Π. Σταθοπούλου
pstath@ece.upatras.gr ή
pstath@upatras.gr

Ομάδα Α΄ (Φοιτητές με μονό αριθμό Μητρώου)

- Διδασκαλία : Παρασκευή 11πμ-13μμ ΗΛ7
- Φροντιστήριο : Δευτέρα 11πμ-12πμ ΗΛ4

Προηγούμενη Διάλεξη

1. Προετοιμασία για το 1ο Εργαστήριο
2. Βασικές Εντολές για την σύνταξη προγραμμάτων σε γλώσσα C
   - Εντολές Προεπεξεργαστή (*#incude*, *#define*)
     - **Τυπικές Βιβλιοθήκες της C**
     - ***<ctype.h> <math.h><stdio.h> <stdlib.h><string.h>***
   - Εντολές εισόδου, εξόδου & τρόπος σύνταξης
     - ***scanf(), printf()***
3. Πρώτη Γνωριμία με
   - τους τελεστές & τον βρόχο **for**

Σήμερα

❶Μεταβλητές, Σταθερές & Τύποι Δεδομένων
(πληροφορία/δεδομένο ⟶ μεταβλητή)
- *αριθμητικοί τύποι μεταβλητών*
- *τελεστής* ***sizeof()***
- *η λέξη_κλειδί (keyword)* ***typedef***

❷Προτάσεις, Εκφράσεις & Τελεστές
(statements, expressions & operators)

❸Απλές Εντολές & Εντολές ελέγχου
(simple & control statements)

Μεταβλητές, Σταθερές & Τύποι Δεδομένων

Ορισμός:

Μεταβλητή *είναι μία ονομαστική θέση αποθήκευσης δεδομένων στην μνήμη του υπολογιστή*

Ο υπολογιστής χρησιμοποιεί ***(RAM)***

Δεδομένα	Απαιτούμενα Bytes
το γράμμα x	*1 byte*
ο αριθμός 800	*2 byte*
ο αριθμός 633,745	*4 byte*
Η φράση "Hello word"	*22 byte*
μία σελίδα	*3000 byte*

Μεταβλητές, Σταθερές & Τύποι Δεδομένων

*Η **διαχείριση** των **μεταβλητών** υπαγορεύει συγκεκριμένους κανόνες*

Κανόνες χειρισμού μεταβλητών

- *Σωστή ονοματολογία*
- *Διάκριση τύπων μεταβλητών*
- *Δήλωση μεταβλητών &*
- *Απόδοση τιμών στις μεταβλητές*

Μεταβλητές, Σταθερές & Τύποι Δεδομένων

Σωστή ονοματολογία

ΝΑΙ	ΟΧΙ
Χρήση Περιγραφικών Ονομάτων	*Λέξεις κλειδιά*
Υιοθέτηση ενός στυλ ονοματολογίας	*Πρώτος ψηφία (0 έως 9)*
Διάκριση πεζών/κεφαλαίων	*Πρώτος χαρακτήρας _*

Μεταβλητές, Σταθερές & Τύποι Δεδομένων

Βασικοί τύποι δεδομένων

- Αριθμητικοί
- Boolean
  - Τιμές true, false
- Χαρακτήρες
- Αλφαριθμητικοί

int ακέραιος
float αριθμός κινητής υποδιαστολής απλής ακρίβειας(32 bits)
double αριθμός κινητής υποδιαστολής απλής ακρίβειας(64 bits)

enum boolean {false, true}

[ ] πίνακας, struct δομές
Union ένωση* δείκτης

char χαρακτήρας

Μεταβλητές, Σταθερές & Τύποι Δεδομένων

Διάκριση τύπων μεταβλητών
(Αριθμητικοί τύποι μεταβλητών)

Τύπος μεταβλητής	Λέξη κλειδί	Απαιτούμενα bytes	Περιοχή
Χαρακτήρας	***char***	***1***	*-128 έως 127*
Μικρός ακέραιος	***short***	***2***	*-32767 έως 32767*
Ακέραιος	***int***	***4***	*-2.147.483.647 έως 2.147.483.647*
Μεγάλος ακέραιος	***long***	***4***	*-2.147.483.647 έως 2.147.483.647*
Κινητής υποδιαστολής απλής ακρίβειας	***float***	***4***	*$1.2E^{-38}$ έως 3.4 E^{38}*
Κινητής υποδιαστολής απλής ακρίβειας	***double***	***8***	*$2.2E^{-308}$ έως 1.8 E^{308}*

Μεταβλητές, Σταθερές & Τύποι Δεδομένων

Τελεστής *sizeof()*

sizeof() *μοναδιαίος τελεστής που εφαρμόζεται σε μεταβλητή ή τύπο και δίνει το μέγεθος σε bytes*

Τελεστής *sizeof()*

```
/* Πρόγραμμα που εκτυπώνει το μέγεθος τύπου μεταβλητής
   sizeof.c---type in bytes */
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  printf("\n\tA Character is %d bytes",sizeof(char));
  printf("\n\tAn Integer  is %d bytes",sizeof(int));
  printf("\n\tA short  is %d bytes",sizeof(short));
  printf("\n\tA long is %d bytes",sizeof(long));
  printf("\n\tA long long is %d bytes",sizeof(long long));
  printf("\n\tA float is %d bytes",sizeof(float));
  printf("\n\tA double is %d bytes",sizeof(double));
  printf("\n\tA long double is %d bytes",sizeof(long double));
  printf("\n"); printf("\n");

  system("PAUSE");
  return 0;
}
```

Σχόλια

Κυρίως Πρόγραμμα

Δηλώσεις μεταβλητών

- *Οι μεταβλητές πρέπει να **δηλώνονται***
- *Η δήλωση μεταβλητής **υποδεικνύει** στον μεταγλωττιστή το **όνομα** και τον **τύπο** της μεταβλητής*
- *συντάσσεται: **nametype var$_1$,var$_2$,...var$_n$;***

  *(π.χ. **int count,number,start;**)*
- *Η θέση των μεταβλητών στον πηγαίο κώδικα είναι **σημαντική.** Καθορίζει την **εμβέλεια των μεταβλητών** σε ολόκληρο το πρόγραμμα*
- *Τώρα → ακριβώς πριν την αρχή της συνάρτησης **main()***

Δηλώσεις μεταβλητών

η λέξη_κλειδί (keyword) *typedef*

- *χρησιμοποιείται για την απόδοση ονόματος σε υπάρχοντα τύπο δεδομένων*
- *συντάσσεται : **typedef int integer;***

  Δημιουργεί ένα συνώνυμο του τύπου δεδομένων

  *(Π.χ. **typedef int integer;***

  ***integer count;**)*
- *συνηθέστερη χρήση στις δομές*

Απόδοση αρχικών τιμών σε μεταβλητές

Γενικά μια μεταβλητή μπορεί να έχει τιμή

- *μηδέν*
- *μια γνωστή τιμή*

οι τιμές μεταβλητών καθορίζονται με τον τελεστή εκχώρησης (=)

Αρχική τιμή μεταβλητών = Σταθερά

Απόδοση αρχικών τιμών σε μεταβλητές

Σταθερά ≠ Μεταβλητής

*Η τιμή της σταθεράς δεν **αλλάζει** σε όλο το πρόγραμμα*

Κατηγορίες Σταθερών

- *Συμβολικές σταθερές*
- *Κυριολεκτικές σταθερές*

Απόδοση αρχικών τιμών σε μεταβλητές

Κυριολεκτική σταθερά *είναι μία μεταβλητή που της αποδίδεται τιμή άμεσα στον κώδικα κάθε φορά που χρειάζεται*

Παράδειγμα ***Κυριολεκτικής σταθεράς*** →

```
int count=20;
float tax_rate=0.28;
```

Συμβολική σταθερά *είναι μία μεταβλητή που της αποδίδεται καταρχήν ένα όνομα (σύμβολο) και στην συνέχεια η τιμή της.*

Απόδοση αρχικών τιμών σε μεταβλητές

Οι Συμβολικές σταθερές ορίζονται:

- με την εντολή προεπεξεργαστή **#define**

  ή

- με την λέξη κλειδί **const**

συντάσσονται δε ως εξής:

- #define constname value
- const datatype constname=value;

Απόδοση αρχικών τιμών σε μεταβλητές

Η διαφορά της **#define** από **const** υφίσταται στην εμβέλεια των μεταβλητών και στους δείκτες

Παράδειγμα ***Συμβολικής σταθεράς***

```
#define   PI   3.14
circumference = PI*(2*radius);
area= PI*(radius)*(radius);
```

```
const int count= 100;
const int x=50, float rate=0.21;
```


Προτάσεις, Εκφράσεις & Τελεστές

Προτάσεις *(Statements)*

Πρόταση:

είναι μία ολοκληρωμένη οδηγία που κατευθύνει τον υπολογιστή να διεξάγει μια συγκεκριμένη εργασία.

Κάθε πρόταση

- ☞ *τελειώνει με το ελληνικό ερωτηματικό (;)*
- ☞ *εκτείνεται σε μία ή περισσότερες γραμμές*

Προτάσεις, Εκφράσεις & Τελεστές

Είδη Προτάσεων - *(Παραδείγματα)*

- *Πρόταση δήλωσης*
- *Πρόταση ανάθεσης*
- *Σύνθετη πρόταση*
- *Πρόταση ελέγχου ροής*
- *Πρόταση κλήσης συνάρτησης*

Είδη Προτάσεων - *(Παραδείγματα)*

```
#include <stdio.h>
#include <stdlib.h>

int num1,num2,sum;
int add(int x, int y);

int main(int argc, char *argv[])
{
  printf("\n\t\t Give the First number:");
  scanf("%d",&num1);
  printf("\n\t\t Give the Second number:");
  scanf("%d",&num2);
if(num1>num2)
        {
        printf("\n\t\t The First number is greater the Second");
        sum=add(num1,num2);
        printf("\n\t\  The SUM of two number is %d\n\n",sum);
        }
    else
    printf("\n\t\t The Second number is greater or equal the First \n\n");

system("PAUSE");
  return 0;
}

int add(int x, int y)
{
        return (x+y);
}
```

Προτάσεις δήλωσης

Προτάσεις ελέγχου ροής

Προτάσεις ανάθεσης & κλήσης συνάρτησης

Σύνθετη πρόταση

Προτάσεις δομής συνθήκης

Η πρόταση if

απλή πρόταση → if
σύνθετη πρόταση → if - else
σύνθετη πρόταση → if – else if

Η πρόταση switch-case

Πρόταση συνθήκης *if*

απλή πρόταση

```
if (έκφραση)
{
  πρόταση1;
}
επόμενη_πρόταση;
```

σύνθετη πρόταση

```
if (έκφραση)
  {
    πρόταση1;
  }
  else
  {
    πρόταση2;
  }
  επόμενη_πρόταση;
```

σύνθετη πρόταση

```
if (έκφραση1)
      πρόταση1;
   else if (έκφραση2)
      πρόταση2;
 else
      πρόταση3;
 επόμενη_πρόταση;
```

Πρόταση συνθήκης *if*

Οι ***προτάσεις συνθήκης*** χρησιμοποιούν τους ***σχεσιακούς τελεστές*** για να δομήσουν τον έλεγχο του προγράμματος

Σχεσιακοί Τελεστές			
Σύμβολο	**Τελεστής**	**Ερώτηση**	**Παράδειγμα**
==	Ίσον	Τελεσταίος1 ίσον Τελεσταίο2 ?	x==y
>	Μεγαλύτερο	Τελεσταίος1 μεγαλύτερος Τελεσταίου2 ?	x>y
<	Μικρότερο	Τελεσταίος1 μικρότερος Τελεσταίου2 ?	x<y
>=	Μεγαλύτερο ή ίσον	Τελεσταίος1 μεγαλύτερος ή ίσον Τελεσταίου2 ?	x>=y
<=	Μικρότερο ή ίσον	Τελεσταίος1 μικρότερος ή ίσον Τελεσταίου2 ?	x<=y
!=	Άνισο	Τελεσταίος1 δεν ισούται Τελεσταίο2	x!=y

Σχεσιακοί Τελεστές

Προτεραιότητα Σχεσιακών Τελεστών

Γενικά οι προτάσεις & οι εκφράσεις μπορούν να περιέχουν ***περισσότερους από ένα τελεστές.***

Η σειρά εκτέλεσης των πράξεων που υπαγορεύουν οι τελεστές καθορίζεται με συγκεκριμένη ***προτεραιότητα***

Προτεραιότητα σχεσιακών τελεστών	
Τελεστές	**Σχετική προτεραιότητα**
< <= > >=	1
!= ==	2

Προτάσεις επαναληπτικής δομής ελέγχου

Η πρόταση for ***Η πρόταση while***

Η πρόταση do ....while

Είναι δομές προγραμματισμού
της C που εκτελούν ένα μπλοκ μιας ή περισσοτέρων προτάσεων **για ορισμένες φορές**

Πρόταση *for*

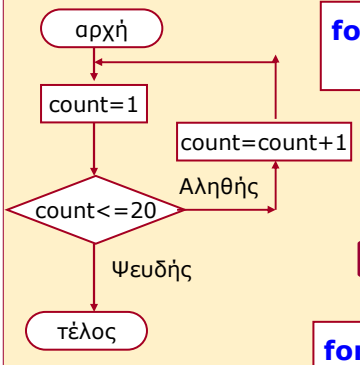
Η εντολή **for** (ή βρόχος **for**) χρησιμοποιείται για να περιγράψει επαναληπτικές διαδικασίες

Συντάσσεται ως εξής:

for (αρχική; συνθήκη; αύξηση)

- Η έκφραση *αρχική* είναι μία πρόταση εκχώρησης
- Η έκφραση *συνθήκη* είναι μία σχεσιακή έκφραση
  - Εάν η *συνθήκη* είναι ψευδής η πρόταση **for** τερματίζεται
  - Εάν η *συνθήκη* είναι αληθής εκτελούνται οι επόμενες προτάσεις
- Υπολογίζεται η έκφραση *αύξηση* και η εκτέλεση επιστρέφει στο 2ο βήμα

Πρόταση *for*




```
for (count=1;count<=20; count++)
    printf("%d\n",count);
```

Αυξάνοντας τον μετρητή με βήμα 1

Μειώνοντας τον μετρητή με βήμα 1

```
for (count=100;count>20;count--)
    printf("%d\n",count);
```


Πρόταση *for*

```
for (count=0;count<=100; count +=5)
    printf("%d\n",count);
```

Αυξάνοντας τον μετρητή με βήμα 5

Μειώνοντας τον μετρητή με βήμα 10

```
for (count=0;count<=100; count -=10)
    printf("%d\n",count);
```

Πρόταση *for*

```
count=10;
for (;count<=100; count +=5)
        printf("%d\n",count);
```

Αυξάνοντας τον μετρητή με βήμα 5 & με αρχική τιμή =10

Αυξάνοντας τον μετρητή με βήμα 10 & με αρχική τιμή το 35

```
for (count=30;count<=100;)
        printf("%d\n",count +=5);
```


Πρόταση *for*

Εκφράσεις ελέγχου πολύπλοκες που τερματίζουν τον βρόχο

Τυπώνει όλα τα στοιχεία ενός πίνακα με το όνομα **array** τερματίζοντας όταν έχουν προβληθεί όλα τα στοιχεία ή όταν ένα στοιχείο της array έχει την τιμή 0

```
for (count=0; count<=100 && array[count]!=0;count++)
        printf("%d\n",array[count]);
```

```
for (count=0; count<=100 && array[count]!=0;)
        printf("%d\n",array[count++]);
```

Πρόταση *for*

Εκφράσεις ελέγχου πολύπλοκες που τερματίζουν τον βρόχο

```
for (count=0; count<=1000; array[count]=500)
        printf("%d\n",array[count++]);
```

Εκχωρείται σε όλα τα στοιχεία του πίνακα με το όνομα **array** η **τιμή 500** και εκτυπώνονται τα στοιχεία του πίνακα

```
for (count=0; count<=1000; array[count++]=500)
    ;
```

Μόνο εκχωρείται σε όλα τα στοιχεία του πίνακα με το όνομα **array** η **τιμή 500**

Πρόταση *for*

Εκφράσεις ελέγχου πολύπλοκες που τερματίζουν τον βρόχο

```
for (count=0, j=100; count<=120; count++, j--)
        b[j]=a[count];
```

Ποιες είναι οι τιμές του **j** ?

Η έκφραση αυτή είναι σωστή ?