



Προγραμματισμός Υπολογιστών & Υπολογιστική Φυσική

Ενότητα 5: Δομές Επανάληψης

Νικόλαος Στεργιούλας
Τμήμα Φυσικής



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο



ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο την 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.



ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ & ΥΠΟΛΟΓΙΣΤΙΚΗ ΦΥΣΙΚΗ

Μέρος 5ο

ΝΙΚΟΛΑΟΣ ΣΤΕΡΓΙΟΥΛΑΣ



ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Η ΕΝΤΟΛΗ `for`

Με την εντολή `for` δημιουργούμε βρόχους επανάληψης σε ένα πρόγραμμα. Η σύνταξη είναι:

```
for ( έκφραση_1; συνθήκη; έκφραση_2 )  
{  
  ...  
  /* ομάδα εντολών */  
  ...  
}
```

Για όσο η `συνθήκη = ΑΛΗΘΗΣ`, εκτελείται ξανά και ξανά η ομάδα εντολών μέσα στα άγκιστρα.

Αν κάποια στιγμή η `συνθήκη = ΨΕΥΔΗΣ`, τότε σταματά η εκτέλεση της ομάδας εντολών μέσα στα άγκιστρα.

Η ΕΝΤΟΛΗ `for`

Η πιο συχνή χρήση της `for` γίνεται με κάποια **ακέραια μεταβλητή**. Για παράδειγμα:

```
for ( i=1; i<=10; i++ )  
{  
  ...  
  /* ομάδα εντολών */  
  ...  
}
```

αρχικοποίηση
της
μεταβλητής

έλεγχος
συνθήκης

αλλαγής της
τιμής με κάθε
επανάληψη

Στο παραπάνω παράδειγμα, η μεταβλητή `i` αρχικά λαμβάνει την τιμή `1`. Μετά από κάθε επανάληψη, η τιμή της `i` αυξάνει κατά `1`. Ο βρόχος επαναλαμβάνεται για όσο ισχύει η συνθήκη `i<=10` (άρα ο βρόχος θα εκτελεστεί συνολικά `10` φορές).

ΠΑΡΑΔΕΙΓΜΑ: ΑΥΞΗΣΗ ΚΑΤΑ 1

```
#include <stdio.h>

int main(void)
{
    int i;

    for ( i = 1; i <= 3; i++ )
    {
        printf("Η τιμή της μεταβλητής είναι i = %d.\n", i);
    }

    printf("\n");
    printf("Οι επαναλήψεις σταμάτησαν, διότι\n");
    printf("η τιμή της μεταβλητής έγινε i = %d.\n", i);

    return 0;
}
```

Η τιμή της μεταβλητής είναι i = 1.

Η τιμή της μεταβλητής είναι i = 2.

Η τιμή της μεταβλητής είναι i = 3.

Οι επαναλήψεις σταμάτησαν, διότι

η τιμή της μεταβλητής έγινε i = 4.

ΠΑΡΑΔΕΙΓΜΑ: ΜΕΙΩΣΗ ΚΑΤΑ 1

```
#include <stdio.h>

int main(void)
{
    int i;

    for ( i = 3; i >= 1; i-- )
    {
        printf("Η τιμή της μεταβλητής είναι i = %d.\n", i);
    }

    printf("\n");
    printf("Οι επαναλήψεις σταμάτησαν, διότι\n");
    printf("η τιμή της μεταβλητής έγινε i = %d.\n", i);

    return 0;
}
```

Η τιμή της μεταβλητής είναι $i = 3$.
Η τιμή της μεταβλητής είναι $i = 2$.
Η τιμή της μεταβλητής είναι $i = 1$.

Οι επαναλήψεις σταμάτησαν, διότι
η τιμή της μεταβλητής έγινε $i = 0$.

ΠΑΡΑΔΕΙΓΜΑ: ΑΥΞΗΣΗ ΚΑΤΑ 2

```
#include <stdio.h>

int main(void)
{
    int i;

    for ( i = 1; i <= 9; i += 2 )
    {
        printf("Η τιμή της μεταβλητής είναι i = %d.\n", i);
    }

    printf("\n");
    printf("Οι επαναλήψεις σταμάτησαν, διότι\n");
    printf("η τιμή της μεταβλητής έγινε i = %d.\n", i);

    return 0;
}
```

Η τιμή της μεταβλητής είναι i = 1.
Η τιμή της μεταβλητής είναι i = 3.
Η τιμή της μεταβλητής είναι i = 5.
Η τιμή της μεταβλητής είναι i = 7.
Η τιμή της μεταβλητής είναι i = 9.

Οι επαναλήψεις σταμάτησαν, διότι
η τιμή της μεταβλητής έγινε i = 11.

ΠΑΡΑΔΕΙΓΜΑ: ΜΕΙΩΣΗ ΚΑΤΑ 2

```
#include <stdio.h>

int main(void)
{
    int i;

    for ( i = 9; i >= 1; i -= 2 )
    {
        printf("Η τιμή της μεταβλητής είναι i = %d.\n", i);
    }

    printf("\n");
    printf("Οι επαναλήψεις σταμάτησαν, διότι\n");
    printf("η τιμή της μεταβλητής έγινε i = %d.\n", i);

    return 0;
}
```

Η τιμή της μεταβλητής είναι i = 9.

Η τιμή της μεταβλητής είναι i = 7.

Η τιμή της μεταβλητής είναι i = 5.

Η τιμή της μεταβλητής είναι i = 3.

Η τιμή της μεταβλητής είναι i = 1.

Οι επαναλήψεις σταμάτησαν, διότι
η τιμή της μεταβλητής έγινε i = -1.

Η ΕΝΤΟΛΗ `for`

Αν στην εντολή `for` η συνθήκη είναι εξ' αρχής **ΨΕΥΔΗΣ**, τότε δε θα εκτελεστεί **καμία φορά** η ομάδα εντολών μέσα στα άγκιστρα. Π.χ.

```
for ( i=1; i>5; i++ )  
{  
    ... /* δε θα εκτελεστεί */  
}
```

ΕΞ' ΑΡΧΗΣ ΨΕΥΔΗΣ
διότι η αρχική τιμή είναι
 $i = 1 < 5$

Η ΕΝΤΟΛΗ `for`

Αν στην εντολή `for` η συνθήκη είναι πάντοτε **ΑΛΗΘΗΣ**, τότε οι επαναλήψεις **δεν τερματίζονται ποτέ** (ατέρμονος βρόχος). Π.χ.

```
for ( i=1; i>0; i++ )  
{  
... /* θα επαναλαμβάνονται συνεχώς */  
}
```

ΠΑΝΤΑ ΑΛΗΘΗΣ
διότι $i = 1, 2, 3, \dots > 0$

Η ΕΝΤΟΛΗ `for`

Τα ορίσματα της εντολής `for` μπορούν να περιέχουν περισσότερες από μία εντολές, χωρισμένες με κόμμα.

Π.χ.

```
for ( i=1, j=5; i*j <= 8; i++, j-- )  
{  
  ...  
  /* ομάδα εντολών */  
  ...  
}
```

δύο
εντολές
αρχικοποίησης

συνθήκη
δύο
μεταβλητών

αλλαγής της
τιμής σε δύο
μεταβλητές

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int i, j;

    for ( i = 1, j = 5; (i*j >= -20) && (i*j <= 10); i++, j-- )
        printf("Οι τιμές είναι i = %d, j = %d, i*j = %d.\n",
               i, j, i*j);

    printf("\n");
    printf("Οι επαναλήψεις σταμάτησαν, διότι i*j = %d.\n", i*j);

    return 0;
}
```

Οι τιμές είναι i = 1, j = 5, i*j = 5.
Οι τιμές είναι i = 2, j = 4, i*j = 8.
Οι τιμές είναι i = 3, j = 3, i*j = 9.
Οι τιμές είναι i = 4, j = 2, i*j = 8.
Οι τιμές είναι i = 5, j = 1, i*j = 5.
Οι τιμές είναι i = 6, j = 0, i*j = 0.
Οι τιμές είναι i = 7, j = -1, i*j = -7.
Οι τιμές είναι i = 8, j = -2, i*j = -16.

Οι επαναλήψεις σταμάτησαν, διότι i*j = -27.

Η ΕΝΤΟΛΗ `break`

Η εντολή `break` χρησιμοποιείται για τον άμεσο τερματισμό ενός βρόχου επανάληψης, μέσω του ελέγχου κάποιας συνθήκης:

```
for ( έκφραση_1; συνθήκη; έκφραση_2 )  
{  
    ...  
    if (συνθήκη)  
        break;  
    ...  
}
```

άμεσος
τερματισμός
του βρόχου for

Αν ικανοποιείται η συνθήκη, η εντολή `break` τερματίζει τις επαναλήψεις και το πρόγραμμα συνεχίζει μετά το βρόχο. ΠΡΟΣΟΧΗ: όσες εντολές έχουν μείνει μέσα στο βρόχο μετά το `break` δεν εκτελούνται!

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int i;

    for ( i = 1; i <= 5 ; i++ )
    {
        printf("Η τιμή της μεταβλητής είναι i = %d.\n", i);

        if ( i == 3 )
            break;

        printf("Δεν έγινε ακόμη break.\n");
    }

    printf("\nΤο πρόγραμμα συνεχίζει εδώ.\n");

    return 0;
}
```

Η τιμή της μεταβλητής είναι i = 1.
Δεν έγινε ακόμη break.
Η τιμή της μεταβλητής είναι i = 2.
Δεν έγινε ακόμη break.
Η τιμή της μεταβλητής είναι i = 3.
Το πρόγραμμα συνεχίζει εδώ.

Η ΕΝΤΟΛΗ `continue`

Η εντολή `continue` χρησιμοποιείται για τον άμεση έναρξη της επόμενης επανάληψης ενός βρόχου, μέσω του ελέγχου κάποιας συνθήκης. Π.χ.

```
for ( έκφραση_1; συνθήκη; έκφραση_2 )  
{  
    ...  
    if (συνθήκη)  
        continue;  
    ...  
}
```

από εδώ πάει
στην αρχή της
επόμενης
επανάληψης

Η εντολή `continue` αναγκάζει το βρόχο να συνεχίσει από την αρχή της επόμενης επανάληψης. Όσες εντολές έχουν μείνει μέσα στο βρόχο μετά το `continue` δεν εκτελούνται στην τρέχουσα επανάληψη!

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int i;

    for ( i = 1; i <= 5 ; i++ )
    {
        printf("Η τιμή της μεταβλητής είναι i = %d.\n", i);

        if ( i == 3 )
            continue;

        printf("Εδώ εκτελούνται εντολές μετά το if.\n");
    }

    printf("\nΤο πρόγραμμα συνεχίζει εδώ.\n");

    return 0;
}
```

Η τιμή της μεταβλητής είναι i = 1.
Εδώ εκτελούνται εντολές μετά το if.
Η τιμή της μεταβλητής είναι i = 2.
Εδώ εκτελούνται εντολές μετά το if.
Η τιμή της μεταβλητής είναι i = 3.
Η τιμή της μεταβλητής είναι i = 4.
Εδώ εκτελούνται εντολές μετά το if.
Η τιμή της μεταβλητής είναι i = 5.
Εδώ εκτελούνται εντολές μετά το if.

Το πρόγραμμα συνεχίζει εδώ.

όταν i=3 δεν
εκτυπώθηκε το
printf μετά το if

ΕΝΘΕΤΟΙ ΒΡΟΧΟΙ `for`

Ένας βρόχος επανάληψης `for` μπορεί να είναι **ένθετος** μέσα σε ένα άλλο βρόχο `for`. Η σύνταξη είναι:

```
for ( έκφραση_1; συνθήκη; έκφραση_2 )  
{
```

`...`

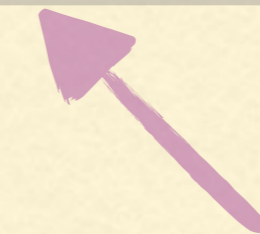
`...`

```
for ( έκφραση_1; συνθήκη; έκφραση_2 )  
{  
    ...  
    ...  
}
```

`...`

`...`

```
}
```



(ένθετος βρόχος)

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int i, j;

    for ( i = 1; i <= 2 ; i++ )
    {
        for ( j = 1; j <= 3 ; j++ )
            printf("i = %d  j = %d  i+j = %d\n", i, j, i+j);
    }

    return 0;
}
```

```
i = 1  j = 1  i+j = 2
i = 1  j = 2  i+j = 3
i = 1  j = 3  i+j = 4
i = 2  j = 1  i+j = 3
i = 2  j = 2  i+j = 4
i = 2  j = 3  i+j = 5
```

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int i, j;

    for ( i = 1; i <= 2 ; i++ )
    {
        for ( j = 1; j <= 3 ; j++ )
            printf("(%d,%d)  ", i, j);

        printf("\n");
    }

    return 0;
}
```

```
(1,1) (1,2) (1,3)
(2,1) (2,2) (2,3)
```

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int i, j;

    for ( i = 1; i <= 5 ; i++ )
    {
        for ( j = 1; j <= i ; j++ )
            printf("(%d,%d)  ", i, j);

        printf("\n");
    }

    return 0;
}
```

```
(1,1)
(2,1) (2,2)
(3,1) (3,2) (3,3)
(4,1) (4,2) (4,3) (4,4)
(5,1) (5,2) (5,3) (5,4) (5,5)
```

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

#define EXPERIMENTS 2
#define MEASUREMENTS 3

int main(void)
{
    int i, j;

    double value, sum;

    for ( i = 1; i <= EXPERIMENTS ; i++ )
    {
        printf("Δώσε τις μετρήσεις για το %do πείραμα:\n", i);

        sum = 0.0;

        for ( j = 1; j <= MEASUREMENTS ; j++ )
        {
            printf("Η %δη μέτρηση είναι: ", j);
            scanf("%lf", &value);
            sum += value;
        }

        printf("\n0 μέσος όρος στο %do πείραμα είναι: %f\n", i, sum/MEASUREMENTS);

        printf("\n");
    }

    return 0;
}
```

ΠΑΡΑΔΕΙΓΜΑ

Εκτελώντας το πρόγραμμα, δίνουμε τις τιμές μία σε κάθε σειρά (πατώντας enter μετά από κάθε τιμή) και παίρνουμε, για παράδειγμα, τα εξής αποτελέσματα:

```
Δώσε τις μετρήσεις για το 1ο πείραμα:
```

```
Η 1η μέτρηση είναι: 3.8
```

```
Η 2η μέτρηση είναι: 4.1
```

```
Η 3η μέτρηση είναι: 3.9
```

```
Ο μέσος όρος στο 1ο πείραμα είναι: 3.933333
```

```
Δώσε τις μετρήσεις για το 2ο πείραμα:
```

```
Η 1η μέτρηση είναι: 9.9
```

```
Η 2η μέτρηση είναι: 9.8
```

```
Η 3η μέτρηση είναι: 10.1
```

```
Ο μέσος όρος στο 2ο πείραμα είναι: 9.933333
```


Η ΕΝΤΟΛΗ `while`

Με την εντολή `while` δημιουργούμε βρόχους επανάληψης βάσει μιας συνθήκης μόνο. Η σύνταξη είναι:

```
while ( συνθήκη )  
{  
  ...  
  ... /* ομάδα εντολών */  
  ...  
}
```

Για όσο η `συνθήκη = ΑΛΗΘΗΣ`, εκτελείται ξανά και ξανά η ομάδα εντολών μέσα στα άγκιστρα.

Αν κάποια στιγμή η `συνθήκη = ΨΕΥΔΗΣ`, τότε σταματά η εκτέλεση της ομάδας εντολών μέσα στα άγκιστρα.

Η ΕΝΤΟΛΗ `while`

Η εντολή `while` χρησιμοποιείται κυρίως όταν δεν γνωρίζουμε εκ των προτέρων τον ακριβή αριθμό των επαναλήψεων που θα χρειαστεί να γίνουν.

Αν η συνθήκη είναι πάντα ΑΛΗΘΗΣ, τότε οι επαναλήψεις δεν σταματούν ποτέ και ο βρόχος είναι ατέρμονος.

Αν η συνθήκη είναι εξ' αρχής ΨΕΥΔΗΣ, τότε οι ομάδα εντολών μέσα στη `while` δε θα εκτελεστεί ποτέ.

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    double value = 0;

    while ( value >= 0 )
    {
        printf("\nΔώσε έναν θετικό αριθμό: ");

        scanf("%lf", &value);

        if( value >= 0 )
            printf("\nΤο τετράγωνο του αριθμού είναι: %f\n", value*value);
    }

    return 0;
}
```

Δώσε έναν θετικό αριθμό: 32.89

Το τετράγωνο του αριθμού είναι: 1081.752100

Δώσε έναν θετικό αριθμό: 12.98

Το τετράγωνο του αριθμού είναι: 168.480400

Δώσε έναν θετικό αριθμό: -1

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int dig = 0, a = 12345678;

    while ( a > 0 )
    {
        a /= 10;

        printf("dig = %d  a = %d\n", dig, a);

        dig++;
    }

    return 0;
}
```

```
dig = 0  a = 12345678
dig = 1  a = 1234567
dig = 2  a = 123456
dig = 3  a = 12345
dig = 4  a = 1234
dig = 5  a = 123
dig = 6  a = 12
dig = 7  a = 1
dig = 8  a = 0
```

ΠΑΡΑΔΕΙΓΜΑ

Γράψτε ένα πρόγραμμα στο οποίο δίνουμε ακέραιους αριθμούς, μέχρι να εισάγουμε 0. Στο τέλος, εκτυπώνεται το πλήθος των θετικών και αρνητικών αριθμών (χωρίς το 0).

```
#include <stdio.h>

int main(void)
{
    int value = 1, positives = 0, negatives = 0;

    while ( value != 0 )
    {
        printf("Δώσε έναν μη-μηδενικό ακέραιο: ");
        scanf("%d", &value);

        if( value == 0)
            break;
        else if ( value > 0 )
            positives++;
        else
            negatives++;
    }

    printf("Το πλήθος των θετικών αριθμών ήταν: %d\n", positives);
    printf("Το πλήθος των αρνητικών αριθμών ήταν: %d", negatives);

    return 0;
}
```

Η ΕΝΤΟΛΗ `do - while`

Η εντολή `while` μπορεί να βρίσκεται στο τέλος ενός βρόχου επανάληψης, εάν στην αρχή υπάρχει η εντολή `do`. Η σύνταξη είναι:

```
do
{
...
... /* ομάδα εντολών */
...
}while ( συνθήκη );
```

Για όσο η `συνθήκη = ΑΛΗΘΗΣ`, εκτελείται ξανά και ξανά η ομάδα εντολών μέσα στα άγκιστρα.

Αν κάποια στιγμή η `συνθήκη = ΨΕΥΔΗΣ`, τότε σταματά η εκτέλεση της ομάδας εντολών μέσα στα άγκιστρα.

Η ΕΝΤΟΛΗ `do - while`

Η εντολή `do - while` χρησιμοποιείται λιγότερο συχνά από την `while` ή την `for` και είναι εναλλακτική αυτών.

Όποιο πρόβλημα λύνεται με τη χρήση της `do - while` μπορεί να λυθεί εναλλακτικά και με την `while` ή την `for`.

Η ομάδα εντολών που υπάρχουν μέσα στα άγκιστρα εκτελούνται **τουλάχιστον μια φορά**, καθώς ο έλεγχος της συνθήκης γίνεται στο τέλος.

Στο τέλος της συνθήκης `while` πρέπει να υπάρχει ;

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    double value;

    do
    {
        printf("\nΔώσε έναν θετικό αριθμό: ");
        scanf("%lf", &value);

        if( value >= 0 )
            printf("\nΤο τετράγωνο του αριθμού είναι: %f\n", value*value);
    } while ( value >= 0 );

    return 0;
}
```

Δώσε έναν θετικό αριθμό: 32.89

Το τετράγωνο του αριθμού είναι: 1081.752100

Δώσε έναν θετικό αριθμό: 12.98

Το τετράγωνο του αριθμού είναι: 168.480400

Δώσε έναν θετικό αριθμό: -1

Η ΕΝΤΟΛΗ goto

Η εντολή `goto` χρησιμοποιείται για να μεταφέρει τη ροή εκτέλεσης ενός προγράμματος, από κάποιο σημείο σε κάποιο άλλο σημείο (της ίδιας όμως συνάρτησης, π.χ. εντός της `main`).

```
goto ΤΟΠΟΘΕΣΙΑ;
```

```
...
```

```
... /* ομάδα εντολών */
```

```
...
```

```
ΤΟΠΟΘΕΣΙΑ:
```

```
...
```

```
... /* ομάδα εντολών */
```

```
...
```

Όταν το πρόγραμμα συναντήσει το `goto ΤΟΠΟΘΕΣΙΑ`, θα συνεχίσει από το μοναδικό σημείο που ορίστηκε με το όνομα `ΤΟΠΟΘΕΣΙΑ:` (είναι απαραίτητη η `:`).

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int i=1;

    double value;

LOCATION1:

    while ( i<=5 )
    {
        printf("Δώσε έναν θετικό αριθμό: ");
        scanf("%lf", &value);

        if( value < 0.0 ) goto LOCATION2;

        printf("Το τετράγωνο του %dου αριθμού είναι: %f \n", i, value*value);
        i++;
    }

LOCATION2:

    if( i <= 5 )
    {
        printf("0 αριθμός ήταν αρνητικός!\n");
        goto LOCATION1;
    }

    return 0;
}
```

ΠΑΡΑΔΕΙΓΜΑ

Το πρόγραμμα συνεχίζει να ρωτά, μέχρι να δώσουμε 5 θετικούς αριθμούς. Αν δώσουμε αρνητικό αριθμό, εκτυπώνει προειδοποίηση.

```
Δώσε έναν θετικό αριθμό: 3.2
Το τετράγωνο του 1ου αριθμού είναι: 10.240000
Δώσε έναν θετικό αριθμό: -1.2
Ο αριθμός ήταν αρνητικός!
Δώσε έναν θετικό αριθμό: 4.3
Το τετράγωνο του 2ου αριθμού είναι: 18.490000
Δώσε έναν θετικό αριθμό: -3.1
Ο αριθμός ήταν αρνητικός!
Δώσε έναν θετικό αριθμό: 4.9
Το τετράγωνο του 3ου αριθμού είναι: 24.010000
Δώσε έναν θετικό αριθμό: 5.8
Το τετράγωνο του 4ου αριθμού είναι: 33.640000
Δώσε έναν θετικό αριθμό: -8.1
Ο αριθμός ήταν αρνητικός!
Δώσε έναν θετικό αριθμό: 8.1
Το τετράγωνο του 5ου αριθμού είναι: 65.610000
```

Η ΕΝΤΟΛΗ `goto`

Επειδή το πρόγραμμα περιπλέκεται αν χρησιμοποιήσουμε πολλές εντολές `goto` το καλύτερο είναι να αποφεύγουμε εντελώς αυτή την εντολή! Η μόνη περίπτωση που δικαιολογείται, είναι για να βγούμε εντελώς μέσα από ένθετους βρόχους, διότι η `break` μας βγάζει μόνο μέσα από ένα βρόχο τη φορά. Π.χ.

```
for ( i = 1; i <= 5; i++ )
{
    ...
    for ( j = 1; j <= 10; j++ )
    {
        ...
        if (συνθήκη) goto ΤΟΠΟΘΕΣΙΑ;
    }
}
ΤΟΠΟΘΕΣΙΑ:
```

Σημείωμα Αναφοράς

Copyright Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, **Νικόλαος Στεργιούλας**
«Προγραμματισμός Υπολογιστών & Υπολογιστική Φυσική». Έκδοση: 1.0.
Θεσσαλονίκη 2014. Διαθέσιμο από τη δικτυακή διεύθυνση:
http://opencourses.auth.gr/eclass_courses.



Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά - Παρόμοια Διανομή [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».



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

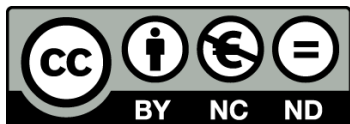
[1] <https://creativecommons.org/licenses/by-nc-nd/4.0/>





Τέλος ενότητας

Επεξεργασία: Νικόλαος Τρυφωνίδης
Θεσσαλονίκη, 20/09/2015



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο



ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης

ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



ΕΣΠΑ
2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

ΣΗΜΕΙΩΜΑΤΑ

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.

