ΜΑΘΗΜΑ 10

ΑΣΚΗΣΕΙΣ - ΔΟΜΕΣ ΕΠΑΝΑΛΗΨΗΣ

Γιώργος Διάκος - Full Stack Developer

Γ. Ασκήσεις Εφαρμογή 2

- Γράψε ένα πρόγραμμα C++ το οποίο θα λαμβάνει ως είσοδο δυο ακέραιους αριθμούς και θα τυπώνει τον μεγαλύτερο από τους 2.

Γ. Ασκήσεις Εφαρμογή 3

- Γράψε ενα πρόγραμμα στην C++ το οποίο θα λαμβάνει ως είσοδο τρείς ακεραίους αριθμούς και θα τυπώνει τον μεγαλύτερο από τους 3.

Γ. Ασκήσεις Εφαρμογή 4

- Γράψε ενα πρόγραμμα στην C++ το οποίο θα λαμβάνει ως είσοδο τρείς ακεραίους αριθμούς και θα τους τυπώνει σε αύξουσα σειρά.
- Output example :

Eisagete ton prwto arithmo:8

Eisagete ton deutero arithmo:5

Eisagete ton trito arithmo : 9

H diataksi einai :5,8,9

Γ. Ασκήσεις Εφαρμογή 5

- Γράψε ενα πρόγραμμα στην C++ το οποίο
  - Να δέχεται απο τον χρήστη σαν είσοδο εναν ακέραιο αριθμό , που θα απεικονίζει δευτερόλεπτα.
  - Να υπολογίζει πόσες ώρες , λεπτά και δευτερόλεπτα είναι η είσοδος του χρήστη.

- Output example :

Eisagete to plhthos twn deuteroleptwn : 5000

Wres:1

Lepta : 23

Deyterolepta :20

Υπόδειξη :

Χρησιμοποίησε τους τελεστές / και % για να επιτύχεις το επιθυμητό αποτέλεσμα.

Γ. Ασκήσεις Εφαρμογή 6

Σε μια (εσφαλμένη) εκδοχή του μπαρμπουτιού δυο παίχτες ρίχνουν δυο ζάρια (ο καθένας) και νικάει ο παίχτης με το μεγαλύτερο άθροισμα στα ζάρια.

Γράψε ενα πρόγραμμα στην C++ που θα ζητάει από τον χρήστη να εισάγει το αποτέλεσμα κάθε ζαριού κάθε παίχτη (έστω Α και Β) και να τυπώνει στην οθόνη τον παίχτη που νίκησε.

Γ. Ασκήσεις Εφαρμογή 7

- Ένα έτος είναι δίσεκτο αν είναι :
- Πολλαπλάσιο του 4 με την ειδική περίπτωση ότι
- Αν είναι πολλαπλάσιο του 100 και οχι του 400 τότε δεν είναι δίσεκτο .
- Γράψε ενα πρόγραμμα C++ που θα δέχεται σαν είσοδο χρήστη το έτος και θα τυπώνει αν είναι δίσεκτο ή οχι.
- Παραθέτο εκτελέσεις του προγράμματος :

Eisagete etos : 2018

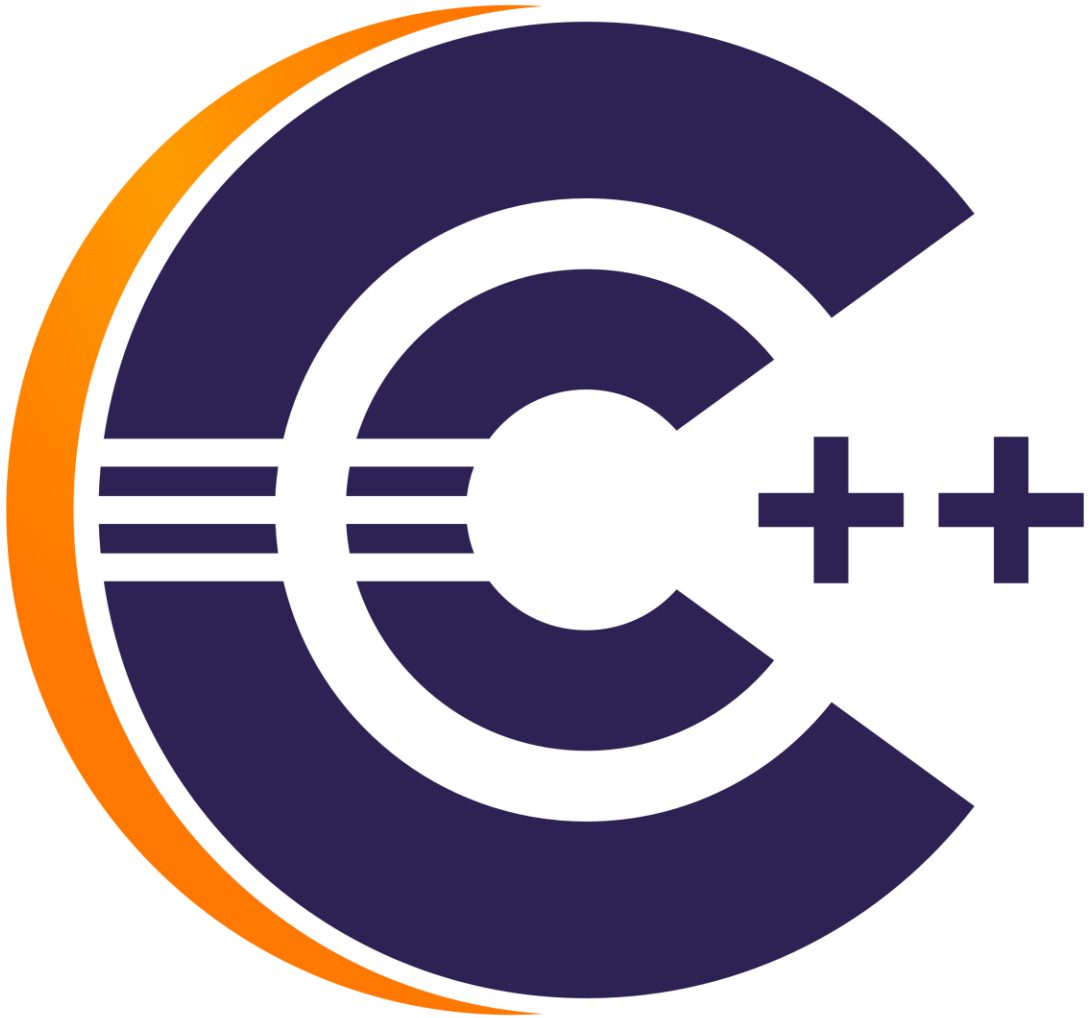
Einai disekto!

Eisage etos : 2100

Den einai disekto!

- Υπόδειξη :
- Για να ελέγξουμε αν ενας αριθμός Χ είναι π.χ πολλαπλάσιο του 4 , θα πρέπει το υπόλοιπο της διαίρεσης του Χ με το 4 να είναι 0.

ΜΟΝΟΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ



1. Εισαγωγή στους Πίνακες

1. Μονοδιάστατοι Πίνακες

- Ένας **πίνακας** είναι μια σειρά από μεταβλητές ίδιου τύπου αποθηκευμένες στην μνήμη.
- Για παράδειγμα ένας πίνακας 10 ακεραίων είναι 10 ακέραιες μεταβλητές αποθηκευμένες στην σειρα (η μία μετά την άλλη) στην μνήμη.
  - Ένας πίνακας θα δηλώνεται (στο τμήμα δήλωσης μεταβλητών) με εντόλη της μορφής:

```
Τύπος_Δεδομένων ΟΝΟΜΑ_ΠΙΝΑΚΑ[ΠΛΗΘΟΣ];
```

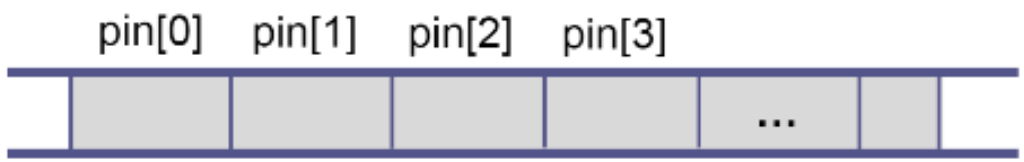
  - Όπου:
    - Τύπος_Δεδομένων: Ο τύπος δεδομένων των μεταβλητών του πίνακα.
    - ΟΝΟΜΑ_ΠΙΝΑΚΑ: Το όνομα που επιλέγουμε εμείς για τον πίνακα.
    - ΠΛΗΘΟΣ (προσέξτε ότι είναι μέσα σε αγκύλες): Πόσες μεταβλητές θα περιέχει ο πίνακας

1. Μονοδιάστατοι Πίνακες

- Για παράδειγμα με την δήλωση:

```
int pin[4];
```

  - Δηλώνουμε έναν πίνακα με 4 ακέραιες μεταβλητές.
  - Με την εντολή δήλωσης, δεσμεύεται στην μνήμη χώρος για τον πίνακα και είναι σημαντικό ότι οι θέσεις αυτές είναι σε μια σειρά.
- Τα ονόματα των 4μεταβλητών που κατασκευάσαμε είναι:
  - pin[0], pin[1], pin[2], pin[3]
    - Παρατηρούμε ότι η αρίθμηση ξεκινά από το 0 έως και ένα λιγότερο από τον αριθμό που δηλώσαμε.
    - Άρα μετά την εντολή δήλωσης, έχουν δεσμευτεί 4 **διαδοχικές** (προσοχή!) θέσεις στην μνήμη για να αποθηκεύσουν τιμές σε αυτές τις μεταβλητές:



1. Εισαγωγή στους Πίνακες

2. Παράδειγμα

- Μετά τη δήλωση του πίνακα είναι σαν να έχουμε 4 μεταβλητές που μπορούμε να χρησιμοποιήσουμε στο πρόγραμμά μας.
- Δείτε για παράδειγμα το εξής απλό πρόγραμμα:
- Οι μεταβλητές διαχειρίζονται όπως οι τυπικές ακέραιες μεταβλητές

```
/* array.c: Aplo programma me pinaka */
#include <stdio.h>

int main()
{
   int pin[3];
   int sum;

   pin[0]=1;
   pin[1]=3;
   pin[2]=4;
   sum=pin[0]+pin[1]+pin[2];
   printf("\n%d+%d+%d=%d",pin[0],pin[1],pin[2],sum);
}
```

- Προσοχή, να μην γράψουμε αριθμό για όριο του πίνακα που είναι εκτός του επιτρεπτού. Π.χ. Αν έχουμε δήλωση int pin[4]; και γράψουμε για όνομα μεταβλητής pin[5] θα έχουμε σφάλμα όταν θα εκτελέσουμε το πρόγραμμα!
- Περισσότερα για τους πίνακες θα δούμε σε επόμενο μάθημα, όπου θα δούμε και πολυδιάστατους πίνακες (διδιάστατους, τριδιάστατους κ.λπ.)
- Εδώ κάνουμε μια απλή αναφορά στους πίνακες για να τους χρησιμοποιήσουμε με τη δομή επανάληψης στις ασκήσεις. Πολύ περισσότερα σε επόμενο μάθημα.

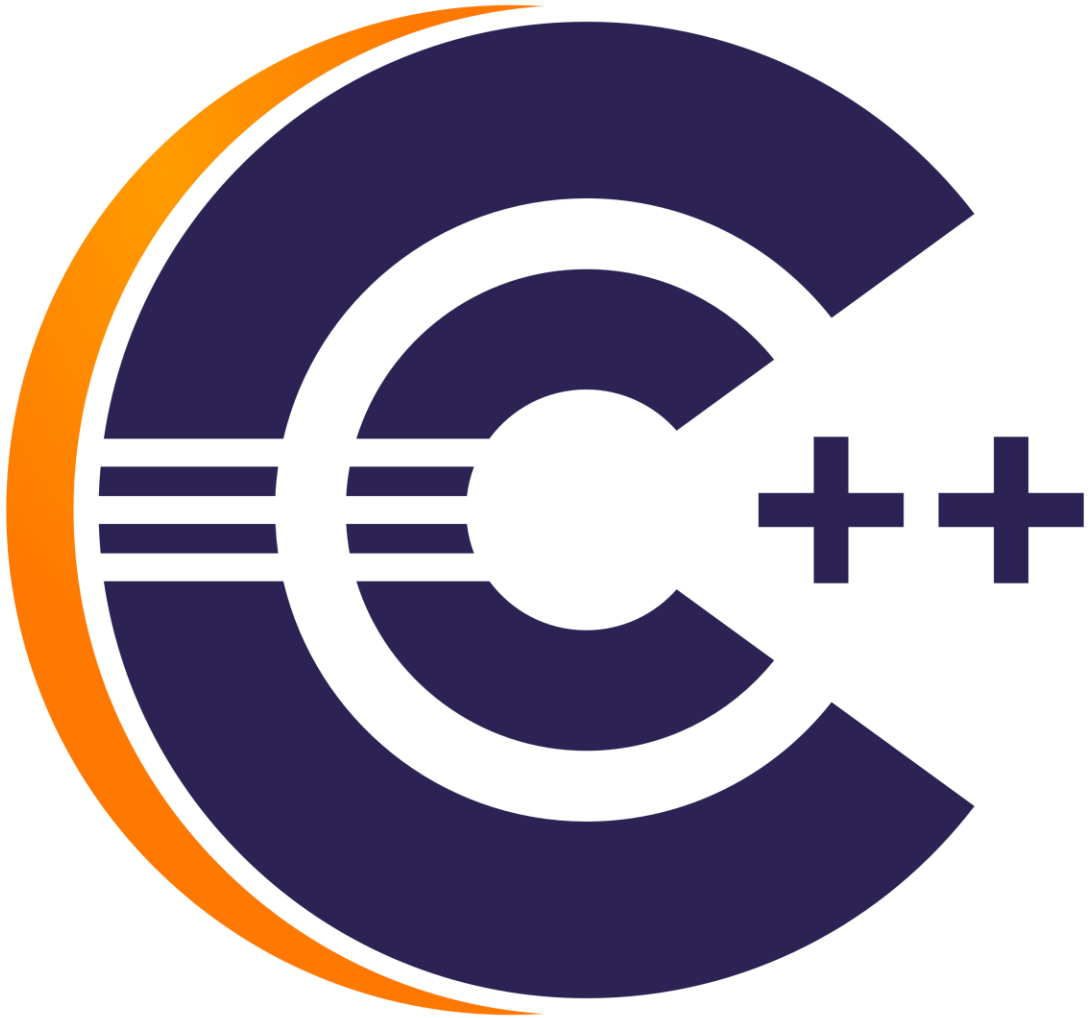
1. Γενικά

- Η εντολή επανάληψης είναι η σημαντικότερη δομή σε ένα πρόγραμμα.
  - ...διότι μας επιτρέπει να εκτελέσουμε ένα τμήμα κώδικα πολλές φορές, το οποίο είναι το κύριο χαρακτηριστικό του προγραμματισμού.

- Στη γλώσσα C που μαθαίνουμε, υπάρχουν τρεις τρόποι για να κάνουμε επανάληψη της εκτέλεσης ενός τμήματος κώδικα:
  - Η εντολή **for (…; … ; …) { … }** στην οποία ρητά αναφέρουμε πόσες φορές θέλουμε να εκτελεστεί ένα τμήμα κώδικα.
  - Η εντολή **while(…) { … }**
  - Η εντολή **do{…} while(…);**

- Θα αναλύσουμε τους τρεις τρόπους επανάληψης και πότε χρησιμοποιούμε τον καθένα

ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ for



2. Η δομή for

1. Συντακτικό της for

- Το συντακτικό της εντολής for είναι:

```
for( αρχική; συνθήκη; βήμα)
    εντολή;
```

```
for( αρχική; συνθήκη; βήμα)
{
    (εντολές)
}
```

- Δείτε ότι αν τρέχουμε μία εντολή τότε δεν είμαστε αναγκασμένοι να χρησιμοποιήσουμε άγκιστρα, αλλιώς αν τρέχουμε παραπάνω από μία εντολή πρέπει να βάλουμε τις εντολές σε άγκιστρα (όπως στην if, ακολουθεί την for ένα μπλοκ κώδικα).

- Με το παραπάνω συντακτικό γίνονται οι εξής ενέργειες:
  - 1. Τρέχει η αρχική: είναι μια εντολή που τρέχει μία μόνο φορά στην αρχή. Αναφέρεται και σαν εντολή αρχικοποίησης
  - 2. Ελέγχεται η συνθήκη: συνήθως είναι μια σύγκριση τιμών.
    - 2.1 Αν η συνθήκη είναι ψευδής, τότε βγαίνουμε από την for και πάμε στην αμέσως επόμενη εντολή.
    - 2.2. Αλλιώς αν η συνθήκη είναι αληθής, εκτελείται η εντολή αύξηση και μεταβαίνουμε και πάλι στο βήμα 2.

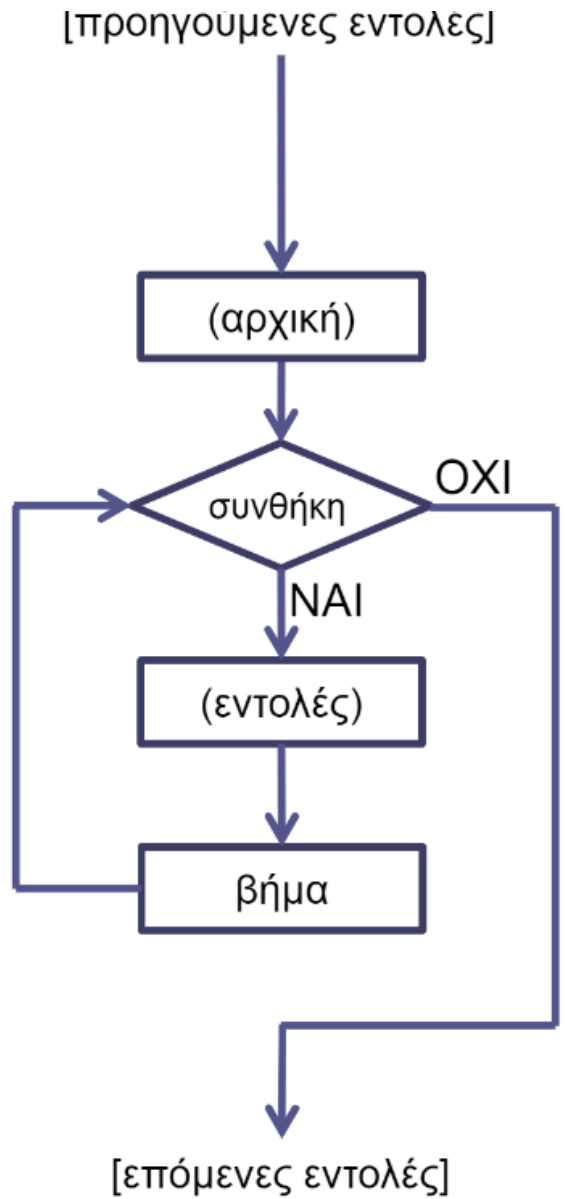
2. Η δομή for

2. Διάγραμμα Ροής Προγράμματος

- Και το διάγραμμα ροής προγράμματος:

```
[προηγούμενες εντολές]

for( αρχική; συνθήκη; βήμα)
    εντολή;

[επόμενες εντολές]
```

```
[προηγούμενες εντολές]

for( αρχική; συνθήκη; βήμα)
{
    (εντολές)
}

[επόμενες εντολές]
```



3. Παραδείγματα Εκτέλεσης

- Παράδειγμα 1:
  - Τι κάνει το ακόλουθο τμήμα κώδικα;

```
for (i=1; i<=5; i++)
   printf("\nKalimera");
```

  - Απάντηση:
    - Εκτελείται η εντολή i=1.
    - Ελέγχεται αν `i<=5`. Είναι αληθές (`1<=5`), άρα τυπώνεται «Καλημέρα» και εκτελείται η αύξηση `i++`, άρα το i γίνεται 2.
    - Ελέγχεται αν `i<=5`. Είναι αληθές (`2<=5`), άρα τυπώνεται «Καλημέρα» και εκτελείται η αύξηση `i++`, άρα το i γίνεται 3.
    - Ελέγχεται αν `i<=5`. Είναι αληθές (`3<=5`), άρα τυπώνεται «Καλημέρα» και εκτελείται η αύξηση `i++`, άρα το i γίνεται 4.
    - Ελέγχεται αν `i<=5`. Είναι αληθές (`4<=5`), άρα τυπώνεται «Καλημέρα» και εκτελείται η αύξηση `i++`, άρα το i γίνεται 5.
    - Ελέγχεται αν `i<=5`. Είναι αληθές (`5<=5`), άρα τυπώνεται «Καλημέρα» και εκτελείται η αύξηση `i++`, άρα το i γίνεται 6.
    - Ελέγχεται αν `i<=5`. Είναι ψευδές (`6<=5`), άρα μεταβαίνουμε στην αμέσως επόμενη εντολή μετά την for.
  - Άρα το τμήμα κώδικα τυπώνει 5 φορές στην οθόνη τη λέξη ΚΑΛΗΜΕΡΑ

2. Η δομή for

3. Παραδείγματα Εκτέλεσης

- Συνήθως στις εφαρμογές χρησιμοποιείται η τιμή της μεταβλητής που τροποποιείται στην αύξηση. Είναι σημαντικό ότι σε κάθε επανάληψη η τιμή της μεταβλητής είναι διαφορετική

- Παράδειγμα 2:
  - Τι κάνει το ακόλουθο τμήμα κώδικα;

```
for (i=1; i<=3; i++)
   printf("\n%d",i);
```

  - Απάντηση:
    - Εκτελείται η εντολή i=1.
    - Ελέγχεται αν `i<=3`. Είναι αληθές (`1<=3`), άρα τυπώνεται «`1`» και εκτελείται η αύξηση `i++`, άρα το `i` γίνεται 2.
    - Ελέγχεται αν `i<=3`. Είναι αληθές (`2<=3`), άρα τυπώνεται «`2`» και εκτελείται η αύξηση `i++`, άρα το `i` γίνεται 3.
    - Ελέγχεται αν `i<=3`. Είναι αληθές (`3<=3`), άρα τυπώνεται «`3`» και εκτελείται η αύξηση `i++`, άρα το `i` γίνεται 4.
    - Ελέγχεται αν `i<=3`. Είναι ψευδές (`4<=3`), άρα μεταβαίνουμε στην αμέσως επόμενη εντολή μετά την for.
  - Και το πρόγραμμα τερματίζει

2. Η δομή for

3. Παραδείγματα Εκτέλεσης

- Συνήθως, χρησιμοποιώντας την τιμή της μεταβλητής κάνουμε ακόμη και περίπλοκους υπολογισμούς.

- Παράδειγμα 3:
  - Τι κάνει το ακόλουθο τμήμα κώδικα (οι j,x είναι ακέραιες μεταβλητες);

```
for (j=0; j<2; j++)
{
   x=j*j-1;
   printf("\n%d",x);
}
```

  - Απάντηση: Οι εντολές που περικλείονται ανάμεσα στα άγκιστρα εκτελούνται για τις διαδοχικές τιμές της μεταβλητής. Άρα:
    - Εκτελείται η εντολή `j=0`.
    - Ελέγχεται αν `j<2`. Είναι αληθές (`0<2`), άρα υπολογίζεται `x=0*0-1=-1` άρα τυπώνεται «`-1`» και εκτελείται η αύξηση `j++`, άρα το `j` γίνεται 1.
    - Ελέγχεται αν `j<2`. Είναι αληθές (`1<2`), άρα υπολογίζεται `x=1*1-1=0` άρα τυπώνεται «`0`» και εκτελείται η αύξηση `j++`, άρα το `j` γίνεται 2.
    - Ελέγχεται αν `j<2`. Είναι ψευδές (`2<2`), άρα μεταβαίνουμε στην επόμενη εντολή μετά τη for.

2. Η δομή for

3. Παραδείγματα Εκτέλεσης

- Ας δούμε και ένα παράδειγμα που η μεταβλητή μειώνεται αντί να αυξάνεται.
- Παράδειγμα 4:
  - Τι κάνει το ακόλουθο τμήμα κώδικα (οι j,x είναι ακέραιες μεταβλητές);

```
for (j=2; j>=0; j--)
{
   x=j*j-1;
   printf("\n%d",x);
}
```

  - Απάντηση: Οι εντολές που περικλείονται ανάμεσα στα άγκιστρα εκτελούνται για τις διαδοχικές τιμές της μεταβλητής. Άρα:
    - Εκτελείται η εντολή `j=2`.
    - Ελέγχεται αν `j>=0`. Είναι αληθές (`2>=0`), άρα υπολογίζεται `x=2*2-1=3` άρα τυπώνεται «`3`» και εκτελείται η μείωση `j--`, άρα το `j` γίνεται 1.
    - Ελέγχεται αν `j>=0`. Είναι αληθές (`1>=0`), άρα υπολογίζεται `x=1*1-1=0` άρα τυπώνεται «`0`» και εκτελείται η μείωση `j--`, άρα το `j` γίνεται 0.
    - Ελέγχεται αν `j>=0`. Είναι αληθές (`0>=0`), άρα υπολογίζεται `x=0*0-1=0` άρα τυπώνεται «`-1`» και εκτελείται η μείωση `j--`, άρα το `j` γίνεται `-1`.
    - Ελέγχεται αν `j>=0`. Είναι ψευδές `(-1>=0)`, άρα μεταβαίνουμε στην επόμενη εντολή μετά τη for.

2. Η δομή for

3. Παραδείγματα Εκτέλεσης

- Μπορούμε να έχουμε και οποιοδήποτε βήμα αύξησης επιθυμούμε με το κατάλληλο συντακτικό.
- Παράδειγμα 5:
  - Τι κάνει το ακόλουθο τμήμα κώδικα (η `i` είναι ακέραια μεταβλητή);

```
for (i=0; i<=6; i+=2)
   printf("\n%d",i*i);
```

  - Απάντηση:
    - Εκτελείται η εντολή `i=0`.
    - Ελέγχεται αν `i<=6`. Είναι αληθές `(0<=6)`, υπολογίζεται `0*0=0` τυπώνεται «`0`» και εκτελείται η αύξηση `i+=2`, άρα το i γίνεται `2`.
    - Ελέγχεται αν `i<=6`. Είναι αληθές `(2<=6)`, υπολογίζεται `2*2=4` τυπώνεται «`4`» και εκτελείται η αύξηση `i+=2`, άρα το i γίνεται `4`.
    - Ελέγχεται αν `i<=6`. Είναι αληθές `(4<=6)`, υπολογίζεται `4*4=16` τυπώνεται «`16`» και εκτελείται η αύξηση `i+=2`, άρα το i γίνεται `16`.
    - Ελέγχεται αν `i<=6`. Είναι αληθές `(6<=6)`, υπολογίζεται `6*6=36` τυπώνεται «`36`» και εκτελείται η αύξηση `i+=2`, άρα το i γίνεται `8`.
    - Ελέγχεται αν `i<=6`. Είναι ψευδές `(8<=6)`, άρα μεταβαίνουμε στην επόμενη εντολή μετά τη for.

2. Η δομή for

3. Παραδείγματα Εκτέλεσης

- Αν και ο συνηθισμένος τρόπος χρήσης της for, είναι κάποιος από αυτούς που είδαμε, έχουμε επιπλέον δικαίωματα:
  - Να βάλουμε παραπάνω απο μία αρχικές εντολές (χωρισμένες με κόμμα)
  - Να βάλουμε παραπάνω από μια εντολές αύξησης (χωρισμένες με κόμμα)
  - Να έχουμε πιο σύνθετες συνθήκες (π.χ. και με λογικό έλεγχο.)
- Εκτελέστε το παρακάτω πρόγραμμα:

```
/* complex_for.c: Deixnei to oloklиromeno sintaktiko tis for */
#include <stdio.h>

main()
{
   int i,j;

   for (i=0,j=0; i<5 && j<5; i++,j+=2)
      printf("\ni=%d,j=%d: ",i,j);

}
```

ΤΕΛΟΣ ΜΑΘΗΜΑΤΟΣ

ΑΣΚΗΣΕΙΣ - ΔΟΜΕΣ ΕΠΑΝΑΛΗΨΗΣ

Γιώργος Διάκος - Full Stack Developer