ΕΠΛ 033: ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΓΙΑ ΜΗΧΑΝΙΚΟΥΣ

Μάριος Belk, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου

Email: belk@cs.ucy.ac.cy

Βασικοί Τύποι Δεδομένων C

Τύπος	Μέγεθος	Πεδίο Τιμών	Μοναδικές
char	1byte	‘a’ .. ‘z’ ‘A’ .. ‘Z’ ‘0’ .. ‘9’	2^8 ή 256
int	4 bytes	$-2^{31}..2^{31}-1$	2^{32}
float,	4 bytes	$10^{-37}..10^{38}$	2^{32}
double	8 bytes	$10^{-307}..10^{308}$	2^{64}
δείκτης	4 bytes	διευθύνσεις $(0..2^{32}-1)$	2^{32}

Σε αυτό το κεφάλαιο θα μάθετε

- Να ορίζετε και να αρχικοποιείτε μεταβλητές διεύθυνσης (δείκτες)
- Τελεστές για δείκτες
- Να ορίζετε εκφράσεις με δείκτες και αριθμητική με δείκτες

Ορισμός Μεταβλητών Διεύθυνσης (Δείκτες)

- Μια μεταβλητή τύπου δείκτη
  - Είναι μια μεταβλητή που **σαν τιμή** μπορεί να πάρει **μόνο διεύθυνση μνήμης**
  - Χρειάζεται ΠΑΝΤΑ να ξέρει ποιος είναι ο τύπος της τιμής που υπάρχει σε εκείνη τη διεύθυνση

  - Εάν κάθε μεταβλητή είναι ένα σπίτι, τότε μια μεταβλητή τύπου δείκτη είναι ένα σπίτι όπου μέσα απλά υπάρχει μια διεύθυνση προς ένα άλλο σπίτι

    →Για να πάω στο σπίτι στο οποίο δείχνει η διεύθυνση, ΠΡΕΠΕΙ να ξέρω τον τύπο της τιμής (της διεύθυνσης) έτσι ώστε να την ερμηνεύσω: στην προκειμένη περίπτωση οδηγίες (οδός, αριθμός, ΤΚ, κτλ)
  - Κυνήγι θησαυρού (ράλλυ παλαιού αυτοκινήτου/ στίβος τοπογραφίας/):

    →Ξεκινάμε από ένα αρχικό σημείο το οποίο περιέχει οδηγίες/διεύθυνση (πιθανώς συντεταγμένες) κάποιου άλλου σημείου

    →Η διαδικασία επαναλαμβάνεται μέχρι το σημείο τερματισμού

    → Για να μεταβώ από ένα σημείο σε κάποιο άλλο (το επόμενο), πρέπει να ξέρω τον τύπο της τιμής έτσι ώστε να την ερμηνεύσω: στην προκειμένη περίπτωση συντεταγμένες

Ορισμός Μεταβλητών Διεύθυνσης (Δείκτες)

- Μια μεταβλητή τύπου δείκτη ορίζεται με το *
- Ταυτόχρονα δηλώνεται ο τύπος των μεταβλητών στις οποίες επιτρέπεται να δείχνει ο δείκτης αυτός
  - πχ. το παρακάτω ορίζει έναν δείκτη ο οποίος επιτρέπεται να λάβει διεύθυνση μίας μεταβλητής ακέραιων int

```
int *myPtr;
```

- Είναι δυνατό να ορίσετε **δείκτες σε οποιοδήποτε τύπο δεδομένων**.
- Οι δυνατές αρχικές τιμές ενός δείκτη είναι είτε **NULL** είτε **μια διεύθυνση** στην μνήμη

Πως παίρνουμε τη διεύθυνση μιας μεταβλητής;

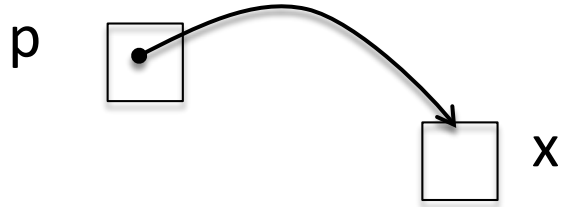
- με τον τελεστή διεύθυνσης &
  - Επιστρέφει την διεύθυνση μιας μεταβλητής, π.χ.

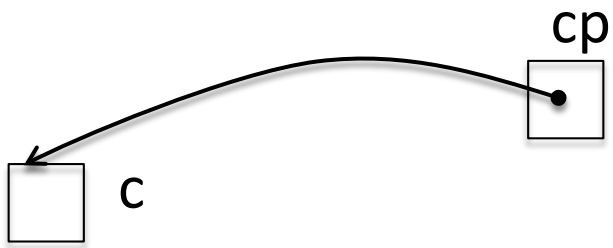
```
int y = 5;
int *yPtr;
yPtr = &y;      /* η yPtr παίρνει σαν τιμή τη διεύθυνση το y */
```

Λέμε ότι η μεταβλητή yPtr “δείχνει στο” y

Πως παίρνουμε τη διεύθυνση μιας μεταβλητής;

```
float y = 5;
float *yPtr;
yPtr = &y;

float x;
float *p;
p = &x;

char c;
char *cp;
cp = &c;
```

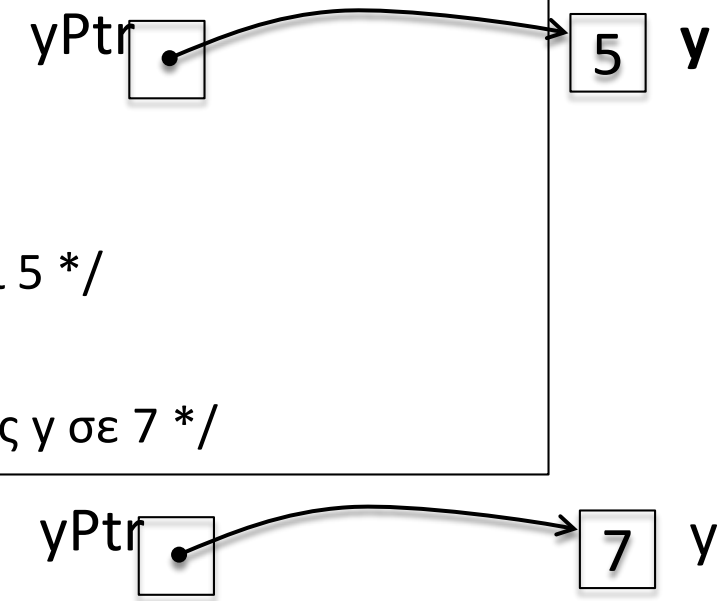




Πως χρησιμοποιούμε τους δείκτες;

- Με τον τελεστή *
  - Ο τελεστής αυτός μας πηγαίνει στη διεύθυνση που του δίνουμε (ταξιτζής), πχ.

```
int y = 5;
int *yPtr;
yPtr = &y;

printf("%d", *yPtr); /* τυπώνει 5 */

*yPtr = 7; /* αλλάζει την τιμή της y σε 7 */
```



Άρα, όποτε γράφω *yPtr θα είναι σαν να γράφω y!

Προτεραιότητα Τελεστών

Operators	Associativity	Type
`()` `[]`	left to right	highest
`+` `-` `++` `--` `!` `*` `&` (*type*)	right to left	unary
`*` `/`	left to right	multiplicative
`+` `-`	left to right	additive
`<` `<=` `>` `>=`	left to right	relational
`==` `!=`	left to right	Equality
`&&`	left to right	logical and
`||`	left to right	logical OR
`?:`	right to left	conditional
`=` `+=` `-=` `*=` `/=` `%=`	right to left	assignment
`,`	left to right	comma

Χαρακτήρας *

- Τελεστής γινομένου
  - a*b;
- Δήλωση Δείκτη
  - int *p;
- Τελεστής Έμμεσης Αναφοράς *
  - x = *p + 1;

Παράδειγμα 1

- int *p, y, *z;

- y = 6;
- p = &y;
- z = p; // Ο δείκτης z δείχνει στη διεύθυνση που δείχνει και ο δείκτης p
- *p = *p + 1;
- y = *z + 1;

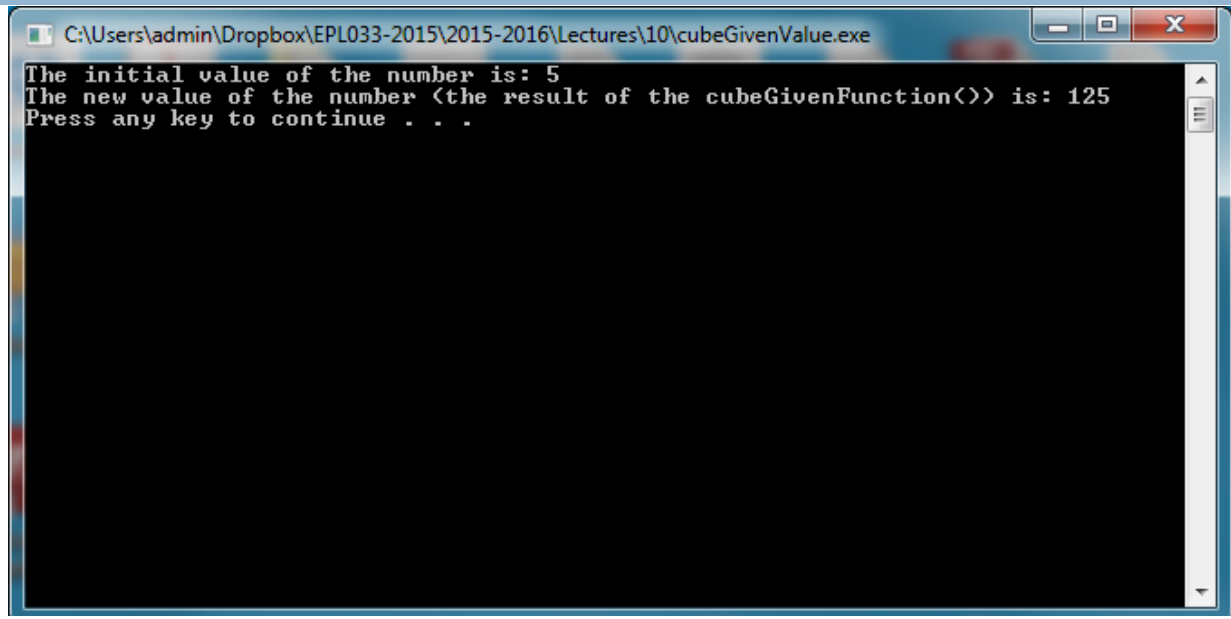
Η τιμή όλων των μεταβλητών είναι 8 αφού p και z δείχνουν στην διεύθυνση του y

Παράμετροι

- **Επιτρέπουν την επικοινωνία μεταξύ συναρτήσεων**
- **Πέρασμα δια τιμής (*τιμή*)**
  - Διοχέτευση πληροφοριών στην συνάρτηση
  - Μεταβλητή στην συνάρτηση κλήσης δεν επηρεάζεται - αντιγράφεται η τιμή της
- **Πέρασμα/επιστροφή δια αναφοράς (διεύθυνση)**
  - Διοχέτευση πληροφοριών από την συνάρτηση (και είσοδο)
  - Μεταβλητή στην συνάρτηση κλήσης μπορεί να της ανατεθούν τιμές στην καλούμενη συνάρτηση (π.χ. scanf με τελεστή διεύθυνσης)
  - Επιστροφή πολλών τιμών μέσω διευθύνσεων

Παράδειγμα χρήσης: πέρασμα διά τιμής


```c
#include <stdio.h>


int cubeGivenValue(int n){
   n = n * n * n;
   return n;
}



int main() {

  // Dilwsi metavlitis kai arxikopoiisi
  int number = 5;

  // Typwse arxiki timi tis number
  printf("%d", number);

  // Kalese tin synartisi me orisma TIN TIMI tis number
  // kai anathese tin timi epistrofis stin number
  number = cubeGivenValue(number);

  // Typwse tin timi tis number
  printf("%d", number);

  return 0;
}
```



The initial value of the number is: 5
The new value of the number (the result of the cubeGivenFunction()) is: 125

Παράδειγμα χρήσης δεικτών

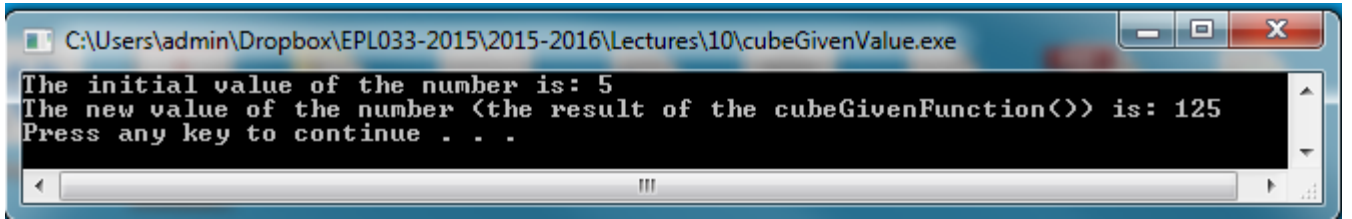

```
#include <stdio.h>

void cubeGivenAddress(int *p){
   *p = *p * *p * *p;
}

int main() {
   // Dilwsi metavlitis kai arxikopoiisi
   int number = 5;
   // Typwse arxiki timi tis number
   printf("%d", number);

   // Kalese tin synartisi me orisma TIN DIEYTHINSI tis number
   cubeGivenAddress( &number );

   // Typwse tin timi tis number
   printf("%d", number);

   return 0;
}
```

Εδώ περνάμε σαν παράμετρο μια ΔΙΕΥΘΥΝΣΗ προς ακέραια μεταβλητή

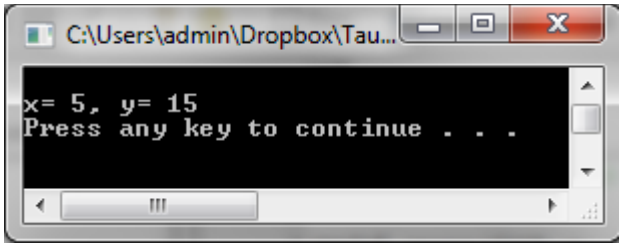
Κλήση της διαδικασίας cubeGivenAddress.

Η τιμή της παραμέτρου είναι η διεύθυνση της μεταβλητής number στην μνήμη

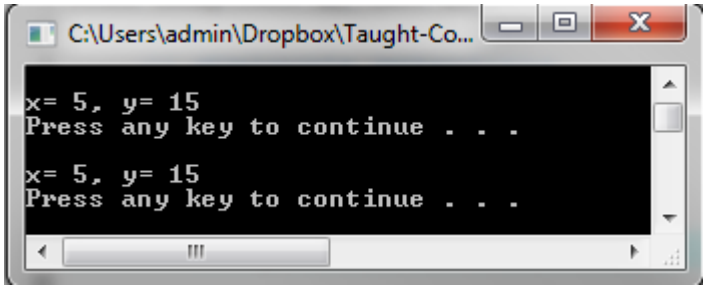
Η τιμή στη θέση της μνήμης όπου δείχνει ο δείκτης p αλλάζει.



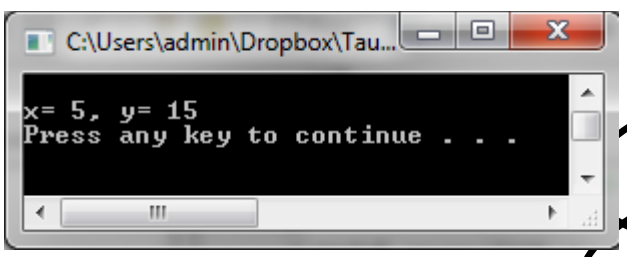
```
#include <stdio.h>
void swap(int a, int b){

    int temp;
    temp = a;
    a = b;
    b = temp;

    return;
}

int main() {

    int x = 5, y = 15;

    printf("x=%d, y=%d", x, y);

    swap( x, y );

    printf("x=%d, y=%d", x, y);

    return 0;
}
```

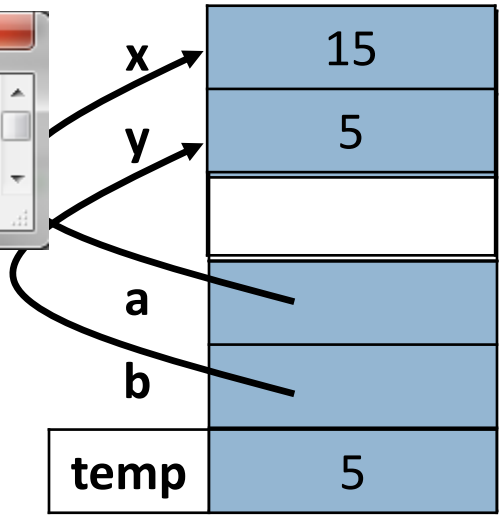


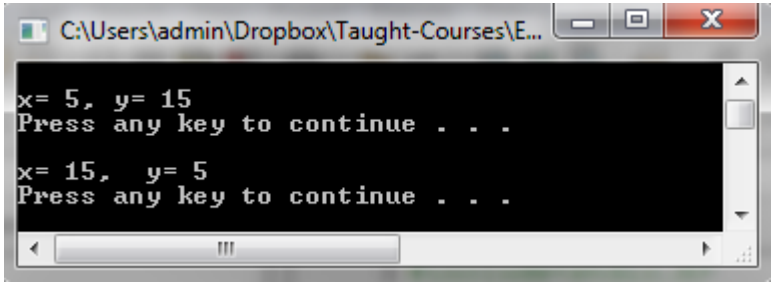
x	5
y	15
a	15
b	5
temp	5



```c
#include <stdio.h>
void swap(int *a, int *b){

   int temp;
   temp = *a;
   *a = *b;
   *b = temp;

   return;
}

int main() {

   int x = 5, y = 15;

   printf("x=%d, y=%d", x, y);

   swap( &x, &y );

   printf("x=%d, y=%d", x, y);

  return 0;
}
```



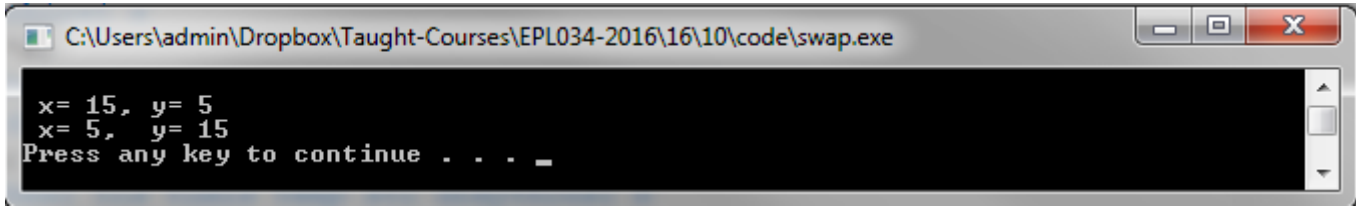




Αλλαγή τιμών μεταβλητών εντός συνάρτησης


```c
#include <stdio.h>

/* synartisi poy antallassei tis times poy vriskontai
 * stis dieythinseis poy dexetai ws orismata */
void swap(int *a, int *b){

   // dimiouria metavlitis
   int temp;
   // arxikopoiisi metavlitis me tin timi poy vrisketai
   // sti dieythinsi a
   temp = *a;
   // anathesi tis timis poy vrisketai sti dieythinsi b
   // sti dieythinsi a
   *a = *b;
   // anathesi tis timis temp sti dieythinsi b
   *b = temp;

   return;
}
```

```c
int main() {
   // Dilwsi metavlitwn kai arxikopoiisi
   int x = 15, y = 5;

   // Typwse arxikes times twn metavlitwn
   printf("x=%d, y=%d", x, y);

   // Kalese tin synartisi me orismata TIS
DIEYTHINSEIS twn metavlitwn
   swap( &x, &y );

   // Typwse tin timi tis number
   printf("x=%d, y=%d", x, y);

   return 0;
}
```



Παράδειγμα 2

- Γράψτε τη συνάρτηση sort2 που θα ταξινομεί δύο αριθμούς a και b ώστε μετά την εκτέλεση a<=b
- Χρειάζονται δείκτες

Λύση 1

```
void
sort2(int *a, int *b) {
    if (*a>*b)
        swap(&a,&b);
}
```

```
int main(){
    int x, y;
    x = 14;
    y = 5;
    sort2(&x,&y);
    return 0;
}
```

ΣΩΣΤΗ;

Λύση 2

```
void
sort2(int *a, int *b)
{
    if (*a>*b)
        swap(a,b);
}
```

ΣΩΣΤΗ;