Προγραμματισμός Ι (ΗΥ120)

Διάλεξη 10:

Δείκτες

Χρήστος Δ. Αντωνόπουλος
9/11/2016

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Δείκτες

- Υπάρχουν περιπτώσεις που δεν ενδιαφέρει το περιεχόμενο αλλά η **διεύθυνση** μιας μεταβλητής.
  - Χρειάζεται κατάλληλος μηχανισμός αναφοράς και επεξεργασίας τιμών που είναι **διευθύνσεις** μνήμης.
- Υπάρχουν μεταβλητές ειδικού τύπου: **δείκτες**.
  - Μια μεταβλητή δείκτης μπορεί να χρησιμοποιηθεί για την αποθήκευση μιας ακέραιας τιμής που ερμηνεύεται σαν **διεύθυνση μνήμης**.
  - Μέσω μιας μεταβλητής δείκτη μπορούμε να διαβάσουμε και να αλλάξουμε την τιμή που βρίσκεται αποθηκευμένη στην διεύθυνση που αυτή περιέχει.

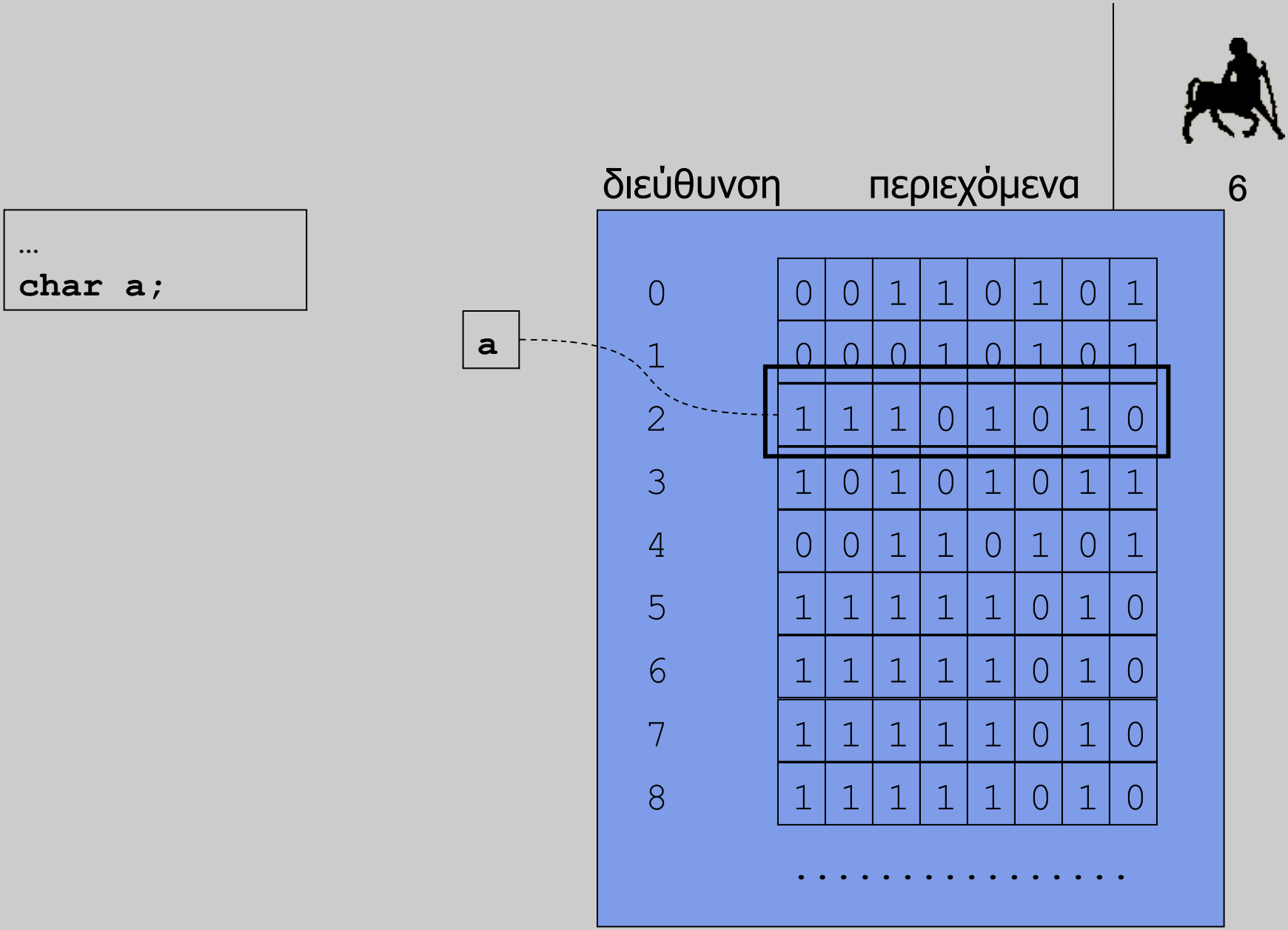
Δείκτες και τύποι δεδομένων

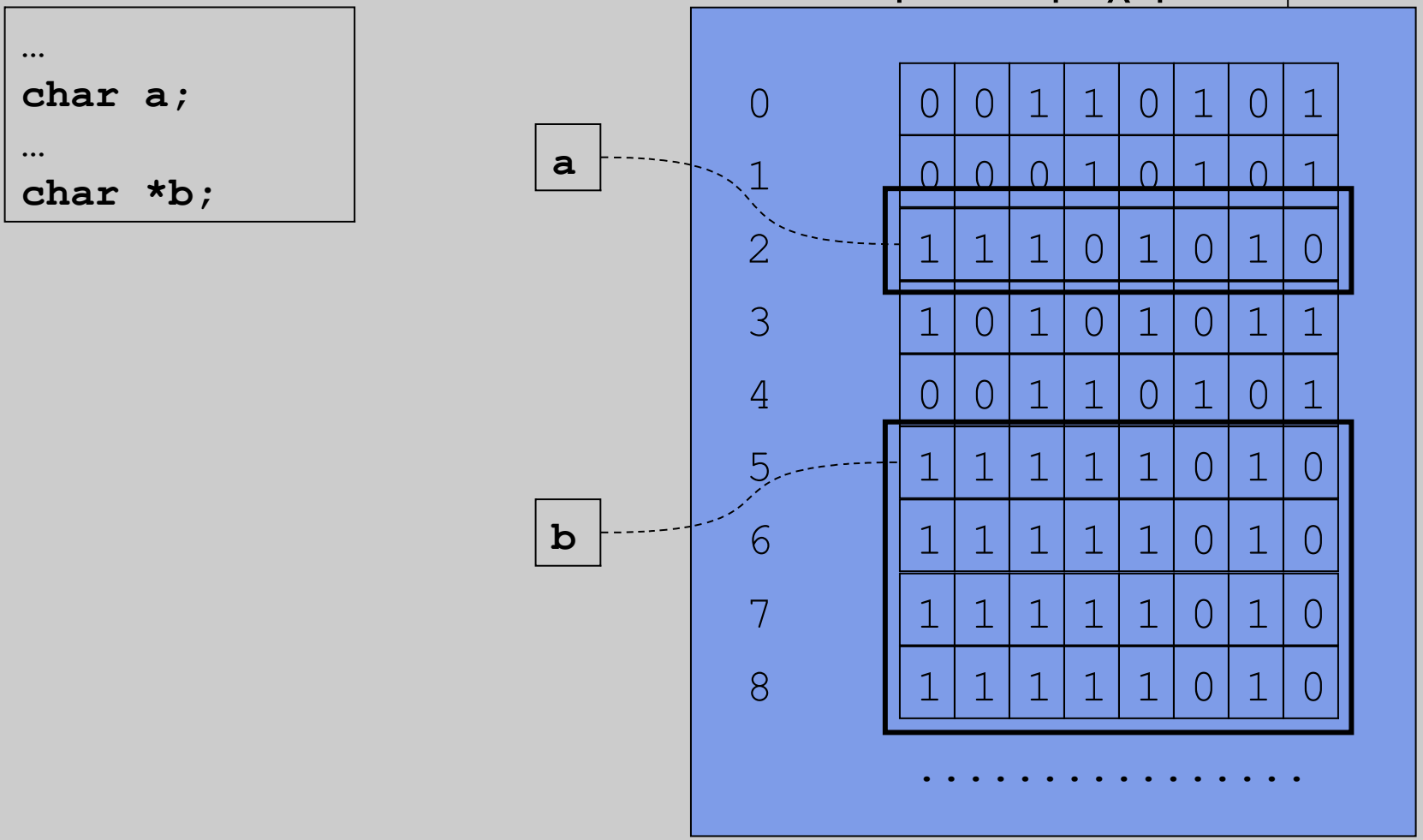
- Η έννοια του δείκτη μπορεί να εφαρμοστεί σε κάθε τύπο T για να οριστεί ο τύπος δείκτης-σε-T.
- Υπάρχει **συντακτική** συμβατότητα ανάμεσα:
  - στην τιμή που βρίσκεται στην διεύθυνση που περιέχει μια μεταβλητή δείκτης-σε-T και την τιμή μιας μεταβλητής τύπου T
  - στην τιμή μιας μεταβλητής δείκτης-σε-T και την διεύθυνση μιας μεταβλητής τύπου T
- Η ειδική τιμή NULL (0) σημαίνει «καμία διεύθυνση», και είναι η συνηθισμένη τιμή αρχικοποίησης μιας μεταβλητής δείκτη.
  - Η προσπάθεια προσπέλασης της διεύθυνσης 0 έχει σαν αποτέλεσμα τον τερματισμό του προγράμματος.

Τελεστές που αφορούν δείκτες

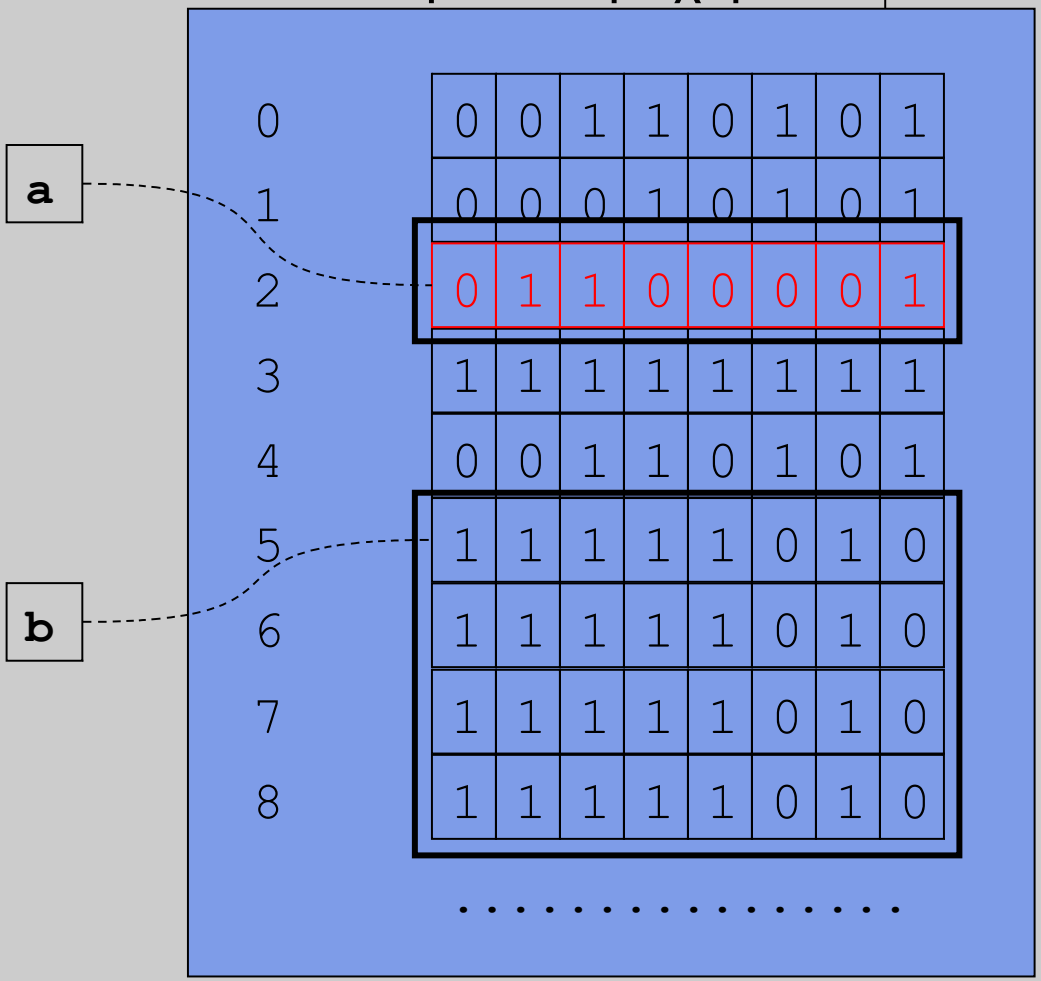
- Ο τελεστής **&** (reference) εφαρμόζεται σε μια μεταβλητή και επιστρέφει την διεύθυνση της.
- Ο τελεστής ***** (dereference) εφαρμόζεται σε μια μεταβλητή δείκτη-σε-Τ ώστε αυτή να ερμηνευτεί σαν μεταβλητή τύπου Τ.
  - Μπορεί να χρησιμοποιηθεί σε οποιαδήποτε έκφραση αποτίμησης ή/και ανάθεσης όπου αναμένεται μια μεταβλητή (ή τιμή) τύπου Τ.
  - Μέσω ενός δείκτη που περιέχει την διεύθυνση μιας (συμβατικής) μεταβλητής μπορούμε να διαβάσουμε / αλλάξουμε **έμμεσα** την τιμή της μεταβλητής, **χωρίς** να φαίνεται στον κώδικα το όνομα της.
    - Αυτό είναι μια μορφή **παρενέργειας**!


```
int a;                  /* ακέραιος a */

int *b,*c;              /* δείκτες σε ακέραιο b,c */

a = 1;


b = &a;                 /* b γίνεται διεύθυνση του a */

*b = 2;                 /* a γίνεται 2 */

*b = *b+1;              /* a γίνεται 3 */

c = b;                  /* c γίνεται διεύθυνση του a */

*c = *c+2;              /* a γίνεται 5 */
```

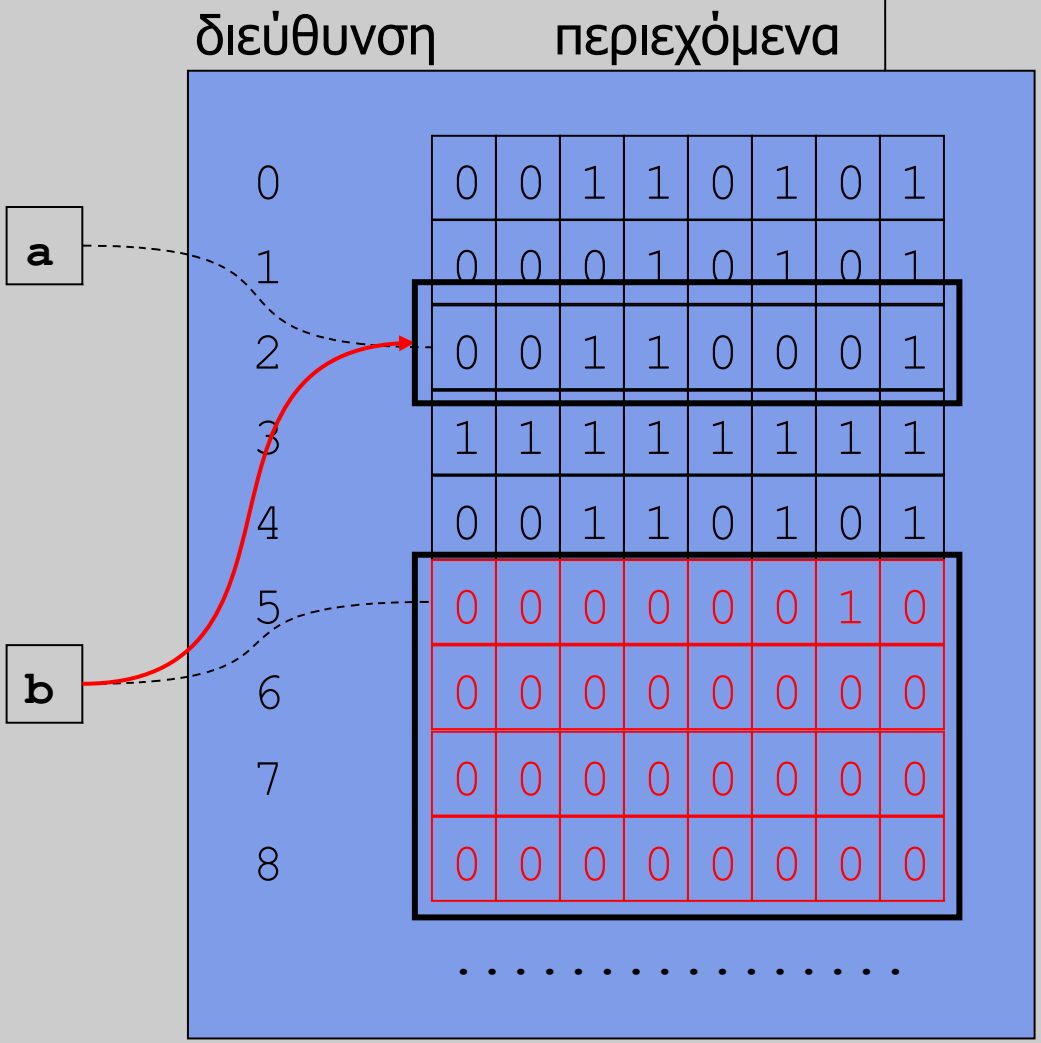

διεύθυνση
περιεχόμενα
6
…
char a;
a
0 0 1 1 0 1 0 1
1 1 1 0 1 0 1 0
1 0 1 0 1 0 1 1
0 0 1 1 0 1 0 1
1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0
0
1
2
3
4
5
6
7
8

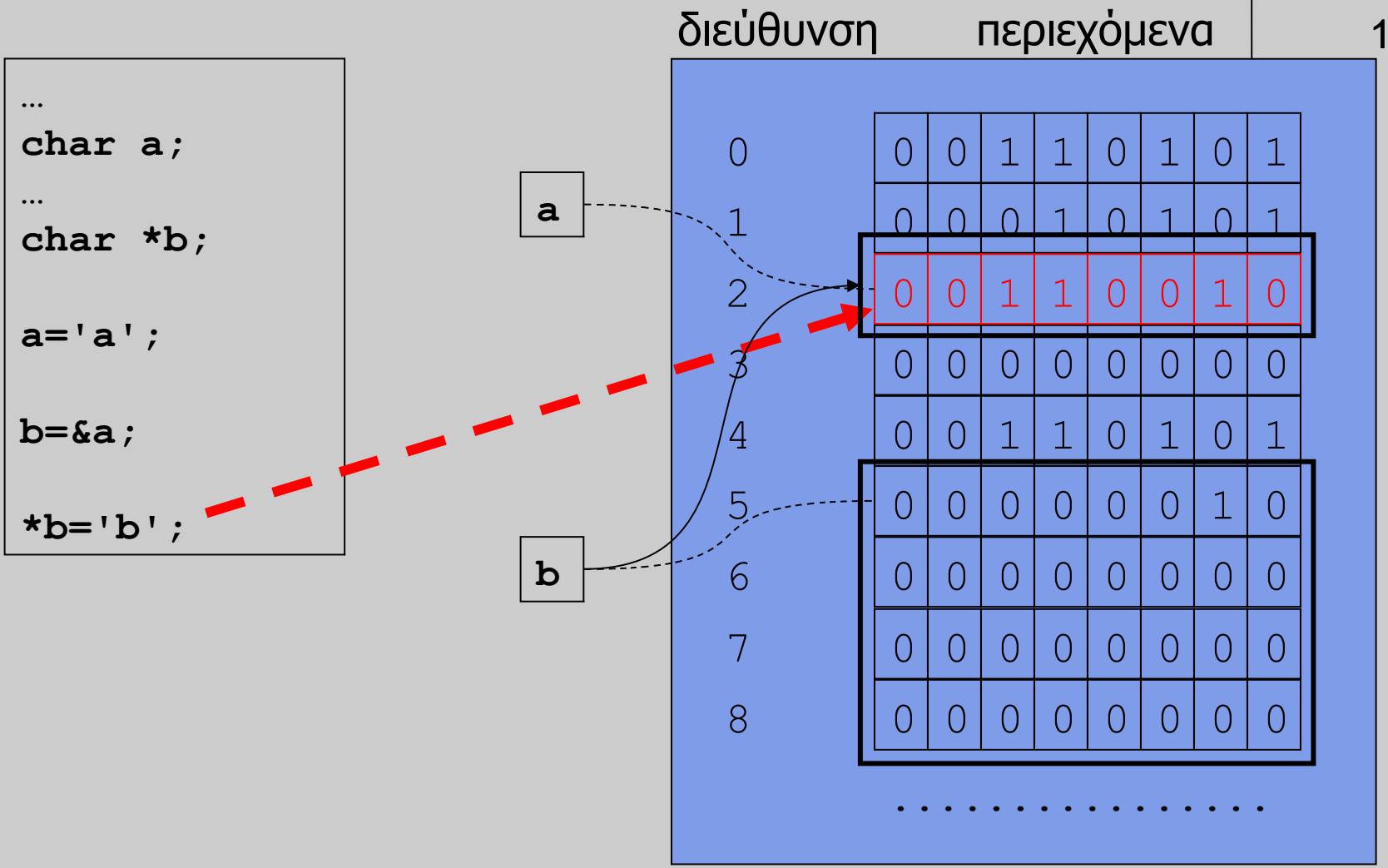

…
char a;
…
char *b;
διεύθυνση
περιεχόμενα
7
a
b
0
1
2
3
4
5
6
7
8
0 0 1 1 0 1 0 1
0 0 0 1 0 1 0 1
1 1 1 0 1 0 1 0
1 0 1 0 1 0 1 1
0 0 1 1 0 1 0 1
1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0
. . . . . .

```
…
char a;
…
char *b;

a='a';
```



```
…
char a;
…
char *b;

a='a';

b=&a;
```

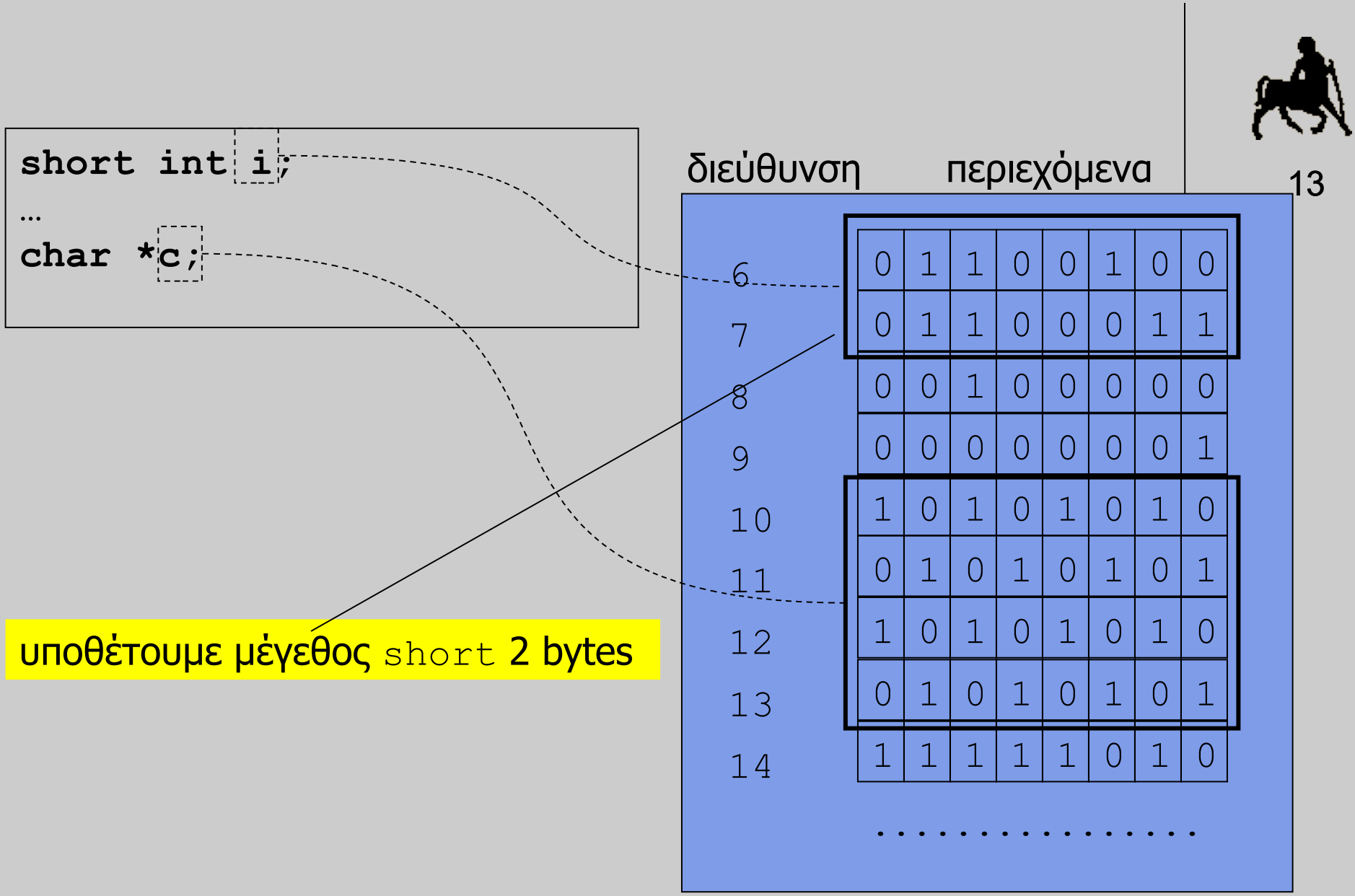



διεύθυνση
περιεχόμενα
…
char a;
…
char *b;
a='a';
b=&a;
*b='b';
a
b
0
1
2
3
4
5
6
7
8
0 0 1 1 0 1 0 1
0 0 0 1 0 1 0 1
0 0 1 1 0 0 1 0
0 0 0 0 0 0 0 0
0 0 1 1 0 1 0 1
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
. . . . . .

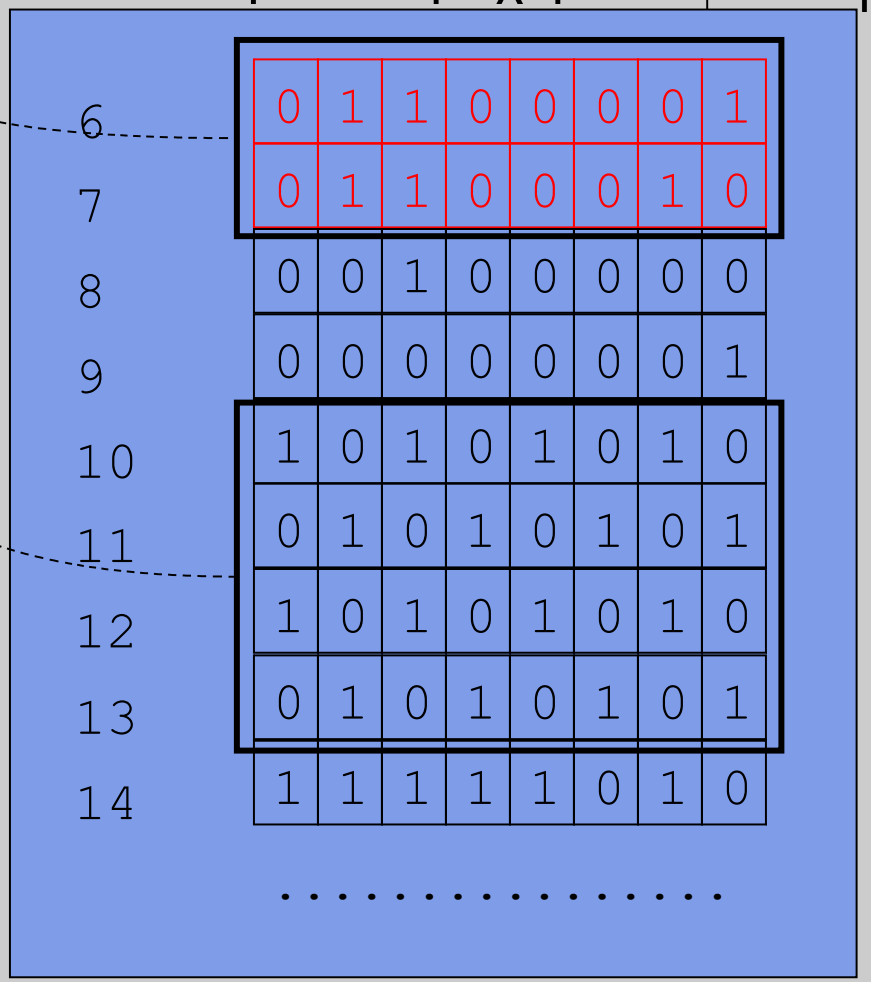
Παράκαμψη συντακτικού ελέγχου

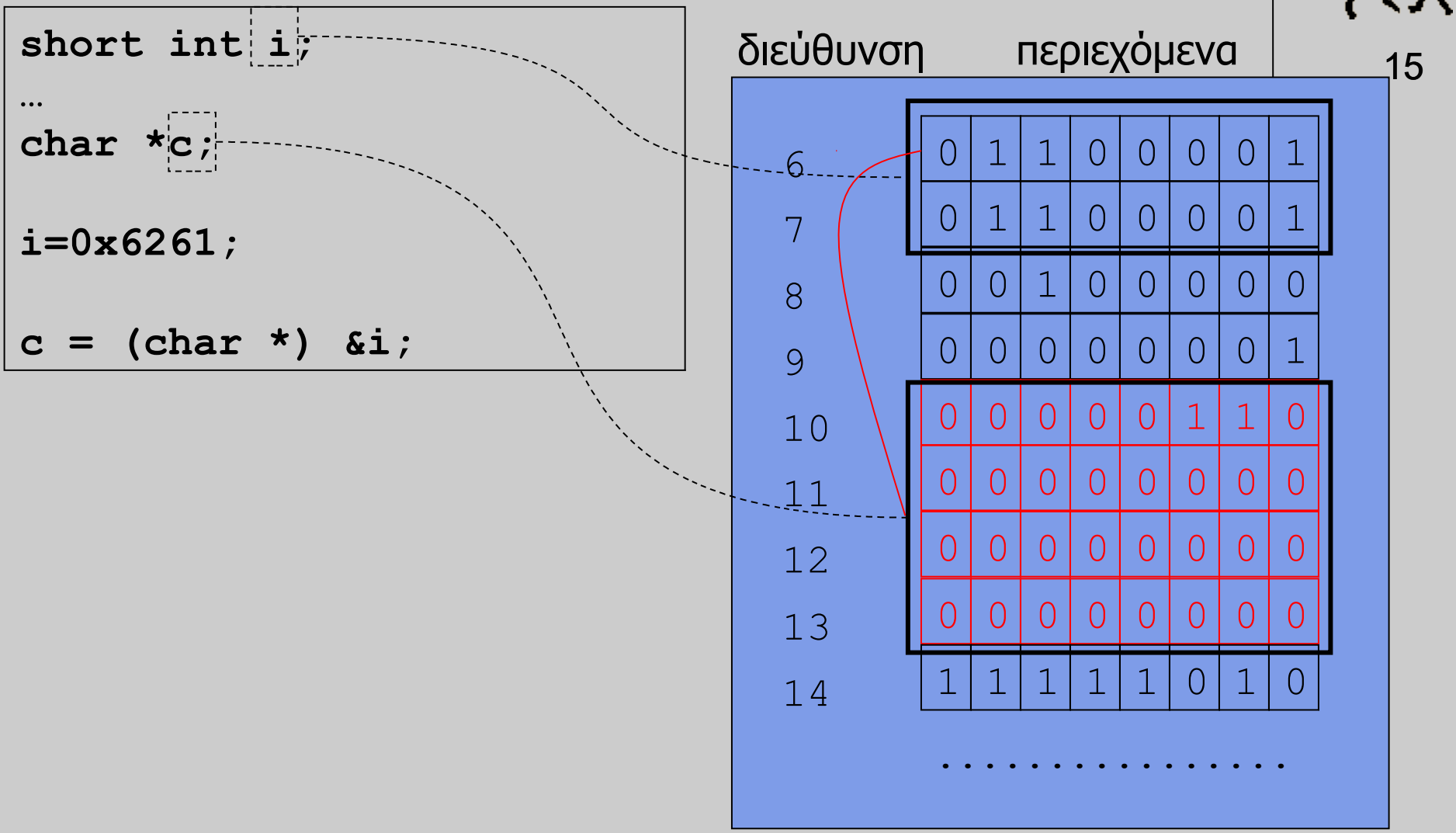
- Ο συντακτικός έλεγχος μπορεί να **παρακαμφθεί** με χρήση type casting (δηλωμένης μετατροπής τύπου).
  - Έτσι, μια **οποιαδήποτε ακέραια τιμή** να ανατεθεί (χωρίς μετατροπή) σε μια μεταβλητή δείκτη.
- Επιτρέπεται **αριθμητική με δείκτες**
  - π.χ. Αν `p` είναι μια μεταβλητή δείκτης-σε-Τ, τότε η έκφραση `p+i` μεταφράζεται (**αυτόματα**) ως `p+i*sizeof(T)`.
- Δεν υπάρχει τρόπος για τον μεταφραστή να ελέγξει αν μια τιμή που ανατίθεται σε μια μεταβλητή δείκτη-σε-Τ (μέσω type cast ή αριθμητικής) αντιστοιχεί πραγματικά στη διεύθυνση ενός αντικειμένου Τ.
  - **Την ευθύνη** ανάθεσης κατάλληλων τιμών σε μεταβλητές δείκτη **την έχει ο προγραμματιστής.**


```
short int a;        /* ακέραιος a (έστω 2 bytes) */

char *b;            /* δείκτης σε χαρακτήρα b */

b=(char*)&a;        /* b γίνεται διεύθυνση του a,
                       αλλά η πρόσβαση μέσω b γίνεται
                       με βάση τον τύπο char, ανά byte */

*b = *b+1;          /* το πρώτο byte του a αυξάνεται κατά 1 */

b++;                /* b γίνεται διεύθυνση του a + 1 */

*b = *b+1;          /* το δεύτερο byte του a αυξάνεται κατά 1 */

b++;                /* b γίνεται διεύθυνση του a + 2 */

*b = *b+1;          /* το τρίτο(;) byte του a αυξάνεται κατά 1 */
```

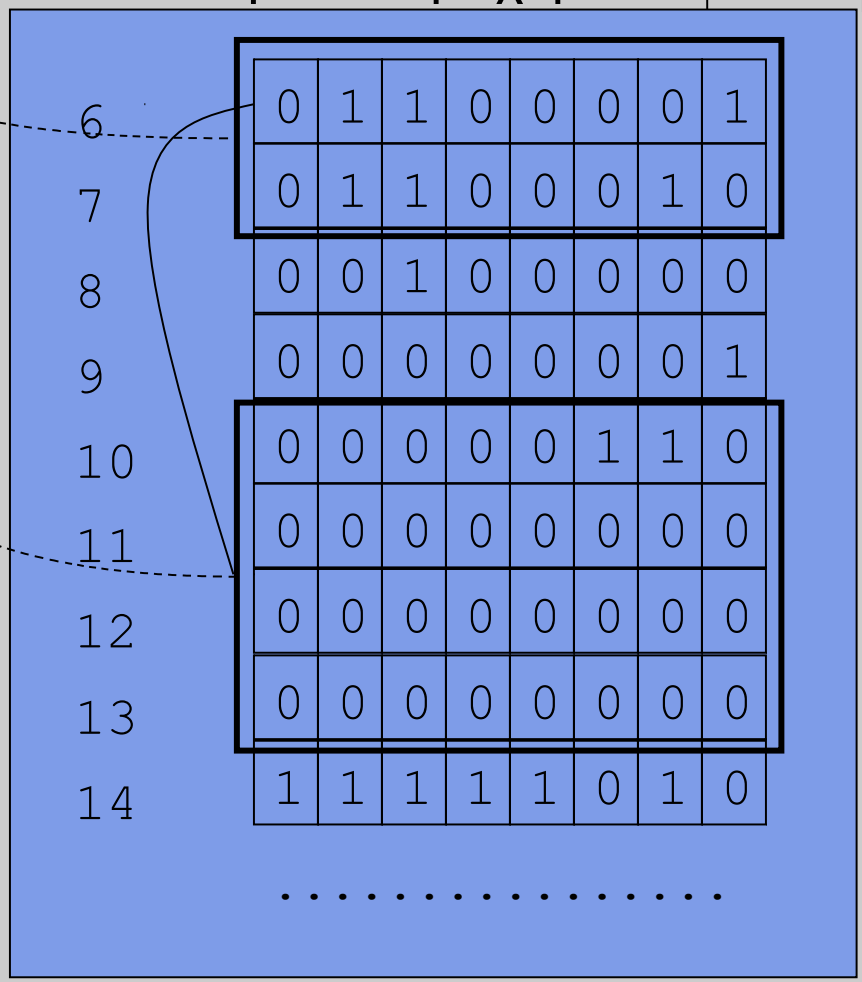

short int i;
…
char *c;
διεύθυνση
περιεχόμενα
6
7
8
9
10
11
12
13
14
0 1 1 0 0 1 0 0
0 1 1 0 0 0 1 1
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1
1 1 1 1 1 0 1 0
. . . . . .
υποθέτουμε μέγεθος short 2 bytes

```
short int i;
…
char *c;

i=0x6261;
```

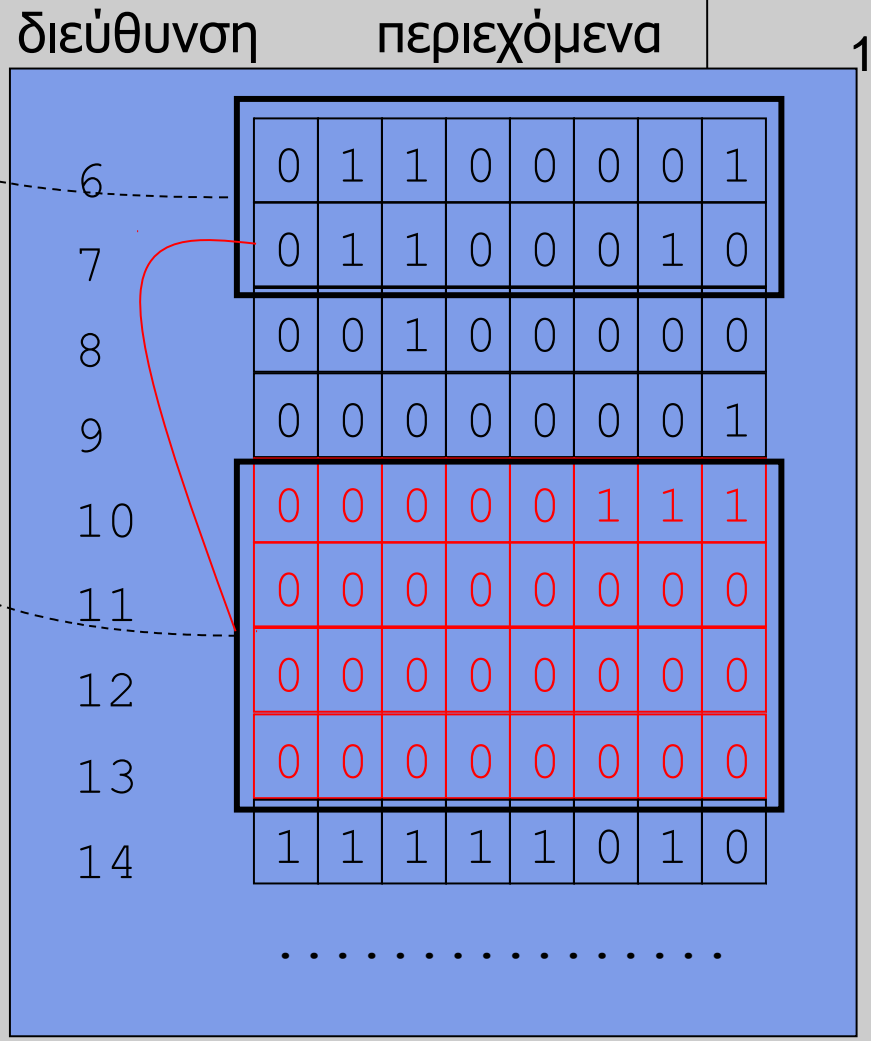



short int i;
…
char *c;
i=0x6261;
c = (char *) &i;
διεύθυνση
περιεχόμενα
6
7
8
9
10
11
12
13
14
0 1 1 0 0 0 0 1
0 1 1 0 0 0 0 1
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 1 1 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
1 1 1 1 1 0 1 0
. . . . . .

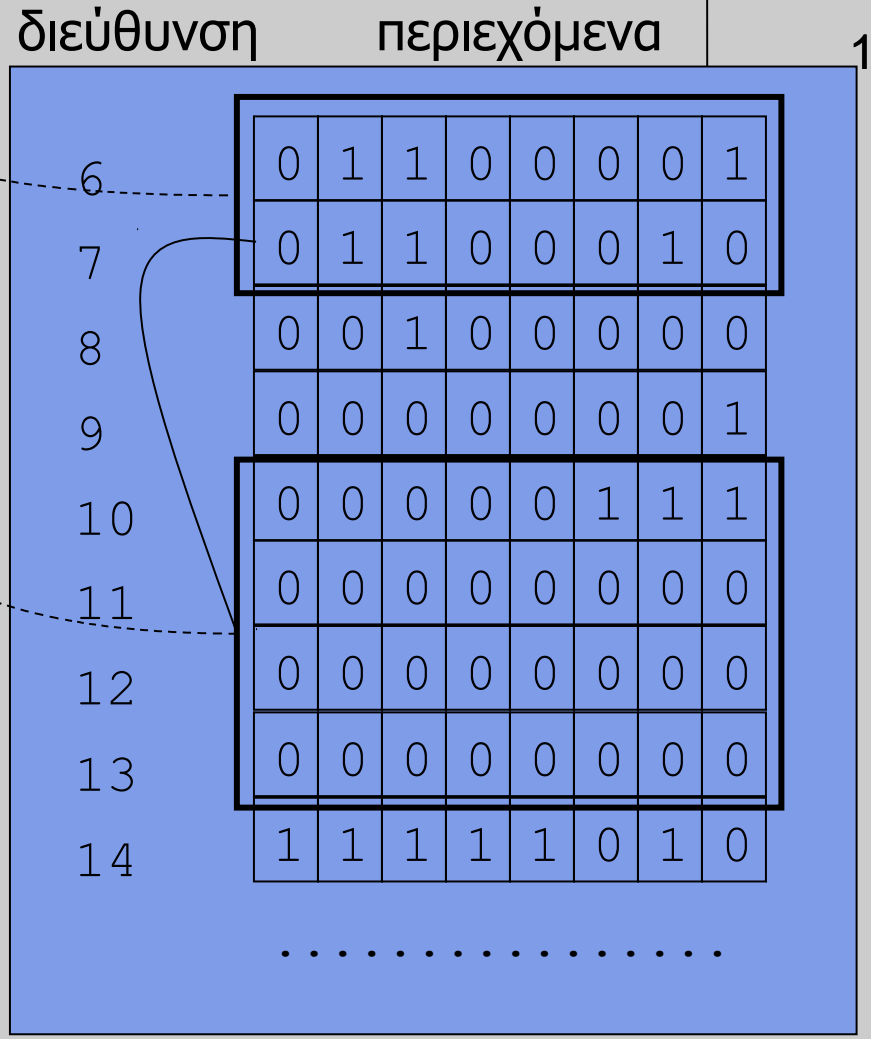
```
short int i;
…
char *c;

i=0x6261;

c = (char *) &i;

putchar(*c);  /* -> 'a' */
```

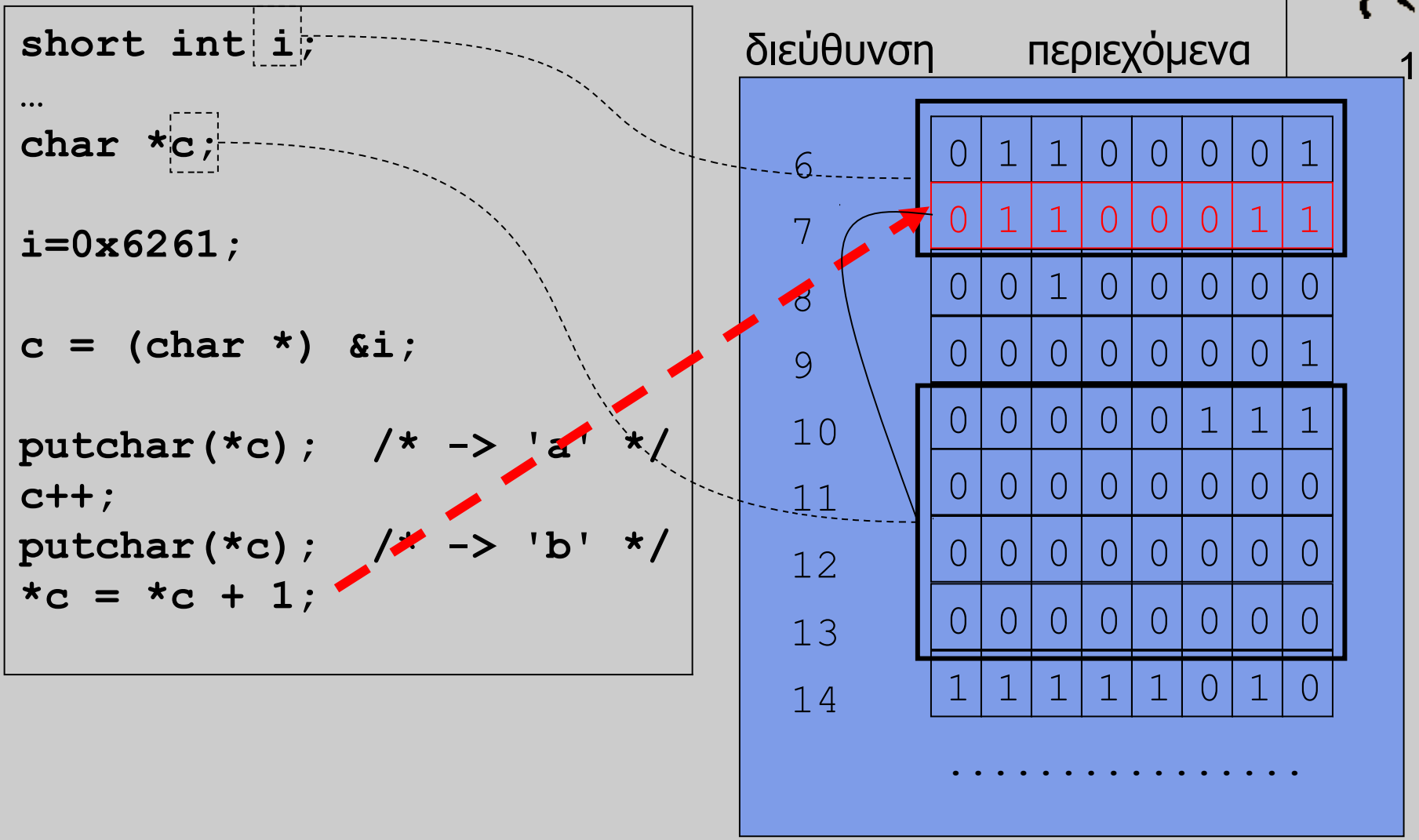


```
short int i;
…
char *c;

i=0x6261;

c = (char *) &i;

putchar(*c);  /* -> 'a' */
c++;
```

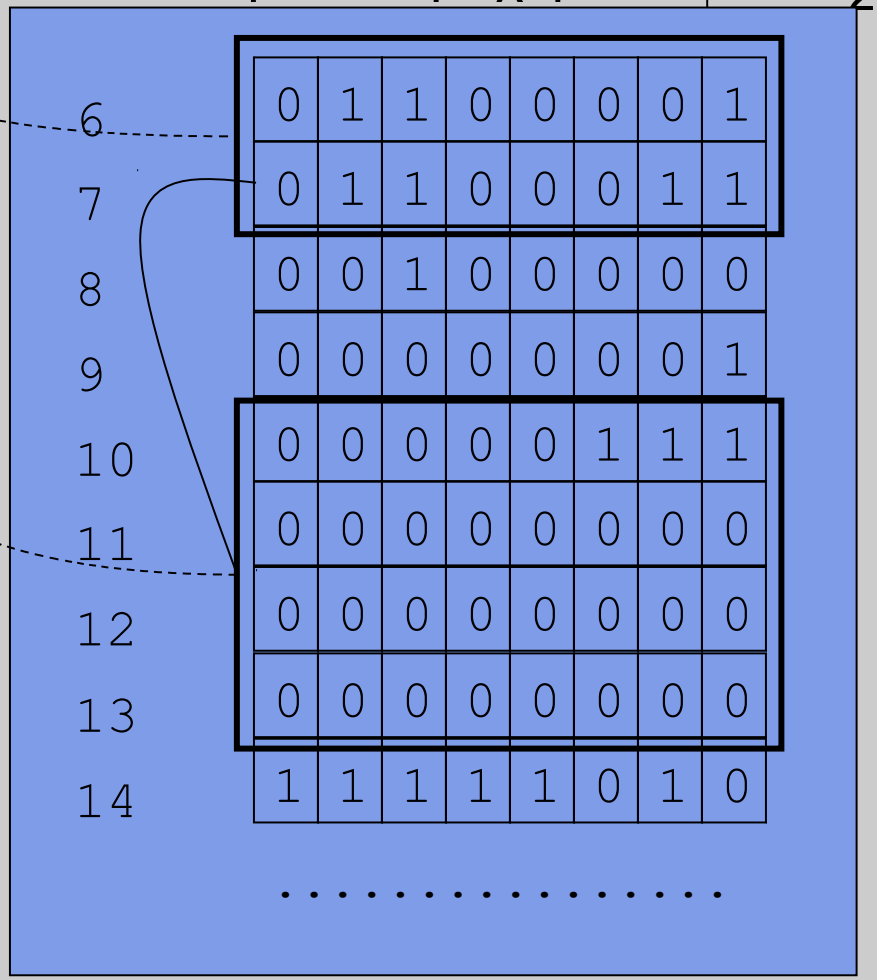


```
short int i;
…
char *c;

i=0x6261;

c = (char *) &i;

putchar(*c);  /* -> 'a' */
c++;
putchar(*c);  /* -> 'b' */
```

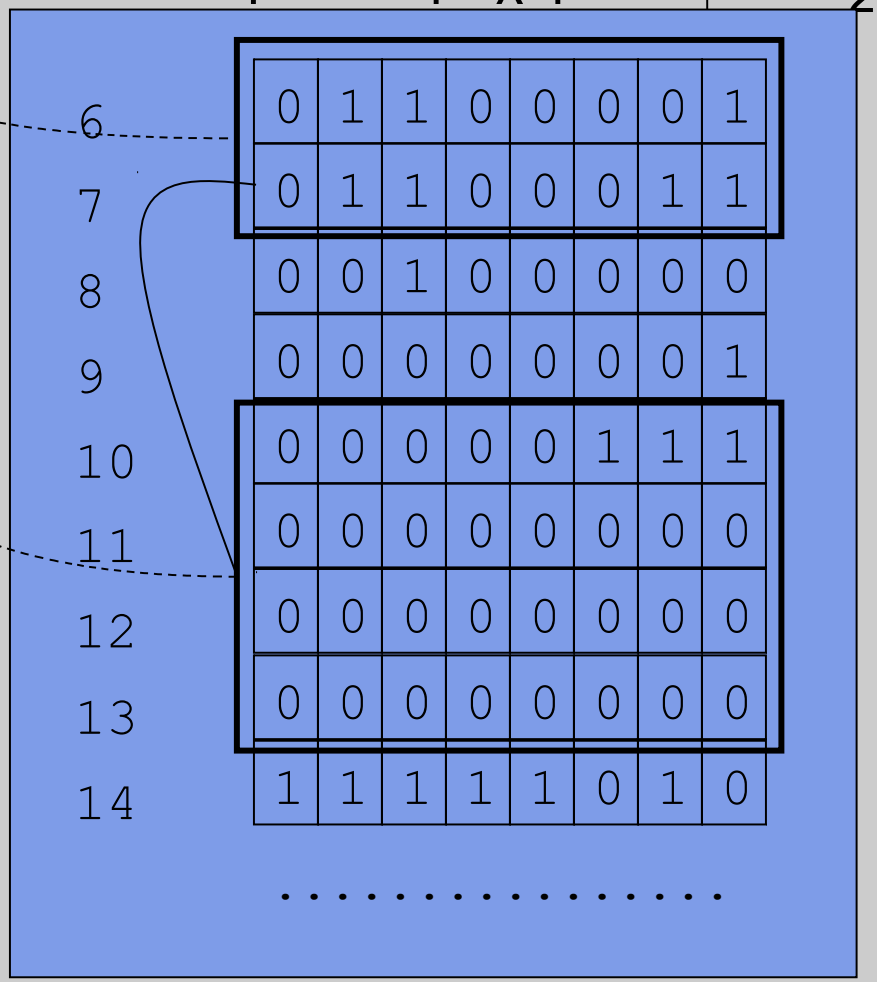



short int i;
…
char *c;
i=0x6261;
c = (char *) &i;
putchar(*c); /* -> 'a' */
c++;
putchar(*c); /* -> 'b' */
*c = *c + 1;
διεύθυνση
περιεχόμενα
6
7
8
9
10
11
12
13
14
0 1 1 0 0 0 0 1
0 1 1 0 0 0 1 1
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 1 1 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
1 1 1 1 1 0 1 0
.....

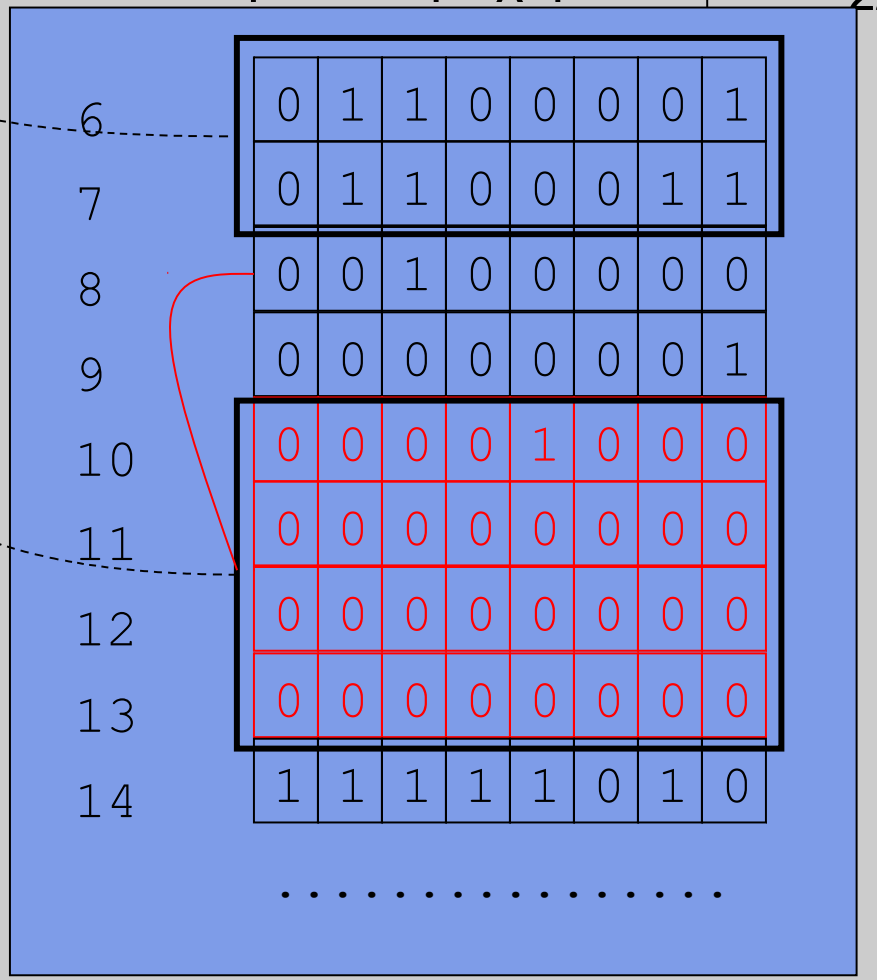
```
short int i;
…
char *c;

i=0x6261;

c = (char *) &i;

putchar(*c);  /* -> 'a' */
c++;
putchar(*c);  /* -> 'b' */
*c = *c + 1;
putchar(*c);  /* -> 'c' */
```



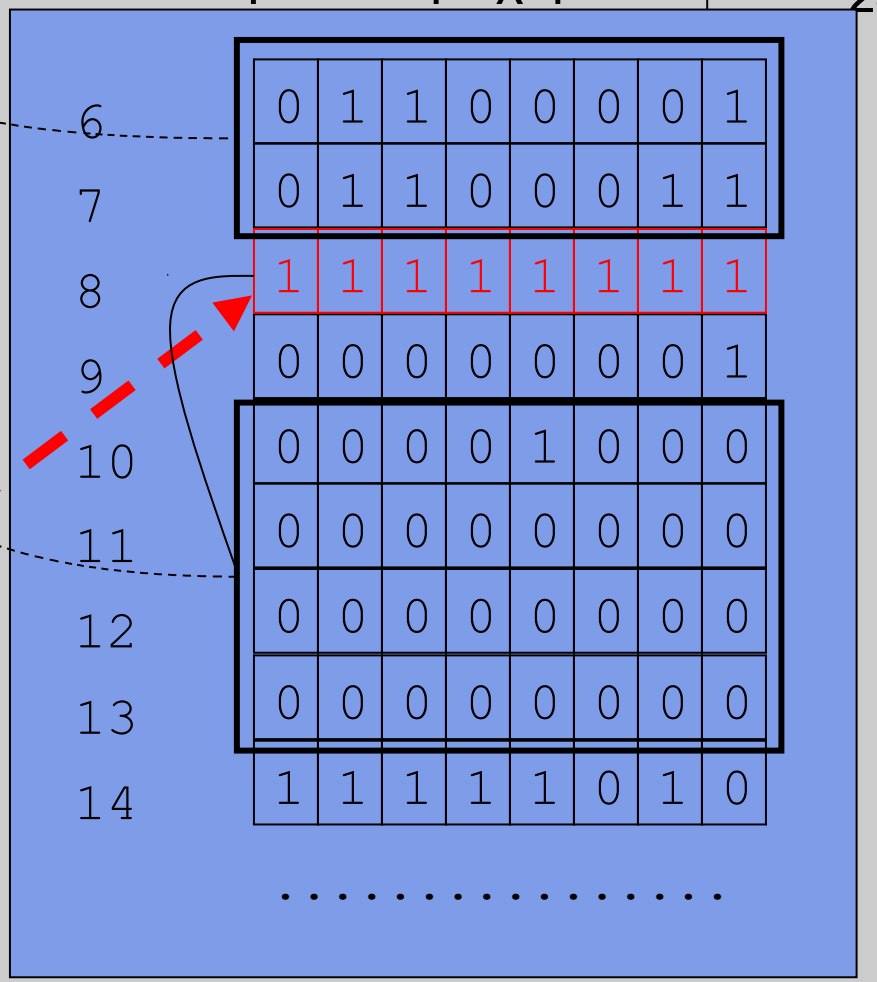
```
short int i;
…
char *c;

i=0x6261;

c = (char *) &i;

putchar(*c);  /* -> 'a' */
c++;
putchar(*c);  /* -> 'b' */
*c = *c + 1;
putchar(*c);  /* -> 'c' */
printf("%x",i); /* 6361 */
```



```
short int i;
…
char *c;

i=0x6261;

c = (char *) &i;

putchar(*c);  /* -> 'a' */
c++;
putchar(*c);  /* -> 'b' */
*c = *c + 1;
putchar(*c);  /* -> 'c' */
printf("%x",i); /* 6361 */
c++;
```



```
short int i;
…
char *c;

i=0x6261;

c = (char *) &i;

putchar(*c);  /* -> 'a' */
c++;
putchar(*c);  /* -> 'b' */
*c = *c + 1;
putchar(*c);  /* -> 'c' */
printf("%x",i); /* 6361 */
c++;
*c = 0xFF;
```



Προσοχή

- Η ανάθεση τιμών σε μεταβλητές δείκτες γίνεται με **αποκλειστική** ευθύνη του προγραμματιστή
  - Χωρίς να γίνεται κάποιος (ολοκληρωμένος) έλεγχος από τον μεταφραστή.
  - Το **παραμικρό** λάθος όταν χρησιμοποιούμε δείκτες, μπορεί να οδηγήσει σε **απρόβλεπτα** αποτελέσματα.
- Χωρίς να το επιθυμούμε (ούτε να το αντιληφθούμε) μπορεί να αλλάξουμε κατά λάθος τιμές μεταβλητών, ακόμα και μεταβλητών-δεικτών ...
- Όπως και στην παράβαση των ορίων ενός πίνακα, τέτοια σφάλματα **δύσκολα** εντοπίζονται
  - Αν έχουμε τύχη, τερματίζεται η εκτέλεση του προγράμματος.