάποιο `break`

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - switch - 2ο Παράδειγμα

```
1.  // αναγνώριση άρτιων και περιττών ψηφίων
2.
3.  import java.util.Scanner;
4.
5.  class EvenOrOddNumber
6.  {
7.     public static void main(String args[])
8.     {
9.        System.out.println("Please insert a digit ");
10.       Scanner input = new Scanner(System.in);
11.       int digit = input.nextInt();
12.       switch (digit) {
13.               case 0: case 2: case 4: case 6: case 8:
14.                        System.out.println("it is odd");
15.                        break;
16.               case 1: case 3: case 5: case 7: case 9:
17.                        System.out.println("it is even");
18.                        break;
19.               default:
20.                        System.out.println("not a digit");
21.                        break;
22.       }
23.    }
24. }
```

σημείωση

ο κώδικας που παρουσιάζεται

- αποτελεί ένα παράδειγμα χρήσης της εντολής **switch**
- δεν αποτελεί την καλύτερη λύση του προβλήματος αναγνώρισης άρτιων και περιττών αριθμών (ή ψηφίων)

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - switch - 3ο Παράδειγμα

αναπτύξτε ένα πρόγραμμα που

- εύχεται "καλημέρα"
- σε τρεις διαφορετικές γλώσσες
- ανάλογα με την επιλογή του χρήστη
  - "GR"
  - "EN"
  - "FR"

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - switch - 3ο Παράδειγμα

```
1.  // Καλημέρα σε τρεις διαφορετικές γλώσσες
2.
3.  import java.util.Scanner;
4.
5.  class MultiLanguageGoodMorning
6.  {
7.      public static void main(String args[])
8.      {
9.          Scanner input = new Scanner(System.in);
10.         String option = input.next();
11.
12.         switch (option) {
13.             case "GR":                                  // if (option.equals("GR"))
14.                 System.out.println("Kalimera!");
15.                 break;
16.             case "EN":                                  // else if (option.equals("EN"))
17.                 System.out.println("Good morning!");
18.                 break;
19.             case "FR":                                  // else if (option.equals("FR"))
20.                 System.out.println("Bonjour!");
21.                 break;
22.             default:                                    // else
23.                 System.out.println("I do not speak this language.\n Greek, English, and French only.");
24.                 break;
25.         }
26.     }
27. }
```

Πώς μπορούμε να υποστηρίξουμε και τα μικρά γράμματα;

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - switch - 3ο Παράδειγμα II

```
1.  // Καλημέρα σε τρεις διαφορετικές γλώσσες
2.
3.  import java.util.Scanner;
4.
5.  class MultiLanguageGoodMorning
6.  {
7.     public static void main(String args[])
8.     {
9.        Scanner input = new Scanner(System.in);
10.       String option = input.next();
11.
12.       switch (option) {
13.           case "GR":
14.           case "gr":
15.               System.out.println("Kalimera!");
16.               break;
17.           case "EN": case "en":
18.               System.out.println("Good morning!");
19.               break;
20.           case "FR": case "fr":
21.               System.out.println("Bonjour!");
22.               break;
23.           default:
24.               System.out.println("I do not speak this language.\n Greek, English, and French only.");
25.               break;
26.       }
27.    }
28. }
```

Πώς μπορούμε να υποστηρίξουμε ανάμεικτα μικρά και μεγάλα γράμματα;

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επιλογή - switch - 3ο Παράδειγμα III

```
1.  // Καλημέρα σε τρεις διαφορετικές γλώσσες
2.
3.  import java.util.Scanner;
4.
5.  class MultiLanguageGoodMorning
6.  {
7.      public static void main(String args[])
8.      {
9.          Scanner input = new Scanner(System.in);
10.         String option = input.next();
11.
12.         switch (option.toLowerCase()) {
13.             case "gr":
14.                 System.out.println("Kalimera!");
15.                 break;
16.             case "en":
17.                 System.out.println("Good morning!");
18.                 break;
19.             case "fr":
20.                 System.out.println("Bonjour!");
21.                 break;
22.             default:
23.                 System.out.println("I do not speak this language.\n Greek, English, and French only.");
24.                 break;
25.         }
26.     }
27. }
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επανάληψη - while

- το πρώτο πρόγραμμα που έτρεξε σε έναν υπολογιστή (David Wheeler, University of Cambridge, 1949)
  - *(όχι, δε γράφτηκε σε Java ☺)*

```
1.  // υπολογισμός και εκτύπωση ενός πίνακα τετραγώνων για τους αριθμούς 0-99
2.
3.  class FirstComputerProgram
4.  {
5.     public static void main(String args[])
6.     {
7.        int i = 0;                                          // αρχική τιμή μεταβλητής βρόχου, η οποία
8.                                                  // παρακολουθεί πόσες φορές έχει εκτελεστεί ο βρόχος
9.        while (i < 100 ) {                                  // συνθήκη τερματισμού βρόχου
10.           System.out.println(i + '\t' + square(i) + '\n');  // εντολή βρόχου
11.           ++i;                                            // αύξηση μεταβλητής βρόχου κατά 1
12.       }
13.    }
14. }
```

Χαρακτηριστικά γλώσσας
Εντολή (statement) - Επανάληψη - while II

- τι χρειαζόμαστε:
  - μία μεταβλητή βρόχου (ή μεταβλητή ελέγχου); `i`
  - αρχικοποίηση μεταβλητής βρόχου; `int i = 0;`
  - κριτήριο τερματισμού; `if i<100 is false, terminate`
  - αύξηση μεταβλητής βρόχου; `++i;`
  - κάτι να κάνουμε σε κάθε επανάληψη `System.out.println(…);`

```
1.      int i = 0;
2.      while (i < 100 ) {
3.         System.out.println(i + '\t' + square(i) + '\n');
4.         ++i;
5.      }
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Μπλοκ ή σύνθετη εντολή

μία ακολουθία από εντολές που ορίζονται από τα άγκιστρα `{` και `}`

- ένα μπλοκ είναι ένα είδος εντολής
  - π.χ. η ακολουθία εντολών που εκτελείται από το `while`

```
1.      while (i < 100 ) {
2.         System.out.println(i + '\t' + square(i) + '\n');
3.         ++i;
4.      }
```

- το κενό μπλοκ είναι χρήσιμο μερικές φορές για να εκφράσουμε ότι τίποτα δεν πρέπει να γίνει

```
1.      if (a <= b) {      // μην κάνεις τίποτα
2.      }
3.      else {             // αντιμετάθεσε τα a και b
4.         int t = a;
5.         a = b;
6.         b = t;
7.      }
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - while & μπλοκ ή σύνθετη εντολή - Παράδειγμα

```
1.  // Επανάληψη όσο ο χρήστης πληκτρολογεί "YES"
2.
3.  import java.util.Scanner;
4.
5.  class WhileDoBlockCodeExample
6.  {
7.     public static void main(String args[])
8.     {
9.        Scanner input = new Scanner(System.in);
10.
11.       System.out.println("Do you want to continue?");
12.       String option = input.next();
13.
14.       while (option.equals("YES")) {
15.              System.out.println("Do you want to continue?");
16.              option = input.next();
17.       }
18.    }
19. }
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - do…while & μπλοκ ή σύνθετη εντολή - Παράδειγμα

```
1.  // Επανάληψη όσο ο χρήστης πληκτρολογεί "YES"
2.
3.  import java.util.Scanner;
4.
5.  class DoWhileBlockCodeExample
6.  {
7.     public static void main(String args[])
8.     {
9.        Scanner input = new Scanner(System.in);
10.
11.       do {
12.               System.out.println("Do you want to continue?");
13.               String option = input.next();
14.       } while (option.equals("YES"));
15.    }
16. }
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - Επανάληψη - for

- η εντολή **for** είναι παρόμοια με την εντολή **while**, με τη διαφορά ότι η διαχείριση της μεταβλητής ελέγχου επικεντρώνεται στην αρχή → ευκολότερο να καταλάβουμε τι συμβαίνει

```
1.    for (int i = 0; i < 100; ++i)
2.      System.out.println(i + '\t' + square(i) + '\n');
```

```
1.    int i = 0;
2.      while (i < 100 ) {
3.        System.out.println(i + '\t' +
                    square(i) + '\n');
4.        ++i;
5.      }
```

- συντακτικό εντολής **for**:

```
for (initialize; condition; increment)
    statement
```

Χαρακτηριστικά γλώσσας
Εντολή (statement) - Επανάληψη - for

- ? *παρατηρείτε κάποιο πρόβλημα στον παρακάτω κώδικα;*

```
1.  for (int i = 0; i < 100; ++i)       // για κάθε i στην περιοχή [0:100)
2.  {
3.     System.out.println(i + '\t' + square(i) + '\n');
4.     ++i;
5.  }
```

- δεν τροποποιούμε ποτέ την εντολή βρόχου στο μπλοκ της **for** → δυσνόητος κώδικας
  - εάν θέλετε να αυξήσετε το μετρητή κατά 2, πείτε το!

```
1.    // υπολογισμός και εκτύπωση ενός πίνακα τετραγώνων άρτιων αριθμών στην περιοχή [0:100)
2.
3.    for (int i = 0; i < 100; i+=2)
4.       System.out.println(i + '\t' + square(i) + '\n');
```

Άσκηση

αναπτύξτε ένα πρόγραμμα που:

- παίρνει σαν είσοδο ένα αριθμό και υλοποιεί μια αντίστροφη μέτρηση
- αν ο αριθμός είναι θετικός, η αντίστροφη μέτρηση γίνεται προς τα κάτω μέχρι το μηδέν
- αν ο αριθμός είναι αρνητικός, η αντίστροφη μέτρηση γίνεται προς τα πάνω μέχρι το μηδέν
- η διαδικασία επαναλαμβάνεται μέχρι ο χρήστης να δώσει την τιμή μηδέν

Χαρακτηριστικά γλώσσας
Εντολή (statement) - break & continue

`break`

- έχει ως αποτέλεσμα την έξοδο του προγράμματος από τον (πιο εσωτερικό) βρόχο ή switch στο οποίο αυτή περιέχεται

`continue`

- επιστρέφει τη ροή του προγράμματος στον έλεγχο της συνθήκης σε ένα βρόχο (τον πιο εσωτερικό)
- βολικό για
  1. τον έλεγχο συνθηκών πριν ξεκινήσει η εκτέλεση του βρόχου, ή
  2. για πρόωρη επιστροφή στον έλεγχο της συνθήκης

✍ βρόχοι: **`for`**, **`while`**, **`do … while`**

Χαρακτηριστικά γλώσσας

Εντολή (statement) - break - 1ο Παράδειγμα

```
while (… && !StopFlag)
{
    <some code>

     if (I should stop){
         StopFlag = true;
     }else{
         <some more code>
     }
} // end of while loop
```

```
while (…)
{
   <some code>

   if (I should stop){
       break;
   }

   < some more code>
} // end of while loop
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - continue - Παράδειγμα

```
while (…)
{
    if (everything is ok){

        < rest of code>

    } // end of if
} // end of while loop
```

```
while (…)
{
  if (I don’t like something){
      continue;
  }

  < rest of code>

} // end of while loop
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - break - 2ο Παράδειγμα

```
1.  // Παράδειγμα χρήστης break
2.
3.  class Break2ndExample
4.  {
5.     public static void main(String args[])
6.     {
7.        int counter = 0;
8.
9.        while (true) {                    // ατέρμων βρόχος
10.           counter++;
11.           if (counter == 100)           // αύξηση counter έως το 100
12.              break;                     // έξοδος από τον ατέρμων βρόχο
13.       }
14.    }
15. }
```

Χαρακτηριστικά γλώσσας

Εντολή (statement) - break - 3ο Παράδειγμα

```
1.  // Παράδειγμα χρήστης break
2.
3.  class Break3rdExample
4.  {
5.     public static void main(String args[])
6.     {
7.        int counter1 = 0, counter2 = 0;
8.
9.        while (true) {                        // 1ος ατέρμων βρόχος
10.           counter1++;
11.           if (counter1 == 100)              // αύξηση counter1 έως το 100
12.               break;                        // έξοδος από τον 1ο ατέρμων βρόχο
13.
14.           counter2 = 0;
15.           while (true) {                    // 2ος ατέρμων βρόχος
16.               counter2+=2;
17.               if (counter2 == 200)          // αύξηση counter2 έως το 200
18.                   break;                    // έξοδος από τον 2ο ατέρμων βρόχο
19.            }
20.       }
21.    }
22. }
```

Χαρακτηριστικά γλώσσας
Συνάρτηση (function)

- ? τι ήταν το `square(i)` στα προηγούμενα παραδείγματα;
  - μία κλήση στη συνάρτηση `square(i)`

```
1.    int square (int x)        // τύπος επιστρεφόμενης τιμής, όνομα, παράμετροι
2.    {
3.       return x * x;          // επιστροφή τιμής (τύπου int)
4.    }
```

- ορίζουμε μια συνάρτηση όταν θέλουμε να χωρίσουμε έναν υπολογισμό επειδή
  - είναι λογικά ξεχωριστός
  - καθιστά το κείμενο του προγράμματος σαφέστερο (με την ονομασία του υπολογισμού)
  - είναι χρήσιμος σε περισσότερες από μία θέσεις στο πρόγραμμά μας
  - διευκολύνει τη δοκιμή, τη διανομή εργασίας και τη συντήρηση

Χαρακτηριστικά γλώσσας

Συνάρτηση (function) - 1ο Παράδειγμα

```
1.      int square (int x)              // τύπος επιστρεφόμενης τιμής, όνομα, παράμετροι
2.      {
3.         return x * x;                // επιστροφή τιμής (τύπου int)
4.      }
```

ορθή χρήση

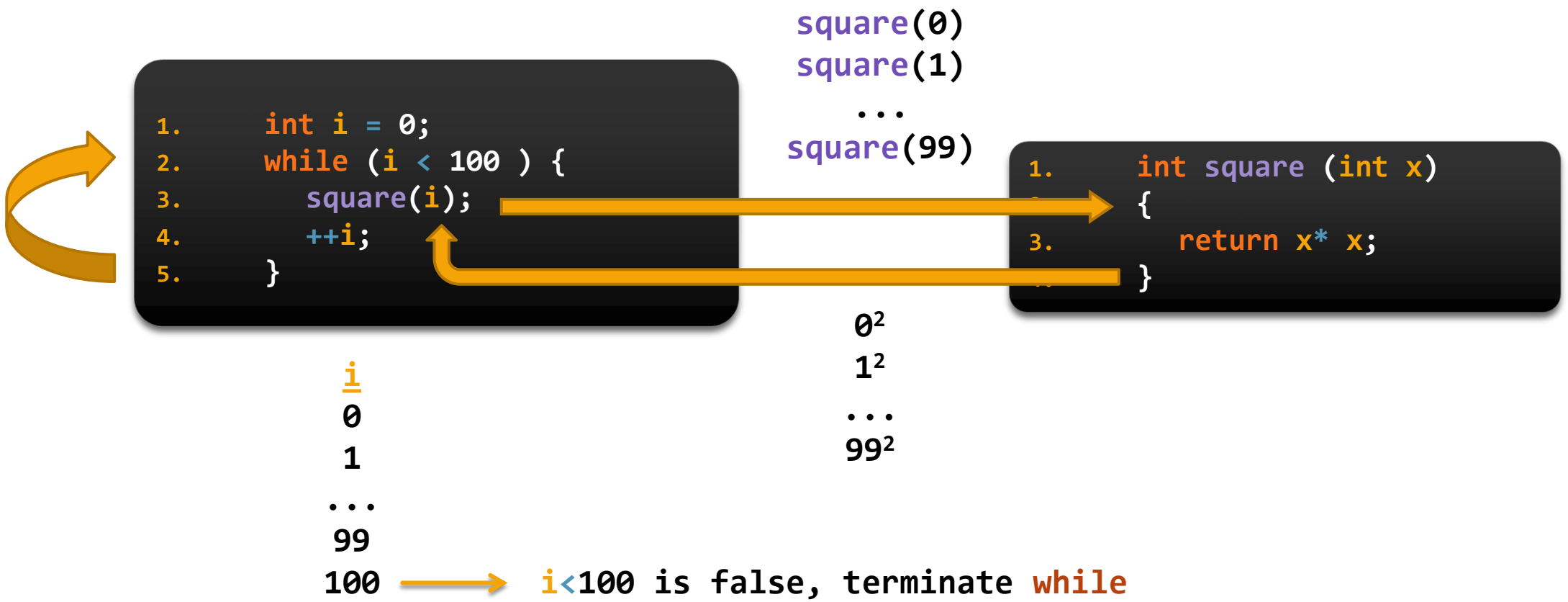
```
1.      public static void main(String args[])
2.      {
3.         System.out.println(square(2));
4.         System.out.println(square(10));
5.         square(20);          // πιθανώς σφάλμα: τιμή επιστροφής που δε χρησιμοποιείται
6.      }
```

λάθος χρήση

```
1.      public static void main(String args[])
2.      {
3.         int v1 = square();              // σφάλμα: λείπει όρισμα
4.         int v2 = square;                // σφάλμα: λείπουν παρενθέσεις
5.         int v3 = square(1,2);           // σφάλμα: πάρα πολλά ορίσματα
6.         int v4 = square("two");         // σφάλμα: λάθος τύπος ορίσματος
7.      }
```

Σειριακή εκτέλεση

Ροή ελέγχου (Control flow)



Χαρακτηριστικά γλώσσας
Συνάρτηση (function) II

- συντακτικό συνάρτησης

```
return_type function_name (parameter list) // (type name, etc.)
{
    // use each parameter in code
    return some_value;     // of type return_type
}
```

  - παράδειγμα

```
1.    int square (int x)          // τύπος επιστρεφόμενης τιμής, όνομα, παράμετροι
2.    {
3.       return x * x;            // επιστροφή τιμής (τύπου int)
4.    }
```

Χαρακτηριστικά γλώσσας

Συνάρτηση (function) - 2ο Παράδειγμα

```
1.  // υπολογισμός του μέγιστου ακεραίου, με χρήση συνάρτησης
2.
3.  class MaxNumber
4.  {
5.     int max (int a, int b)                  // κώδικας συνάρτησης max
6.     {
7.        if (a < b)
8.           return b;
9.        else
10.          return a;
11.    }
12.
13.    public static void main(String args[])
14.    {
15.       int x = max(7,9);                    // το x παίρνει την τιμή 9
16.       int y = max(19,-27);                 // το y παίρνει την τιμή 19
17.       int z = max(20,20);                  // το z παίρνει την τιμή 20
18.    }
19. }
```

Χαρακτηριστικά γλώσσας
Εμβέλεια (scope) μεταβλητών

- κάθε μεταβλητή που ορίζουμε έχει εμβέλεια (scope) μέσα στο block το οποίο ορίζεται
  - τοπική μεταβλητή μέσα στο block
- μόλις βγούμε από το block η μεταβλητή χάνεται
  - ο μεταγλωττιστής (compiler) δημιουργεί ένα χώρο στη μνήμη για το block το οποίο εκτελούμε, ο οποίος εξαφανίζεται όταν το block τελειώσει
- ένα block μπορεί να περιλαμβάνει κι άλλα φωλιασμένα blocks
  - η μεταβλητή έχει εμβέλεια και μέσα στα φωλιασμένα blocks
  - δε μπορούμε να ορίσουμε μια άλλη μεταβλητή με το ίδιο όνομα σε ένα φωλιασμένο block

Χαρακτηριστικά γλώσσας

Εμβέλεια (scope) μεταβλητών - 1ο Παράδειγμα

```
1.  // Παράδειγμα εμβέλειας (scope) μεταβλητών
2.
3.  class ScopeExample
4.  {
5.     public static void main(String args[])
6.     {
7.        int y = 1;
8.        int x = 2;
9.        for (int i = 0; i < 3; i++)
10.       {
11.              y = i;
12.              double x = i+1;                    // Σφάλμα! το x έχει δηλωθεί ξανά στο πατρικό scope (γραμμή 8)
13.              int z = x+y;
14.              System.out.println("i = " + i);
15.              System.out.println("y = " + y);
16.              System.out.println("z = " + z);
17.       }
18.       System.out.println("x = " + x);
19.       System.out.println("y = " + y);
20.       System.out.println("z = " + z);          // Σφάλμα! το z δεν έχει δηλωθεί στο τρέχων scope
21.       System.out.println("i = " + i);          // Σφάλμα! το i δεν έχει δηλωθεί στο τρέχων scope
22.    }
23. }
```

Χαρακτηριστικά γλώσσας

Εμβέλεια (scope) μεταβλητών - 2ο Παράδειγμα

```
public static void main(String[] args)
{
  ... ...
  {
    ... ...
    {
        int y = 1;
        ... ...
        {
            {
                ... ...
            }
            ... ...
        }
         ... ...
    }
    ... ...
  }
  ... ...
}
```

το μπλέ είναι ο χώρος εκτός της εμβελείας του **y**

- στο **μπλε** δε μπορούμε να χρησιμοποιήσουμε τη μεταβλητή **y**

Η εμβέλεια του **y**

στο **κίτρινο**

- μπορούμε να χρησιμοποιήσουμε την μεταβλητή **y**
- δε μπορούμε να ορίσουμε άλλη μεταβλητή με το όνομα **y**

- κάθε block έχει το δικό του χώρο μνήμης
- σε ένα χώρο μνήμης μια μεταβλητή μπορεί να οριστεί μόνο μία φορά
- τα φωλιασμένα blocks έχουν και τις μεταβλητές των προγόνων

Σύνοψη

- υπολογισμός
  - τι είναι υπολογίσιμο;
  - αφαίρεση, αλγόριθμοι, δομές δεδομένων
- χαρακτηριστικά γλώσσας
  - σειριακή εκτέλεση
  - εκφράσεις και εντολές
  - επιλογή
  - επανάληψη
  - συνάρτηση
  - εμβέλεια μεταβλητών