Η ΓΛΩΣΣΑ C

Μάθημα 8:

Δείκτες

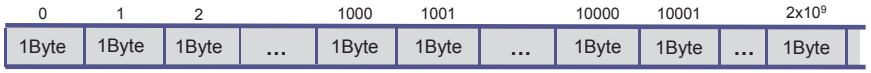
Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

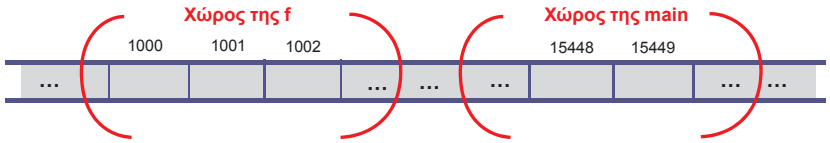
Α. Δείκτες
1. Η Μνήμη του Υπολογιστή

- Έχουμε ήδη δει κάποιες πληροφορίες για την οργάνωση της μνήμης σε ένα πρόγραμμα.
- Σε μια απλουστευμένη (αλλά επαρκή) προσέγγιση μπορούμε να φανταστούμε την μνήμη σαν μια ταινία με χώρους αποθήκευσης δεδομένων.
  - Κάθε χώρος αποθήκευσης είναι 1 byte (δηλαδή 8 bits).
  - Προκειμένου να μπορούμε να εντοπίσουμε κάθε χώρο αποθήκευσης, έχει έναν αναγνωριστικό αύξων αριθμό (που λέγεται διεύθυνση μνήμης).
  - Έτσι η εικόνα που πρέπει να έχουμε για την μνήμη θα είναι κελιά, το κάθε ένα με έναν αναγνωριστικό αριθμό, το οποίο θα έχει μέγεθος 1 byte:



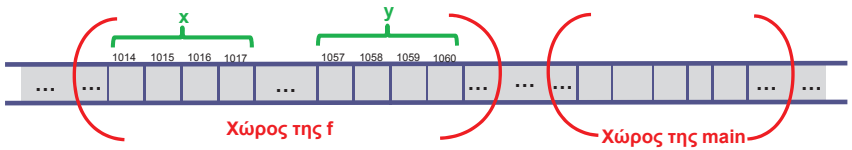
Α. Δείκτες
1. Η Μνήμη του Υπολογιστή

- Είδαμε ότι μια συνάρτηση έχει τον δικό της χώρο αποθήκευσης μεταβλητών.
  - Όταν πρόκειται να εκτελεστεί μια συνάρτηση, δεσμεύεται ο χώρος της μνήμης για τις μεταβλητές της συνάρτησης.
  - Έτσι αν για παράδειγμα η main καλεί μια συνάρτηση που έχουμε γράψει (έστω με όνομα f), τότε ο μεταγλωττιστής θα μεριμνήσει ώστε κάθε συνάρτηση να έχει τον χώρο της στην μνήμη.
  - Ο χώρος που δεσμεύεται είναι αρκετά μεγάλος για να χωρέσει πάρα πολλές μεταβλητές (το ακριβές πλήθος καθορίζεται από το λειτουργικό σύστημα, το μεταγλωττιστή κ.λπ.)



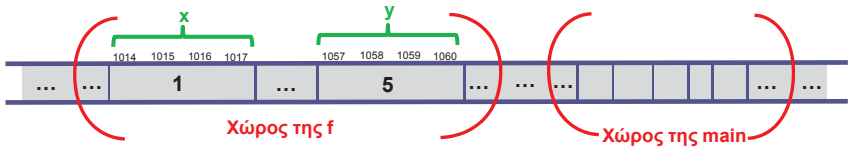
Α. Δείκτες
1. Η Μνήμη του Υπολογιστή

- Όταν δηλώνουμε μεταβλητές σε μια συνάρτηση, ο μεταγλωττιστής επιλέγει κάποιον από τον ελεύθερο χώρο της συνάρτησης για να αποθηκεύσει την μεταβλητή.
- Έτσι αν π.χ. η f δηλώνει στον χώρο δήλωσης μεταβλητών της δύο ακέραιες μεταβλητές, έστω την x και την y (θεωρούμε ότι μια ακέραια μεταβλητή δεσμεύει 4 bytes όπως συζητήσαμε σε προηγούμενο μάθημα)
- Τότε η εικόνα της μνήμης θα είναι η εξής



Α. Δείκτες
1. Η Μνήμη του Υπολογιστή

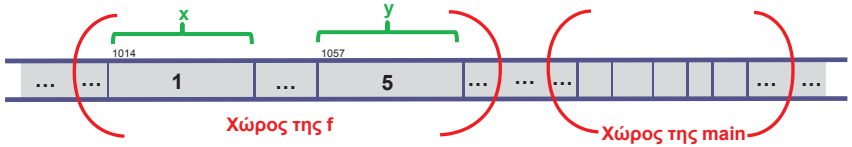
- Έτσι αν γράψουμε x=1 και y=5 με εντολές καταχώρησης στο σώμα της f, τότε αποθηκεύονται στις μεταβλητές οι αντίστοιχες τιμές.
- Ο τρόπος αποθήκευσης στο δυαδικό σύστημα των αριθμών αυτών δεν μας ενδιαφέρει όταν γράφουμε το πρόγραμμά μας.



- Συνοψίζοντας αντιλαμβανόμαστε ότι μία μεταβλητή μπορεί να εντοπιστεί πλήρως από δύο δεδομένα: τον τύπο δεδομένων της (για να ξέρουμε πόσα bytes δεσμεύει στην μνήμη) και το πρώτο byte στο οποίο είναι αποθηκευμένη. Π.χ. Για την x αν ξέρω ότι είναι ακέραια και το 1ο byte της είναι το 1014, τότε μπορώ να εξάγω ότι θα είναι αποθηκευμένη στα bytes 1014-1017.

Α. Δείκτες
2. Η Έννοια του Δείκτη

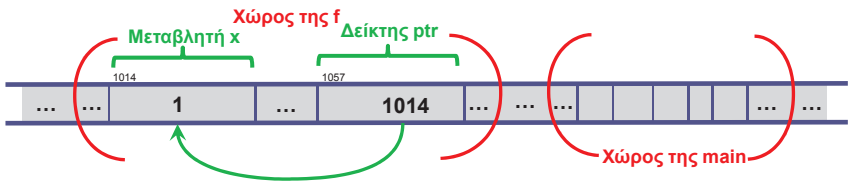
- Έτσι λοιπόν, η εικόνα που θα πρέπει να έχουμε στο μυαλό μας, είναι ότι κάθε μεταβλητή έχει έναν χώρο αποθήκευσης (ίσου με το μέγεθος της μεταβλητής σε bytes) με αναγνωριστικό τη διεύθυνση μνήμης του 1ου byte που αυτή δεσμεύει.



- Από την στιγμή όμως που η διεύθυνση μνήμης είναι ένας αριθμός, μπορεί να αντιμετωπιστεί σαν οποιοσδήποτε άλλος αριθμός στην C. Άρα για παράδειγμα μπορούμε να δηλώσουμε μια μεταβλητή στην οποία να αποθηκεύουμε την διεύθυνση μνήμης μιας άλλης μεταβλητής!

Α. Δείκτες
2. Η Έννοια του Δείκτη

- Συνεπώς μπορούμε να δηλώσουμε μια μεταβλητή (έστω με όνομα ptr) που θα αποθηκεύσουμε π.χ. την διεύθυνση της μεταβλητής x.
- Αυτή η μεταβλητή, η οποία αποθηκεύει ως τιμή την διεύθυνση μνήμης άλλων μεταβλητών ονομάζεται δείκτης:



- Δείτε λοιπόν ότι ο δείκτης ptr έχει ως τιμή την διεύθυνση της μεταβλητής x. Θα λέμε για το λόγο αυτό, ότι ο δείκτης ptr δείχνει στην μεταβλητή x

Α. Δείκτες
3. Ορισμός Δείκτη

- Ένας δείκτης δηλώνεται με την εντολή δήλωσης:

```
Τύπος_Δεδομένων *Ονομα_Δείκτη
```

  - Δηλώνουμε έναν δείκτη με όνομα «Ονομα_Δείκτη» που θα δείχνει σε μια μεταβλητή τύπου δεδομένων «Τύπος_Δεδομένων»
- Παραδείγματα:

```
int *p;
```

  - Εδώ δηλώνουμε έναν δείκτη σε ακέραιο με όνομα p.

```
double *ptr;
```

  - Εδώ δηλώνουμε έναν δείκτη σε double με όνομα ptr

- Διαπιστώστε ότι ένας δείκτης είναι ακόμη ένας τύπος δεδομένων. Με την ιδιαιτερότητα ότι οι μεταβλητές που θα αποθηκεύει θα είναι διευθύνσεις μνήμης.
- Ο τύπος δεδομένων συμβολικά είναι (int *) και (double *) αντίστοιχα στις δηλώσεις που είδαμε.

Α. Δείκτες
4. Απόδοση Τιμής σε Δείκτη

- Για να έχει νόημα ένας δείκτης πρέπει να τον συσχετίσουμε με μια μεταβλητή, ή όπως λέμε να τον βάλουμε να δείχνει στην μεταβλητή. Αυτό γίνεται με τον εξής τρόπο:

```
int x;
int *p;

p=&x; // Βάλε τον δείκτη p να δείχνει στην x.
```

  - Ο τελεστής & (θα τον διαβάζουμε «διεύθυνση του») μπαίνει μπροστά από μια μεταβλητή και μας επιστρέφει την διεύθυνση του.
  - Άρα:
    - αν η x είναι αποθηκευμένη π.χ. στην διεύθυνση 10333,
    - τότε η τιμή &x (διεύθυνση του x) είναι 10333,
    - άρα η τιμή της p μετά την εντολή ανάθεσης είναι 10333.

Α. Δείκτες
5. Απόδοση Τιμής μέσω Δείκτη (Ο τελεστής *)

- Ο τελεστής * (που ονομάζεται τελεστής έμμεσης διευθυνσιοδότησης) έχει δύο χρήσεις:
  - Είδαμε την πρώτη που είναι να δηλώσουμε μια μεταβλητή τύπου δείκτη.
  - Η δεύτερη χρήση είναι η εξής:
    - Αν σε μία εντολή, βάλουμε το * μπροστά από έναν δείκτη, τότε αναφερόμαστε στην μεταβλητή που δείχνει ο δείκτης.
- Βλέπουμε το παράδειγμα:

```
int x;
int *p;

p=&x;

*p=5; //Η μεταβλητή που δείχνει το *p (άρα το x) παίρνει την τιμή 5
```

- Μετά από αυτό το παράδειγμα αντιλαμβανόμαστε ότι έχουμε δύο τρόπους για να έχουμε πρόσβαση στην μεταβλητή x:
  - Απευθείας με το όνομα της x (άμεση πρόσβαση)
  - Μέσω του δείκτη που δείχνει στην x (έμμεση πρόσβαση)

Α. Δείκτες
6. Παράδειγμα Χρήσης Δεικτών

- Μεταγλωττίστε, εκτελέστε και **μελετήστε** το πρόγραμμα:

```
/* pointers.c: Deixnei tin vasiki xrisi twn deiktwn */

#include <stdio.h>

main()
{
   int x;
   int *ptr;

   ptr=&x;

   x=5;
   printf("x=%d\t *ptr=%d\t &x=%d\t ptr=%d",x,*ptr,&x,ptr);
   *ptr=8;
   printf("\nx=%d\t *ptr=%d\t &x=%d\t ptr=%d",x,*ptr,&x,ptr);
   x=9;
   printf("\nx=%d\t *ptr=%d\t &x=%d\t ptr=%d",x,*ptr,&x,ptr);
}
```

Β. Δείκτες και Πίνακες
1. Το Όνομα ενός Πίνακα είναι Δείκτης

- Οι δείκτες είναι πολύ χρήσιμοι σε συνδυασμό με τους πίνακες!
- Αυτό συμβαίνει διότι εξ' ορισμού το όνομα κάθε πίνακα είναι ένας δείκτης με τύπο δεδομένων τον τύπο δεδομένων του πίνακα.
- Ωστόσο είναι ένας δείκτης που δεν μπορούμε να αλλάξουμε την τιμή του.
  - Μπορούμε όμως να ορίσουμε έναν δείκτη να δείχνει στην αρχή του πίνακα με τον εξής τρόπο:

```
int pinakas[100];
int *ptr;

ptr=pinakas;
```

  - Ο δείκτης θα δείχνει στον πρώτο ακέραιο αριθμό του πίνακα.
  - Ο δείκτης ptr μπορεί να αλλάξει!
  - Σημειώστε εδώ ότι θα μπορούσαμε να βάλουμε τον δείκτη να δείχνει στο πρώτο στοιχείο του πίνακα και με την εντολή

```
int pinakas[100];
int *ptr;

ptr=&pinakas[0];
```

  - Ωστόσο είναι σημαντικό να γνωρίζουμε τον πρώτο τρόπο. ptr=&pinakas[0]

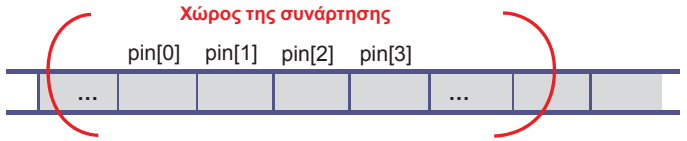
Β. Δείκτες και Πίνακες
2. Αποθήκευση ενός Πίνακα στη Μνήμη

- Στο μάθημα 8, είδαμε ότι τα στοιχεία ενός πίνακα αποθηκεύονται σε διαδοχικές θέσεις μνήμης.
- Έφτασε η ώρα να το διαπιστώσουμε και πειραματικά.

- Είδαμε ότι με την δήλωση:

```
int pin[4];
```

  - Δημιουργείται ένας πίνακας 4 διαδοχικών θέσεων στην μνήμη



  - Γνωρίζουμε πλέον ότι &pin[0],&pin[1],&pin[2],&pin[3] είναι οι διευθύνσεις μνήμης των 4 μεταβλητών του πίνακα.
  - Το πρόγραμμα της επόμενης διαφάνειας δείχνει ότι όντως οι μεταβλητές αυτές αποθηκεύονται σε διαδοχικές θέσεις μνήμης.

Β. Δείκτες και Πίνακες
2. Αποθήκευση ενός Πίνακα στη Μνήμη

- Μεταγλωττίστε, εκτελέστε και μελετήστε το πρόγραμμα:

```
/* matrix_pointers.c: Deixnei oti oi theseis mnimis enos pinaka einai
diadoxikes */

#include <stdio.h>

#define N 4

main()
{
   int pin[N];
   int i;

   for (i=0; i<N; i++)
      printf("\nStoixeio:%d, Diefthinsi Thesis Mnimis: %d",i, &pin[i]);
}
```


Β. Δείκτες και Πίνακες
3. Αριθμητική Δεικτών

- Μεγάλη προσοχή. Οι δείκτες έχουν μια ιδιαίτερη αριθμητική έτσι ώστε να δουλεύουν αποδοτικά σε συνδυασμό με τους πίνακες:
- Έστω το πρόγραμμα

```
int pinakas[100];
int *ptr;

ptr=pinakas;
```

- Η εντολή:

```
ptr++;
```

  - Δεν αυξάνει την τιμή του δείκτη κατά 1!
  - Αλλά αυξάνει την τιμή του δείκτη κατά τόσα bytes όσα είναι και ο τύπος δεδομένων του δείκτη (για ακέραιο: κατά 4!).

- Ενώ η εντολή:

```
ptr--;
```

  - Μειώνει την τιμή του δείκτη κατά τόσα bytes όσα είναι και ο τύπος δεδομένων του δείκτη (για ακέραιο: κατά 4!).

Β. Δείκτες και Πίνακες
3. Αριθμητική Δεικτών

- Το παρακάτω πρόγραμμα δείχνει πως μπορούμε να διατρέξουμε τα στοιχεία ενός πίνακα μέσω

```
/* pointer_calc.c: Pws exoume prosvasi sta stoixeia enos pinaka mesw deikti */

#include <stdio.h>

#define N 10

main()
{
   int pin[N];
   int i;
   int *ptr;

   ptr=pin;
   for (i=0; i<N; i++)
   {
      printf("\nStoixeio:%d, Diefthinsi Thesis Mnimis: %d",i, ptr);
      ptr++;
   }
}
```


Β. Δείκτες και Πίνακες
3. Αριθμητική Δεικτών

- Για την ώρα μπορεί να μην φαίνεται χρήσιμο, γιατί να κάνουμε πρόσβαση στα στοιχεία του πίνακα με αυτόν τον τρόπο!
- Αργότερα όμως θα δούμε πραγματικά χρήσιμες εφαρμογές αυτής της ιδιότητας.
- Ας δούμε όμως ένα ακόμη παράδειγμα:

```
double pinakas[100];
double *ptr;

ptr=pinakas;
ptr+=2;
```

  - Η εντολή `ptr+=2` λοιπόν θα αυξήσει την τιμή της `ptr` κατα 2 x (το μέγεθος του double), άρα θα αυξήσει τη διεύθυνση κατά 16, αφού μία double μεταβλητή δεσμεύει 8 bytes.
  - Μετά το πέρας της εκτέλεσης αυτών των εντολών ο δείκτης `ptr` θα δείχνει στο στοιχείο `pinakas[2]`.

Β. Δείκτες και Πίνακες
4. Ισοδύναμος Συμβολισμός για την Πρόσβαση σε Πίνακα

- Με βάση την αριθμητική δεικτών, η παράσταση:

```
*(array+2)
```

  - Μπορεί να εξηγηθεί ως εξής:
    - Το (array+2) δίνει την διεύθυνση του 3ου στοιχείου του πίνακα.
    - Άρα με το * μπροστά, μας δίνει την τιμή του 3ου στοιχείου του πίνακα.
- Έτσι για παράδειγμα ο βρόχος:

```
for (i=0; i<N; i++)
   printf("%d",array[i]);
```

  - μπορεί να γραφεί ισοδύναμα:

```
for (i=0; i<N; i++)
   printf("%d",*(array+i));
```


Γ. Δείκτες και Συναρτήσεις
1. Διοχέτευση Δείκτη σε Συνάρτηση

- Από τη στιγμή που ένας δείκτης μπορεί να ειδωθεί σαν μια μεταβλητή (στην οποία αποθηκεύονται διευθύνσεις μεταβλητών), έχουμε κάθε δικαίωμα να τον περάσουμε ως όρισμα σε μια συνάρτηση.
  - Π.χ. Το ακόλουθο πρωτότυπο συνάρτησης:

```
int func(int *p, int x)
```

  - Δέχεται ως ορίσματα δύο πράγματα:
    - Έναν δείκτη σε ακέραιο (το p) και έναν ακέραιο (το x)
- Μπορεί βεβαίως και από μία συνάρτηση να επιστραφεί ένας δείκτης σε μια μεταβλητή. Το πρωτότυπο μιας τέτοιας συνάρτησης θα είναι:

```
int *func2(float y)
```

  - Όπου στο σώμα της συνάρτησης θα ορίζουμε ότι επιστρέφεται ένας δείκτης σε ακέραια μεταβλητή.
- Μας ενδιαφέρει αυτό;
  - ΠΑΡΑ ΠΟΛΥ! Το πως περνάμε μια μεταβλητή (μέσω μεταβλητής ή μέσω δείκτη) είναι από τις πιο σημαντικές προγραμματιστικές τεχνικές της C!

Γ. Δείκτες και Συναρτήσεις

2. Διοχέτευση Ορίσματος σε Συνάρτηση Μέσω Τιμής

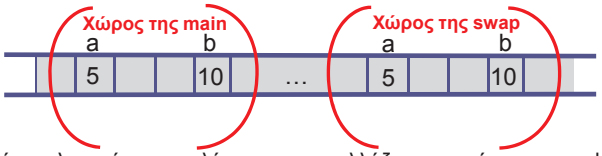
- Στο μάθημα 6 «Συναρτήσεις», είδαμε ότι κάθε συνάρτηση έχει τον δικό της χώρο μεταβλητών στην μνήμη.
- Ας μελετήσουμε για να θυμηθούμε τις ιδιότητες αυτές ένα απλό πρόγραμμα.

```
/*byvalue.c: Perasma orismatwn mesw timis */
#include <stdio.h>

void swap(int a,int b);

main()
{
  int a=5,b=10;
  printf("\nMain: a=%d,b=%d",a,b);
  swap(a,b);
  printf("\nMain: a=%d,b=%d",a,b);
}
void swap(int a, int b)
{
  int k;
  k=a;
  a=b;
  b=k;
  printf("\nSwap: a=%d,b=%d",a,b);
}
```

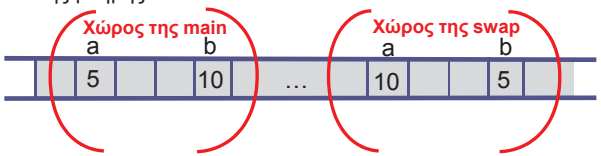

Γ. Δείκτες και Συναρτήσεις

2. Διοχέτευση Ορίσματος σε Συνάρτηση Μέσω Τιμής

- Ας θυμηθούμε πως δουλεύει το πρόγραμμα αυτό.
  - Πριν την κλήση της συνάρτησης swap, οι μεταβλητές a και b έχουν πάρει τις τιμές a=5 και b=10.
  - Συνεπώς όταν γίνεται η κλήση της swap, τα ορίσματα a, και b της συνάρτησης swap έχουν αντίστοιχα τιμές 5 και 10.
  - Η εικόνα της μνήμης είναι:



  - Αφού εκτελεστούν οι εντολές της swap, αλλάζουν οι τιμές των a και b στην swap, αλλά επειδή αυτές είναι τοπικές μεταβλητές στην swap η τιμή των a,b στην main δεν αλλάζει!
  - Η εικόνα της μνήμης είναι:



Γ. Δείκτες και Συναρτήσεις

2. Διοχέτευση Ορίσματος σε Συνάρτηση Μέσω Τιμής

- Πρακτικά, όταν περνάμε ορίσματα μέσω μεταβλητών, οι αλλαγές που γίνονται στις μεταβλητές που διοχετεύουμε ως ορίσματα, δεν θα εξακολουθούν να υφίστανται όταν ο έλεγχος του προγράμματος επιστρέψει στη καλούσα συνάρτηση.
  - Αυτός ο τρόπος για να περάσουμε ένα όρισμα σε μια συνάρτηση λέγεται διοχέτευση ορίσματος σε συνάρτηση μέσω τιμής (**by value**) υπό την έννοια ότι στην συνάρτηση έχουμε στα χέρια μας την τιμή της μεταβλητής και όχι την ίδια την μεταβλητή.
- Είναι προφανές ότι σε κάποιες συναρτήσεις θα επιθυμούμε να αλλάζει η τιμή της μεταβλητής μέσα στην συνάρτηση και αυτό να μπορεί να το «δει» και η καλούσα συνάρτηση.
  - Αυτός είναι ο δεύτερος τρόπος για να περάσουμε ένα όρισμα σε μια συνάρτηση, λέγεται διοχέτευση ορίσματος σε συνάρτηση μέσω αναφοράς (**by reference**) και όπως θα δούμε στην συνέχεια δεν διοχετεύουμε ως όρισμα την μεταβλητή, αλλά δείκτη στην μεταβλητή!

Γ. Δείκτες και Συναρτήσεις

3. Διοχέτευση Ορίσματος σε Συνάρτηση Μέσω Αναφοράς

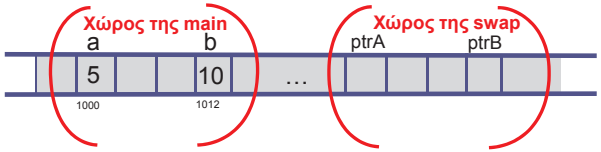
- Άρα τροποποιούμε την συνάρτηση μας ως εξής:

```
/*byreference.c: Perasma orismatwn mesw anaforas */
#include <stdio.h>

void swap(int *ptrA, int *ptrB);

main()
{
  int a=5,b=10;
  printf("\nMain: a=%d,b=%d",a,b);
  swap(&a,&b);
  printf("\nMain: a=%d,b=%d",a,b);
}
void swap(int *ptrA, int *ptrB)
{
  int k;
  k=*ptrA;
  *ptrA=*ptrB;
  *ptrB=k;
  printf("\nSwap: a=%d,b=%d",*ptrA,*ptrB);
}
```

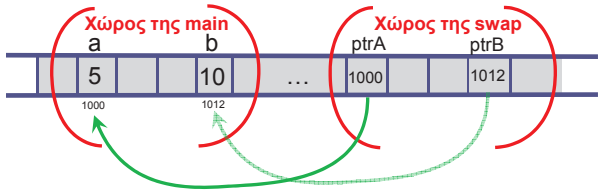
Γ. Δείκτες και Συναρτήσεις

3. Διοχέτευση Ορίσματος σε Συνάρτηση Μέσω Αναφοράς

- Να δούμε πως δουλεύει το πρόγραμμα αυτό.
  - Πριν την κλήση της συνάρτησης swap, οι μεταβλητές a και b έχουν πάρει τις τιμές a=5 και b=10.
  - Ας υποθέσουμε ότι η διεύθυνση της a είναι 1000 και της b 1012.



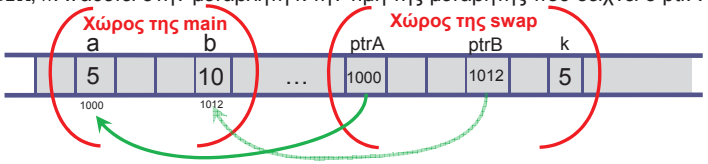
  - Η τιμή &a, υπολογίζεται στο 1000, ενώ η τιμή &b υπολογίζεται στο 1012.
    - Συνεπώς όταν αρχίζει η εκτέλεση της swap η εικόνα της μνήμης είναι:



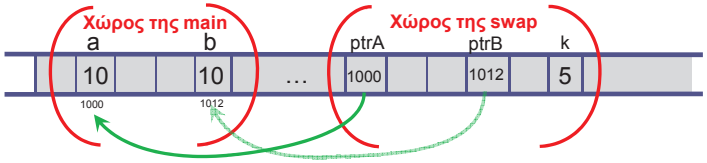
Γ. Δείκτες και Συναρτήσεις

3. Διοχέτευση Ορίσματος σε Συνάρτηση Μέσω Αναφοράς

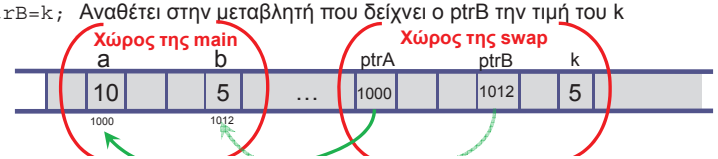
- Ας δούμε τώρα αναλυτικά τις εντολές που τρέχουν στην swap:
  - `k=*ptrA;` //Αναθέτει στην μεταβλητή k την τιμή της μεταβήτης που δείχνει ο ptrA



  - `*ptrA=*ptrB;` Αναθέτει στην μεταβλητή που δείχνει ο ptrA την τιμή που δείχνει ο ptrB



  - `*ptrB=k;` Αναθέτει στην μεταβλητή που δείχνει ο ptrB την τιμή του k



Γ. Δείκτες και Συναρτήσεις

3. Διοχέτευση Ορίσματος σε Συνάρτηση Μέσω Αναφοράς

- Τι παρατηρούμε:
  - Ότι με έμμεσο τρόπο η συνάρτηση προσπελαύνει τις ίδιες τις μεταβλητές της main!
  - Άρα για τον λόγο αυτό, οι αλλαγές που κάνουμε στις μεταβλητές παραμένουν και στην main!

- Συνοψίζοντας:
  - Αν θέλουμε να αλλάξει η τιμή μια μεταβλητής σε μια συνάρτηση και το αποτέλεσμα αυτό να διατηρηθεί και στην καλούσα συνάρτηση, διοχετεύουμε το όρισμα μέσω αναφοράς (δηλαδή μέσω δείκτη)
  - Αλλιώς διοχετεύουμε το όρισμα απλά μέσω τιμής.
- Επιπλέον:
  - Δείτε ότι με τον τρόπο αυτό μπορούμε να έχουμε περισσότερα από ένα επιστρεφόμενα αποτελέσματα.
  - Το ένα επιστρεφόμενο αποτέλεσμα θα το πάρουμε από την return και το άλλο μέσω δείκτη!
  - Ή ακόμη πιο απλά μπορούμε να πάρουμε και τις δύο επιστρεφόμενες τιμές μέσω δείκτη.

Δ. Παρατηρήσεις

1. Διοχέτευση Πίνακα ως Όρισμα σε Συνάρτηση

- Σημείωση:
  - Επειδή είναι πολύ συνηθισμένο να διοχετεύουμε έναν πίνακα ως όρισμα, και για να είναι πιο εμφανές στην ανάγνωση ότι πρόκειται περί πίνακα και όχι δείκτη σε μεταβλητή, η C μας δίνει την δυνατότητα:
    - Αντί να γράψουμε το πρωτότυπο:

```
void print(int *pinakas, int n);
```

    - Να γράψουμε το πρωτότυπο:

```
void print(int pinakas[], int n);
```

  - Οποιοδήποτε από τα δύο πρωτότυπα και να χρησιμοποιήσουμε θα έχουμε το ίδιο αποτέλεσμα.

- Με βάση τα παραπάνω εξ' ορισμού το πέρασμα ενός πίνακα σε μία συνάρτηση γίνεται μέσω αναφοράς.
- Άρα θα θυμόμαστε ότι οποιαδήποτε αλλαγή γίνεται σε έναν πίνακα που διοχετεύουμε ως όρισμα, παραμένει και στην καλούσα συνάρτηση.
- Αν θέλουμε να περάσουμε αντίγραφο του πίνακα ως όρισμα σε μία συνάρτηση, θα πρέπει να κατασκευάσουμε πρώτα ένα αντίγραφό του.

Δ. Παρατηρήσεις
2. Η Σταθερά NULL

- Ένα πολύ συνηθισμένο λάθος είναι να καταχωρούμε τιμή σε έναν δείκτη που δεν τον έχουμε συσχετίσει με μια μεταβλητή. Π.χ. ο κώδικας:

```
int x;
int *p;

*p=15;
```

  - Θα δημιουργήσει σφάλμα εκτέλεσης! Το p δεν δείχνει πουθενά, άρα όταν πάμε να εκτελέσουμε την εντολή δεν επηρεάζουμε κάποια μεταβλητή, άρα το πρόγραμμα έχει μια απροσδόκητη συμπεριφορά!
- Συνεπώς πάντα όταν δηλώνουμε ένα δείκτη, θα πρέπει να τον συσχετίζουμε με μια μεταβλητή η οποία θα είναι αποθηκευμένη στην μνήμη.

- Θα υπάρξουν και κάποιες περιπτώσεις που θα θέλουμε ο δείκτης να μην έχει τιμή. Για τον λόγο αυτό στην C είναι ορισμένη μία συμβολική σταθερά, το **NULL**, την οποία θα την χρησιμοποιούμε συμβολικά για να λέμε ότι «ο δείκτης δεν δείχνει πουθενά». Παράδειγμα:

```
int *p;

p=NULL; /* Ο p δεν δείχνει πουθενά */
```


Ε. Ασκήσεις
1. Υπολογισμός Ριζών Β'Βάθμιας Εξίσωσης

1. Γράψτε μια συνάρτηση με όνομα:

```
int rizes(float a, float b, float c, float *x1, float *x2);
```

   - Η οποία να υπολογίζει τις ρίζες της εξίσωσης $ax^2+bx+c=0$. Η συνάρτηση να επιστρέφει (μέσω ορίσματος) στις μεταβλητές x1 και x2 τις πραγματικές ρίζες της εξίσωσης και να επιστρέφει με return το πλήθος των λύσεων της εξίσωσης
     - Βοήθεια: Θα χρειαστείτε την συνάρτηση:

```
double sqrt(double x);
```

       - Η οποία είναι ορισμένη στο αρχείο κεφαλίδας math.h
     - Βοήθεια: Ο προσδιοριστής των printf, scanf για τύπο δεδομένων float είναι **%f** (αντί για %d στους ακεραίους)
2. Έπειτα γράψτε μια συνάρτηση main, που θα διαβάζει από την είσοδο τις τιμές των a,b,c (συντελεστές της εξίσωσης) και αφού εκτελέσει την συνάρτηση rizes θα εκτυπώνει τα κατάλληλα μηνύματα με τις ρίζες της β' βάθμιας εξίσωσης.

Ε. Ασκήσεις
2. Συνάρτηση init_array

1. Γράψτε την συνάρτηση: `void init_array(int *pinakas, int n, int a, int b)` που να αρχικοποιεί έναν πίνακα n ακεραίων, με τυχαίους αριθμούς στο διάστημα [a..b]
2. Ελέγξτε την ορθότητα της συνάρτησής σας με κατάλληλα παραδείγματα

Ε. Ασκήσεις
3. Συνάρτηση print_array

Επεκτείνετε το πρόγραμμα της προηγούμενης άσκησης:

1. Γράψτε την συνάρτηση: `void print_array(int *pinakas, int n)` που να τυπώνει τα περιεχόμενα του πίνακα ακεραίων pinakas.
2. Ελέγξτε την ορθότητα της συνάρτησής σας με κατάλληλα παραδείγματα

Ε. Ασκήσεις

4. Συνάρτηση max_array

Επεκτείνετε το πρόγραμμα της προηγούμενης άσκησης:

1. Γράψτε την συνάρτηση: `int max_array(int *pinakas, int n)` που να επιστρέφει τον μέγιστο στοιχείο του πίνακα ακεραίων pinakas
2. Ελέγξτε την ορθότητα της συνάρτησής σας με κατάλληλα παραδείγματα

Ε. Ασκήσεις

5. Ένα Πολύπλοκο Πρόγραμμα

Συνεχίζοντας το προηγούμενο παράδειγμα, γράψτε μία συνάρτηση main η οποία:

1. Να δηλώνει έναν πίνακα ακεραίων 1000 θέσεων
2. Μέσα από ένα επαναλαμβανόμενο βρόχο να δίνει 6 επιλογές στον χρήστη:
    1. Να εισάγει το μέγεθος του πίνακα
    2. Να αρχικοποιεί με τυχαίους αριθμούς από το a έως το b τα στοιχεία του πίνακα, όπου a και b είναι αριθμοί που ζητούνται από το χρήστη.
    3. Να βρίσκει τον μέγιστο του πίνακα και να τον τυπώνει
    4. Να τυπώνει όλα τα στοιχεία του πίνακα
    5. Να κάνει έξοδο από τον επαναλαμβανόμενο βρόχο