Εισαγωγή στην Πληροφορική & στον Προγραμματισμό

Αρχές Προγραμματισμού Η/Υ (με τη γλώσσα C)

Διάλεξη #4

17 & 18 Μαρτίου 2022

Παναγιώτης Παύλου

c-programming-23@allos.gr

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.

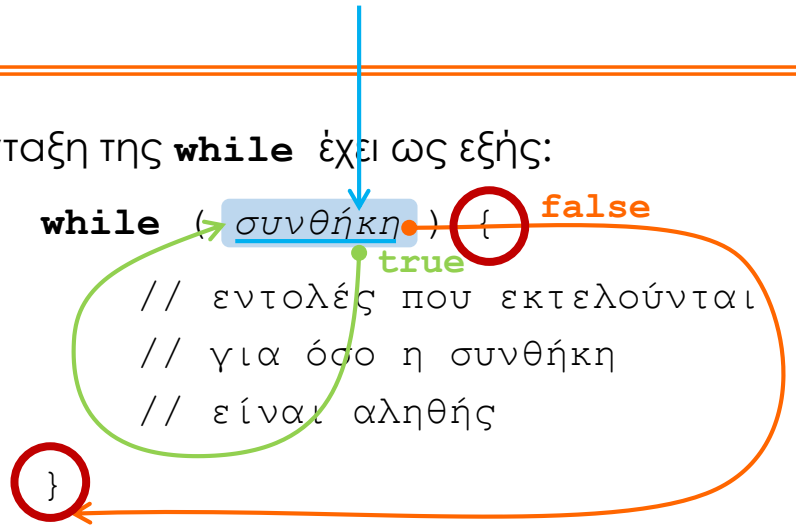
Με τους **βρόχους** είναι δυνατή η επιστροφή σε προηγούμενο σημείο του κώδικα και η επανάληψη της εκτέλεσης των ίδιων εντολών. Η επανάληψη αυτή αρχίζει και συνεχίζεται εφόσον ισχύει κάποια **συνθήκη**.

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


```
int i=1;
while (i <= 5) {
    printf("C is the best language!");
    i = i + 1;
}
```

```
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.


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

2*i

με αποτέλεσμα:

```
1 2 3 4 5
```

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 <u>επίθεμα</u>

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

ή

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

είναι σαν να είχαμε γράψει

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

k=10

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

ενώ όταν γράφεται ως <u>πρόθεμα</u>

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

ή

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

είναι σαν να γράφουμε

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

k=11

k=9

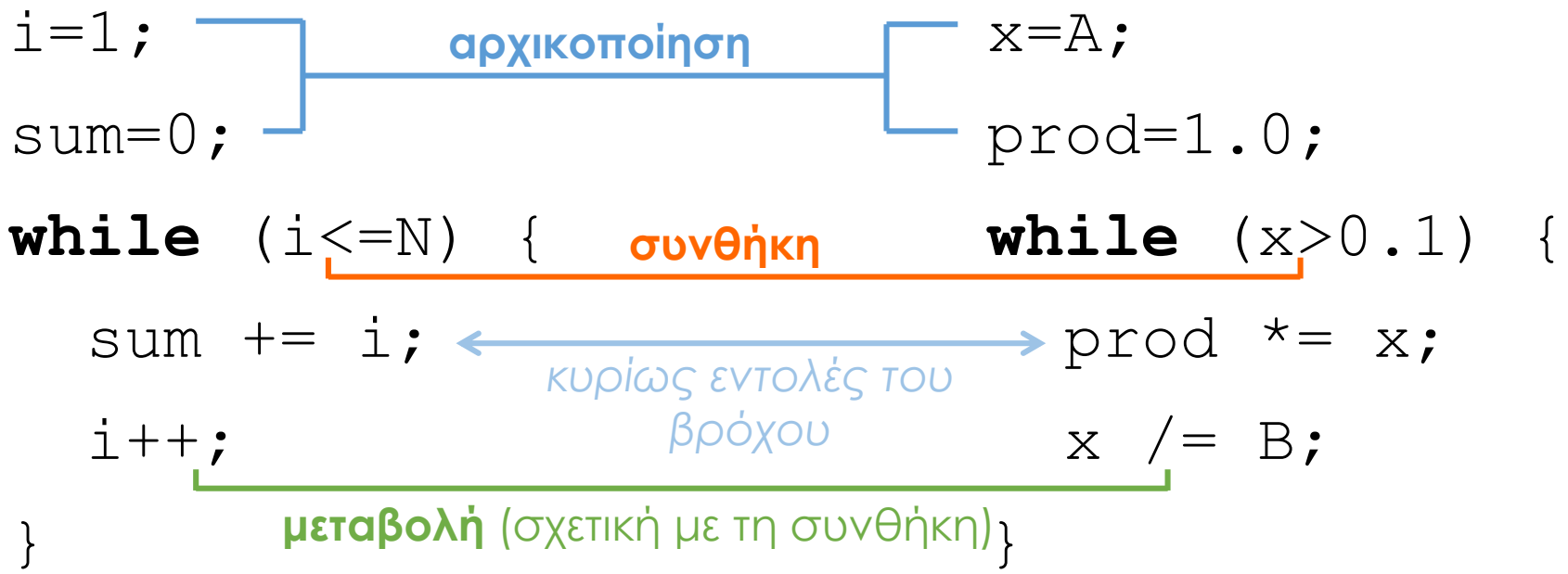
```
i--;
k = i;
```

Μια άλλη μορφή βρόχου

Μια πιο συμπτυγμένη μορφή επαναλήψεων με τη χρήση του for

Γενικό σχήμα επανάληψης

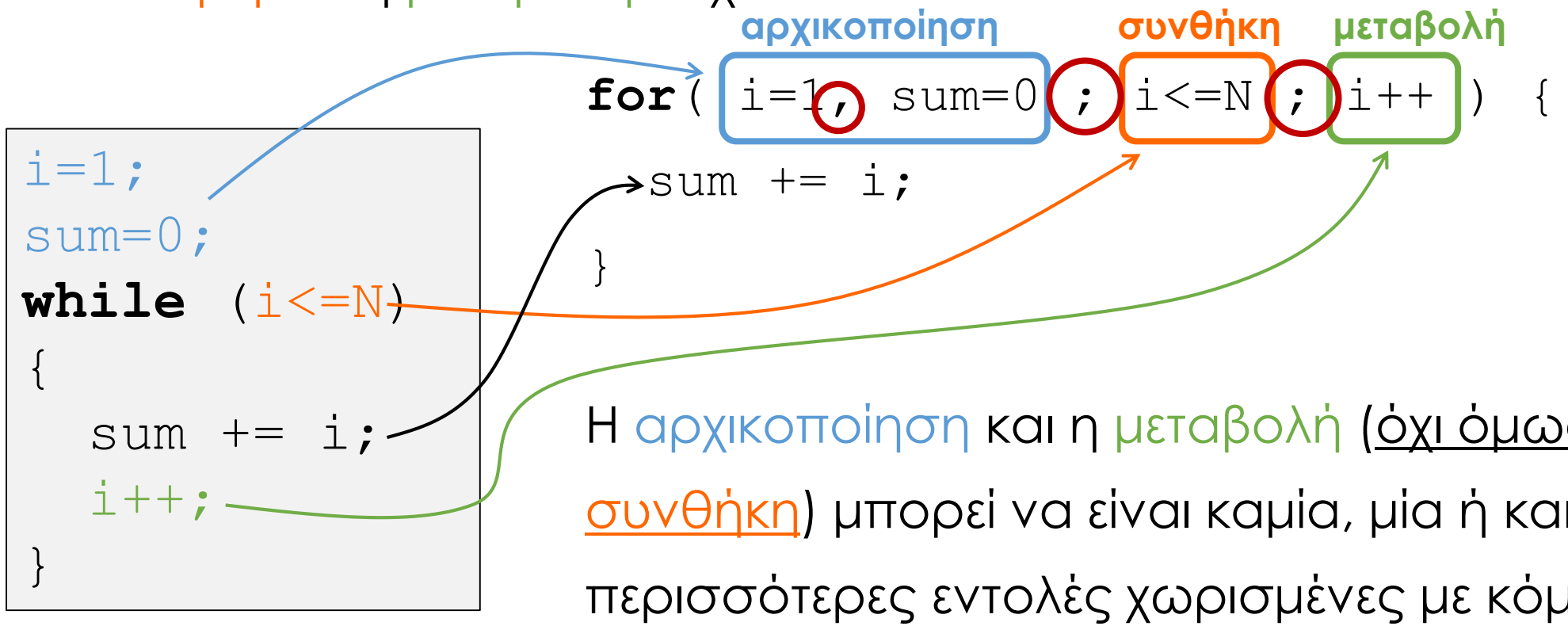
Στις περισσότερες περιπτώσεις η δομή ενός βρόχου γράφεται με βάση το ακόλουθο σχήμα



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.

Παράδειγμα – Ανάλυση παραγόντων

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

```
void printFactors(long n) {
    for (long factor = 2; factor <= n/factor; factor++) {
        while (n % factor == 0) {
            n /= factor;
            printf("%ld ", factor);
        }
    }
    if (n > 1) {
        printf("%ld ", n);
    }
    printf("\n");
}
```

n	factor
60	2
30	3
15	4
5	

2 2 3 5

Χρόνος

Άλλες μορφές ελέγχου ροής

Πιο σπάνια χρησιμοποιούμενες μορφές ελέγχου ροής

Παραλλαγές ελέγχου ροής do/while

Κάποιες – λίγες φορές – συμφέρει να **εκτελούνται οι εντολές** του while τουλάχιστον **μία φορά πριν ελεγχθεί η συνθήκη**.

Γι' αυτές τις περιπτώσεις υπάρχει η ακόλουθη μορφή:

```
do {
      // σώμα του do/while
} while( συνθήκη ) ;
```

όπου θέλει προσοχή το σύμβολο του τερματισμού της εντολής (;) που είναι υποχρεωτικό να γραφεί μόνο του μετά την while.

Παραλλαγές ελέγχου ροής switch/case (1/2)

Κάποιες φορές στα πολλαπλά `if/else if/else` οι συγκρίσεις είναι μόνο **ισοτικές** και γίνονται ανάμεσα **στην ίδια παράσταση** σε σχέση με διάφορες, **γνωστές εκ των προτέρων σταθερές τιμές**.

Γι' αυτές τις περιπτώσεις υπάρχει η switch/case η οποία συντάσσεται όπως δίπλα.

Η διάταξη με την οποία παρουσιάζεται δίπλα η switch/case μοιάζει με τη δομή της `if/else if/else`, όπου το **`default`** αντιστοιχεί στο **`else`**.

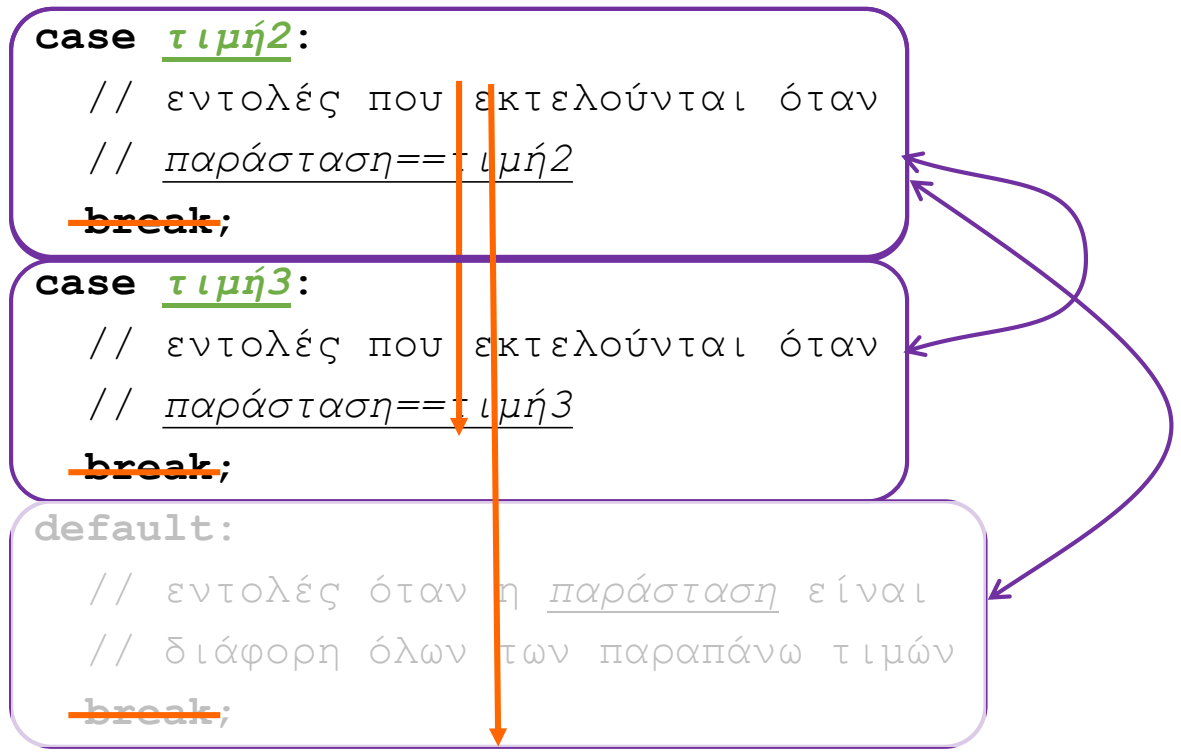
```
switch( παράσταση ) {
  case τιμή1:
    // εντολές που εκτελούνται όταν
    // παράσταση==τιμή1
    break;
  case τιμή2:
    // εντολές που εκτελούνται όταν
    // παράσταση==τιμή2
    break;
  case τιμή3:
    // εντολές που εκτελούνται όταν
    // παράσταση==τιμή3
    break;
  default:
    // εντολές όταν η παράσταση είναι
    // διάφορη όλων των παραπάνω τιμών
    break;
}
```

Παραλλαγές ελέγχου ροής switch/case (2/2)

Η switch/case όμως είναι **γενικότερη** και μερικές φορές πιο **δυνατή**, επειδή:

1. Όταν λείπει η **break**, η εκτέλεση συνεχίζεται στις εντολές που ακολουθούν κανονικά στο επόμενο case ή default, μέχρι να βρεθεί κάποιο break ή να κλείσει η switch. Αυτό δεν μπορούσε να γίνει με τα πολλαπλά `if/else if/else`.
2. Η σειρά των **case** δεν παίζει ρόλο, μπορεί να επιλεγεί βάσει του #1
3. Η θέση της **default** δεν παίζει ρόλο , μπορεί να επιλεγεί βάσει του #1
4. Η χρήση της default είναι προαιρετική, όπως και του else

```
switch( παράσταση ) {
  case τιμή1:
    // εντολές που εκτελούνται όταν
    // παράσταση==τιμή1
    break;
  case τιμή2:
    // εντολές που εκτελούνται όταν
    // παράσταση==τιμή2
    break;
  case τιμή3:
    // εντολές που εκτελούνται όταν
    // παράσταση==τιμή3
    break;
  default:
    // εντολές όταν η παράσταση είναι
    // διάφορη όλων των παραπάνω τιμών
    break;
}
```



Σημαντικά σημεία

Μετά από τη σημερινή διάλεξη θα πρέπει να γνωρίζετε:

- Τις εντολές while και for με τις οποίες μπορείτε να επαναλαμβάνετε την εκτέλεση τμημάτων του κώδικα
- Τους τελεστές ανάθεσης που σας επιτρέπουν να γράφετε τις μεταβολές των τιμών μεταβλητών με ξεκάθαρο και αποδοτικό τρόπο
- Να έχετε υπόψη σας τη διαφορά της χρήσης των ++ και -- ως πρόθεμα και ως επίθεμα
- Την ύπαρξη και λειτουργία της do/while
- Τη χρήση και τα πλεονεκτήματα της switch/case/default

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-23@allos.gr

- Εάν έχετε **πρόβλημα** με κάποιο κώδικα στείλτε μαζί τον κώδικα και τα μηνύματα λάθους από το CLion ως κείμενα με copy/paste. Εάν θεωρείτε ότι επιπλέον βοηθά και ένα στιγμιότυπο οθόνης, είναι καλοδεχούμενο.
- Τονίζουμε : Μην στείλετε **ποτέ κώδικα ως εικόνα**, είναι παντελώς άχρηστος!