ΕΠΛ232 – Προγραμματιστικές Τεχνικές και Εργαλεία

Εισαγωγή: Μορφοποίηση, Εκφράσεις, Επιλογή, Επανάληψη
(Κεφάλαια 3-4-5-6, KNK-2ED)

Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου

http://www.cs.ucy.ac.cy/courses/EPL232

Το μάθημα αυτό δομήθηκε βάση των διαλέξεων του Αναπλ. Καθηγητή Δημήτρη Ζεϊναλιπούρ

Εισαγωγική Επισήμανση

- Σε αυτή τη διάλεξη θα έχουμε τη δυνατότητα να δούμε πως υλοποιούνται **γνωστές εντολές** (if, for, while, switch, κτλ…) στην γλώσσα C
  - Ευτυχώς, **δεν υπάρχουν πολλές διαφορές** με την JAVA, άρα δε θα χρειαστεί να δώσουμε όλες τις **εισαγωγικές επεξηγήσεις.**
  - Παρακαλώ δείτε το υλικό του ΕΠΛ131, εάν χρειάζεται.
- Θα **δώσουμε** ωστόσο έμφαση στην ακριβή **σύνταξη**, **κοινά λάθη** και **παραλήψεις** όσο και σε **νέους τελεστές** που δεν έχετε διδαχθεί.
  - Δε θα δούμε πολλά παραδείγματα, αλλά θα εμπεδώσετε τις έννοιες μέσω της Άσκησης 1.

Ανάθεση

- Μια μεταβλητή μπορεί να πάρει μια τιμή μέσω ***ανάθεσης*** (assignment)***:***

```
height = 8;
```

Ο αριθμός `8` είναι μια ***σταθερά*** (constant)**.**

- Προτού ανατεθεί μια τιμή σε μια μεταβλητή πρέπει πρώτα να δηλωθεί

Ανάθεση

- Μια σταθερά αντιστοιχισμένη σε `float` μεταβλητή περιέχει συνήθως ένα δεκαδικό ψηφίο

```
profit = 2150.48;
```

- Είναι καλύτερα να προσαρτήσετε το γράμμα `f` σε μια σταθερά κινητής υποδιαστολής, εάν έχει αντιστοιχιστεί σε `float` μεταβλητή:

```
profit = 2150.48f;
```

Εάν δεν συμπεριληφθεί το f μπορεί να προκληθεί προειδοποίηση από το πρόγραμμα μεταγλώττισης.

Ανάθεση

- Όταν μια μεταβλητή έχει αντιστοιχιστεί σε μια τιμή, μπορεί να χρησιμοποιηθεί για στον υπολογισμό της τιμής μιας άλλης μεταβλητής:

```
height = 8;
length = 12;
width = 10;
volume = height * length * width;
  /* volume is now 960 */
```

- Η δεξιά πλευρά μιας ανάθεσης μπορεί να είναι ένας τύπος (***expression***) με σταθερές, μεταβλητές και τελεστές.

Εκτύπωση της τιμής μιας μεταβλητής

- Η `printf` μπορεί να χρησιμοποιηθεί για την εμφάνιση της τρέχουσας τιμής μιας μεταβλητής.
- Για να γράψετε το μήνυμα

  `Height:` *h*

  όπου *h* είναι η τρέχουσα τιμή της μεταβλητής `height`, θα χρησιμοποιήσουμε το ακόλουθο κάλεσμα του `printf`:

```
printf("Height: %d\n", height);
```

- Όπου `%d` είναι ένα σύμβολο κράτησης θέσης που υποδεικνύει που θα εκτυπωθεί η τιμή του `height`.

Εκτύπωση της τιμής μιας μεταβλητής

- Το `%d` λειτουργεί μόνο για μεταβλητές `int`; για να εκτυπώσετε μια μεταβλητή `float`, χρησιμοποιείστε `%f`.
- Από προεπιλογή, το `%f` εμφανίζει έναν δεκαδικό αριθμό με έξι ψηφία μετά την υποδιαστολή.
- Για να αναγκάσετε το `%f` να δείξει *p* ψηφία μετά το δεκαδικό σημείο, τοποθετήστε το `.`*p* μεταξύ `%` και `f`.
- Για να εκτυπώσετε τη γραμμή

```
Profit: $2150.48
```

χρησιμοποιήστε την ακόλουθη κλήσης της `printf`:

```
printf("Profit: $%.2f\n", profit);
```

- Δεν υπάρχει όριο στον αριθμό των μεταβλητών που μπορούν να εκτυπωθούν με μία μόνο κλήση της `printf`:

```
printf("Height: %d  Length: %d\n", height, length);
```

University of Cyprus

Dr. Andreas Aristidou a.aristidou@ieee.org

7

Παράδειγμα 1
Υπολογισμός του βάρους διαστάσεων ενός κουτιού

- Οι ναυτιλιακές εταιρείες συχνά χρεώνουν επιπλέον την μεταφορά κουτιών που είναι μεγάλα αλλά πολύ ελαφριά, ορίζοντας το τέλος χρέωσης βάση του όγκου αντί του βάρους.
- Η συνήθης μέθοδος για τον υπολογισμό του βάρους διαστάσεων (dimensional weight) είναι να διαιρέσετε τον όγκο με το 166 (ο επιτρεπόμενος αριθμός κυβικών ίντσων ανά λίβρα).
- Το πρόγραμμα `dweight.c` υπολογίζει το βάρος διαστάσεων ενός συγκεκριμένου κουτιού:

```
Dimensions: 12x10x8
Volume (cubic inches): 960
Dimensional weight (pounds): 6
```

University of Cyprus

Dr. Andreas Aristidou a.aristidou@ieee.org

8

Παράδειγμα 1
Υπολογισμός του βάρους διαστάσεων ενός κουτιού

- Η διαίρεση αναπαριστάτε με / στην C, οπότε ο προφανής τρόπος για τον υπολογισμό του βάρους διάστασης είναι:

```
weight = volume / 166;
```

- Στην C, ωστόσο, όταν ένας ακέραιος διαιρείται με έναν άλλο, η απάντηση είναι "περικομμένη": όλα τα ψηφία μετά την υποδιαστολή χάνονται.
  - Ο όγκος ενός κουτιού με διαστάσεις 12” × 10” × 8” είναι 960 κυβικές ίντσες.
  - Διαιρώντας τον όγκο με τον αριθμό 166 μας δίνει 5 αντί για 5.783.

Παράδειγμα 1
Υπολογισμός του βάρους διαστάσεων ενός κουτιού

- Μια λύση είναι να προσθέσουμε τον αριθμό 165 στον όγκο προτού το διαιρέσουμε με το 166:

```
weight = (volume + 165) / 166;
```

- Ένα κουτί με όγκο 166 θα μας δώσει βάρος 331/166, δηλαδή 1, ενός αν είχε όγκο 167 θα είχε βάρος 332/166, ή αλλιώς 2.

Παράδειγμα 1
Υπολογισμός του βάρους διαστάσεων ενός κουτιού

dweight.c

```
/* Computes the dimensional weight of a 12" x 10" x 8" box */

#include <stdio.h>

int main(void)
{
  int height, length, width, volume, weight;

  height = 8;
  length = 12;
  width = 10;
  volume = height * length * width;
  weight = (volume + 165) / 166;

  printf("Dimensions: %dx%dx%d\n", length, width, height);
  printf("Volume (cubic inches): %d\n", volume);
  printf("Dimensional weight (pounds): %d\n", weight);

  return 0;
}
```


Αρχικοποίηση

- Μια μεταβλητή που δεν έχει αρχικοποιηθεί με μια τιμή ονομάζεται ***uninitialized.***
- Η προσπάθεια πρόσβασης στην τιμή μιας μεταβλητής που δεν έχει αρχικοποιηθεί οδηγεί σε απρόβλεπτα αποτελέσματα.
  - Σε κάποιους μεταγλωττιστές, μπορεί ακόμα και να προκαλέσουν κατάρρευση του προγράμματος.
- Η αρχική τιμή μιας μεταβλητής μπορεί να συμπεριληφθεί στη δήλωσή της:

```
int height = 8;
```

  Η τιμή `8` λέγεται ότι είναι μια ***initializer.***
- Στην ίδια δήλωση μπορούν να αρχικοποιηθούν πολλές μεταβλητές ταυτόχρονα:

```
int height = 8, length = 12, width = 10;
```

- Each variable requires its own initializer.

```
int height, length, width = 10;
  /* initializes only width */
```

Αρχικοποίηση

- **Uninitialized variables**

```
int Sum(int n) {
      int sum, i;
      for (i = 0; i < n; i++) {
            sum = sum + 1;
      }
return sum;
}
```


Εκτύπωση εκφράσεων

- Η `printf` μπορεί να εμφανίσει την τιμή οποιασδήποτε αριθμητικής παράστασης.
- Οι δηλώσεις

```
volume = height * length * width;
printf("%d\n", volume);
```

θα μπορούσε να αντικατασταθεί με το

```
printf("%d\n", height * length * width);
```

Ανάγνωση εισόδου

- Η `scanf` απαιτεί ένα ***format string*** για να καθορίσετε την εμφάνιση των εισερχόμενων δεδομένων.
- Ένα παράδειγμα χρήσης της `scanf` που διαβάζει μια τιμή `int` :

```
scanf("%d", &i);
/* reads an integer; stores into i */
```

- Το σύμβολο `&` συνήθως είναι απαραίτητο (όχι πάντα).
- Διαβάζοντας μια τιμή `float` είναι περίπου το ίδιο:

```
scanf("%f", &x);
```

- Όπου `"%f"` επισημαίνει στην `scanf` να κοιτάξει για ένα αριθμό `float` (ο αριθμός μπορεί να περιέχει ένα δεκαδικό σημείο).

Παράδειγμα 1 (revised)
Υπολογισμός του βάρους διαστάσεων ενός κουτιού

- Το `dweight2.c` είναι μια βελτιωμένη εκδοχή του προγράμματος για τον υπολογισμό του βάρος διαστάσεων, στο οποίο ο χρήστης εισάγει τις διαστάσεις.
- Σε κάθε κλήση της `scanf` προηγείται ένα κάλεσμα της `printf` που εμφανίζει μια προτροπή στον χρήστη να εισάγει την τιμή**.**

Παράδειγμα 1 (revised)
Υπολογισμός του βάρους διαστάσεων ενός κουτιού

dweight2.c

```
/* Computes the dimensional weight of a box from input provided by the user */

#include <stdio.h>

int main(void)
{
  int height, length, width, volume, weight;

  printf("Enter height of box: ");
  scanf("%d", &height);
  printf("Enter length of box: ");
  scanf("%d", &length);
  printf("Enter width of box: ");
  scanf("%d", &width);
  volume = height * length * width;
  weight = (volume + 165) / 166;

  printf("Volume (cubic inches): %d\n", volume);
  printf("Dimensional weight (pounds): %d\n", weight);

  return 0;
}
```

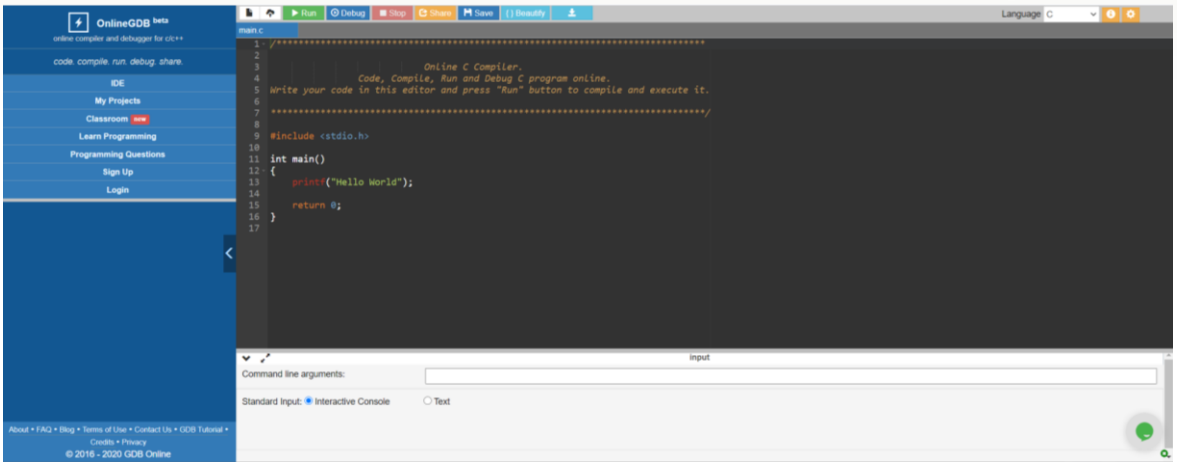
Δεν απαιτείται το σύμβολο για νέα γραμμή

Παράδειγμα 1 (revised)
Υπολογισμός του βάρους διαστάσεων ενός κουτιού

- Δείγμα εξόδου του προγράμματος:

```
Enter height of box: 8
Enter length of box: 12
Enter width of box: 10
Volume (cubic inches): 960
Dimensional weight (pounds): 6
```

Online compiler



https://www.onlinegdb.com/online_c_compiler

Ορισμός ονομάτων για σταθερές

- Οι `dweight.c` και `dweight2.c` βασίζονται στην σταθερά 166, του οποίου η έννοια μπορεί να μην είναι ξεκάθαρη σε κάποιον που διαβάζει το πρόγραμμα.
- Η Χρήση του ***macro definition,*** μπορεί να δώσει όνομα στην σταθερά:

```
#define INCHES_PER_POUND 166
```

- Κατά την προεπεξεργασία, η δήλωση

```
weight = (volume + INCHES_PER_POUND - 1) / INCHES_PER_POUND;
```

θα γίνει

```
weight = (volume + 166 - 1) / 166;
```

- Η τιμή μιας μακροεντολής μπορεί να είναι μια έκφραση:

```
#define RECIPROCAL_OF_PI (1.0f / 3.14159f)
```

Χρήση μόνο κεφαλαία γράμματα

Απαραίτητη η χρήση παρένθεσης

Περιεχόμενα Διάλεξης

- **Μορφοποιήση I/O – Formatted I/O (Κεφ. 3)**
  - printf, scanf, ορίσματα
- **Εκφράσεις – Expressions (Κεφ. 4)**
  - Αριθμητικοί Τελεστές, Τελεστές Ανάθεσης, Μοναδιαίοι & Σχεσιακοί Τελεστές, Λογικοί Τελεστές
- **Εντολές Επιλογής – Selection (Κεφ. 5)**
  - If-then-else, macros, switch
- **Εντολές Επανάληψης – Repetition (Κεφ. 6)**
  - while, do, for, break, continue, goto

Μορφοποίηση Εξόδου της `printf`

- Η **`printf`** μας επιτρέπει να εκτυπώσουμε στην έξοδο, π.χ.,
  `printf("Age: %d", 32);`
- **Προδιαγραφή Μετατροπής (Conversion Specification)**
  - Όπου *m* ***(minimum field width)*** και *p* ***(precision)*** είναι σταθερές και ***X (conversion specifier)*** *είναι γράμμα*
  - *%m . pX* : δεξιά στοίχιση,π.χ.,`("%5.3d", 40)=>|_ _040|`
  - *%–m . pX*: αριστερή, π.χ., `("%-5.3d", 40)=>|040_ _|`

    Νόμιμες Δηλώσεις: ***m.p*** *ή* ***m*** *ή* ***.p*** *και για* ***X έχουμε:***
    - *d (decimal, π.χ., 0-9) o (octal 0-7) και x (hexadecimal 0-F)*
      - *i (integer): Σε scanf σαρώνει σε* ***d (56), o (056)*** *ή x (0x56)*
    - *e (exponential, π.χ., 8.392e+02)* ***Εκθετική***
    - *f (float in fixed decimal, π.χ., 8.7)* ***Πραγματική***
    - *g (exponential | fixed decimal ανάλογα με το μέγεθος).*

Παράδειγμα
Μορφοποίηση Εξόδου της **printf**

```
/* Prints int and float values in various formats */
  #include <stdio.h>

  int main(void)
  {
    int i;
    float x;

    i = 40;
    x = 839.21f;

    printf("|%d|%5d|%-5d|%5.3d|%-5.3d|\n", i, i, i, i, i);
    printf("|%10.3f|%10.3e|%-10g|\n", x, x, x);

    return 0;
  }
```

- Output:

```
|40|   40|40   |  040|040  |
|   839.210| 8.392e+02|839.21    |
```

Ακολουθίες Escape

- Οι **ακολουθίες Escape** επιτρέπουν σε συμβολοσειρές να περιέχουν **μη-εκτυπώσιμους χαρακτήρες (ελέγχου)**
  - Alert (bell) `\a`
  - Backspace `\b`
  - New line `\n`
  - Horizontal tab `\t`

  ή **χαρακτήρες** με **ειδικό νόημα**
  - **\"** : `printf("\"Hello!\"");` => "Hello!"
  - \\ : `printf("\\");` => \

Ακολουθίες Escape

- Μια συμβολοσειρά μπορεί να περιέχει οποιονδήποτε αριθμό από ακολουθίες διαφυγής:

```
printf("Item\tUnit\tPurchase\n\tPrice\tDate\n");
```

- Η εκτέλεση αυτής της δήλωσης εκτυπώνει τα ακόλουθα:

```
Item    Unit    Purchase
        Price   Date
```


Η Συνάρτηση **`scanf`** (Λογική Σάρωσης)

- Είχαμε πει ότι η συνάρτηση `scanf` διαβάζει δεδομένα από το Standard Input με ένα συγκεκριμένο (προσδιορισμένο) format, π.χ.,
  - `int i, j; float x, y;`
  - `scanf("%d%d%f%f", &i, &j, &x, &y);`

*Η χρήση του & θα εξηγηθεί στη **διάλεξη 5**. Μέχρι τότε να δίνεται σε όλες τις μεταβλητές*

- Ουσιαστικά, η scanf επιχειρεί να βρει τους τύπους αγνοώντας τα ***white-space χαρακτήρες*** (`\n` newline, `\t` horizontal tab, `\v` vertical tab και form-feed `\f`.)
- Παράδειγμα με Δεδομένα σε Πολλαπλές γραμμές όπου ¤ το \n:

Τα \v και \t είχαν χρήση παλιά σε εκτυπωτές αλλά όχι σήμερα.

```
   1
-20   .3
   -4.0e3
```

```
••1¤-20•••.3¤•••-4.0e3¤
ssrsrrrssssrrssssrrrrrr  (s = skipped; r = read)
```

- Η `scanf` δεν διαβάζει το τελευταίο \n, άρα παραμένει στο input buffer για την επόμενη εντολή που διαβάζει από το stdin

Η Συνάρτηση `scanf` (Λογική Σάρωσης)

- **Ακέραιος (`%d`):** διαβάζει ένα-ένα χαρακτήρα μέχρι την εύρεση **χαρακτήρα ψηφίου (0-9) ή χαρακτήρα προσήμου (+,-).** Στη συνέχεια διαβάζει από το stdin μέχρι να βρει ένα χαρακτήρα που δεν είναι ψηφίο.
  - π.χ., abc: **+3432** ac
- **Συμβολοσειρά (`%s`):** διαβάζει ένα-ένα χαρακτήρα μέχρι την εύρεση του χαρακτήρα NUL (\0).
- **Αριθμός Κινητής Υποδιαστολής (%f %e %g):**
  - **Χαρακτήρας προσήμου** (προαιρετικός), μετά
  - **Ψηφία** (ενδεχομένως με δεκαδική ακρίβεια), μετά
  - **Έκθετης** (προαιρετικό): Χαρακτήρας `e` (ή `E`), προαιρετικό πρόσημο και 1 ή περισσότερα ψηφία
  - Π.χ., **`+8.392e+02`**
- **Σημείωση:** Πολλοί προγραμματιστές προτιμούν να σαρώνουν την είσοδο ως συμβολοσειρά και στη συνέχεια να κάνουν τις ανάλογες μετατροπές (για καλύτερο έλεγχο και ασφάλεια)

Η Συνάρτηση `scanf` (Λογική Σάρωσης)

- **Παράδειγμα 1:**

  `scanf("%d%d%f%f", &i, &j, &x, &y);`

  Είσοδος: `1-20.3-4.0e3¤`

  Πιο κάτω δείχνουμε τι θα έκανε η `scanf`:
  - `%d`: Αποθηκεύει 1 στο `i` και βάζει το – πίσω στο stdin.
  - `%d`: Αποθηκεύει -20 στο `j` και βάζει το `.` πίσω στο stdin.
  - `%f`: Αποθηκεύει 0.3 στο `x` και βάζει το – πίσω.
  - `%f`: Αποθηκεύει –4.0 × 10^3 στο `y` και βάζει το \n πίσω.
- **Παράδειγμα 2:** `scanf("%d%d", &i, &j);`

  Είσοδος: `2,3 // αγνοούνται μόνο τα whitespace!`

  Πιο κάτω δείχνουμε τι θα έκανε η `scanf`:
- `i=2` και «`,3`» μένει στο stdin για την επόμενη κλήση (το "," δεν είναι whitespace) .

Η Συνάρτηση **scanf** (Λογική Σάρωσης)

- **Προβληματική Κλήση 1:** `scanf("Hello: %d", &i);`

  (Input) Something 34 => Δεν σαρώνει το 34 ☹

  (Input) Hello: 34 => Σαρώνει το 34 στο i ☺

- **Προβληματική Κλήση 2:** `scanf("%d\n", &i);`

  Δηλώσαμε ότι θέλουμε να σαρώσουμε το πρότυπο `"%d\n"` άρα για επιτυχή τερματισμό του προγράμματος πρέπει να δώσουμε: α) <ακέραιο>`\n`, ή β) τουλάχιστο ένα χαρακτήρα και ένα `\n` (για διακοπή).

- **Προβληματική Κλήση 3:** `#define MAX "50" char name[MAX]; scanf("%s", name); printf("%s", name);`

  (Input) Hello34 => Σαρώνει το "Hello34\0" στο name ☺

  (Input) Hello 34 => Σαρώνει το "Hello\0" στο name ☹

  Δεν υπάρχει & μπροστά στο name

  Για σάρωση λέξεων που χωρίζονται με space έχουμε δυο επιλογές:

- `fgets(name, sizeof(name), stdin);`
- `scanf("%" MAX "[^\n]", name);`

University of Cyprus

Dr. Andreas Aristidou a.aristidou@ieee.org

29

Παράδειγμα Υπολογισμού του UPC Ψηφίου Ισοτιμίας (Parity Digit)

- Τα περισσότερα προϊόντα φέρουν ένα Universal Product Code (UPC).
- Παράδειγμα UPC-A =>

- Νόημα των ψηφίων κάτω από τον κωδικό:
  - 0ο ψηφίο: Τύπος Αντικειμένου (π.χ., 3: φάρμακα)
  - 1ο -5ο ψηφία: Κατασκευαστής
  - 6ο -10ο ψηφία: Προϊόν και μέγεθος
  - 11ο Ψηφίο: Ψηφίο Ισοτιμίας που χρησιμοποιείται για διόρθωση ενός ψηφίου λάθους (π.χ., εάν δεν σαρωθεί ορθά ένα από τα πρώτα 11 ψηφία)

University of Cyprus

Dr. Andreas Aristidou a.aristidou@ieee.org

30

Παράδειγμα Υπολογισμού του UPC Ψηφίου Ισοτιμίας (Parity Digit)

- Τρόπος υπολογισμού Ψηφίου Ισοτιμίας στο UPC-A:
  even = $0^o + 2^o + 4^o + 6^o + 10^o$ ψηφίο
  odd = $1^o + 3^o + 5^o + 7^o + 9^o$ ψηφίο
  total = (3*even + odd) – 1
  Check_Digit = 9 – (total % 10);

Παράδειγμα Υπολογισμού του UPC Ψηφίου Ισοτιμίας (Parity Digit)

```
/* Computes a Universal Product Code check digit */

#include <stdio.h>

int main(void)
{
  int i0, i1, i2, i3, i4, i5, i6, i7, i8, i9, i10,
      even, odd, total;

  printf("Enter the first (single) digit: ");
  scanf("%1d", &i0);
  printf("Enter first group of five digits: ");
  scanf("%1d%1d%1d%1d%1d", &i1, &i2, &i3, &i4, &i5);
  printf("Enter second group of five digits: ");
  scanf("%1d%1d%1d%1d%1d", &i6, &i7, &i8, &i9, &i10);
  even = i0 + i2 + i4 + i6 + i8 + i10;
  odd = i1 + i3 + i5 + i7 + i9;
  total = (3 * even) + odd - 1;

  printf("Check digit: %d\n", 9 - (total % 10));

  return 0;
}
```

Σημειώστε πως η scanf ξεχωρίζει τα ψηφία μέσω του ορίσματος πλάτους «%1d»

Σύγχυση της **printf** με **scanf**

- Μολονότι το κάλεσμα της `scanf` και `printf` μπορεί να μοιάζει, υπάρχουν σημαντικές διαφορές μεταξύ των δύο.
- Ένα συνηθισμένο λάθος είναι να βάλεις `&` μπροστά από τις μεταβλητές κατά το κάλεσμα της `printf`:

```
printf("%d %d\n", &i, &j);  /*** WRONG ***/
```


Σύγχυση της **printf** με **scanf**

- Έστω η ακόλουθη κλήση της `scanf`:

```
scanf("%d, %d", &i, &j);
```

  - η `scanf` θα αναζητήσει πρώτα έναν ακέραιο στην είσοδο, τον οποίο θα αποθηκεύσει στη μεταβλητή `i`.
  - η `scanf` στη συνέχεια, θα προσπαθήσει να ταιριάξει το κόμμα με τον επόμενο χαρακτήρα εισόδου.
  - Εάν ο επόμενος χαρακτήρας εισόδου είναι κενό (space), και όχι κόμμα, η `scanf` θα τερματίσει χωρίς να διαβάσει μια τιμή για το `j`.

Περιεχόμενο Διάλεξης

- Μορφοποιημένη Είσοδος/Εξοδ. (Κεφ. 3)
  - printf, scanf, ορίσματα
- **Εκφράσεις & Τελεστές (Κεφ. 4)**
  - Αριθμητικοί Τελεστές, Τελεστές Ανάθεσης, Μοναδιαίοι & Σχεσιακοί Τελεστές, Λογικοί Τελεστές
- **Εντολές Επιλογής (Κεφ. 5)**
  - If-then-else, macros, switch
- **Εντολές Επανάληψης (Κεφ. 6)**
  - while, do, for, break, continue, goto

Εκφράσεις & Τελεστές (Expressions & Operators)

- Ορισμοί
  - **Έκφραση (Expression):** εξίσωση η οποία δείχνει πως θα υπολογιστεί μια τιμή (π.χ., b + c)
  - **Τελεστής (Operator):** πράξη (π.χ., +)
  - **Τελεσταίοι (Operands):** Όροι μιας πράξης (π.χ., b, c)
- Η C έχει όλους τους **τελεστές** που είδατε και στα πλαίσια της JAVA, συμπεριλαμβανομένων:
  - Αριθμητικοί τελεστές: `+, -, *, /, %`
  - Τελεστές ανάθεσης, π.χ., a = b
  - Τελεστές αύξησης / μείωσης, π.χ., a++, a--
  - Λογικοί τελεστές, π.χ., a>3 && b<1
  - κτλ.

Αριθμητικοί Τελεστές

- Εάν `int` και `float` τελεστές αναμειχθούν, το αποτέλεσμα είναι τύπου `float`.
  `9 + 2.5f` έχει την τιμή 11.5, και `6.7f / 2` έχει την τιμή 3.35.
- Η συμπεριφορά των `/` και `%` για ακέραιες τιμές ορίζεται από την υλοποίηση (***implementation-defined***) => καλό να αποφεύγεται!
  `-9/7 = -1` σε μερικά CPU και `-9/7=-2` σε άλλα!
- Η τιμή του `i % j` είναι το υπόλοιπο του `i` όταν διαιρεθεί με το `j`.
  `10 % 3` είναι `1`, and `12 % 4` είναι `0`.
- Χρησιμοποιείται **πάντα παρενθέσεις** σε εκφράσεις, εναλλακτικά θα εφαρμόζεται η προτεραιότητα των τελεστών.

  Highest: `+ −` (unary) , π.χ., b = -a;
  `* / %`
  Lowest: `+ −` (binary)

Τα δεκαδικά στοιχεία δεν υπάρχουν

Ο τελεστής % απαιτεί μόνο ακέραιους


University of Cyprus

Dr. Andreas Aristidou a.aristidou@ieee.org

37

Εταιρικότητα Τελεστών (Operator Associativity)

- *Το θέμα της* ***Εταιρικότητα (Associativity)*** **προκύπτει** όταν έχουμε δυο ή **περισσότερους τελεστές** με **ΙΣΗ προτεραιότητα** (χωρίς **παρενθέσεις**!) π.χ., `i − j − k`
- **Αριστερή Εταιρικότητα (*left associative*):** *εάν ομαδοποιεί τους τελεσταίους από αριστερά στα δεξιά.*
  - Π.χ., οι **δυαδικοί αριθμητικοί** τελεστές (`*`, `/`, `%`, `+`, and `−`)
    - `i − j − k` ισοδύναμο με `(i − j) − k`
    - `i * j / k` ισοδύναμο με `(i * j) / k`
- **Δεξιά Εταιρικότητα (*right associative*):** *εάν ομαδοποιεί τους τελεσταίους από δεξιά στα αριστερά*
  - Π.χ., οι **μοναδιαίοι αριθμητικοί** τελεστές `(+ και −)`
    - `- + i` ισοδύναμο με `-(+i)`
    - `i + j / k` ισοδύναμο με `i + (j / k)`
- Ανεξάρτητα από Εταιρικότητα, η C δεν ορίζει τη **σειρά εκτέλεσης των όρων** μιας έκφρασης, π.χ., `(a + b) * (c − d)`

Χρησιμοποιείτε Παρενθέσεις!


University of Cyprus

Dr. Andreas Aristidou a.aristidou@ieee.org

38

Τελεστής Ανάθεσης (Assignment Operator)

Παραδείγματα

- `int i; i = 72.99f;  /* i is now 72 */`
- `i = j = k = 0; /* μαζική αρχικοποίηση */`

- Ο **τελεστής ανάθεσης** έχει δεξιά Εταιρικότητα:

```
i=(j=(k=0)); /* μαζική αρχικοποίηση */
int i; float f; f = i = 33.3f; // το i έχει τιμή 33 και το f 33.0
```

- **Σύνθετη Ανάθεση** (Compound) έχει δεξιά Εταιρικότητα:

```
+=      -=      *=    /=    %=
i += 2;   /* ίδιο με i = i + 2; */
i =+ 2;   /* Προσοχή: ίδιο i = (+2); */
```


Τελεστής Αύξησης/Μείωσης (Increment/Decrement Operator)

- Ο τελεστής **αύξησης (++)** / **μείωσης (−−)** μπορεί να χρησιμοποιηθεί ως **προθεματικός (prefix)** ή **επιθεματικός (postfix)** τελεστής ΚΑΙ επηρεάζει την τιμή μιας μεταβλητής

- Παράδειγμα:

Άμεση αύξηση `i`: *Πρώτα αύξηση μετά εκτύπωση*

```
i = 1;
printf("i is %d\n", ++i);   /* prints "i is 2" */
printf("i is %d\n", i);     /* prints "i is 2" */
```

Πρώτα εκτύπωση μετά αύξηση

```
i = 1;
printf("i is %d\n", i++);   /* prints "i is 1" */
printf("i is %d\n", i);     /* prints "i is 2" */
```

Αποτίμηση Αριθμητικών Εκφράσεων (Arithmetic Expression Evaluation)

- Πίνακας Τελεστών που συζητήθηκαν μέχρι στιγμής:

Προτεραιότητα	*Όνομα*	*Σύμβολο(α)*	*Εταιρικότητα*
1	increment (postfix)	a++	left-to-right
	decrement (postfix)	a--	
2	increment (prefix)	++a	right-to-left
	decrement (prefix)	--a	
	unary plus	+	
	unary minus	-	
3	multiplicative	* / %	left-to-right
4	additive	+ -	left-to-right
5	assignment	= *= /= %= += -=	right-to-left

Χρησιμοποιείτε Παρενθέσεις!

Τι θα εκτελεστεί; `a = b += c++ - d + --e / -f`

`(a = (b += (((c++) - d) + ((--e) / (-f)))))` Χρησιμοποιείται Παρενθέσεις!

Γενική Παρατήρηση: Οι μοναδιαίοι τελεστές έχουν ψηλότερη προτεραιότητα από τους δυαδικούς τελεστές

University of Cyprus

Dr. Andreas Aristidou a.aristidou@ieee.org

41

Αποτίμηση Αριθμητικών Εκφράσεων (Arithmetic Expression Evaluation)

- Υπάρχουν **ασαφείς (ambiguous)** εκφράσεις στη C που οδηγούν σε απροσδιόριστη συμπεριφορά (***undefined behavior)*** *ενός προγράμματος.*

- Δηλ., διαφορετικοί μεταγλωττιστές => διαφορετική συμπ.

```
a = 5;
c = (b = a + 2) - (a = 1);
```

Αρισ. Εταιρ Έκφρασης

  - Εάν εκτελεστεί πρώτα `(b = a + 2)` τότε `b=7, a=1` και **`c=6 (GCC)`**
  - Εάν εκτελεστεί πρώτα το `(a = 1)` τότε `a=1, b=3` και **`c=2`**
  - *Κάποιοι μεταγλωττιστές μπορεί να δώσουν προειδοποίηση: "operation on 'a' may be undefined" ή* **"warning: unsequenced modification and access to 'a' [-Wunsequenced]"**

- **Συμπέρασμα:**
  - Χρησιμοποιείτε ΠΑΝΤΑ **παρενθέσεις** σε εκφράσεις
  - **Αποφεύγετε ΠΑΝΤΑ έντεχνες εκφράσεις** που μπορεί να οδηγήσουν σε **απροσδιόριστη συμπεριφορά**

University of Cyprus

Dr. Andreas Aristidou a.aristidou@ieee.org

42

Συνήθη λάθη

- Προσέξτε για απροσδόκητα αποτελέσματα σε συνδεδεμένες αναθέσεις:

```
int i;
float f;

f = i = 33.3f;
```

- Στο `i` αντιστοιχεί η τιμή 33, οπότε στο `f` αντιστοιχεί η τιμή 33.0 (όχι η 33.3).
- Είναι παράνομο να τοποθετήσετε οποιοδήποτε άλλο είδος έκφρασης στην αριστερή πλευρά μιας παράστασης ανάθεσης:

```
12 = i;       /*** WRONG ***/
i + j = 0;    /*** WRONG ***/
-i = j;       /*** WRONG ***/
```

Το πρόγραμμα μεταγλώττισης θα παρουσιάσει ένα μήνυμα σφάλματος, π.χ. *"invalid lvalue in assignment."*

Συνήθη λάθη

- **ΠΡΟΣΟΧΗ:** *v* += *e* δεν είναι πάντα "ισοδύναμο" με *v* = *v* + *e*.
  - Π.χ. Ένα πρόβλημα είναι η προτεραιότητα του χειριστή: `i *= j + k` δεν είναι το ίδιο με το `i = i * j + k`.
- Έστω και αν το `i =+ j` θα μεταγλωττίσει ως σωστό, είναι ισοδύναμο με το `i = (+j)`.

Αποτίμηση Λογικών Εκφράσεων (Logical Expression Evaluation)

- **Λογική Έκφραση στη C:** Έκφραση που αποτιμάται σε 0 (false) ή 1 (true)
  - π.χ., `(a>3)` ή `(3<a && a<5)`
- Σε πολλές γλώσσες (όπως η JAVA), μια λογική έκφραση παράγει ένα "**Boolean**" ή **"λογικό"** τύπο (π.χ., βασικός τύπος boolean ή τύπος κλάσης Boolean).
  - Στη C99 υπάρχει όπως είδαμε ο _Bool τύπος που είναι και αυτός ουσιαστικά μια ακέραια τιμή.

Boolean Τιμές στη C (και C99)

- Για να γίνουν τα προγράμματα πιο κατανοητά, οι προγραμματιστές παραδοσιακά όριζαν macros με ονόματα όπως `TRUE` και `FALSE`:

```
#define TRUE 1
#define FALSE 0
if (flag == TRUE)   ή if (flag)
```

Προσοχή:
Zero: 0 FALSE
Positives: ≥ 1 TRUE
Negatives: ≤ -1 TRUE

- Η C99 παρέχει τον τύπο `_Bool` (στην πράξη απλά ένας ακέραιος ο οποίος λαμβάνει μόνο τις τιμές 0, 1).
- Εάν περιλάβουμε την `<stdbool.h>` τότε μπορούμε να χρησιμοποιήσουμε τα macros `true` και `false` αλλά και τις δηλώσεις:

```
bool flag;
```

Αποτίμηση Λογικών Εκφράσεων (Logical Expression Evaluation)

- **Σχεσιακοί Τελεστές (Relational Operators)**

<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
==	equal to
!=	not equal to

- **Τελεστές Ισότητας (Equality Operators)**

==	equal to
!=	not equal to

- **Λογικοί Τελεστές (Logical Operators)**

!	logical negation	(μοναδιαίος τελεστής, δεξιά εταιρικότητα)
&&	logical *and*	*(δυαδικός τελεστής, αριστερή εταιρικότητα)*
||	logical *or* *(δυαδικός τελεστής, αριστερή εταιρικότητα)*	

Προσοχή:
Διαφορετικό από & και | που είναι οι λεγόμενοι δυαδικοί τελεστές που θα δούμε προς το τέλος του μαθήματος.

Αποτίμηση Λογικών Εκφράσεων (Logical Expression Evaluation)

Παραδείγματα

- **Προτεραιότητα λογικών** τελεστών **μικρότερη** από τους **αριθμητικούς** τελεστές:

  π.χ., `i + j < k - 1` σημαίνει `(i + j) < (k - 1)`

- Η **Εταιρικότητα** των **σχεσιακών τελεσ.** είναι **αριστερή**:

  π.χ., `i < j < k` σημαίνει `(i < j) < k`

  - Εάν η πρόθεση μας ήταν να ελέγξουμε ότι `j` μεταξύ `i` και k έπρεπε να γράψουμε: `i < j && j < k`.

- Τόσο το `&&` όσο και το `||` **"βραχυκυκλώνουν" ("short-circuit")** μια αποτίμηση: αποτιμούν το αριστερότερο σκέλος και μετά τα δεξιότερα (εάν χρειάζεται).

  - `(i != 0) && ((j / i) > 0) // εάν δεν ισχύει το πρώτο σκέλος της έκφρασης δεν εκτελείται το δεξιό`

Περιεχόμενο Διάλεξης

- **Μορφοποιημένη Είσοδος/Εξοδ. (Κεφ. 3)**
  - printf, scanf, ορίσματα
- **Εκφράσεις (Κεφ. 4)**
  - Αριθμητικοί Τελεστές, Τελεστές Ανάθεσης, Μοναδιαίοι & Σχεσιακοί Τελεστές, Λογικοί Τελεστές
- **Εντολές Επιλογής (Κεφ. 5)**
  - If-then-else, macros, switch
- **Εντολές Επανάληψης (Κεφ. 6)**
  - while, do, for, break, continue, goto

Η Εντολή Επιλογής `if..else`

- Η εντολή επιλογής `if` επιτρέπει την **εκτέλεση** μιας ή περισσότερων εντολών, εάν η τιμή της **λογικής έκφρασης expression** αποτιμάται σε **μη-μηδενική τιμή**

```
if ( expression ) { stmt1; stmt2; … }
else if (expression) { stmt1; stmt2; … }
else { stmt1; stmt2; … }
```

- **Συνηθισμένο Λάθος I:** `if (i = 0)`, ενώ εννοούσαμε `if (i == 0)`
  - Μπορείτε να γράφετε `(0 == i)`, έτσι ώστε εάν σας ξεφύγει ένα = να πάρετε λάθος μεταγλώττισης (lvalue)
- **Έλεγχος Εύρους**
  - $0 \leq i < n$: `if (0 <= i && i < n) …`
  - $i<0$ ή $n<=i$: `if (i < 0 || n <= i) …`

Η Εντολή Επιλογής **if..else**

- Μια εντολή `if` μπορεί να έχει μια ρήτρα `else` :

  `if (` *expression* `)` *statement* `else` *statement*

- Η δήλωση που ακολουθεί τη λέξη `else` εκτελείται εάν η παράσταση έχει την τιμή 0.
- Παράδειγμα:

```
if (i > j)
  max = i;
else
  max = j;
```

Η Εντολή Επιλογής **if..else**

- Δεν είναι ασυνήθιστο για τις εντολές `if` να είναι ένθετες μέσα σε άλλες `if` :

```
if (i > j)
  if (i > k)
    max = i;
  else
    max = k;
else
  if (j > k)
    max = j;
  else
    max = k;
```

- Ευθυγραμμίζοντας κάθε `else` με το αντίστοιχο `if` διευκολύνει την γραφή του προγράμματος.

Η Εντολή Επιλογής **if..else**

- Για να αποφύγετε τη σύγχυση, μην διστάσετε να προσθέσετε αγκύλες:

```
if (i > j) {
  if (i > k)
    max = i;
  else
    max = k;
} else {
  if (j > k)
    max = j;
  else
    max = k;
}
```

Η Εντολή Επιλογής **if..else**

- **Συνηθισμένο Λάθος II ('dangling else'):**

```
if (y != 0)
  if (x != 0)
    result = x / y;
else
  printf("Error: y is equal to 0\n");
```

  - Σε ποιο if ανηκει το "else";
- **Μάθημα:** Χρησιμοποιείται πάντα { } παρενθέσεις σε εντολές if, while, do, κτλ. !
  - Στις διαφάνεια ίσως να μην χρησιμοποιούνται πάντα για εξοικονόμηση χώρου.
  - Στον κώδικα σας ωστόσο να χρησιμοποιούνται ΠΑΝΤΑ.

Παράδειγμα
Υπολογισμός της αποζημίωσης ενός μεσίτη

- Όταν αγαθά πωλούνται ή αγοράζονται μέσω ενός μεσίτη, η προμήθεια του μεσίτη συχνά εξαρτάται από την αξία των αγαθών που είναι υπό διαπραγμάτευση.
- Ας υποθέσουμε ότι ο μεσίτης χρεώνει τα ποσά που εμφανίζονται στον ακόλουθο πίνακα:

Transaction size	*Commission rate*
Under $2,500	$30 + 1.7%
$2,500–$6,250	$56 + 0.66%
$6,250–$20,000	$76 + 0.34%
$20,000–$50,000	$100 + 0.22%
$50,000–$500,000	$155 + 0.11%
Over $500,000	$255 + 0.09%

- Η ελάχιστη χρέωση είναι $39.

Παράδειγμα
Υπολογισμός της αποζημίωσης ενός μεσίτη

broker.c

```
/* Calculates a broker's commission */

#include <stdio.h>

int main(void)
{
  float commission, value;

  printf("Enter value of trade: ");
  scanf("%f", &value);

  if (value < 2500.00f)
    commission = 30.00f + .017f * value;
  else if (value < 6250.00f)
    commission = 56.00f + .0066f * value;
  else if (value < 20000.00f)
    commission = 76.00f + .0034f * value;
  else if (value < 50000.00f)
    commission = 100.00f + .0022f * value;
  else if (value < 500000.00f)
    commission = 155.00f + .0011f * value;
  else
    commission = 255.00f + .0009f * value;
  if (commission < 39.00f)
    commission = 39.00f;

  printf("Commission: $%.2f\n", commission);

  return 0;
}
```

Η Μακροεντολή – Τριαδικός Τελεστής (expr1) ? expr2 : expr3

- Άλλος τρόπος διατύπωσης If-then-else είναι με τη χρήση **Μακροεντολής (Macro):**
  - **(expr1) ? expr2 : expr3** Ίδιο με:
  - If (expr1 != 0) { return expr2; } else { return expr3; }
  - Έχει το **πλεονέκτημα** ότι το **αποτέλεσμα** μπορεί να ανατεθεί σε **μεταβλητή** αλλά **δυσχεραίνει** τον **δομημένο προγραμματισμό**.
- Παράδειγμα:

```
int i = 1, j = 2, k;
k = i > j ? i : j;            /* k is now 2 */
k = (i >= 0 ? i : 0) + j;     /* k is now 3 */
printf("%d\n", i > j ? i : j);      /* prints 2 */
return i > j ? i : j;               /* returns 2 */
```

Η Εντολή Επιλογής **switch**

- Η εντολή πολλαπλής επιλογής `switch` διαβάζεται ευκολότερα από πολλαπλά `if` και είναι συχνά και γρηγορότερη.

```
switch ( control-expression ) {
  case constant-expression : statements; break;
  …
  case constant-expression : statements; break;
  default : statements
}
```

int or char

Π.χ., 5 ή 5+10

- Θεληματική Παράληψη break

```
switch (grade) {
    case 2:
    case 1:  printf("Passing"); break;
    case 0:  printf("Failing"); break;
    default: printf("Illegal grade"); break;
  }
```

Περιεχόμενο Διάλεξης

- Μορφοποιημένη Είσοδος/Εξοδ. (Κεφ. 3)
  - printf, scanf, ορίσματα
- Εκφράσεις & Τελεστές (Κεφ. 4)
  - Αριθμητικοί Τελεστές, Τελεστές Ανάθεσης, Μοναδιαίοι & Σχεσιακοί Τελεστές, Λογικοί Τελεστές
- Εντολές Επιλογής (Κεφ. 5)
  - If-then-else, macros, switch
- **Εντολές Επανάληψης (Κεφ. 6)**
  - while, do, for, break, continue, goto

Εντολές Επανάληψης της C

- **`while ( expression ) statement;`**
  - *το expression αποτιμάται πριν το loop.*
  - `while ( expr ) { stmt1; stmt2;… }`
  - `while (1)` *Άπειρο (έξοδος με* `break,return,exit`*)*
  - *Κοινό λάθος:* `while (expr); stmt` *ή* `if (expr);`
- **`do statement while ( expression ) ;`**
  - *το statement εκτελείται πάντα τουλάχιστο μια φορά*
  - *Καλή ιδέα να χρησιμοποιούνται ΠΑΝΤΑ οι { } και να υπάρχει στοίχιση.*
- **`for ( expr1 ; expr2 ; expr3 ) statement`**
  - *Διατυπώνεται και με while:* `i = 10; while (i > 0) { i--; }`

Η Εντολή **while**

- Η εντολή `while` :

```
i = 1;
while (i < n)
  i = i * 2;
```

Ορίζει τον τερματισμό

- Για `n` ίσο με 10:

`i = 1;`	`i` is now 1.
Is `i < n`?	Yes; continue.
`i = i * 2;`	`i` is now 2.
Is `i < n`?	Yes; continue.
`i = i * 2;`	`i` is now 4.
Is `i < n`?	Yes; continue.
`i = i * 2;`	`i` is now 8.
Is `i < n`?	Yes; continue.
`i = i * 2;`	`i` is now 16.
Is `i < n`?	No; exit from loop.

University of Cyprus
Copyright © 2008 W. W. Norton & Company.
All rights reserved.

Παράδειγμα

Άθροιση μιας σειράς αριθμών

- Το πρόγραμμα `sum.c` αθροίζει μια σειρά ακεραίων που εισάγονται από το χρήστη:

```
This program sums a series of integers.
Enter integers (0 to terminate): 8 23 71 5 0
The sum is: 107
```

University of Cyprus
Copyright © 2008 W. W. Norton & Company.
All rights reserved.

Παράδειγμα
Άθροιση μιας σειράς αριθμών

sum.c

```
/* Sums a series of numbers */

#include <stdio.h>

int main(void)
{
  int n, sum = 0;

  printf("This program sums a series of integers.\n");
  printf("Enter integers (0 to terminate): ");

  scanf("%d", &n);
  while (n != 0) {
    sum += n;
    scanf("%d", &n);
  }
  printf("The sum is: %d\n", sum);

  return 0;
}
```

University of Cyprus
Copyright © 2008 W. W. Norton & Company.
All rights reserved.
Dr. Andreas Aristidou a.aristidou@ieee.org

Εντολές Επανάληψης της C (Επανάληψη For)

- Για απαρίθμηση **n** στοιχείων
  - *Απαρίθμηση (πάνω) από* 0 *σε* n–1*:* for (i = 0; i < n; i++)
  - *Απαρίθμηση (πάνω) από* 1 *σε* n*:* for (i = 1; i <= n; i++)
  - *Απαρίθμηση (κάτω) από* n–1 *σε* 0*:* for (i = n - 1; i >= 0; i--)
  - *Απαρίθμηση (κάτω) από* n *σε* 1*:* for (i = n; i > 0; i--)
- **ΠΟΤΕ** δεν χρησιμοποιείται **πραγματική τιμή** στον μετρητή για αποφυγή προβλημάτων (π.χ., `f==10.0f`)
- Επεξήγηση Σύνταξης
  - **`for (; i > 0; --i)`** **=>** Χωρίς αρχικοποίηση
  - **`for (; i > 0;)`** **=>** Ίδιο με "while (i>0)"
  - **`for (;;)`** **=>** Infinite Loop
  - `for (int i = 0; i < n; i++)` **=>** OK για C99 (gcc -std=c99 … το i δεν έχει τιμή εκτός του scope του loop)
    - `for (int i = 0, j = 0; i < n; i++)` // όπως και στη JAVA

University of Cyprus
Dr. Andreas Aristidou a.aristidou@ieee.org

Η εντολή **for**

- Για παράδειγμα, έστω

```
for (i = 10; i > 0; --i)
  printf("T minus %d and counting\n", i);
```

- Η αντίστοιχη εντολή με `while` :

```
i = 10;
while (i > 0) {
  printf("T minus %d and counting\n", i);
  --i;
}
```

Η εντολή **for** στη C99

- Στην C99, η πρώτη έκφραση στην εντολή `for` μπορεί να αντικατασταθεί με δήλωση.
- Αυτή η δυνατότητα επιτρέπει στον προγραμματιστή να δηλώσει μια μεταβλητή για χρήση από τον βρόχο:

```
for (int i = 0; i < n; i++)
  …
```

- Η μεταβλητή `i` δεν χρειάζεται να έχει δηλωθεί πριν από την εντολή.

Η εντολή **for** στη C99

- Μια μεταβλητή που δηλώνεται στην `for` δεν είναι προσβάσιμη έξω από το σώμα του βρόχου

```
for (int i = 0; i < n; i++) {
  …
  printf("%d", i);
   /* legal; i is visible inside loop */
  …
}
printf("%d", i);    /*** WRONG ***/
```

Εντολές Επανάληψης της C (Έξοδος από Loops)

- Το κανονικό **σημείο εξόδου** μιας επανάληψης είναι η **αρχή** (`while` ή `for`) ή το **τέλος** (`do`).
- Έλεγχος ροής: `break` (loops, switch), `continue` (μόνο σε loops), `goto` τα οποία δουλεύουν για ένα επίπεδο.

```
while (…) {
  switch (…) {     break;   }
}
```

- Το **goto** μπορεί να ανακατευθύνει τη ροή εκτέλεσης σε οποιαδήποτε έκφραση σε συνάρτηση, η οποία είναι σημειωμένη (labeled).
  - **Ενάντια στον Δομημένο Προγραμματισμό** => **Να αποφεύγεται,** εφόσον οδηγεί σε **κώδικα spaghetti**.

Η εντολή **continue**

- Ένας βρόχος που χρησιμοποιεί την εντολή `continue`:

```
n = 0;
sum = 0;
while (n < 10) {
  scanf("%d", &i);
  if (i == 0)
    continue;
  sum += i;
  n++;
  /* continue jumps to
here */
}
```

- Ο ίδιος βρόχος γραμμένος χωρίς την εντολή `continue`:

```
n = 0;
sum = 0;
while (n < 10) {
  scanf("%d", &i);
  if (i != 0) {
    sum += i;
    n++;
  }
}
```

Dr. Andreas Aristidou a.aristidou@ieee.org