Εισαγωγή στην Πληροφορική & στον Προγραμματισμό

Αρχές Προγραμματισμού Η/Υ (με τη γλώσσα C)

Διάλεξη #3

5 Απριλίου 2024

Παναγιώτης Παύλου

c-programming-24@allos.gr

Έλεγχος ροής εκτέλεσης

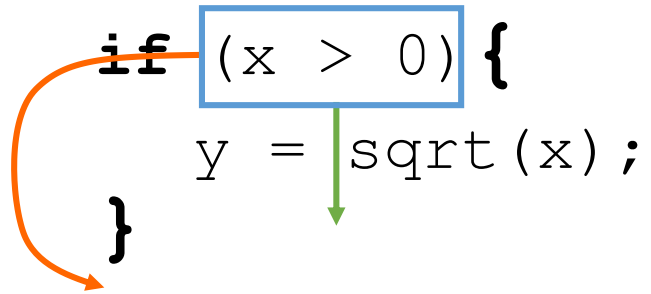
Πως εκτελούνται επιλεκτικά ή επαναλαμβάνονται τμήματα κώδικα

Επιλεκτική εκτέλεση – Εντολή if

Με την μέχρι αυτό το σημείο σειριακή εκτέλεση των εντολών οι δυνατότητες του κώδικα είναι περιορισμένες. Με την εντολή **if** μπορεί ο προγραμματιστής να επιλέξει εάν θα εκτελεστεί κάποιο block κώδικα ή όχι, βάσει της αλήθειας μιας συνθήκης (=μιας λογικής παράστασης).

Αληθής συνθήκη συνεπάγεται εκτέλεση του block. Ψευδής συνθήκη συνεπάγεται παράκαμψη του block.

π.χ.



Η σύνταξη της **if** έχει ως εξής:

```
if ( συνθήκη ) {
    // εντολές που εκτελούνται
    // μόνο αν η συνθήκη ισχύει
}
```

Παρατηρήσεις:

- Ο κώδικας της if είναι στοιχισμένος «πιο μέσα» ώστε να είναι άμεσα εμφανές το ποιες εντολές αφορά.
- Επειδή κάθε μπλοκ εντολών αντιστοιχεί συντακτικά σε μία εντολή, όταν η if καθορίζει την εκτέλεση μίας μόνο εντολής τα άγκιστρα μπορούν να παραληφθούν.

ΠΡΟΣΟΧΗ! Αυτό αποτελεί κακή πρακτική και πρέπει να αποφεύγεται.

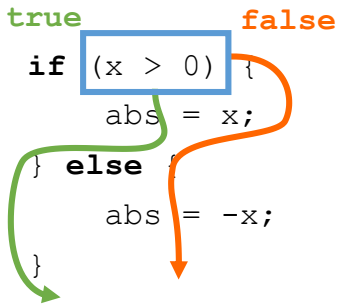
Εναλλακτική εκτέλεση – Εντολή if/else

Κάποιες φορές είναι χρήσιμο να εκτελείται είτε ένα block εντολών (όταν η συνθήκη είναι αληθής), είτε ένα άλλο (όταν η συνθήκη είναι ψευδής). Αυτό γίνεται με τη χρήση της `else` η οποία ακολουθεί το block της `if`. Η `else` συντάσσεται με δικό της block εντολών.

Η σύνταξη της **`if/else`** έχει ως εξής:

```
if ( συνθήκη ) {
    // εντολές για αληθή συνθ.
} else {
    // εντολές για ψευδή συνθ.
}
```

Μόνο ένα από τα δύο block εντολών εκτελείται, αλλά οπωσδήποτε εκτελείται κάποιο



Επειδή τα block εντολών αυτά δεν έχουν κάποια ιδιαιτερότητα, μπορούν να περιέχουν και άλλες εντολές `if/else`. Π.χ.

```
if (age >= 16) {
    if (isMale) {
        // είσαι κύριος
    } else {
        // είσαι κυρία
    }
} else {
    if (isMale) {
        // είσαι αγόρι
    } else {
        // είσαι κορίτσι
    }
}
```

Πολλαπλές εναλλακτικές (1/2)

Συχνά χρειάζεται να εξεταστούν πολλαπλές περιπτώσεις εναλλακτικές μεταξύ τους.

Οπότε, σε κάθε `else` υπάρχει εμφωλευμένο ένα και μόνο `if`. Κατά συνέπεια δεν χρειάζονται και τα άγκιστρα στο `else` αφού το `if` είναι μία εντολή.

Για να είναι πιο ξεκάθαρος ο κώδικας και οι εσοχές να μην προκαλούν οπτικό χάος, το ίδιο γράφεται όπως δεξιά.

```
if (x < 0) {
        // x < 0
} else {

}
```

```
if (x < 0) {
        // x < 0
} else
```

Πολλαπλές εναλλακτικές (2/2)

Ο κώδικας αυτό εκτελείται ως εξής: Ξεκινώντας από πάνω προς τα κάτω, ελέγχονται μία προς μία οι συνθήκες μέχρι να βρεθεί η πρώτη αληθής.

Τότε εκτελείται ο κώδικας που αντιστοιχεί στη συνθήκη που επαληθεύθηκε.

Αν δεν επαληθεύεται καμία συνθήκη τότε (εφόσον υπάρχει η τελική else) εκτελούνται οι εντολές της, αλλιώς καμία.

```
if (x < 0) {
        // x < 0
} else if (x < 50){
        // x >= 0 && x<105
} else if (x < 50){
        // x >= 50&&&x<10
} else {
        // x >= 10
}
```

Εάν γίνει η εναλλαγή προκύπτει λογικό σφάλμα!

Η παράσταση αυτή είναι πάντα **ψευδής**!

Πρέπει να δοθεί πολύ προσοχή στο εξής: Εάν γραφούν οι συνθήκες με σειρά ώστε μία γενική να προηγείται μίας ειδικής, τότε προκύπτει λογικό σφάλμα, καθώς θα επαληθεύεται η γενική και δε θα δίνεται η ευκαιρία να ελεγχθεί η ειδική συνθήκη.

Αυτό στον δίπλα κώδικα είναι προφανές, αλλά δεν είναι πάντα.

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Δίσεκτα έτη

Τα δίσεκτα έτη ξέρουμε όλοι ότι είναι όσα είναι ακέραια πολλαπλάσια του 4.

Υπάρχει όμως μία εξαίρεση: Όσα είναι ακέραια πολλαπλάσια του 100 δεν είναι δίσεκτα (θα περίσσευε μία ημέρα)

Και υπάρχει και η εξαίρεση της εξαίρεσης: Όσα έτη είναι ακέραια πολλαπλάσια του 400 είναι δίσεκτα (θα έλειπε μία ημέρα)

Σημειώστε το προφανές (?) ότι τα ακέραια πολλαπλάσια του 400 είναι ακέραια πολλαπλάσια και του 100 και του 4. Γι'αυτό η σειρά που γράφονται οι συνθήκες χρειάζεται προσοχή!

Αυτό σε κώδικα γράφεται...

```
if (year % 400==0) {
    isLeap = true;  // e.g. 2000
} else if (year % 100==0) {
    isLeap = false; // e.g. 1900
} else if (year % 4==0) {
    isLeap = true;  // e.g. 1984
} else {
    isLeap = false; // e.g. 2019
}
```

Πρακτική #1

Το ζητούμενο σε αυτές τις εφαρμογές είναι να δημιουργηθούν συναρτήσεις:

1. Την maxOf2 που επιστρέφει τη μεγαλύτερη από δύο τιμές A, B

```
function maxOf2(A, B)
```

2. Την maxOf3 που επιστρέφει τη μεγαλύτερη από τρεις τιμές A, B, C:

```
function maxOf3(A, B, C)
```

Τελεστές ανάθεσης

Μια πιο ξεκάθαρη και αποδοτική γραφή των μεταβολών

Τελεστές ανάθεσης

Σε όλους τους βρόχους το συνηθέστερο είναι σε κάθε επανάληψη να μεταβάλλεται τουλάχιστον μία μεταβλητή που εμπλέκεται στη συνθήκη. Για παράδειγμα `i=i-1` ή `N=N+1` . Αυτό με πιο σωστά ονοματισμένες μεταβλητές καταλήγει να είναι δυσανάγνωστο.

Επίσης δεν παράγει αποδοτική γλώσσα μηχανής. Γι'αυτό κάθε παράσταση της μορφής

```
someVariable = someVariable + A;
```

γράφεται και ως

```
someVariable += A;
```

Ανάλογα μπορούν να γραφούν όλοι οι αριθμητικοί τελεστές (και αυτοί των bits). Π.χ.

`X-=4` *ή* `Y*=2` *ή* `Z/=1.44` *ή* `Q%=3`

Μοναδιαίοι τελεστές ανάθεσης (1/3)

Επειδή οι πιο συνηθισμένες χρήσεις των τελεστών ανάθεσης είναι της μορφής **`i+=1`** και **`i-=1`** , δηλαδή όπου Α στην προηγούμενη μορφή, έχουμε την τιμή 1, αλλά και επειδή η γλώσσα μηχανής υποστηρίζει με ειδικές εντολές της αυτές τις αλλαγές, υπάρχει ειδική γραφή γι' αυτές τις περιπτώσεις.

Η γραφή είναι μία από τις παρακάτω. Το

`i=i+1` ή **`i+=1`** γίνεται **`i++`** αλλά και **`++i`**

καθώς και το

`i=i-1` ή **`i-=1`** γίνεται **`i--`** αλλά και **`--i`**

Μοναδιαίοι τελεστές ανάθεσης (2/3)

Όταν πρόκειται για μία απλή εντολή. Π.χ.

`i++;` ή **`++i;`** ή **`i--;`** ή **`--i;`**

τότε δεν υπάρχει καμία διαφορά μεταξύ της χρήσης των **`++`** ή **`--`** ως πρόθεμα ή ως επίθεμα.

Η διαφορά φαίνεται όταν αυτός ο τελεστής (**`++`** ή **`--`**) είναι μέρος μιας πιο σύνθετης παράστασης. Π.χ.

```
k = i++
```

Όταν το **`++`** ή το **`--`** γράφονται πριν ή μετά από τον τελεσταίο τους, αυτό που αλλάζει είναι η τιμή που το αντικαθιστά, εδώ το **`i++`** ή το **`i--`**

Μοναδιαίοι τελεστές ανάθεσης (3/3)

Αν υποθέσουμε ότι **i=10**, όταν ο τελεστής γράφεται ως επίθεμα

```
k = i++;
```

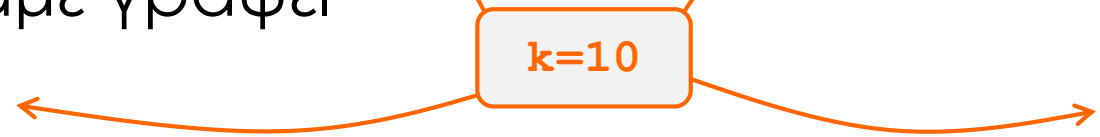
ή

```
k = i--;
```

είναι σαν να είχαμε γράψει

```
k = i;
i++;
```

```
k = i;
i--;
```



ενώ όταν γράφεται ως πρόθεμα

```
k = ++i;
```

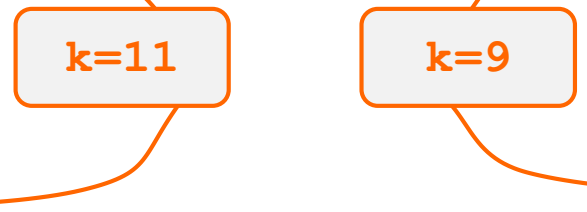
ή

```
k = --i;
```

είναι σαν να γράφουμε

```
i++;
k = i;
```

```
i--;
k = i;
```



Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Βρόχοι/Loops

Επανάληψη εντολών του προγράμματος

Βρόχος while

Μέχρι αυτό το σημείο ροή εκτέλεσης κάθε κώδικα είναι «από πάνω προς τα κάτω», ακόμα και αν είναι υπό συνθήκη.

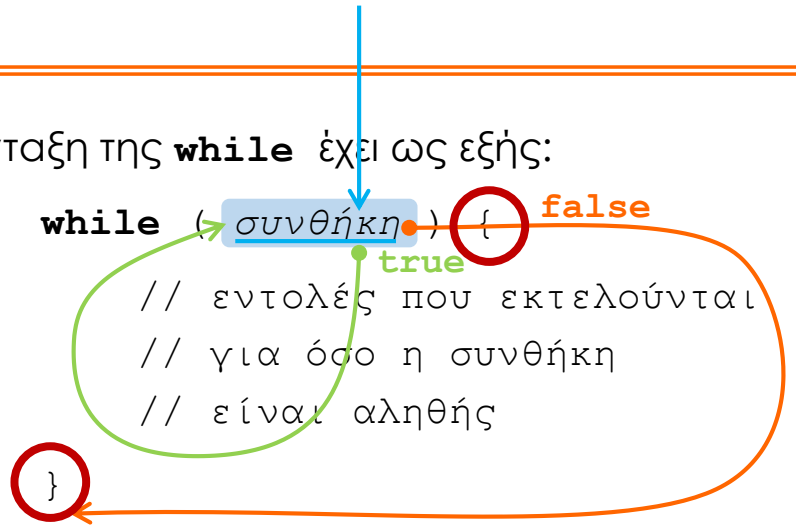
Με τους **βρόχους** είναι δυνατή η επιστροφή σε προηγούμενο σημείο του κώδικα και η επανάληψη της εκτέλεσης των ίδιων εντολών. Η επανάληψη αυτή αρχίζει και συνεχίζεται εφόσον ισχύει κάποια **συνθήκη**.

Η βασική εντολή για επαναληπτική εκτέλεση είναι η **`while`**. Αυτή συντάσσεται ακριβώς όπως η `if`, λειτουργεί όμως διαφορετικά.

Εάν ισχύει η συνθήκη εκτελούνται οι εντολές, αλλά μετά την εκτέλεση τους, η συνθήκη επανεξετάζεται.

Εφόσον ισχύει ακόμα, εκτελούνται ξανά. Αυτή η διαδικασία επαναλαμβάνεται μέχρι η συνθήκη να μην ισχύει.

Η σύνταξη της **`while`** έχει ως εξής:



Προσοχή!

- Εάν η συνθήκη είναι τέτοια που θα είναι πάντα αληθής, τότε το πρόγραμμα δεν μπορεί να τερματιστεί και λέμε ότι «κολλάει».
- Εάν η συνθήκη δεν ισχύει εξαρχής τότε οι εντολές δεν εκτελούνται καθόλου.

1η Δημοτικού

Τιμωρία! Γράψετε 5 φορές το

```
C is the best language!
```

Γράφοντας

```
let i=1;
while (i <= 5) {
    smPrint("C is the best languge!\n");
    i = i + 1;
}
```

i 6

```
C is the best language!
C is the best language!
C is the best language!
C is the best language!
C is the best language!
```

Ας μετρήσουμε μέχρι το 5.

```
let i=1;
while (i <= 5) {
    smPrint("%d ", i);
    i = i + 1;
}
smPrint("\n");
```

i 6

2*i

με αποτέλεσμα:

```
1 2 3 4 5
```

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Πρακτική #2

Το ζητούμενο σε αυτή την εφαρμογή είναι να δημιουργηθεί η συνάρτηση:

```
function factorial(N)
```

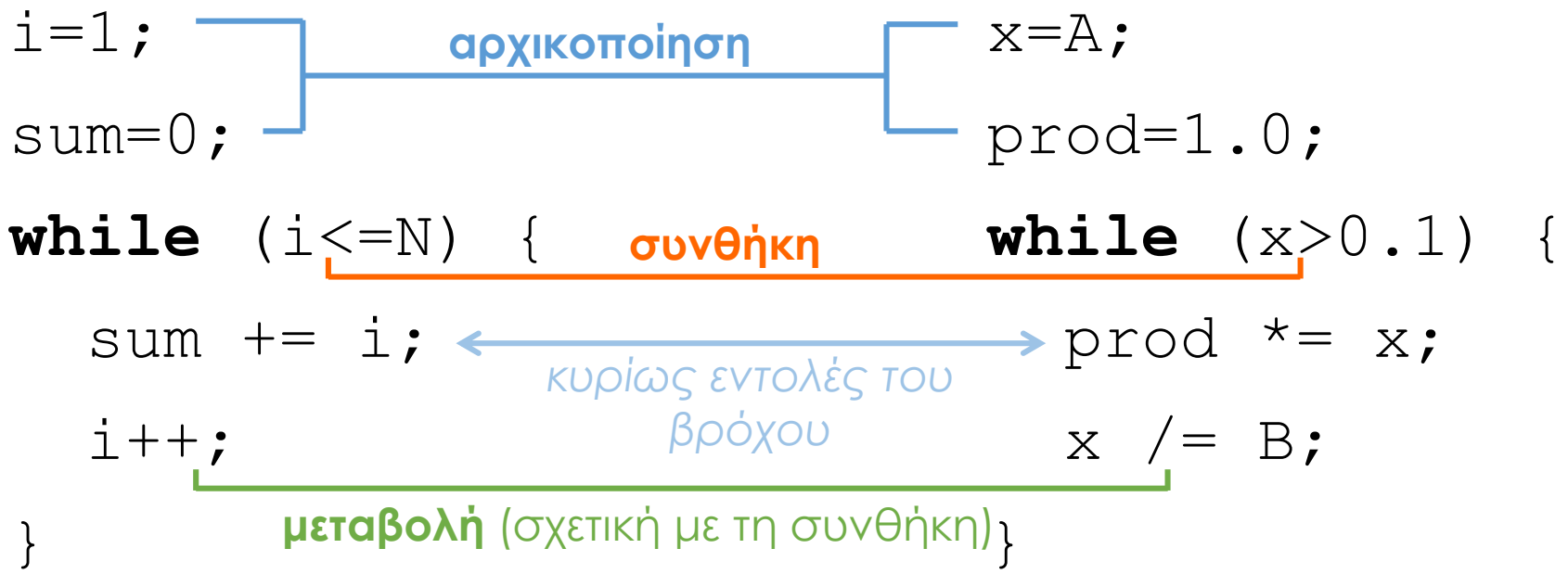
η οποία θα επιστρέφει το παραγοντικό του αριθμού N.

Μια άλλη μορφή βρόχου

Μια πιο συμπτυγμένη μορφή επαναλήψεων με τη χρήση του for

Γενικό σχήμα επανάληψης

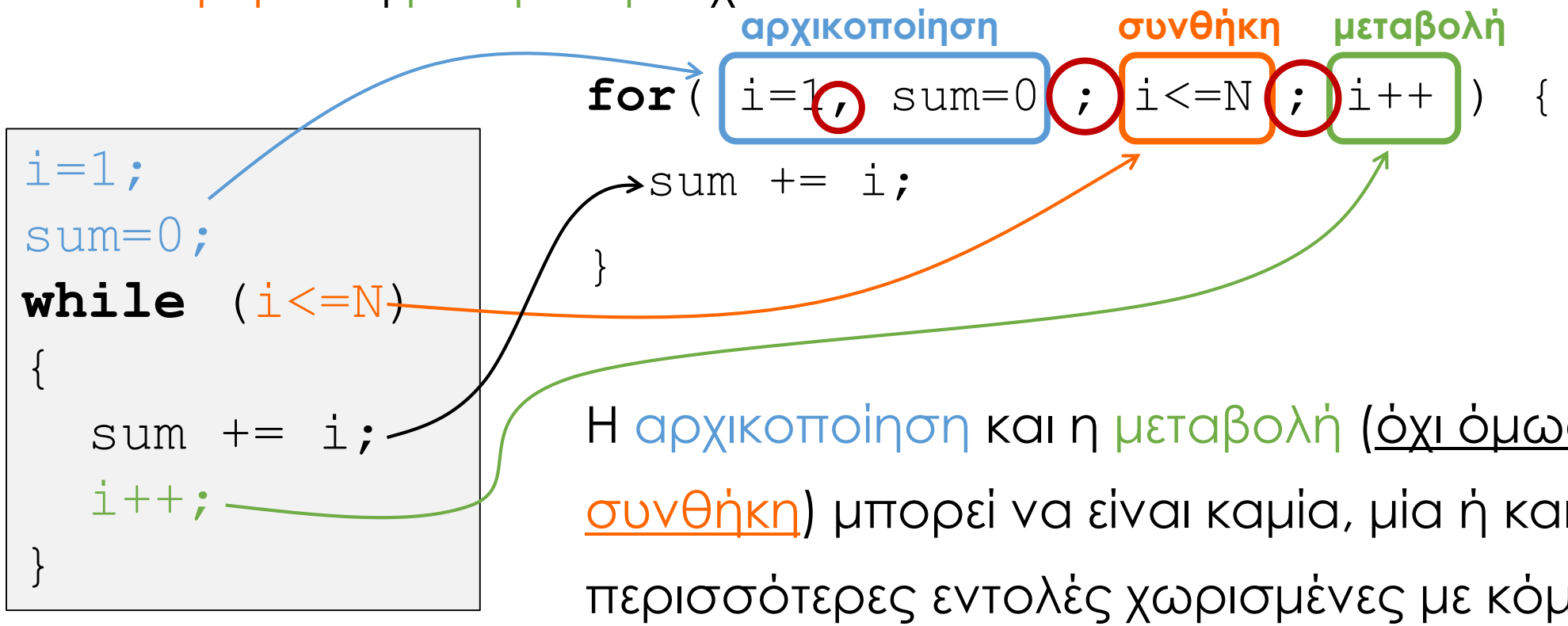
Στις περισσότερες περιπτώσεις η δομή ενός βρόχου γράφεται με βάση το ακόλουθο σχήμα



Αυτό το σχήμα χρησιμοποιείται συνεχώς στους κώδικες.

Βρόχος for

Η δομή αυτή χρησιμοποιείται ισοδύναμα με την while, αλλά μας επιτρέπει να συγκεντρωθούν σε ένα σημείο η αρχικοποίηση, η συνθήκη και η μεταβολή. Π.χ.



Η αρχικοποίηση και η μεταβολή (όχι όμως η συνθήκη) μπορεί να είναι καμία, μία ή και περισσότερες εντολές χωρισμένες με κόμματα.

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!

Πρακτική #3 – Ανάλυση παραγόντων

Πχ οι παράγοντες του n=60 είναι 2,2,3,5

```
function printFactors(n) {
    for (let factor = 2; factor <= n/factor; factor++) {
        while (n % factor == 0) {
            n /= factor;
            smPrint("% ", factor);
        }
    }
    if (n > 1) {
        smPrint("% ", n);
    }
    smPrint("\n");
}
```

n	factor
60	2
30	3
15	4
5	

2 2 3 5

Χρόνος

Αναδρομικές κλήσης

Ένας έμμεσος τρόπος επαναλήψεων χωρίς τη χρήση βρόχων

Αναδρομική κλήση συνάρτησης

Ένας έμμεσος τρόπος επαναληπτικών διαδικασιών είναι η αναδρομική κλήση μιας συνάρτησης, δηλαδή η κλήση μιας συνάρτησης μέσα από την ίδια τη συνάρτηση.

Αυτή η διαδικασία λειτουργεί ακριβώς όπως και η κλήση των υπολοίπων συναρτήσεων, δημιουργώντας ένα νέο αντίγραφο από τις τοπικές μεταβλητές και τις παραμέτρους.

Βέβαια μια τέτοια διαδικασία δεν θα είχε τέλος, θα ήταν ατέρμονη. Γι'αυτό πάντα θα πρέπει να υπάρχει τουλάχιστον μία συνθήκη που θα τερματίζει τη συνάρτηση χωρίς αυτή να καλέσει τον εαυτό της.

Η αναδρομική κλήση μιας συνάρτησης, πάντα μπορεί να αντικατασταθεί από έναν βρόχο, ο οποίος μάλιστα μπορεί να είναι και πιο αποδοτικός. Όμως είναι αρκετές φορές που η αναδρομική γραφή είναι πολύ πιο απλή ως κώδικας και πολύ πιο εύκολη στη σύλληψη.

Ερωτήσεις?

- Διαβάστε τις σημειώσεις, διαβάστε τις διαφάνειες και δείτε τα videos **πριν** ρωτήσετε
- **Συμβουλευτείτε** τη σελίδα ερωταποκρίσεων του μαθήματος

  https://qna.c-programming.allos.gr
- **Στείλτε** τις ερωτήσεις σας πριν και μετά το μάθημα στο

  c-programming-24@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε τον κώδικα ως κείμενο με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Επαναλαμβάνουμε : Μην στείλετε ποτέ κώδικα ως εικόνα μας είναι παντελώς άχρηστος!