

# Διάλεξη 6 - Εντολές και Ροή Ελέγχου

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Εισαγωγή στον Προγραμματισμό

Θανάσης Αυγερινός / Τάκης Σταματόπουλος

## Ανακοινώσεις / Διευκρινίσεις

- Περισσότερα για τις ακολουθίες [aliquot](#)
- Θέλω να ελέγξω πιο γρήγορα την λύση μου; Δοκίμασε pipes!

```
echo -e "138\n0\nf\n" | ./aliquot
```

- Μπορείς να χρονομετρήσεις την λύση σου

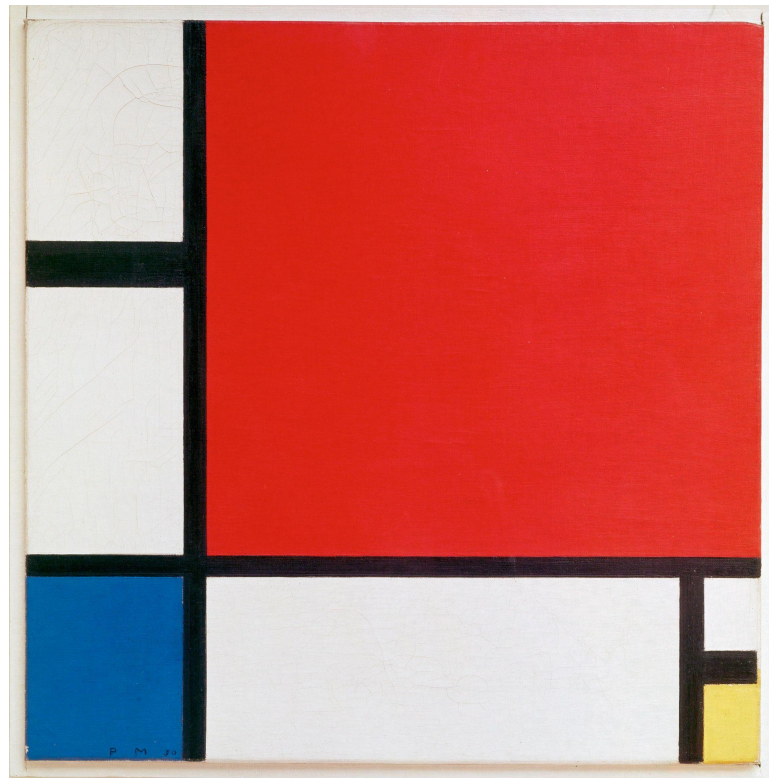
```
echo -e "138\n0\nf\n" | time ./aliquot
```

- Ή ακόμα και να την οπτικοποιήσεις! (μπορεί να χρειαστεί να εγκαταστήσεις το ttyplot)

```
echo -e "2856\n0\nf\n" | ./aliquot | ttyplot
```

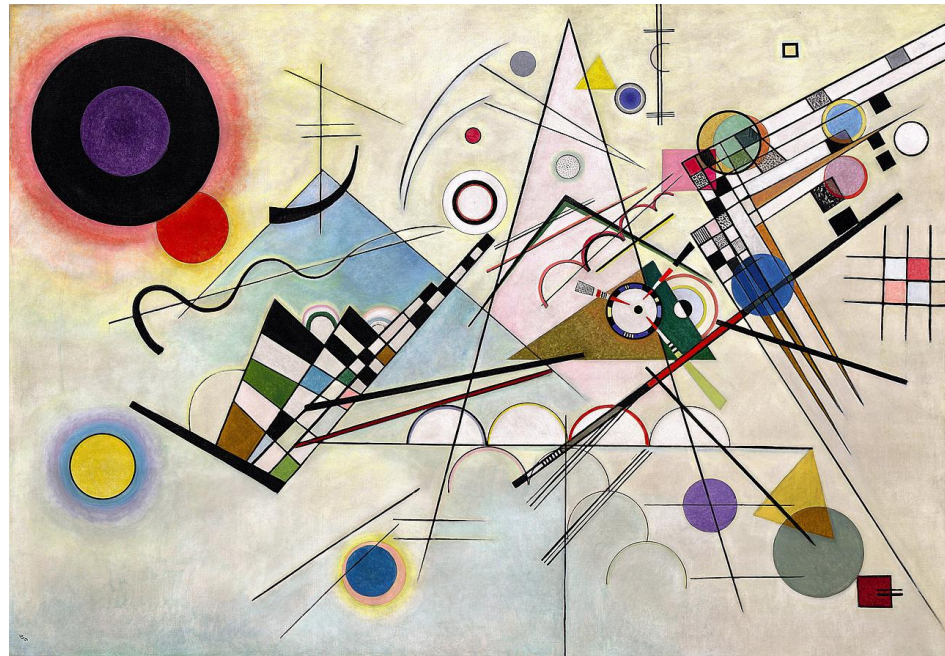
# Την Προηγούμενη Φορά

- Τελεστές
- Εκφράσεις και Εντολές



# Σήμερα

- Εντολές συνέχεια
- Ροή Ελέγχου
- Παραδείγματα Εντολών & Προβλημάτων



# Κατηγοριοποίηση Εντολών

1. Κενές Εντολές
2. Εντολές Έκφρασης
3. Σύνθετες Εντολές
4. Εντολές Συνθήκης
5. Εντολές Επανάληψης

## Κενή Εντολή (Empty / Null Statement)

Η κενή εντολή δεν εκτελεί τίποτε (no-operation ή no-op) και έχει την μορφή:

```
;    // null statement
```

```
;;;;;    // 5 null statements
```

Η χρησιμότητά της φαίνεται σε συνδυασμό με άλλες εντολές.

Σημείωση: χρησιμοποιούμε semicolon (;) ως διαχωριστικό εντολών - για να δείξουμε που τελειώνει η εντολή μας.

# Εντολή Έκφρασης (Expression Statement)

Εντολή έκφρασης στην C ονομάζουμε μια έκφραση που τελειώνει με semicolon (;).  
Παραδείγματα expression statements:

```
x = 4;
```

```
y = 7;
```

```
z = ++y;
```

```
y = z - (x++);
```

```
z = x - (--y);
```

Ποια η τιμή των x, y, z μετά την εκτέλεση αυτών των εντολών έκφρασης;

# Σύνθετη Εντολή (Compound Statement / Block)

Σύνθεση εντολή ή αλλιώς **block**, λέγεται μια σειρά από εντολές όταν περικλείονται από αγκύλες { }.

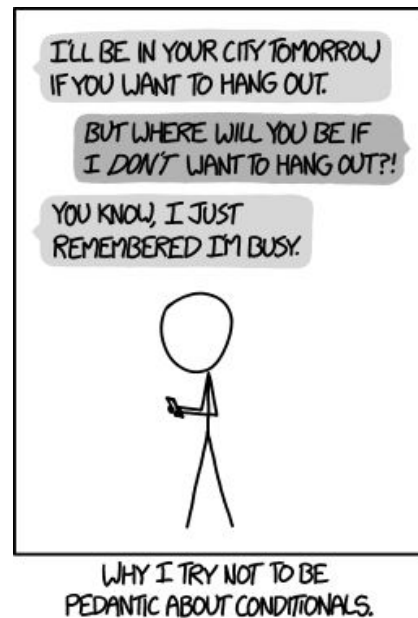
Παράδειγμα:

```
{  
    x = 4;  
    y = 7;  
    z = ++y;  
    y = z - (x++);  
    z = x - (--y);  
}
```



# Εντολές Ροής Ελέγχου (Control Flow Statements)

Η **ροή ελέγχου**, δηλαδή η σειρά με την οποία θα εκτελεστούν οι εντολές σε ένα πρόγραμμα καθορίζεται από τις εντολές ροής ελέγχου. Συνήθως οπτικοποιείται με ένα διάγραμμα.



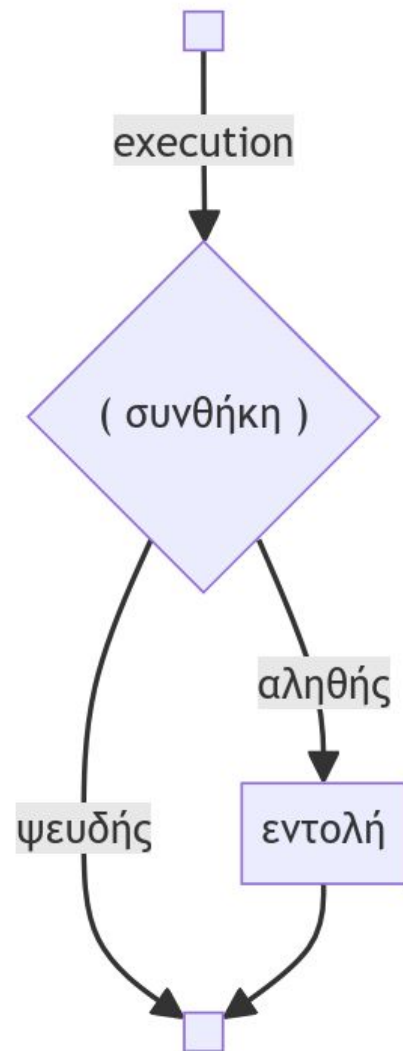
# Εντολή if (if Statement)

Η εντολή if εκτελεί μια εντολή αν μια **λογική** συνθήκη είναι αληθής. Γενική μορφή:

**if ( συνθήκη )**  
**εντολή**

Παραδείγματα:

<pre>if (year &gt; 1)     ;</pre>	<pre>if (year &gt; 1)     year++;</pre>	<pre>if (year &gt; 1) {     year++; }</pre>
-----------------------------------	---	---



# Εντολή if (if Statement)

Η εντολή if εκτελεί μια εντολή αν μια **λογική** συνθήκη είναι αληθής. Γενική μορφή:

**if ( συνθήκη )**  
**εντολή**

**Οι παρενθέσεις είναι υποχρεωτικές!**

Παραδείγματα:

```
if (year > 1)  
    ;
```

```
if (year > 1)  
    year++;
```

```
if (year > 1) {  
    year++;  
}
```



## Εντολή if-else (if-else Statement)

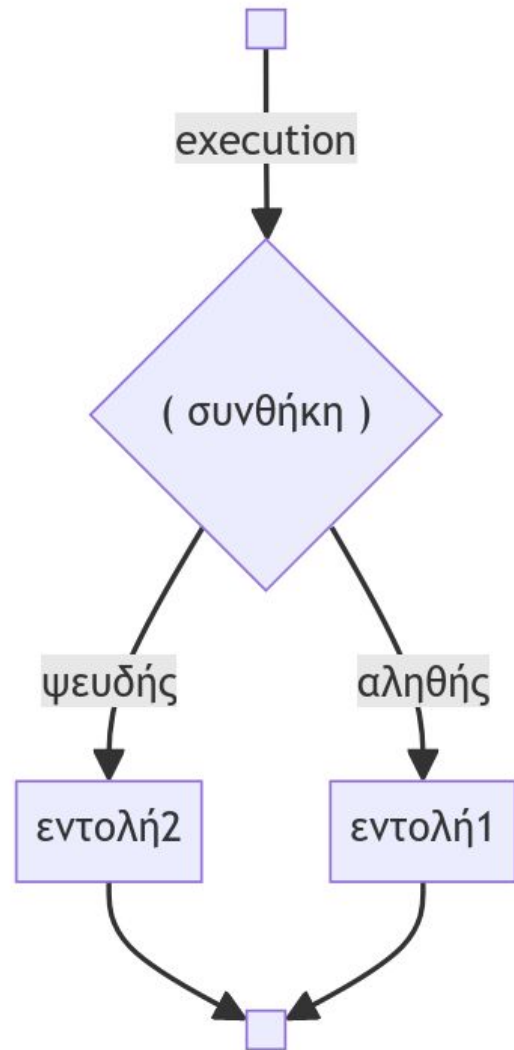
Επέκταση της εντολής if προκειμένου να εκτελέσουμε μια άλλη εντολή αν η λογική συνθήκη είναι ψευδής.

**if ( συνθήκη )**

**εντολή1**

**else**

**εντολή2**



# Εντολή if-else (if-else Statement)

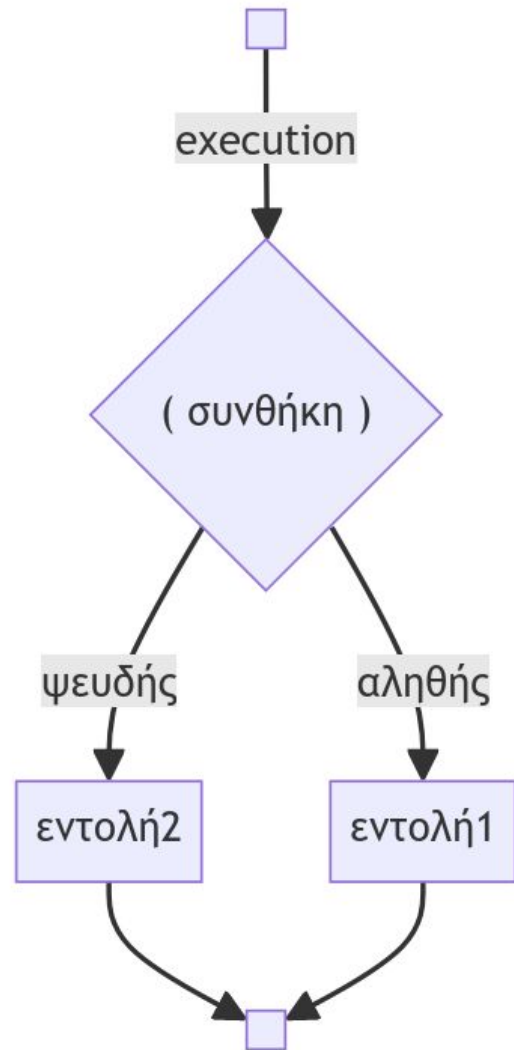
Παραδείγμα:

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

Θα μπορούσαμε να "ζήσουμε" χωρίς else;

Υπάρχει εναλλακτικός τρόπος να γράψουμε το παραπάνω;

Τι κάνουμε όταν έχουμε πάνω από δύο συνθήκες;



# Εμφωλευμένες Εντολές if (Nested if)

Πολλές φορές χρειάζεται να ελέγξουμε πάνω από μια συνθήκες και χρησιμοποιούμε εμφωλευμένες if εντολές:

```
lab = -1;
if (year < 1)
    { /* empty block */ }
else if (year == 1)
    lab = 20;
else
    lab = 0;
```

## To Dangling else πρόβλημα

Πολλές εμφωλευμένες εντολές if μπορεί να οδηγήσουν σε προβλήματα αμφισημίας (κάτι που δεν μας αρέσει καθόλου στο computer science). Είναι τα δύο προγράμματα ισοδύναμα;

```
lab = -1;
if (year < 1)
    { /* empty block */ }
else if (year == 1)
    lab = 20;
else
    lab = 0;
```

```
lab = -1;
if (year <= 1)
    if (year == 1)
        lab = 20;
else
    lab = 0;
```

# Το πρόβλημα με το Dangling else

Πολλές εμφωλευμένες εντολές if μπορεί να οδηγήσουν σε προβλήματα αμφισημίας (κάτι που δεν μας αρέσει καθόλου στο computer science). Είναι τα δύο προγράμματα ισοδύναμα;

```
lab = -1;
if (year < 1)
    { /* empty block */ }
else if (year == 1)
    lab = 20;
else
    lab = 0;
```

```
lab = -1;
if (year <= 1)
    if (year == 1)
        lab = 20;
else
    lab = 0;
```

Αν δεν είμαστε σίγουροι χρησιμοποιούμε αγκύλες { } για να υποδείξουμε τα όρια των σύνθετων εντολών

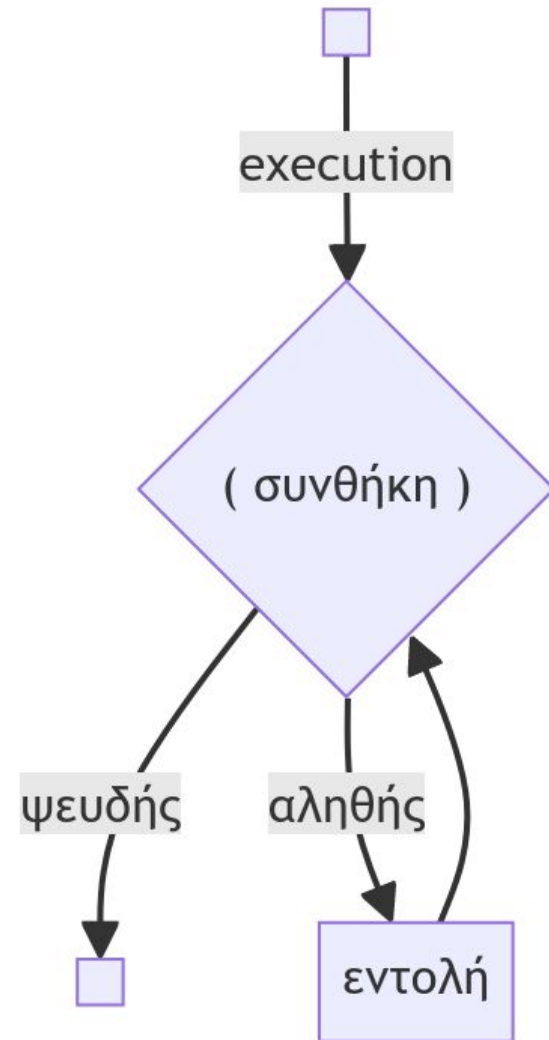


Δομές Επανάληψης / Loops / Βρόχοι

## Εντολή while (while Statement)

Η εντολή while επαναλαμβάνει μια άλλη εντολή όσο η λογική συνθήκη είναι αληθής.

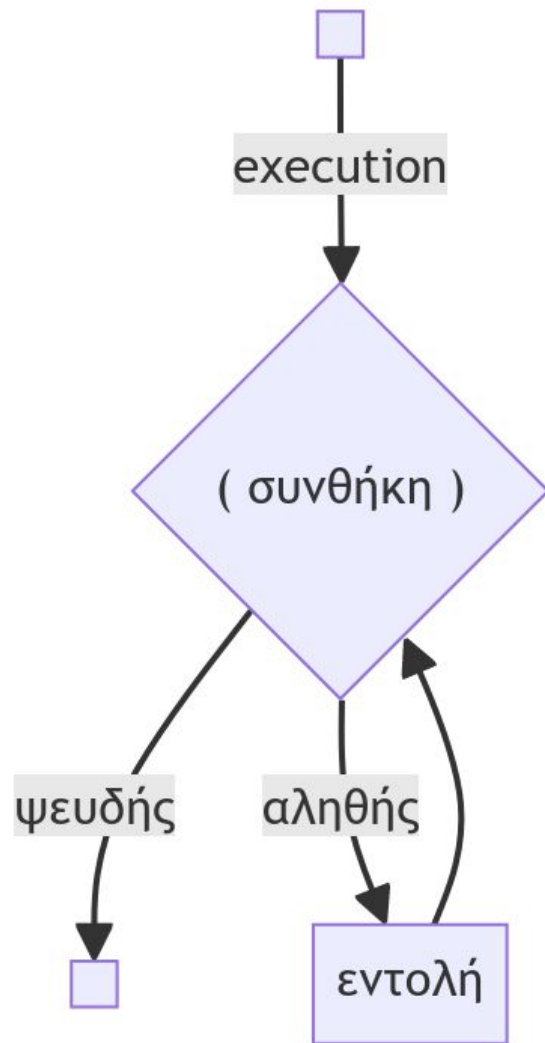
**while ( συνθήκη )**  
**εντολή**



# Εντολή while (while Statement)

Παράδειγμα #1: Τι κάνει το παρακάτω πρόγραμμα;

```
int i = 0;  
int sum = 0;  
while (i < 42) {  
    sum += i;  
    i++;  
}  
printf("%d %d\n", i, sum);
```

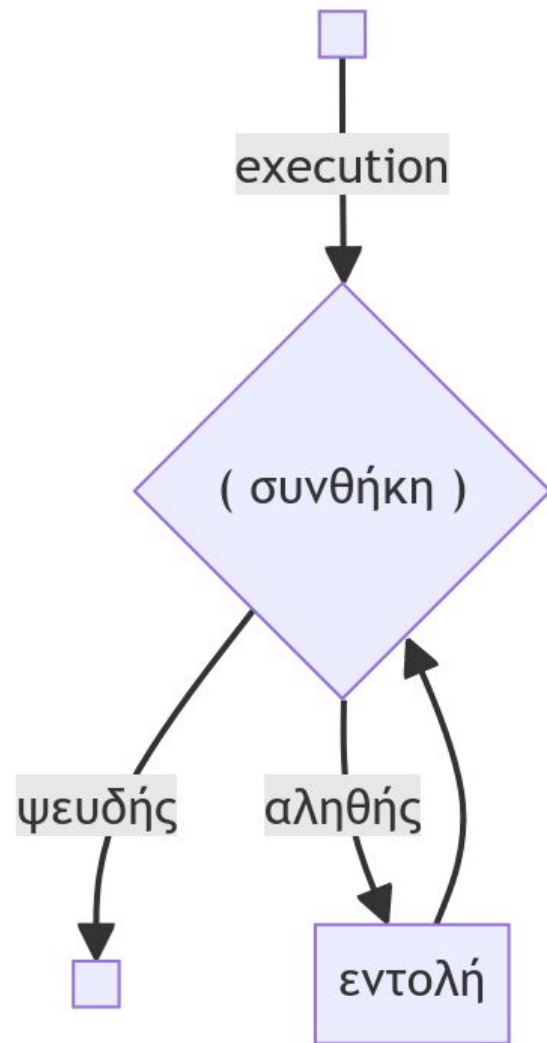


# Εντολή while (while Statement)

Παράδειγμα #2: Τι κάνει το παρακάτω πρόγραμμα;

```
while (1);
```

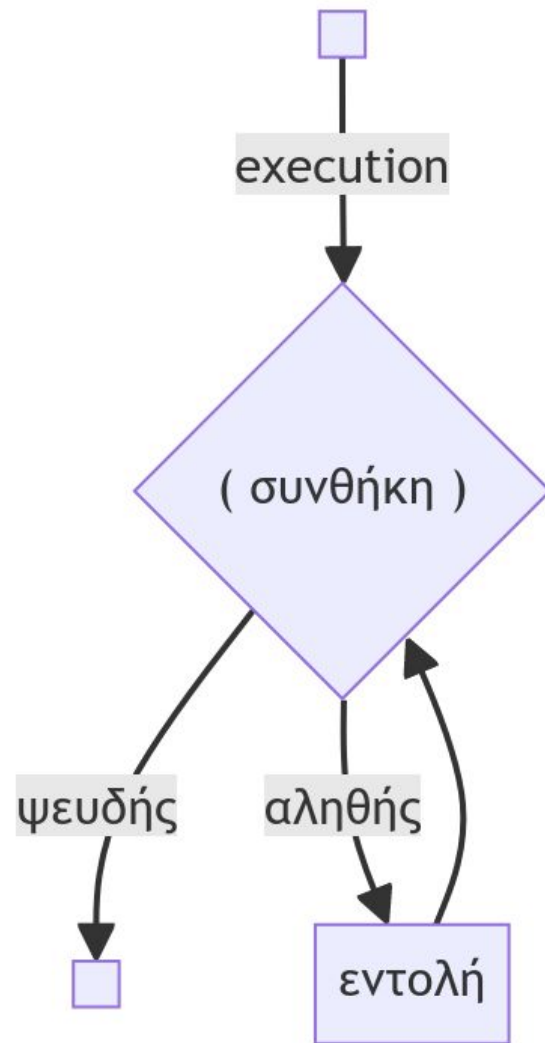
Υπάρχει κάποια χρησιμότητα στο να έχουμε μια λογική συνθήκη που είναι πάντα αληθής;



# Εντολή while (while Statement)

Παράδειγμα #3: Τι κάνει το παρακάτω πρόγραμμα;

```
i = 0;  
while (i < 42);  
    printf("%d\n", i++);
```



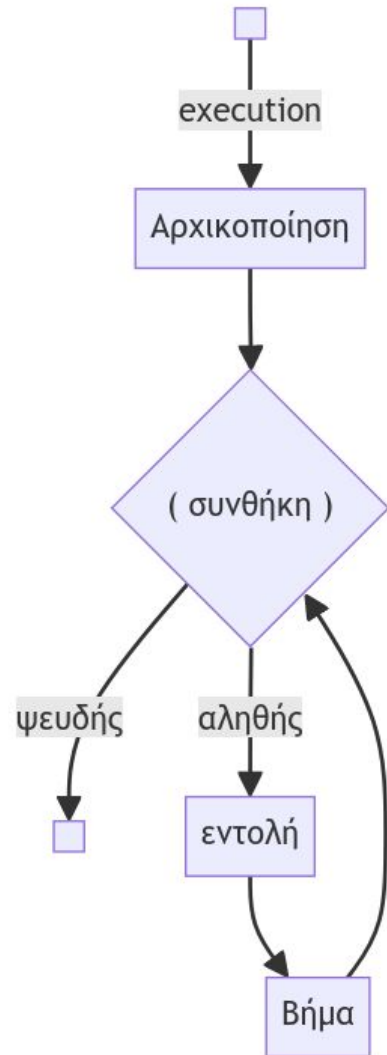
# Εντολή for (for Statement)

Η εντολή for: αρχικοποιεί μεταβλητές, επαναλαμβάνει μια εντολή όσο η λογική συνθήκη είναι αληθής και στο τέλος κάθε επανάληψης εκτελεί το βήμα. Γενική μορφή:

**for ( αρχικοποίηση ; συνθήκη ; βήμα )**  
**εντολή**

Παράδειγμα:

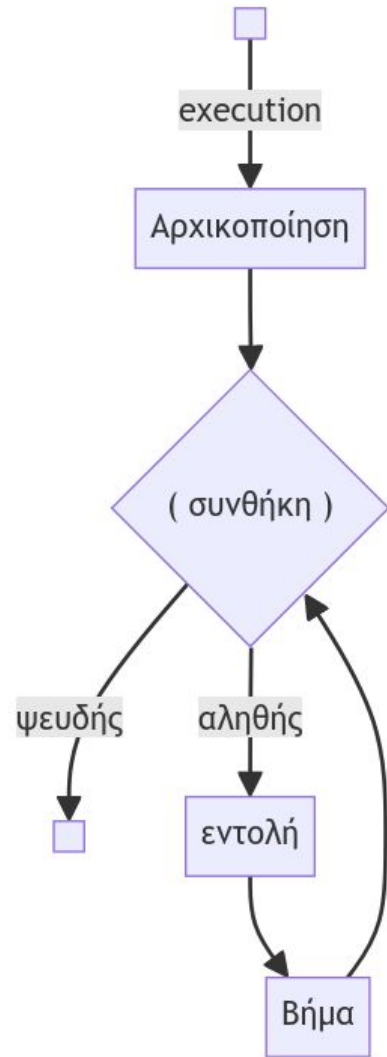
```
int i;  
for ( i = 0 ; i < 100 ; i++ )  
    printf("Hello world\n");
```



# Εντολή for (for Statement)

Τι κάνει το παρακάτω πρόγραμμα;

**for** (;;;)

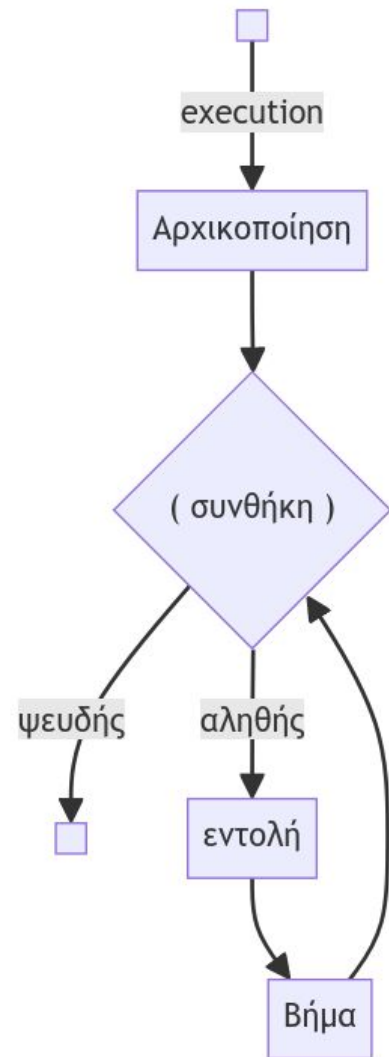


# Εντολή for (for Statement)

Γενικό παράδειγμα:

```
int i;  
for ( i = 0 ; i < N ; i++ ) {  
    printf("%d\n", i);  
}
```

Πόσες φορές θα εκτελεστεί το block εντολών μέσα στην for; Θα τυπωθεί το N; Τι θα συμβεί αν αλλάξω την αρχικοποίηση ή το βήμα;





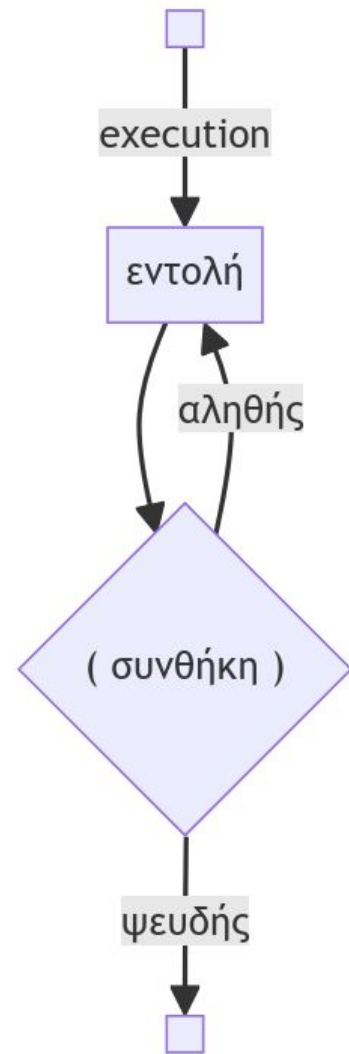
## Εντολή do-while (do-while Statement)

Η εντολή do-while τρέχει πρώτα μια φορά την εντολή και στην συνέχεια ελέγχει την λογική συνθήκη για το αν χρειάζεται άλλη επανάληψη.

**do**

**εντολή**

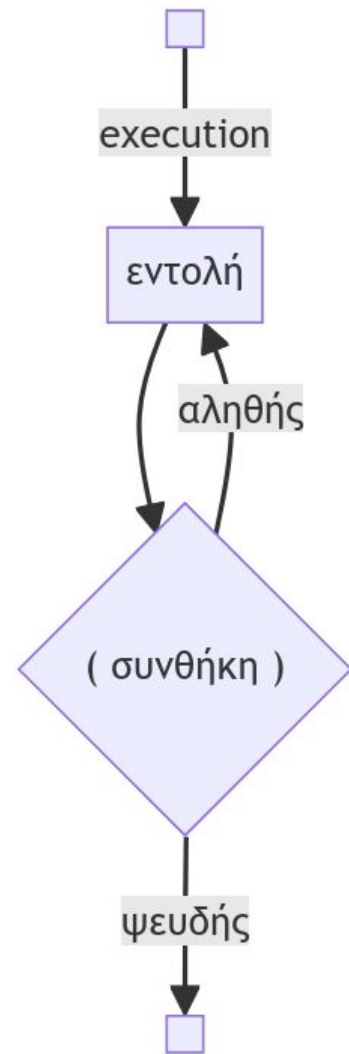
**while ( συνθήκη );**



# Εντολή do-while (do-while Statement)

Παράδειγμα: Τι κάνει το παρακάτω πρόγραμμα;

```
i = 0;  
do  
    i++;  
while (i < 42);  
printf("%d\n", i);
```



**Kahoot Time!**

## Για την Επόμενη Φορά

- Από τις σημειώσεις του κ. Σταματόπουλου συνιστώ να έχετε καλύψει τα πάντα μέχρι την σελίδα 62.
- [Control flow](#)
- [Conditional statements](#) in programming languages
- [While loops](#)
- [For loops](#)

Ευχαριστώ και καλή μέρα εύχομαι!  
Keep (or Start!) coding ;)