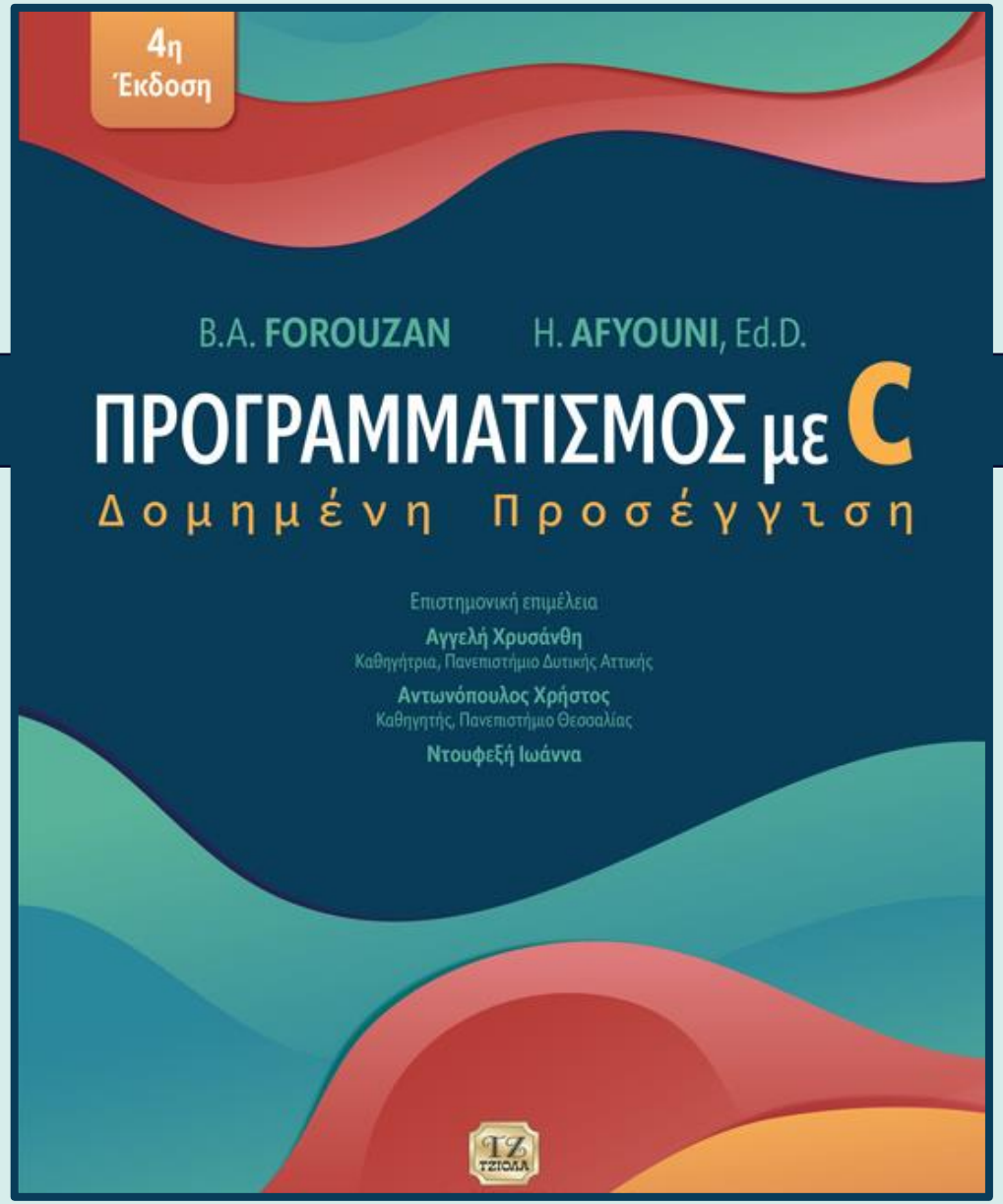


Επιστημονικές Εκδόσεις
ΤΖΙΟΛΑ

ΚΕΦΑΛΑΙΟ 9

Δείκτες

Μαθησιακοί στόχοι (1/2)

Ολοκληρώνοντας αυτό το κεφάλαιο, θα είστε σε θέση:

- 9.1 Να περιγράφετε τον τρόπο δήλωσης, ορισμού και αρχικοποίησης δεικτών στη C
- 9.2 Να γράφετε προγράμματα τα οποία θα χρησιμοποιούν δείκτες ως παραμέτρους και επιστρεφόμενες τιμές
- 9.3 Να ορίζετε έναν δείκτη κατά τρόπο ώστε να δείχνει σε άλλες μεταβλητές τύπου δείκτη
- 9.4 Να περιγράφετε τους τύπους και τη σημασία της συμβατότητας μεταξύ δεικτών
- 9.5 Να διακρίνετε τις εκφράσεις lvalue έναντι των εκφράσεων rvalue στη C

Μαθησιακοί στόχοι (2/2)

Ολοκληρώνοντας αυτό το κεφάλαιο, θα είστε σε θέση:

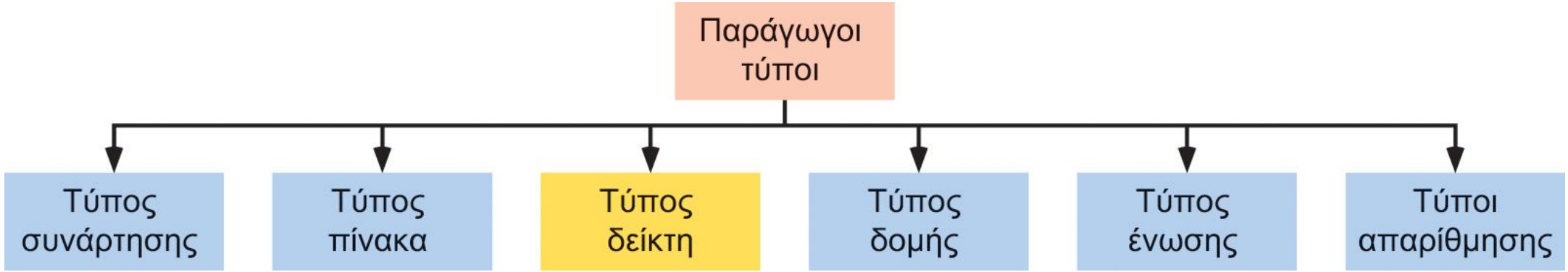
- 9.6 Να εξηγείτε τη σχέση μεταξύ πινάκων και δεικτών
- 9.7 Να περιγράφετε την σχεδίαση και τις έννοιες στις οποίες βασίζεται η «αριθμητική δεικτών»
- 9.8 Να εξηγείτε τον τρόπο με τον οποίο υλοποιείται «πέρασμα» πινάκων σε συναρτήσεις
- 9.9 Να γράφετε προγράμματα C τα οποία θα χρησιμοποιούν στατική και δυναμική δέσμευση μνήμης
- 9.10 Να υλοποιείτε «ακανόνιστους» πίνακες σε προγράμματα C
- 9.11 Να γράφετε προγράμματα με προηγμένες δυνατότητες, χρησιμοποιώντας δείκτες

9.1 Δείκτες: Βασικές έννοιες

9.1 Δείκτες: Βασικές έννοιες (1/2)

- Ένας δείκτης (pointer) είναι μία σταθερά ή μία μεταβλητή η οποία περιέχει μία διεύθυνση που μπορεί να χρησιμοποιηθεί για προσπέλαση των δεδομένων που είναι αποθηκευμένα σε αυτή τη θέση μνήμης
  - Είναι μία ιδιαίτερα αποτελεσματική μέθοδος προσπέλασης δεδομένων
  - Υποστηρίζουν αποτελεσματικές τεχνικές για τον χειρισμό δεδομένων σε πίνακες
  - Χρησιμοποιούνται σε συναρτήσεις ως παράμετροι που περνιούνται δι' αναφοράς
  - Αποτελούν τη βάση για τη δυναμική δέσμευση μνήμης

9.1 Δείκτες: Βασικές έννοιες (2/2)

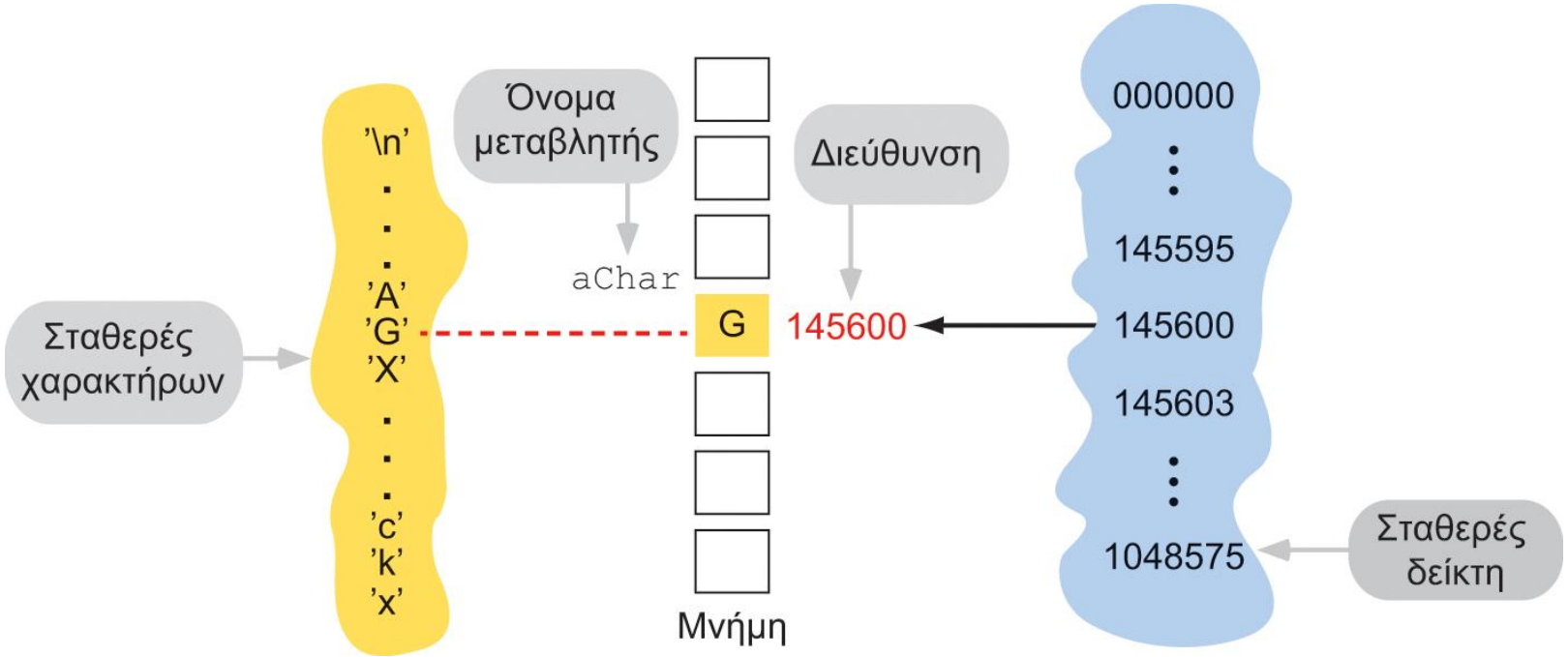


ΣΧΗΜΑ 9-1 Παράγωγοι τύποι δεδομένων.

Σταθερές τύπου δείκτη

- Μία σταθερά τύπου δείκτη ή απλώς σταθερά δείκτη (pointer constant) είναι ένας δείκτης του οποίου τα περιεχόμενα δεν μπορούν να αλλάξουν

ΣΧΗΜΑ 9-3 Σταθερές τύπου δείκτη.

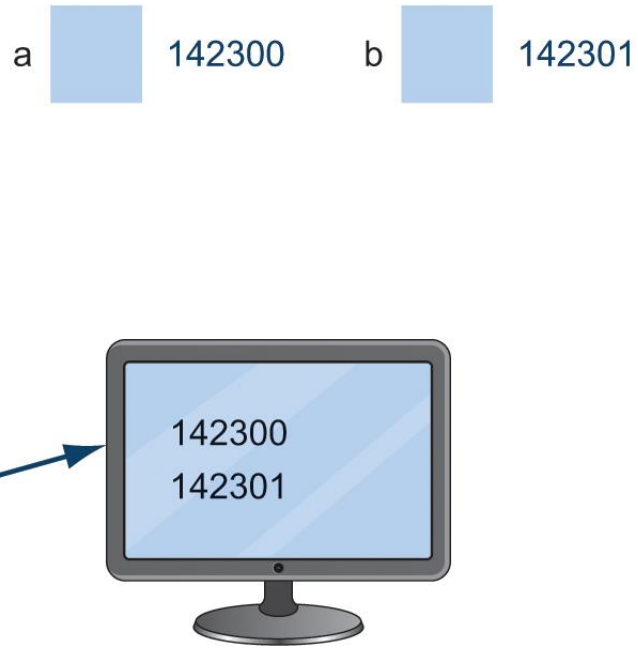


Τιμές δεικτών

- Μία σταθερά δείκτη θα πρέπει να καθορίζεται και να αποθηκεύεται με τη χρήση του τελεστή διεύθυνσης (&)

ΣΧΗΜΑ 9-4 Εκτύπωση διευθύνσεων χαρακτήρων.

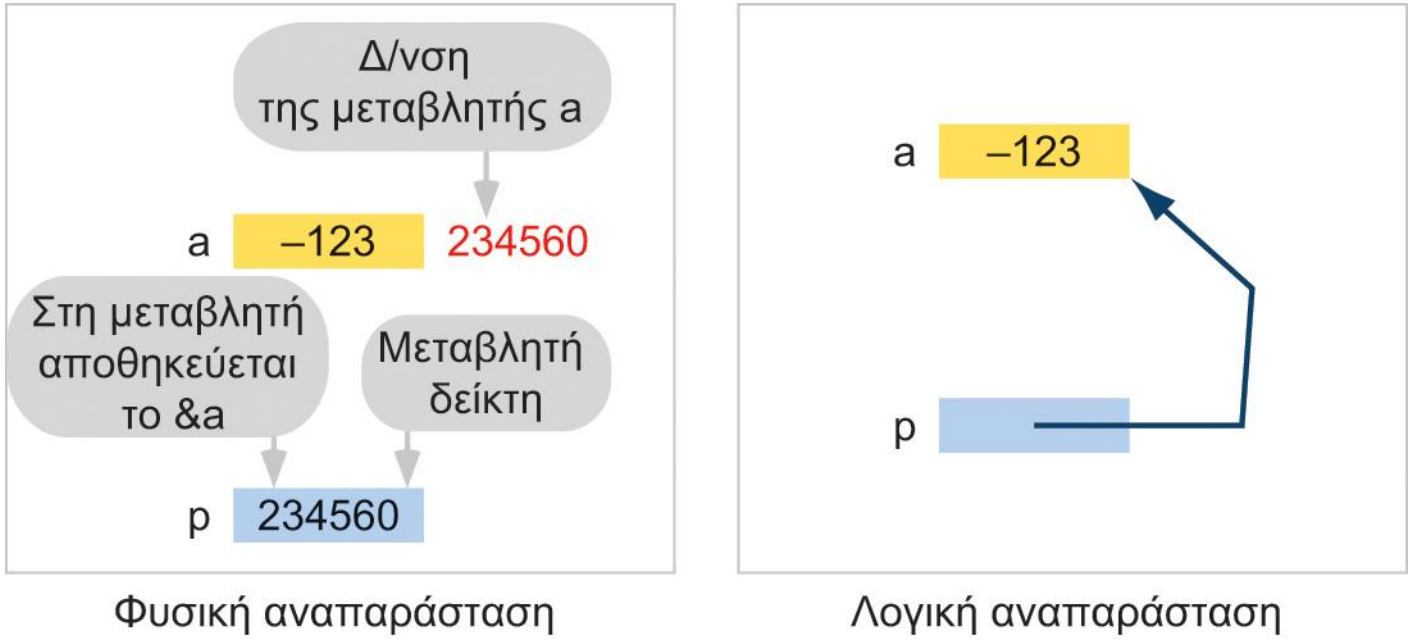
```
//Εκτύπωση διευθύνσεων μνήμης
#include <stdio.h>

int main (void)
{
// Τοπικές δηλώσεις
   char a;
   char b;
// Εντολή
   printf("%p\n %p\n", &a, &b);
   return 0;
} // τέλος main
```



Μεταβλητές τύπου δείκτη

- Εάν έχουμε μία σταθερά δείκτη και μία τιμή δείκτη, τότε μπορούμε να έχουμε μία μεταβλητή δείκτη
  - Επομένως, μπορούμε να αποθηκεύσουμε τη διεύθυνση μιας μεταβλητής σε μία άλλη μεταβλητή, η οποία ονομάζεται μεταβλητή τύπου δείκτη (pointer variable) ή απλά μεταβλητή δείκτη

ΣΧΗΜΑ 9-6 Σχηματική αναπαράσταση μεταβλητής τύπου δείκτη.

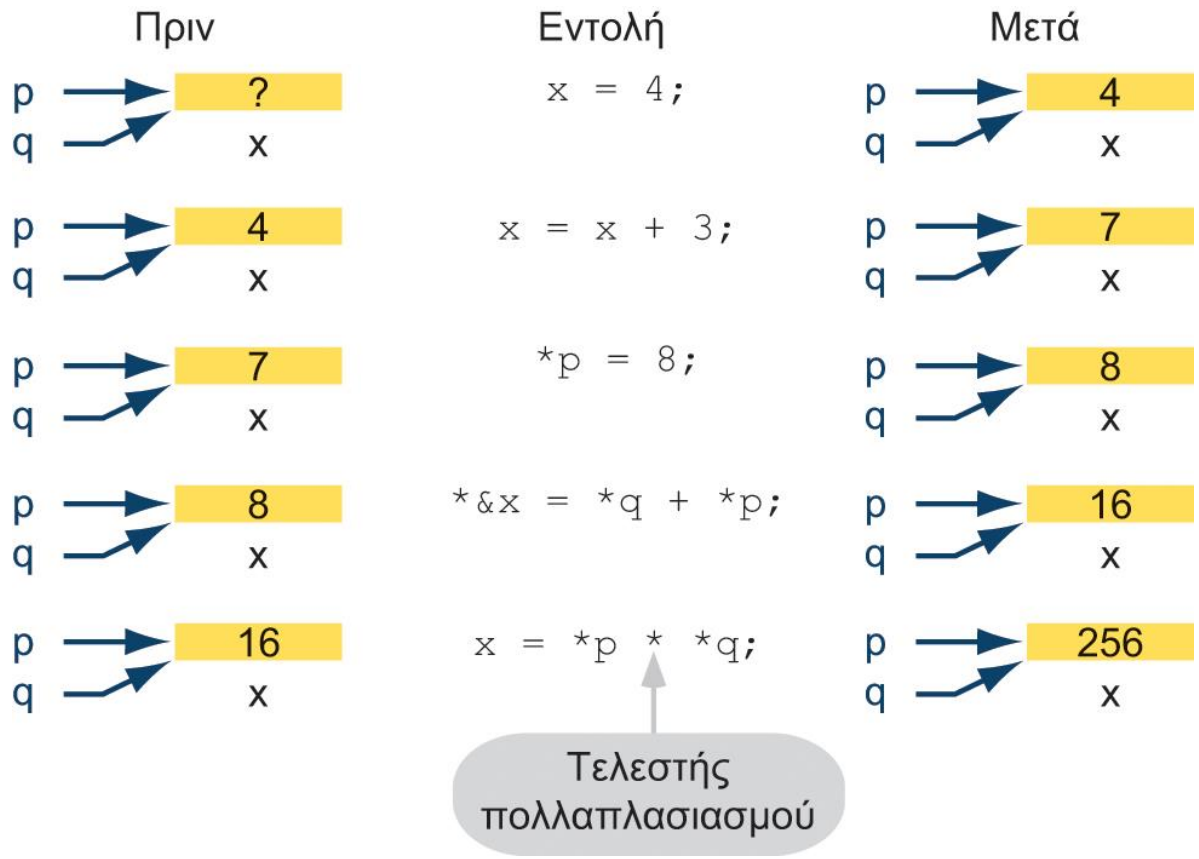


Φυσική αναπαράσταση

Λογική αναπαράσταση

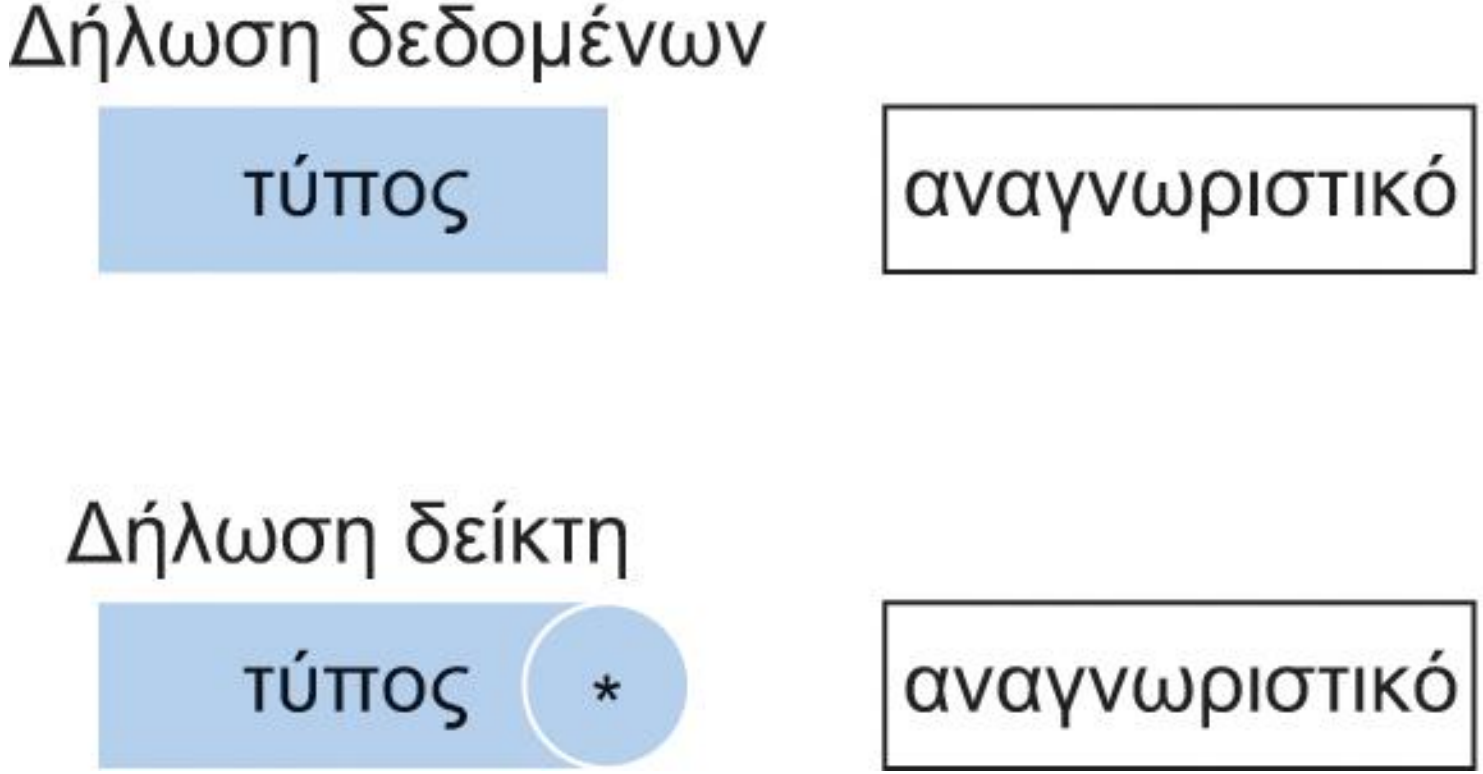
Προσπέλαση μεταβλητών μέσω δεικτών

- Ο τελεστής έμμεσης αναφοράς (*) χρησιμοποιείται για την κωδικοποίηση έμμεσων εκφράσεων
  - Ένας από τους τύπους έκφρασης της κατηγορίας μοναδιαίων εκφράσεων



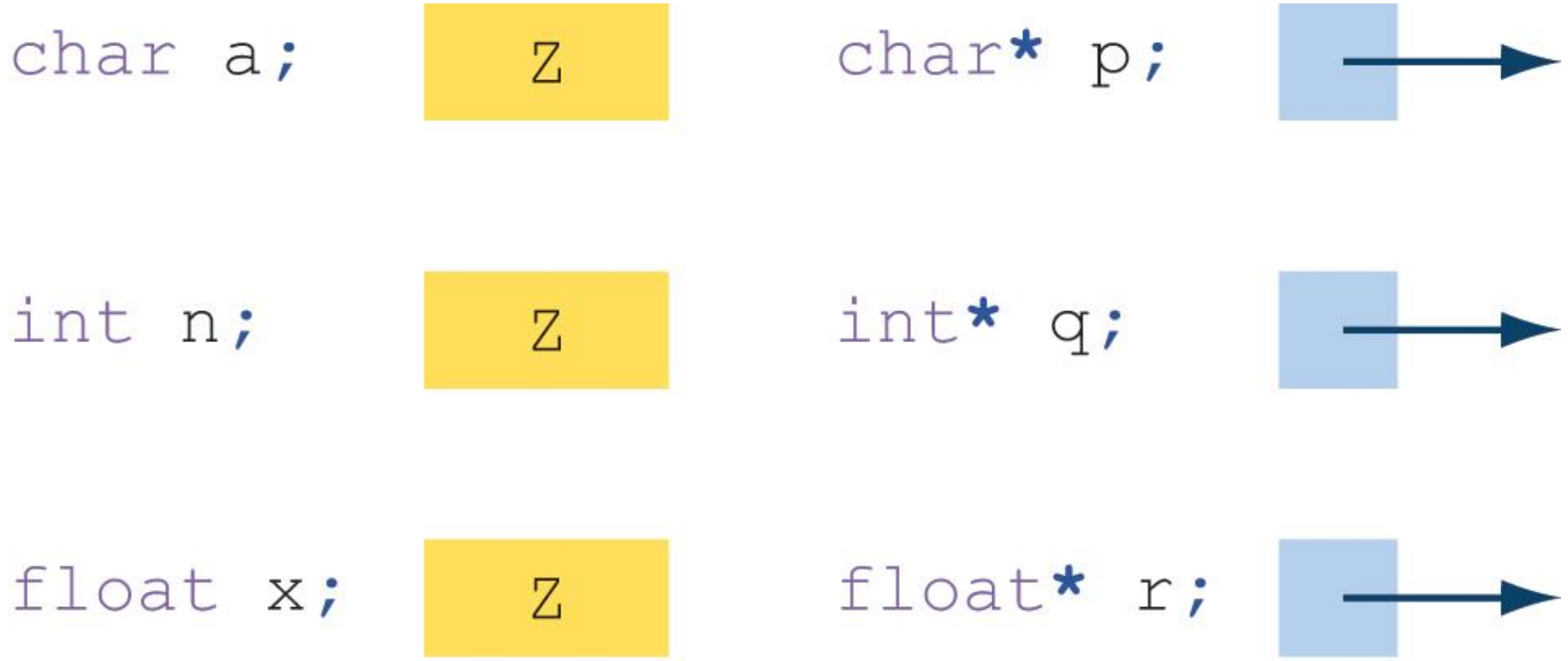
ΣΧΗΜΑ 9-8 Προσπέλαση μεταβλητών μέσω δεικτών.

Δήλωση και ορισμός δεικτών (1/2)



ΣΧΗΜΑ 9-10 Δήλωση μεταβλητής τύπου δείκτη.

Δήλωση και ορισμός δεικτών (2/2)



ΣΧΗΜΑ 9-11 Τρόποι με τους οποίους μπορούν να δηλωθούν μεταβλητές τύπου δείκτη.

Δήλωση έναντι έμμεσης αναφοράς

- Έχουμε χρησιμοποιήσει τον τελεστή * (αστερίσκο) σε δύο διαφορετικά πλαίσια: για δήλωση και για ανακατεύθυνση
  - Όταν ο τελεστής * χρησιμοποιείται για δήλωση ενός δείκτη, συσχετίζεται με έναν τύπο δεδομένων

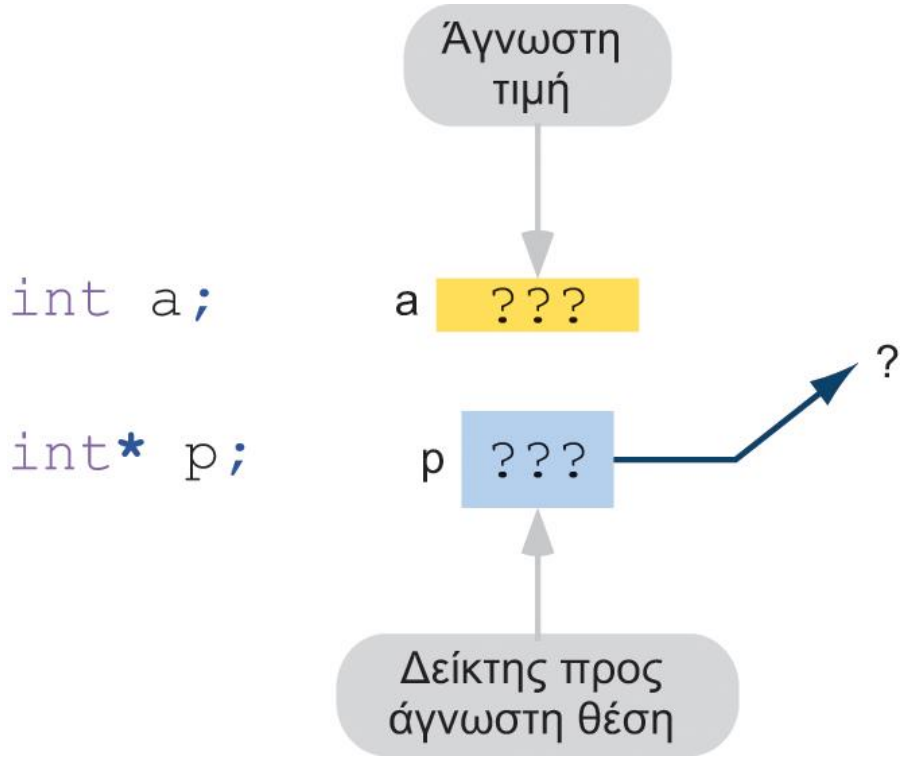
```
int * pa ;

int * pb ;
```

  - Όταν χρησιμοποιείται ως τελεστής ανακατεύθυνσης ανακατευθύνει τη ζητούμενη λειτουργία από τη μεταβλητή δείκτη σε μία μεταβλητή δεδομένων

```
sum = * pa + * pb ;
```

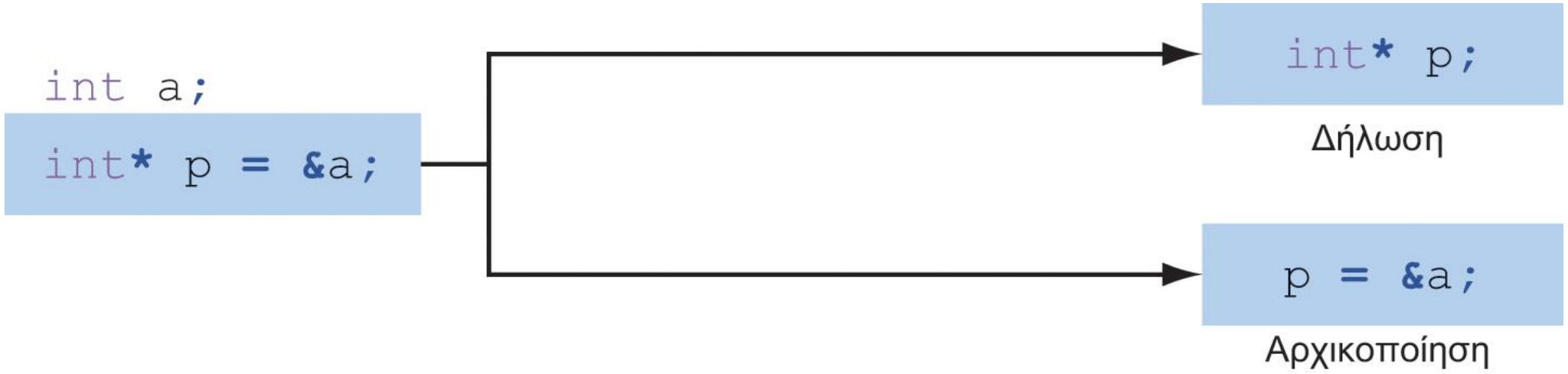
Αρχικοποίηση μεταβλητών τύπου δείκτη (1/2)

- Όταν εκκινεί το πρόγραμμα, οι μη αρχικοποιημένοι δείκτες θα έχουν κάποια άγνωστη διεύθυνση μνήμης μέσα τους
  - Μία άγνωστη τιμή που θα ερμηνεύεται ως θέση μνήμης
- Μία από τις πιο συνηθισμένες αιτίες σφαλμάτων στον προγραμματισμό, είναι οι μη αρχικοποιημένοι δείκτες
  - Αυτά τα σφάλματα μπορεί να είναι πολύ δύσκολο να εντοπιστούν, επειδή η επίδραση του σφάλματος συχνά δεν εμφανίζεται παρά μόνο πολύ αργότερα κατά την εκτέλεση του προγράμματος



ΣΧΗΜΑ 9-12
Μη αρχικοποιημένοι δείκτες.

Αρχικοποίηση μεταβλητών τύπου δείκτη (2/2)



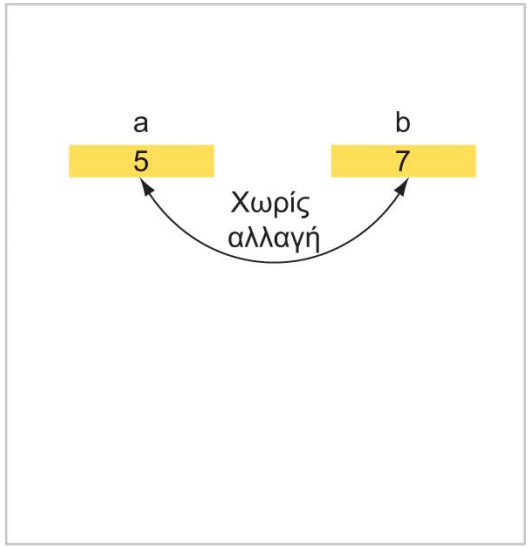
ΣΧΗΜΑ 9-13 Αρχικοποίηση μεταβλητών τύπου δείκτη.

9.2 Χρήση δεικτών για επικοινωνία μεταξύ συναρτήσεων

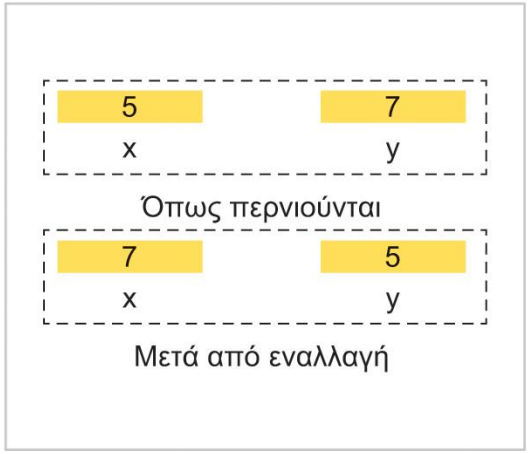
Πέρασμα διευθύνσεων (1/2)

- Όταν χρησιμοποιούμε την επικοινωνία προς τα κάτω, τα δεδομένα εναλλάσσονται στην καλούμενη συνάρτηση, αλλά τίποτα δεν αλλάζει στο καλούν πρόγραμμα

```
// Δηλώσεις συναρτήσεων
void exchange (int x, int y);

int main (void)
{
   int a = 5;
   int b = 7;
   exchange (a, b);
   printf("%d %d\n", a, b);
   return 0;
} // τέλος main
```




```
void exchange (int x, int y)
{
   int temp;

   temp = x;
   x    = y;
   y    = temp;
   return;
} // τέλος exchange
```



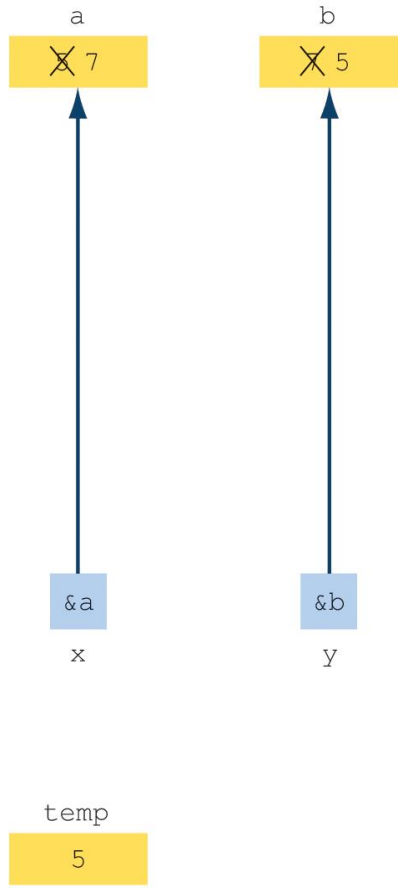
ΣΧΗΜΑ 9-17 Μη λειτουργική έκδοση της συνάρτησης `exchange`.

Πέρασμα διευθύνσεων (2/2)

ΣΧΗΜΑ 9-18 Η συνάρτηση `exchange` με χρήση δεικτών.

```
// Δηλώσεις συναρτήσεων
void exchange (int*, int*);

int main (void)
{
   int a = 5;
   int b = 7;

   exchange (a, b);
   printf("%d %d\n", a, b);
   return 0;
} // τέλος main
```

```
void exchange (int* px, int* py)
{
   int temp;

     temp = *px;
   *px = *py;
   *py = temp;

   return;
} // τέλος exchange
```



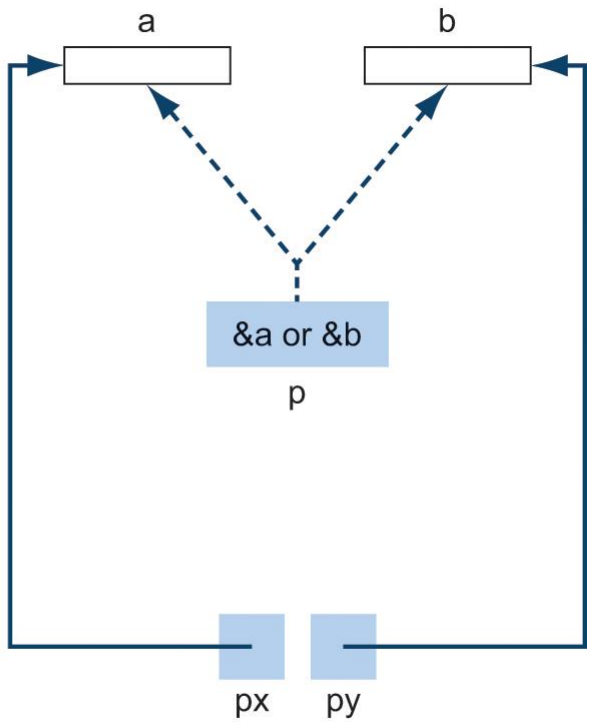
Συναρτήσεις που επιστρέφουν δείκτες (1/2)

- Τίποτα δεν εμποδίζει μία συνάρτηση να επιστρέφει έναν δείκτη στην καλούσα συνάρτηση
  - Είναι αρκετά συνηθισμένο οι συναρτήσεις να επιστρέφουν δείκτες
  - Όταν επιστρέφουμε έναν δείκτη, αυτός πρέπει να δείχνει σε δεδομένα στη συνάρτηση κλήσης ή σε μία συνάρτηση υψηλότερου επιπέδου

Συναρτήσεις που επιστρέφουν δείκτες (2/2)

```
// Δηλώσεις πρωτοτύπων
int* smaller(int* p1, int* p2);

int main (void)
   int a;
   int b;
   int* p;
   ...
   scanf ( "&d &d ", &a, &b );
   p = smaller (&a, &b);
...
```

```
int* smaller(int* px, int* py)
{
   return (*px <*py ? px : py);
} // τέλος smaller
```



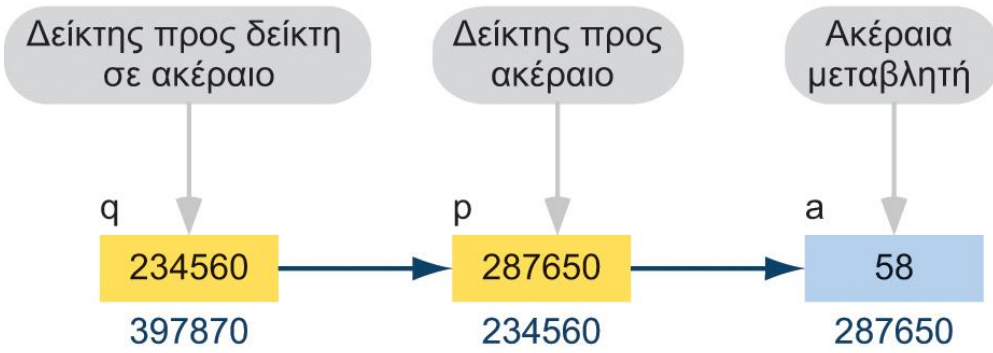
ΣΧΗΜΑ 9-19 Συναρτήσεις που επιστρέφουν δείκτες.

9.3 Δείκτες προς άλλους δείκτες

9.3 Δείκτες προς άλλους δείκτες

- Δεν υπάρχει όριο ως προς το πόσα επίπεδα ανακατεύθυνσης μπορούμε να χρησιμοποιήσουμε, αλλά, πρακτικά, σπάνια χρησιμοποιούμε περισσότερα από δύο
- Κάθε επίπεδο έμμεσης αναφοράς μέσω δείκτη (pointer indirection) απαιτεί έναν ξεχωριστό τελεστή έμμεσης αναφοράς όταν γίνεται η ανάκλησή του

```
// Τοπικές δηλώσεις
int   a;
int*  p;
int** q;
```




```
// Εντολές
a = 58;
P = &a;
q = &p;

printf(" %3d",   a);
printf(" %3d",  *p);
printf(" %3d", **q);
```

ΣΧΗΜΑ 9-20 Δείκτες προς άλλους δείκτες.

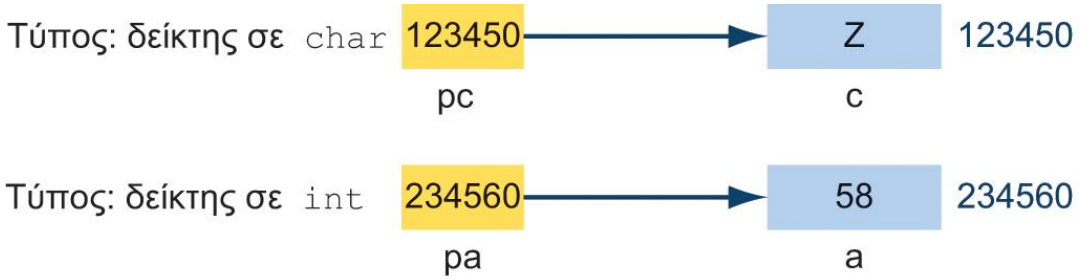
9.4 Ζητήματα συμβατότητας

Συμβατότητα ως προς το μέγεθος δείκτη

- Το μέγεθος όλων των δεικτών είναι το ίδιο
  - Κάθε μεταβλητή δείκτη κατέχει τη διεύθυνση μιας θέσης μνήμης στον υπολογιστή
  - Το μέγεθος της μεταβλητής στην οποία παραπέμπει ο δείκτης μπορεί να είναι διαφορετικό· παίρνει τα χαρακτηριστικά του τύπου στον οποίο παραπέμπει

Συμβατότητα ως προς τον τύπο έμμεσης αναφοράς (1/2)

- Ο τύπος έμμεσης αναφοράς είναι ο τύπος της μεταβλητής στην οποία αναφέρεται ο δείκτης
  - Με μία εξαίρεση· είναι άκυρο να αναθέσετε έναν δείκτη ενός τύπου σε έναν δείκτη άλλου τύπου, παρόλο που οι τιμές και στις δύο περιπτώσεις είναι διευθύνσεις μνήμης και, επομένως, φαίνεται να είναι πλήρως συμβατές




```
char c;
char* pc;

int  a;
int* pa;

pc = &c; // Ορθό και έγκυρο
pa = &a; // Ορθό και έγκυρο

pc = &a; // Σφάλμα: Διαφορετικοί τύποι
pa =  a; // Σφάλμα: Διαφορετικά επίπεδα
```

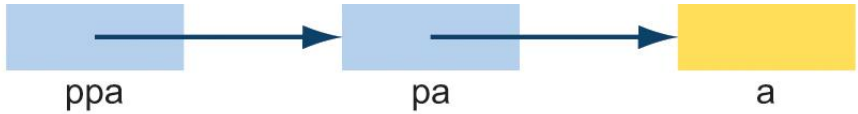
ΣΧΗΜΑ 9-22 Συμβατότητα τύπου έμμεσης αναφοράς.

Συμβατότητα ως προς τον τύπο έμμεσης αναφοράς (2/2)

- Δείκτης σε void
  - Η εξαίρεση στον κανόνα συμβατότητας τύπου αναφοράς είναι ο δείκτης σε void
  - Ένας δείκτης σε void είναι ένας γενικός τύπος που δεν συνδέεται με κάποιον τύπο αναφοράς, δηλαδή δεν είναι η διεύθυνση ενός χαρακτήρα, ενός ακέραιου, ενός πραγματικού ή οποιουδήποτε άλλου τύπου
    - Ωστόσο, είναι συμβατός, μόνο για σκοπούς ανάθεσης, με όλους τους άλλους τύπους δεικτών
- Μετατροπή τύπου δεικτών
  - Το πρόβλημα της ασυμβατότητας ως προς τον τύπο δεδομένων μπορεί να λυθεί εάν χρησιμοποιήσουμε ρητή μετατροπή τύπου (cast)
  - Για αναθέσεις μεταξύ ασύμβατων τύπων δεικτών μπορούμε να χρησιμοποιήσουμε ρητή μετατροπή τύπου

Συμβατότητα ως προς το επίπεδο έμμεσης αναφοράς

- Βασική προϋπόθεση συμβατότητας είναι επίσης η συμβατότητα ως προς το επίπεδο έμμεσης αναφοράς




```
int      a;          // τύπος: int
int      b;          // τύπος: int
int*     pa;         // τύπος: δείκτης σε int
int**    ppa;        // τύπος: δείκτης σε δείκτη σε int

pa   = &a;           // Έγκυρο: ίδιο επίπεδο
ppa  = &pa;          // Έγκυρο: ίδιο επίπεδο
b    = **pa;         // Έγκυρο: ίδιο επίπεδο

pa   = a;            // Άκυρο: διαφορετικό επίπεδο
ppa  = pa;           // Άκυρο: διαφορετικό επίπεδο
b    = *ppa;         // Άκυρο: διαφορετικό επίπεδο
```

ΣΧΗΜΑ 9-23 Συμβατότητα ως προς το επίπεδο έμμεσης αναφοράς.

9.5 Εκφράσεις lvalue και rvalue (αριστερής και δεξιάς τιμής)

9.5 Εκφράσεις lvalue και rvalue (αριστερής και δεξιάς τιμής)

- Η τιμή σε μία έκφραση (μετά την αποτίμηση) μπορεί να χρησιμοποιηθεί με δύο διαφορετικούς τρόπους

  1. Μία έκφραση lvalue πρέπει να χρησιμοποιείται κάθε φορά που το αντικείμενο λαμβάνει μία τιμή, δηλαδή τροποποιείται

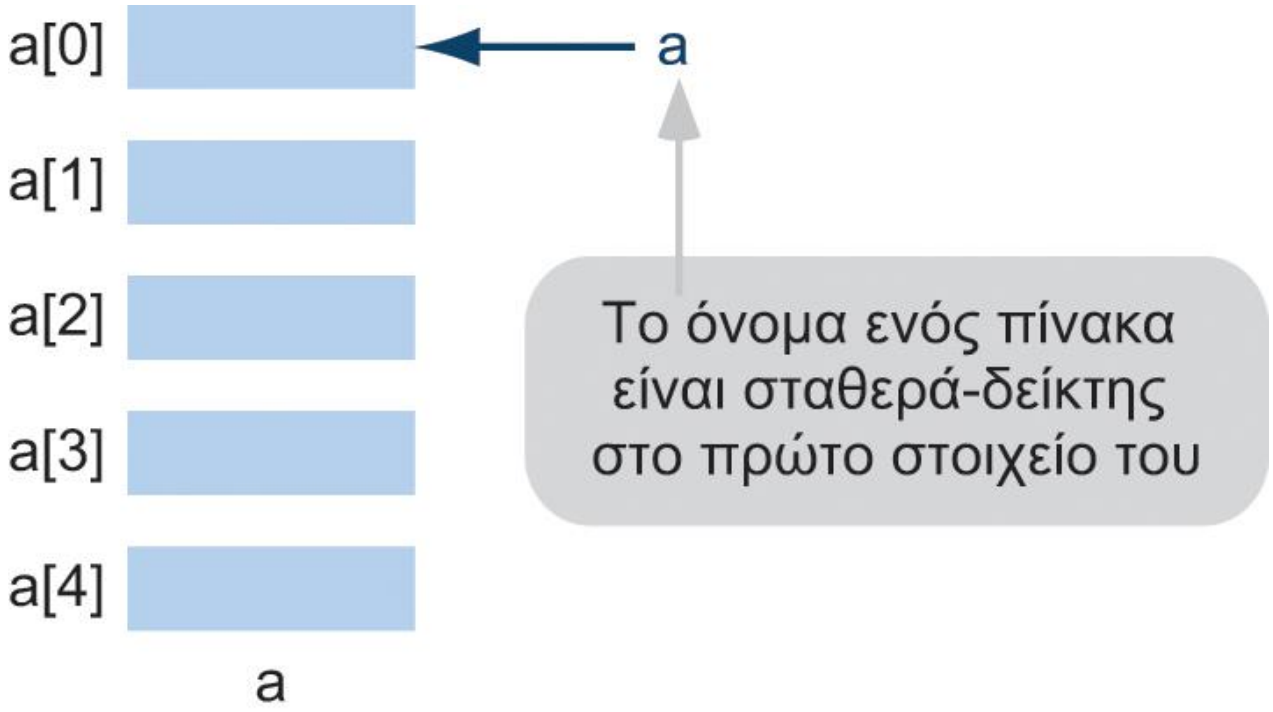
  2. Μία έκφραση rvalue μπορεί να χρησιμοποιηθεί για να παρέχει μία τιμή για περαιτέρω χρήση, δηλαδή για να εξετάσει ή να αντιγράψει την τιμή της

9.6 Πίνακες και δείκτες

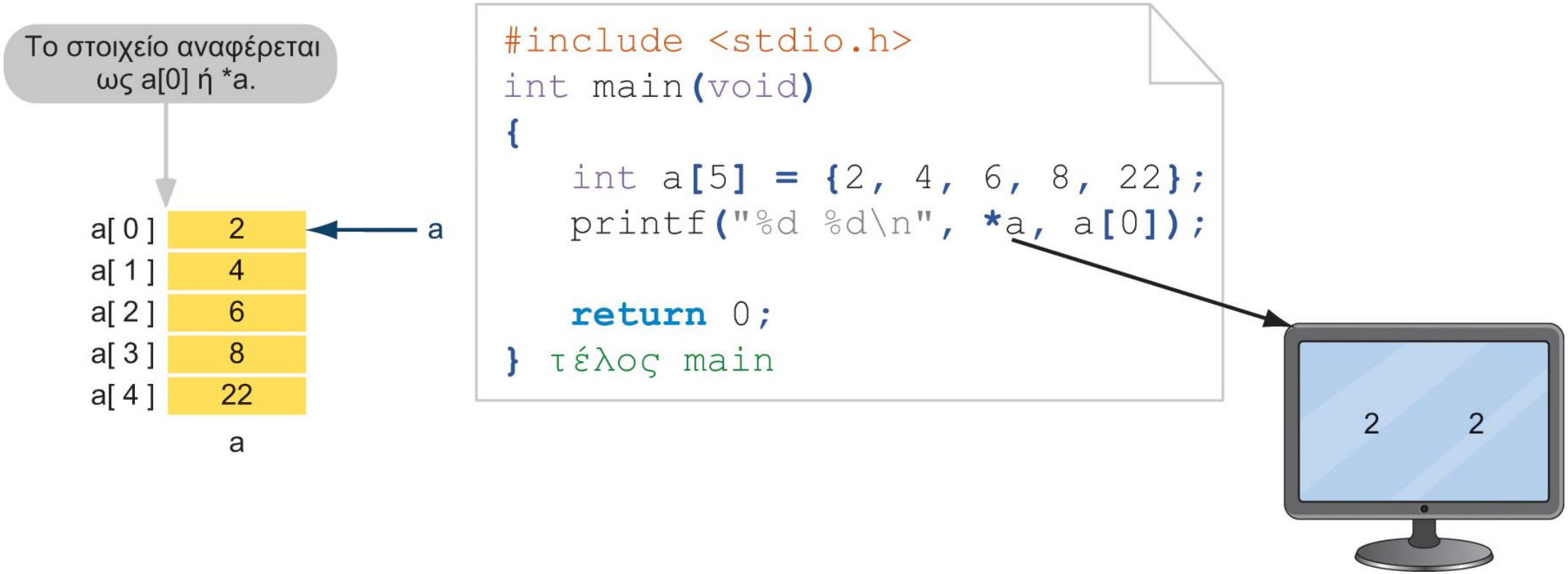
9.6 Πίνακες και δείκτες (1/4)

- Το όνομα ενός πίνακα είναι μία σταθερά δείκτη προς το πρώτο στοιχείο του πίνακα
  - Επειδή το όνομα του πίνακα είναι μία σταθερά δείκτη, η τιμή του δεν μπορεί να αλλάξει

ΣΧΗΜΑ 9-26
Το όνομα του πίνακα ως δείκτης.

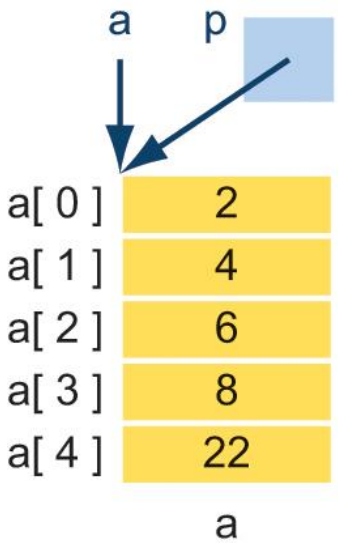


9.6 Πίνακες και δείκτες (2/4)



ΣΧΗΜΑ 9-27 Το όνομα του πίνακα ως έμμεση αναφορά στο πρώτο στοιχείο του.

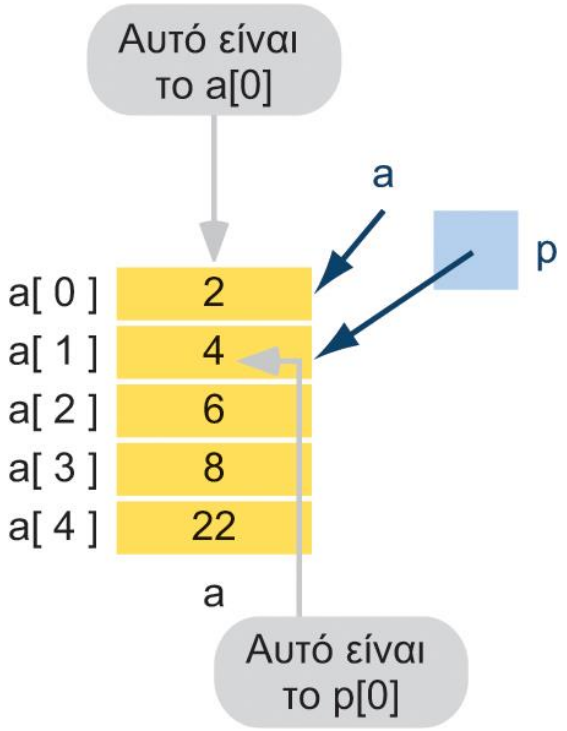
9.6 Πίνακες και δείκτες (3/4)




```c
#include <stdio.h>
int main(void)
{
    int a[5] = {2, 4, 6, 8, 22};
    int* p = a;
    ...
    printf("%d %d\n", a[0], *p);
     ...
    return 0;
} τέλος main
```

ΣΧΗΜΑ 9-28 Το όνομα πίνακα ως δείκτης.

9.6 Πίνακες και δείκτες (4/4)




```c
#include <stdio.h>
int main(void)
{
    int a[5] = {2, 4, 6, 8, 22};
    int* p;
    ...
    p = &a[1];
    pr1ntf("%d %d", a[0], p[-1]);
    printf("\n");
    printf("%d %d", a[1], p[0]);
    ...
} // τέλος main
```

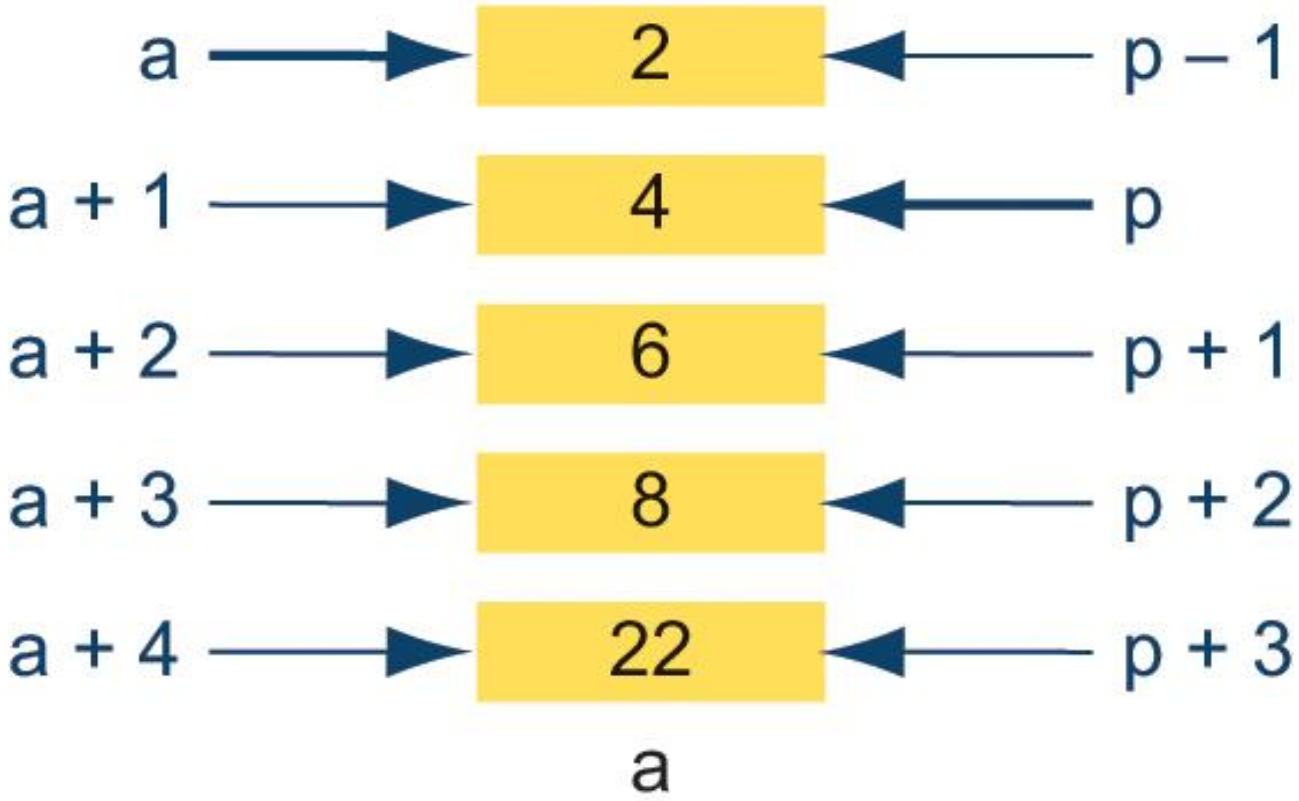
ΣΧΗΜΑ 9-29 Πολλαπλοί δείκτες σε πίνακα.

9.7 Αριθμητική δεικτών και πίνακες

9.7 Αριθμητική δεικτών και πίνακες

- Η αριθμητική δεικτών (pointer arithmetic) είναι μία άλλη ισχυρή μέθοδος μετακίνησης μέσα σε έναν πίνακα
  - Η αριθμητική δεικτών προσφέρει ένα περιορισμένο σύνολο αριθμητικών τελεστών για το χειρισμό των διευθύνσεων στους δείκτες
  - Είναι ιδιαίτερα ισχυρή όταν πρέπει να κινηθούμε μέσα σε έναν πίνακα από στοιχείο σε στοιχείο, όπως όταν διεξάγουμε γραμμικές αναζητήσεις σε έναν πίνακα

Δείκτες και μονοδιάστατοι πίνακες



ΣΧΗΜΑ 9-30 Αριθμητική δεικτών.

Αριθμητικές πράξεις σε δείκτες

- Οι αριθμητικές πράξεις που μπορούν να εφαρμοστούν σε δείκτες είναι πολύ περιορισμένες
  - Η πρόσθεση μπορεί να χρησιμοποιηθεί όταν ο ένας τελεστέος είναι δείκτης και ο άλλος ακέραιος
  - Η αφαίρεση μπορεί να χρησιμοποιηθεί μόνο όταν και οι δύο τελεστέοι είναι δείκτες, ή όταν ο πρώτος τελεστέος είναι δείκτης και ο δεύτερος είναι δείκτης προς ακέραιος
  - Μπορούμε επίσης να χειριστούμε έναν δείκτη με τους επιθεματικούς και μοναδιαίους τελεστές αύξησης και μείωσης τιμής

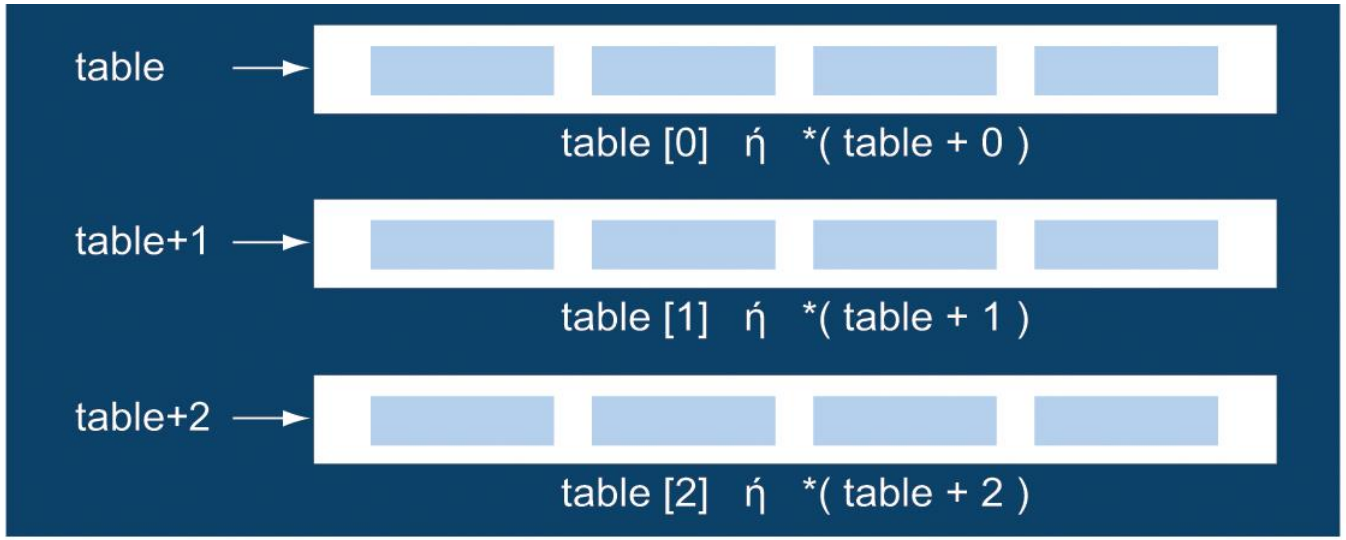
Χρήση αριθμητικής δεικτών

- Παραδείγματα προγραμμάτων που επιδεικνύουν τη μετακίνηση μέσω πινάκων με χρήση δεικτών
  - Εκτύπωση πίνακα
  - Αναζήτηση με δείκτες

Δείκτες και διδιάστατοι πίνακες (1/2)

- Ας υποθέσουμε ότι έχουμε έναν διδιάστατο πίνακα ακέραιων αριθμών
  - Όταν αναφερόμαστε έμμεσα στο όνομα του πίνακα, δεν παίρνουμε έναν ακέραιο, αλλά έναν πίνακα ακεραίων
  - Με άλλα λόγια, η έμμεση αναφορά στο όνομα ενός διδιάστατου πίνακα είναι ένας δείκτης σε έναν μονοδιάστατο πίνακα

Δείκτες και διδιάστατοι πίνακες (2/2)



ΣΧΗΜΑ 9-34
Δείκτες σε διδιάστατους πίνακες.

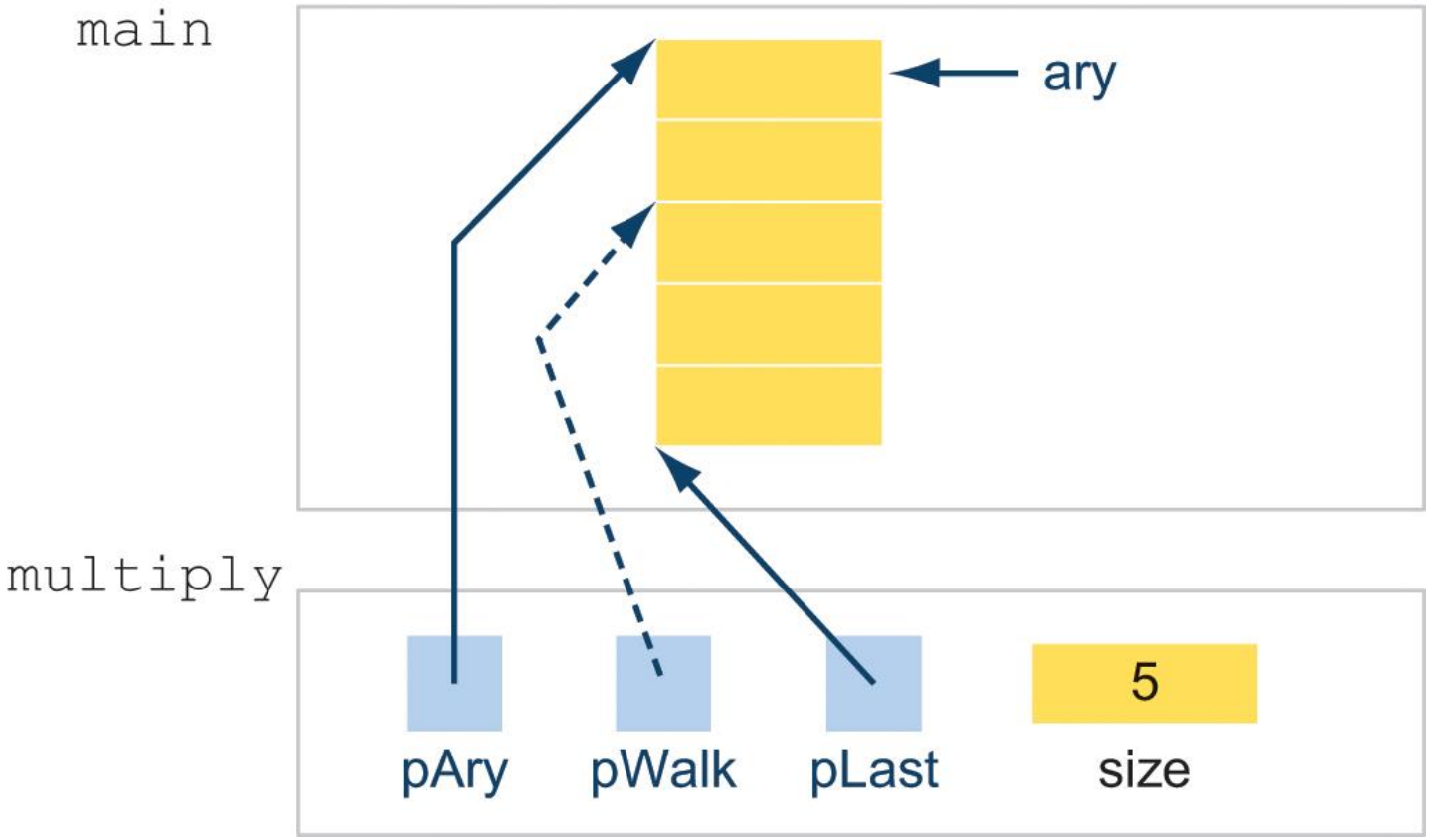
```
for (i=0; i<3; i++)
{
    for (j=0; j<4; j++)
       pr1ntf("%6d", *(*(table + i) + j));
   printf("\n");
} // τέλος for i
```

9.8 Πέρασμα πινάκων σε συναρτήσεις

9.8 Πέρασμα πινάκων σε συναρτήσεις (1/2)

- Επειδη το όνομα ενός πίνακα είναι στην πραγματικότητα ένας δείκτης στο πρώτο στοιχείο, μπορούμε να στείλουμε το όνομα του πίνακα σε μία συνάρτηση για επεξεργασία
- Το καλούμενο πρόγραμμα μπορεί να δηλώσει τον πίνακα με έναν από τους δύο τρόπους
  - Χρησιμοποιώντας την παραδοσιακή σημειογραφία πινάκων
  - Μπορούμε επίσης να δηλώσουμε τον πίνακα στην επικεφαλίδα της συνάρτησης ως απλό δείκτη

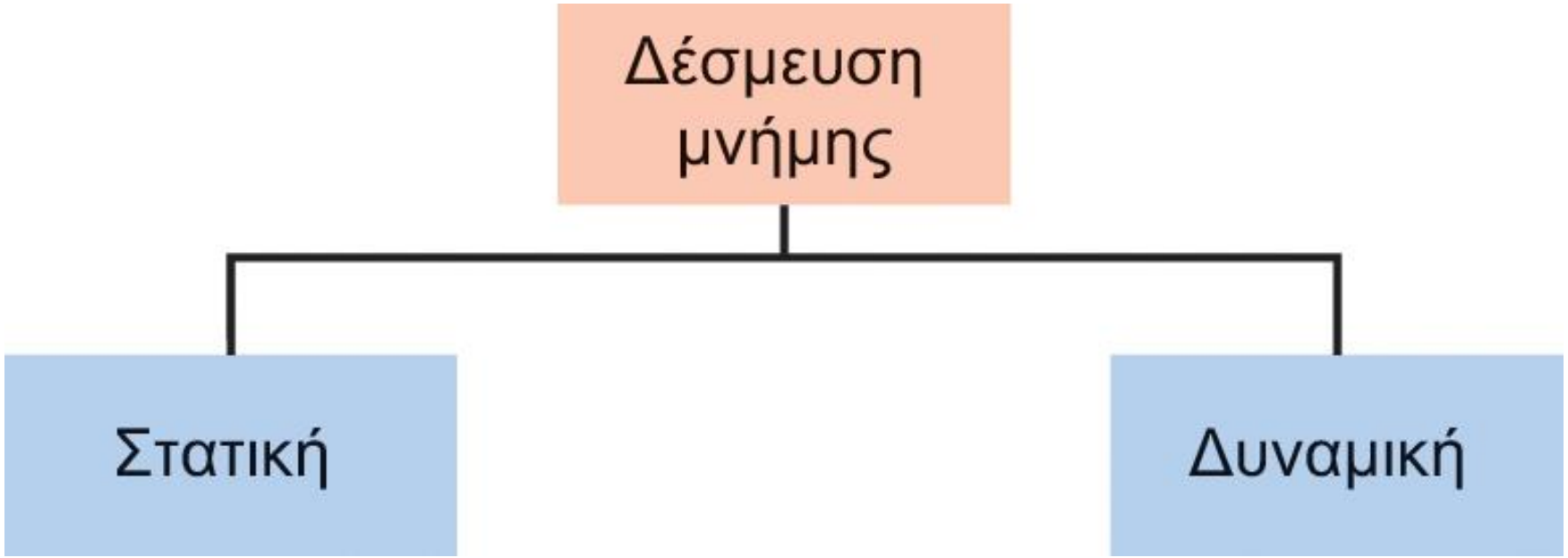
9.8 Πέρασμα πινάκων σε συναρτήσεις (2/2)



ΣΧΗΜΑ 9-35 Μεταβλητές για το πρόγραμμα πολλαπλασιασμού στοιχείων πίνακα επί 2.

9.9 Συναρτήσεις δέσμευσης μνήμης

9.9 Συναρτήσεις δέσμευσης μνήμης

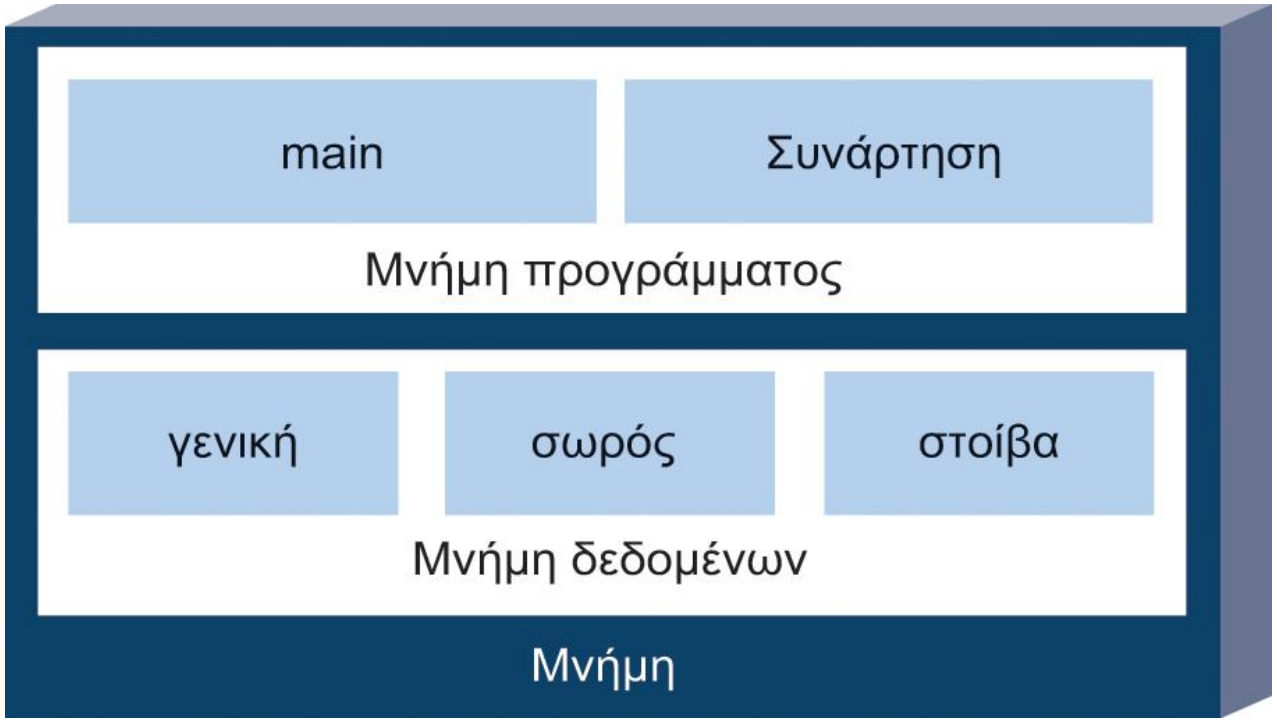


ΣΧΗΜΑ 9-36 Προσεγγίσεις δέσμευσης μνήμης.

Χρήση της μνήμης

- Για να καταλάβουμε πώς λειτουργεί η δυναμική δέσμευση μνήμης, πρέπει να μελετήσουμε πώς χρησιμοποιείται η μνήμη
- Εννοιολογικά, η μνήμη χωρίζεται σε μνήμη προγράμματος και μνήμη δεδομένων

ΣΧΗΜΑ 9-37 Εννοιολογική αναπαράσταση της μνήμης.



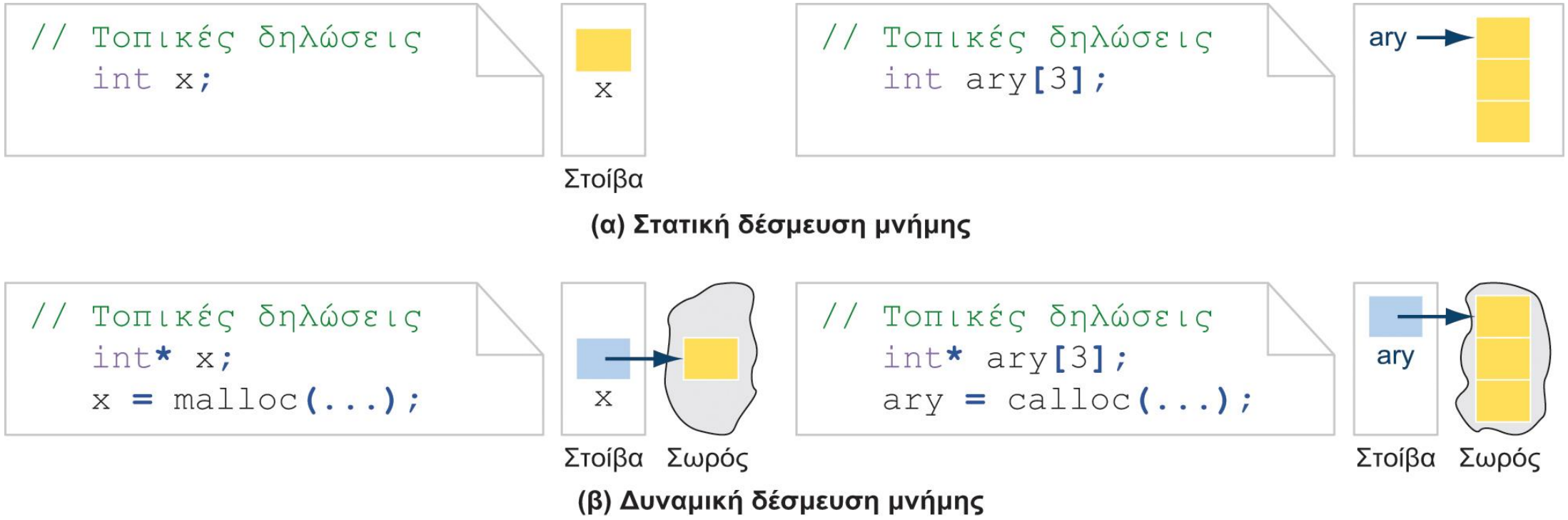
Στατική δέσμευση μνήμης

- Η προσέγγιση της στατικής δέσμευσης μνήμης (static memory allocation), προϋποθέτει ότι η ποσότητα μνήμης που απαιτείται για δηλώσεις και ορισμούς καθορίζεται πλήρως στο πηγαίο πρόγραμμα
  - Ο αριθμός των bytes μνήμης που δεσμεύονται δεν μπορεί να αλλάξει κατά τη διάρκεια της εκτέλεσης του προγράμματος
  - Αυτή είναι η τεχνική που χρησιμοποιήσαμε μέχρι τώρα για τον ορισμό μεταβλητών, πινάκων, δεικτών και ροών

Δυναμική δέσμευση μνήμης (1/2)

- Η προσέγγιση της δυναμικής δέσμευση μνήμης (static memory allocation) χρησιμοποιεί προκαθορισμένες συναρτήσεις για τη δέσμευση και αποδέσμευση μνήμης για δεδομένα κατά τη διάρκεια εκτέλεσης του προγράμματος
  - Ουσιαστικά αναβάλλει τον ορισμό των δεδομένων, αλλά όχι τη δήλωση των δεδομένων, για το χρόνο εκτέλεσης

Δυναμική δέσμευση μνήμης (2/2)

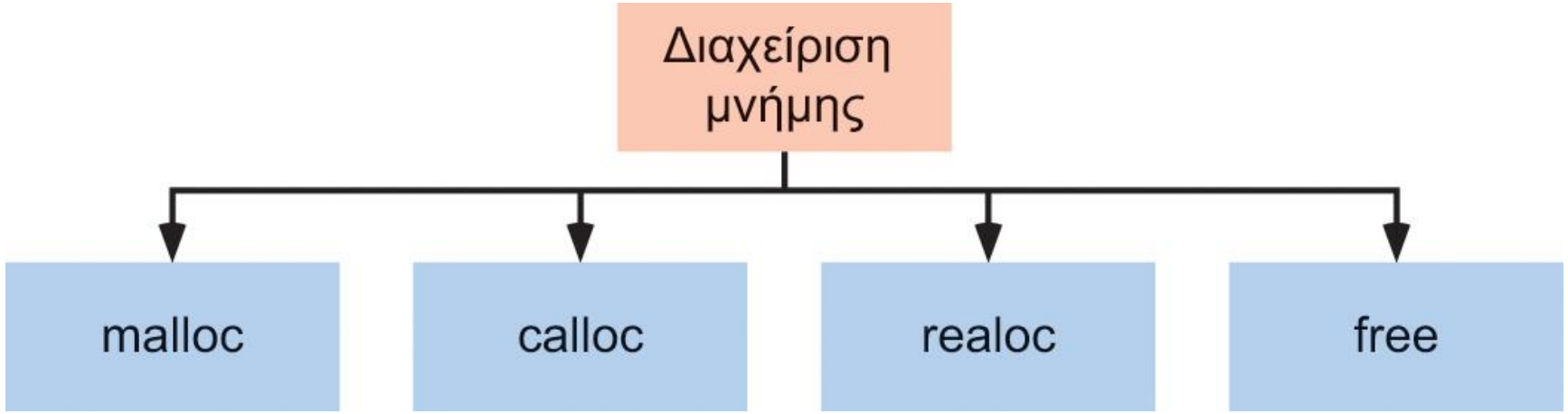


ΣΧΗΜΑ 9-38 Προσπέλαση δυναμικής μνήμης.

Συναρτήσεις δέσμευσης μνήμης

- Η C παρέχει τέσσερις συναρτήσεις για τη διαχείριση δυναμικής μνήμης.
  - Δέσμευση μπλοκ μνήμης (malloc)
    - Δεσμεύει ένα μπλοκ μνήμης που περιέχει τον αριθμό των bytes που καθορίζεται στην παράμετρό της
  - Δέσμευση συνεχούς μνήμης (calloc)
    - Χρησιμοποιείται κυρίως για την δέσμευση μνήμης για πίνακες

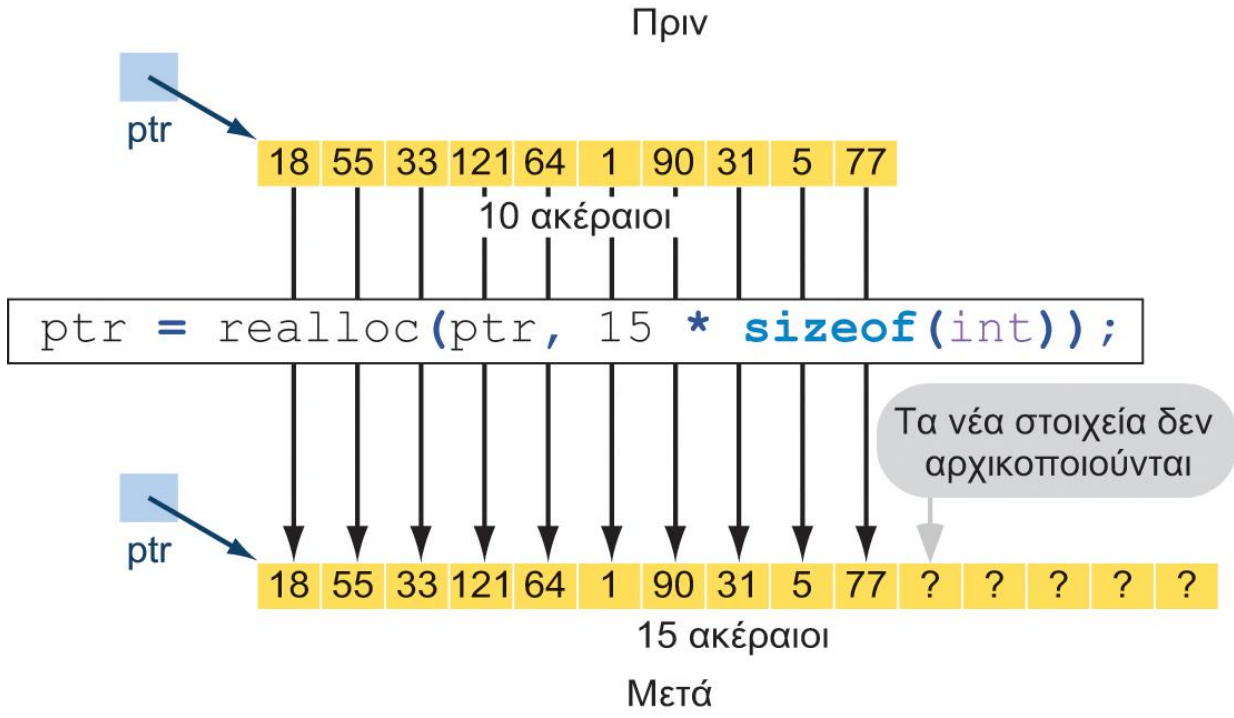
ΣΧΗΜΑ 9-39 Συναρτήσεις διαχείρισης μνήμης.



Εκ νέου δέσμευση μνήμης (`realloc`)

- Η συνάρτηση realloc μπορεί να είναι εξαιρετικά αναποτελεσματική και επομένως θα πρέπει να χρησιμοποιείται με προσοχή
- Αλλάζει το μέγεθος του μπλοκ διαγράφοντας ή επεκτείνοντας τη μνήμη στο τέλος του μπλοκ

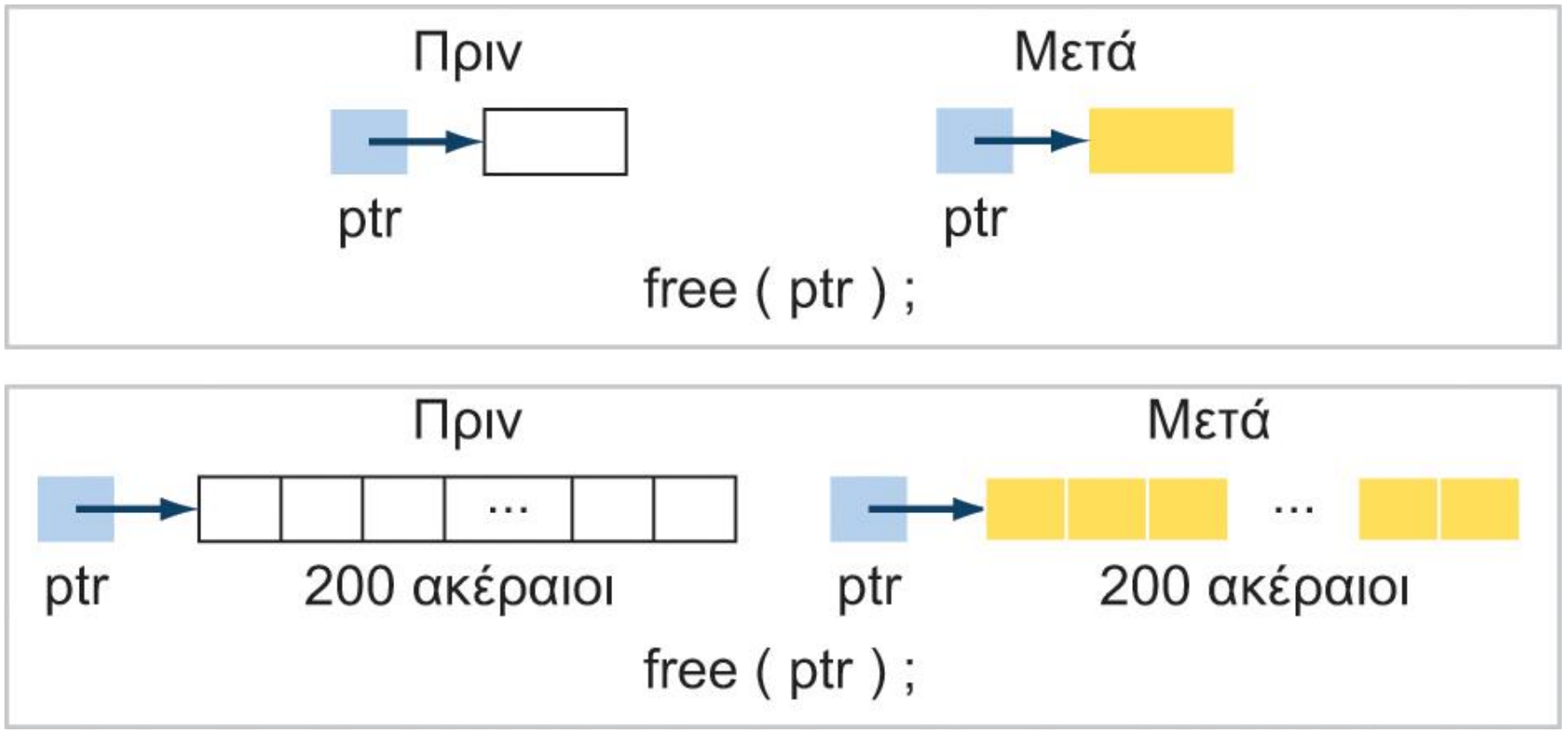
ΣΧΗΜΑ 9-42
Σχηματική αναπαράσταση της συνάρτησης realloc.



Ελευθέρωση μνήμης (`free`) (1/2)

- Όταν οι θέσεις μνήμης που έχουν δεσμευτεί με malloc, calloc ή realloc δεν χρειάζονται πλέον, θα πρέπει να ελευθερωθούν χρησιμοποιώντας την προκαθορισμένη συνάρτηση free
- Αποτελεί σφάλμα η αποδέσμευση μνήμης με κενό (null) δείκτη, με δείκτη που δεν είναι το πρώτο στοιχείο ενός δεσμευμένου μπλοκ, με δείκτη που είναι διαφορετικού τύπου από τον δείκτη για τον οποίο δεσμεύτηκε η μνήμη
- Είναι επίσης πιθανό σφάλμα η αναφορά στη μνήμη μετά την αποδέσμευσή της

Ελευθέρωση μνήμης (`free`) (2/2)



ΣΧΗΜΑ 9-43 Αποδέσμευση μνήμης.

9.10 Πίνακες δεικτών

9.10 Πίνακες δεικτών (1/3)

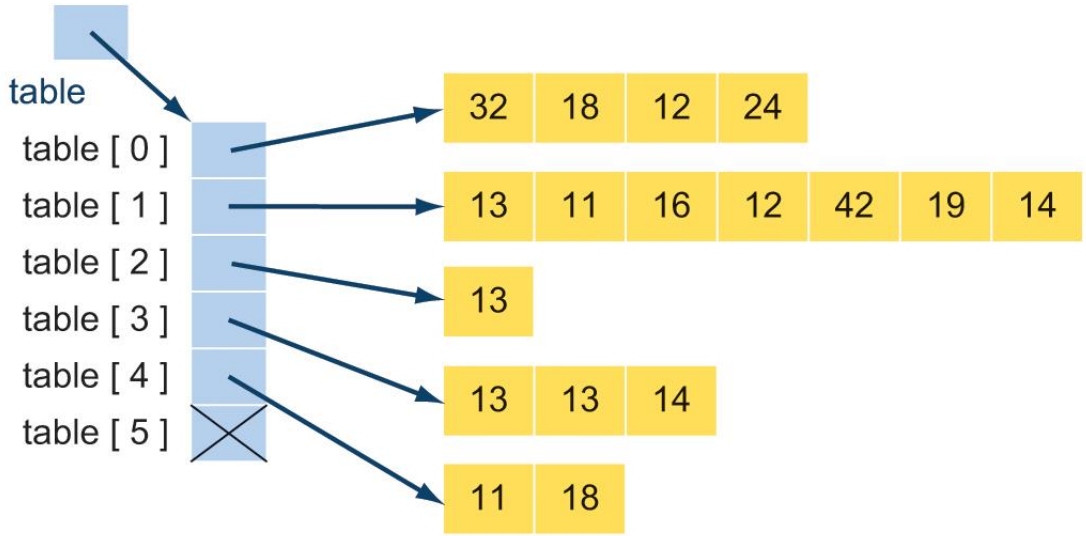
- Μία άλλη χρήσιμη δομή που χρησιμοποιεί πίνακες και δείκτες είναι ένας πίνακας δεικτών (array of pointers)
  - Αυτή η δομή είναι ιδιαίτερα χρήσιμη όταν ο αριθμός των στοιχείων του πίνακα είναι μεταβλητός
  - Γενικά, οι πίνακες στους οποίους μπορεί να λείπουν ένα ή περισσότερα στοιχεία από το τέλος μιας ή περισσότερων γραμμών αναφέρονται ως «ακανόνιστοι» πίνακες (ragged arrays), επειδή η απουσία στοιχείων οδηγεί σε «ακανόνιστο» δεξιό όριο

9.10 Πίνακες δεικτών (2/3)

ΠΙΝΑΚΑΣ 9-6 «Ακανόνιστος» πίνακας

32	18	12	24			
13	11	16	12	42	19	14
22						
13	13	14				
11	18					

9.10 Πίνακες δεικτών (3/3)



ΣΧΗΜΑ 9-44 Σχηματική αναπαράσταση και δήλωση «ακανόνιστου» πίνακα.

```
table = (int**)calloc (rowNum + 1, sizeof(int*));

table[0] = (int*)calloc (4, sizeof(int));
table[1] = (int*)calloc (7, sizeof(int));
table[2] = (int*)calloc (1, sizeof(int));
table[3] = (int*)calloc (3, sizeof(int));
table[4] = (int*)calloc (2, sizeof(int));
table[5] = NULL;;
```

Σύνοψη

- Κάθε υπολογιστής διαθέτει «διευθυνσιοδοτήσιμες» (προσπελάσιμες μέσω διεύθυνσης) θέσεις μνήμης

- Ένας δείκτης είναι ένας παράγωγος τύπος δεδομένων που αποτελείται από διευθύνσεις που είναι διαθέσιμες στον υπολογιστή

- Μία σταθερά δείκτη είναι μία διεύθυνση που υπάρχει από μόνη της. Δεν μπορούμε να την αλλάξουμε. Μπορούμε μόνο να τη χρησιμοποιήσουμε

- Ο τελεστής διεύθυνσης (&) δημιουργεί μία τιμή δείκτη από μία σταθερά δείκτη. Για να πάρουμε τη διεύθυνση μιας μεταβλητής, χρησιμοποιούμε απλά αυτόν τον τελεστή μπροστά από το όνομα της μεταβλητής

- Ο τελεστής διεύθυνσης μπορεί να χρησιμοποιηθεί μόνο μπροστά από μία έκφραση lvalue. Το αποτέλεσμα είναι μία έκφραση rvalue

- Μία μεταβλητή δείκτη είναι μία μεταβλητή που μπορεί να αποθηκεύσει μία διεύθυνση

Σύνοψη

- Ο τελεστής έμμεσης αναφοράς, ή ανακατεύθυνσης (*), προσπελάζει μία μεταβλητή μέσω ενός δείκτη που περιέχει τη διεύθυνσή της

- Οι μεταβλητές δείκτη μπορούν να αρχικοποιηθούν όπως ακριβώς και οι μεταβλητές δεδομένων. Η τιμή αρχικοποίησης πρέπει να είναι η διεύθυνση μιας προηγουμένως ορισμένης μεταβλητής ή μιας άλλης μεταβλητής δείκτη

- Ένας δείκτης μπορεί να οριστεί να δείχνει σε άλλες μεταβλητές δείκτη (δείκτες προς δείκτες)

- Η τιμή μιας μεταβλητής δείκτη μπορεί να αποθηκευτεί σε μία άλλη μεταβλητή εάν είναι συμβατές, δηλαδή του ίδιου τύπου

- Μία από τις πιο χρήσιμες εφαρμογές των δεικτών είναι σε συναρτήσεις

- Εάν θέλουμε μία καλούμενη συνάρτηση να έχει πρόσβαση σε μία μεταβλητή της καλούσας συνάρτησης, περνάμε τη διεύθυνση της μεταβλητής αυτής στην καλούμενη συνάρτηση και χρησιμοποιούμε τον τελεστή έμμεσης αναφοράς για να προσπελάσουμε τη μεταβλητή

Σύνοψη

- Όταν πρέπει να επιστρέψουμε περισσότερες από μία τιμές από μία συνάρτηση, πρέπει να χρησιμοποιούμε δείκτες

- Εάν ένα στοιχείο δεδομένων δεν πρόκειται να αλλάξει, μεταβιβάζεται στην καλούμενη συνάρτηση ως τιμή. Εάν ένα στοιχείο δεδομένων πρόκειται να αλλάξει, η διεύθυνσή του μεταβιβάζεται για να μπορέσει η καλούμενη συνάρτηση να αλλάξει την τιμή του

- Μία συνάρτηση μπορεί επίσης να επιστρέψει μία τιμή δείκτη

- Οι πίνακες και οι δείκτες έχουν στενή σχέση. Το όνομα ενός πίνακα είναι μία σταθερά δείκτη στο πρώτο στοιχείο του πίνακα

- Το όνομα ενός πίνακα είναι ένας δείκτης στο πρώτο στοιχείο μόνο, όχι για ολόκληρο τον πίνακα

- Μία μεταβλητή δείκτη στο πρώτο στοιχείο ενός πίνακα μπορεί να χρησιμοποιηθεί οπουδήποτε επιτρέπεται το όνομα του πίνακα, όπως με ένα δείκτη

Σύνοψη

- Στην αριθμητική δεικτών, εάν το `ptr` δείχνει σε ένα συγκεκριμένο στοιχείο ενός πίνακα, το `ptr + n` είναι η τιμή του δείκτη n στοιχεία μακριά

- Το όνομα ενός διδιάστατου πίνακα είναι ένας δείκτης στην πρώτη γραμμή του που είναι ένας μονοδιάστατος πίνακας

- Σε έναν πολυδιάστατο πίνακα, οι ακόλουθες δύο εκφράσεις είναι ισοδύναμες

```
*(*(a + i) + j) a[i][j]
```

- Μπορούμε να περάσουμε έναν πίνακα σε μία συνάρτηση με πολλούς τρόπους. Ένας τρόπος είναι να περάσουμε το όνομα του πίνακα ως δείκτη

- Ένας πίνακας δεικτών μπορεί να χρησιμοποιηθεί για εξοικονόμηση χώρου όταν δεν είναι γεμάτες όλες οι γραμμές του πίνακα

Σύνοψη

- Η μνήμη ενός υπολογιστή μπορεί να χωριστεί σε μνήμη προγράμματος και μνήμη δεδομένων. Η μνήμη δεδομένων μπορεί να χωριστεί σε περιοχή «γενικών δηλώσεων», σωρό και στοίβα

- Η στατική δέσμευση της μνήμης απαιτεί να προσδιορίζεται πλήρως η δήλωση και ο ορισμός της μνήμης κατά τη μεταγλώττιση

- Η δυναμική δέσμευση της μνήμης γίνεται κατά τη διάρκεια του χρόνου εκτέλεσης μέσω της χρήσης προκαθορισμένων συναρτήσεων

- Η C διαθέτει τέσσερις προκαθορισμένες συναρτήσεις δέσμευσαης-δέσμευσης μνήμης: `malloc`, `calloc`, `realloc` και `free`

- Για να διαβάσουμε και να ερμηνεύσουμε μία σύνθετη δήλωση, μπορούμε να χρησιμοποιήσουμε τον κανόνα δεξιά-αριστερά

Σύνοψη

- Στη μηχανική λογισμικού, οι παράγοντες ποιότητας αναφέρονται στα χαρακτηριστικά που πρέπει να έχει ένα κομμάτι λογισμικού για να γίνει ποιοτικό λογισμικό

- Ως παράγοντες ποιότητας ορίζονται η λειτουργικότητα, η συντηρησιμότητα και η δυνατότητα μεταφοράς

- Ένα από τα πιο σημαντικά σημεία αναφορικά με την ποιότητα ενός συστήματος λογισμικού έχει να κάνει με το γεγονός ότι η ποιότητα πρέπει να εντάσσεται εξαρχής στο σύστημα, από τη σχεδίασή του. Δεν μπορεί να προστεθεί σε αυτό εκ των υστέρων. Για να εντάξουμε την ποιότητα στη σχεδίαση ενός συστήματος μπορούμε να χρησιμοποιήσουμε ένα εργαλείο το οποίο ονομάζεται κύκλος ποιότητας

Το παρόν συνοδευτικό έργο **παρέχεται** αποκλειστικά και μόνο στους διδάσκοντες που έχουν επιλέξει το αντίστοιχο βιβλίο μέσω του συστήματος του **Ευδόξου** ως διδακτικό σύγγραμμα για τη διδασκαλία των μαθημάτων τους και την αξιολόγηση της εκμάθησης των φοιτητών και για όσο χρονικό διάστημα διατηρείται η επιλογή του συγκεκριμένου συγγράμματος. Η **διάδοση, δημοσίευση, αναπαραγωγή** ή **πώληση** οπουδήποτε μέρους αυτού του έργου με οποιοδήποτε μέσο καταστρέφει την ακεραιότητα του έργου **και για σκοπούς διαφορετικούς από αυτούς για τους οποίους έχει χορηγηθεί η σχετική άδεια δεν επιτρέπεται**. Το περιεχόμενο του έργου **δεν θα πρέπει να διατίθεται** στους φοιτητές παρά μόνο όταν αυτό αναφέρεται ρητά ότι μπορεί να γίνει. Η **χρήση** του παρόντος έργου γίνεται αποκλειστικά από τον διδάσκοντα στα πλαίσια των εκπαιδευτικών καθηκόντων του και με τη χρήση των μέσων που αυτός διαχειρίζεται και έχει στη διάθεσή του από τις **επίσημες υπηρεσίες του εκπαιδευτικού ιδρύματος** όπου ανήκει, διασφαλίζοντας τη μη περαιτέρω διάδοση, δημοσίευση και αναπαραγωγή του έργου προς τρίτους. Ο διδάσκων δύναται να **προσαρμόζει** μέρος του υλικού των διαφανειών σύμφωνα με τις διδακτικές του ανάγκες, αναφερόμενος όμως πάντοτε στην αρχική πηγή αναφοράς. Το παρόν έργο προστατεύεται από την ελληνική και ευρωπαϊκή νομοθεσία περί πνευματικής ιδιοκτησίας καθώς και από τη νομοθεσία του κράτους όπου ανήκει η ξενόγλωσση πρωτότυπη έκδοση του έργου.