

Προγραμματισμός I (HY120)

Διάλεξη 10:
Δείκτες





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;  
  
int *b,*c;  
  
a = 1;  
  
b = &a;  
  
*b = 2;  
  
*b = *b+1;  
  
c = b;  
  
*c = *c+2;
```

```
/* ακέραιος a */  
  
/* δείκτες σε ακέραιο b,c */  
  
/* b γίνεται διεύθυνση του a */  
  
/* a γίνεται 2 */  
  
/* a γίνεται 3 */  
  
/* c γίνεται διεύθυνση του a */  
  
/* a γίνεται 5 */
```



διεύθυνση περιεχόμενα

```
...  
char a;
```

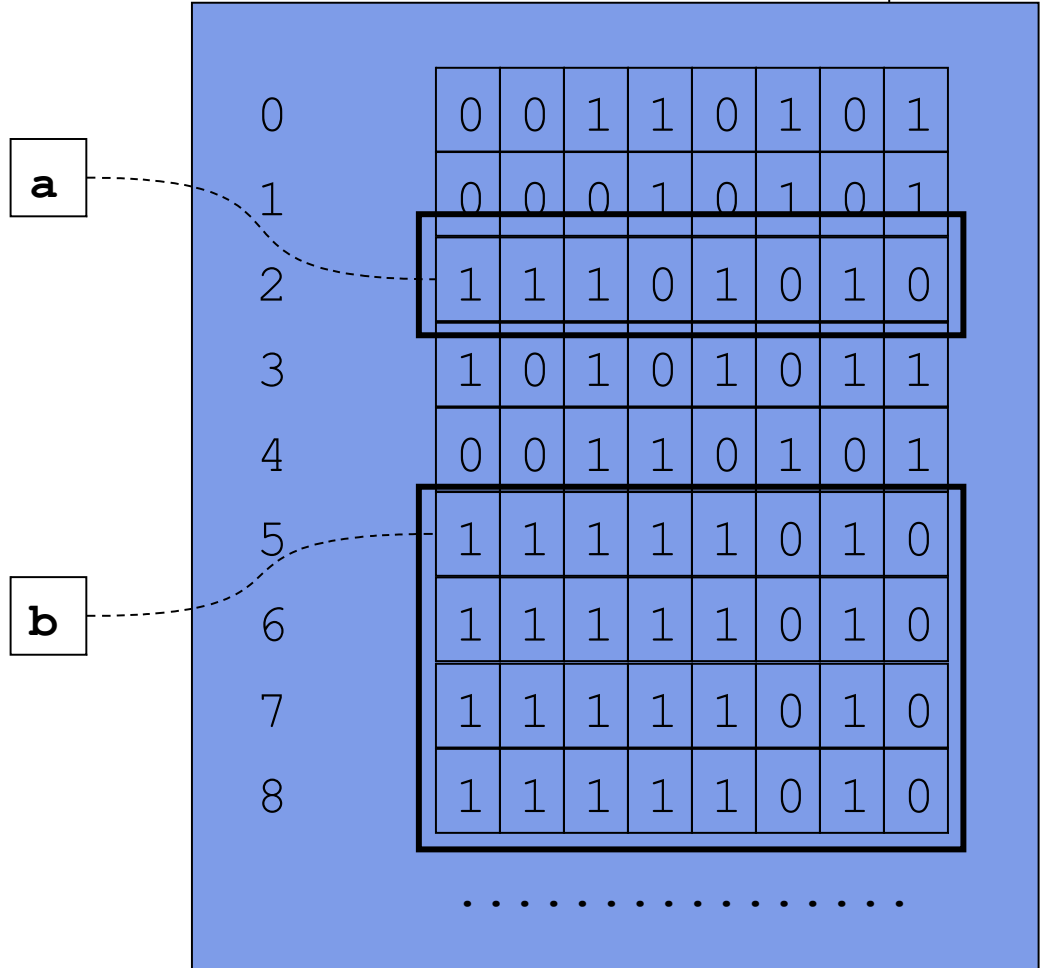
a

0	0	0	1	1	0	1	0	1
1	0	0	0	1	0	1	0	1
2	1	1	1	0	1	0	1	0
3	1	0	1	0	1	0	1	1
4	0	0	1	1	0	1	0	1
5	1	1	1	1	1	0	1	0
6	1	1	1	1	1	0	1	0
7	1	1	1	1	1	0	1	0
8	1	1	1	1	1	0	1	0
.....								



```
...  
char a;  
...  
char *b;
```

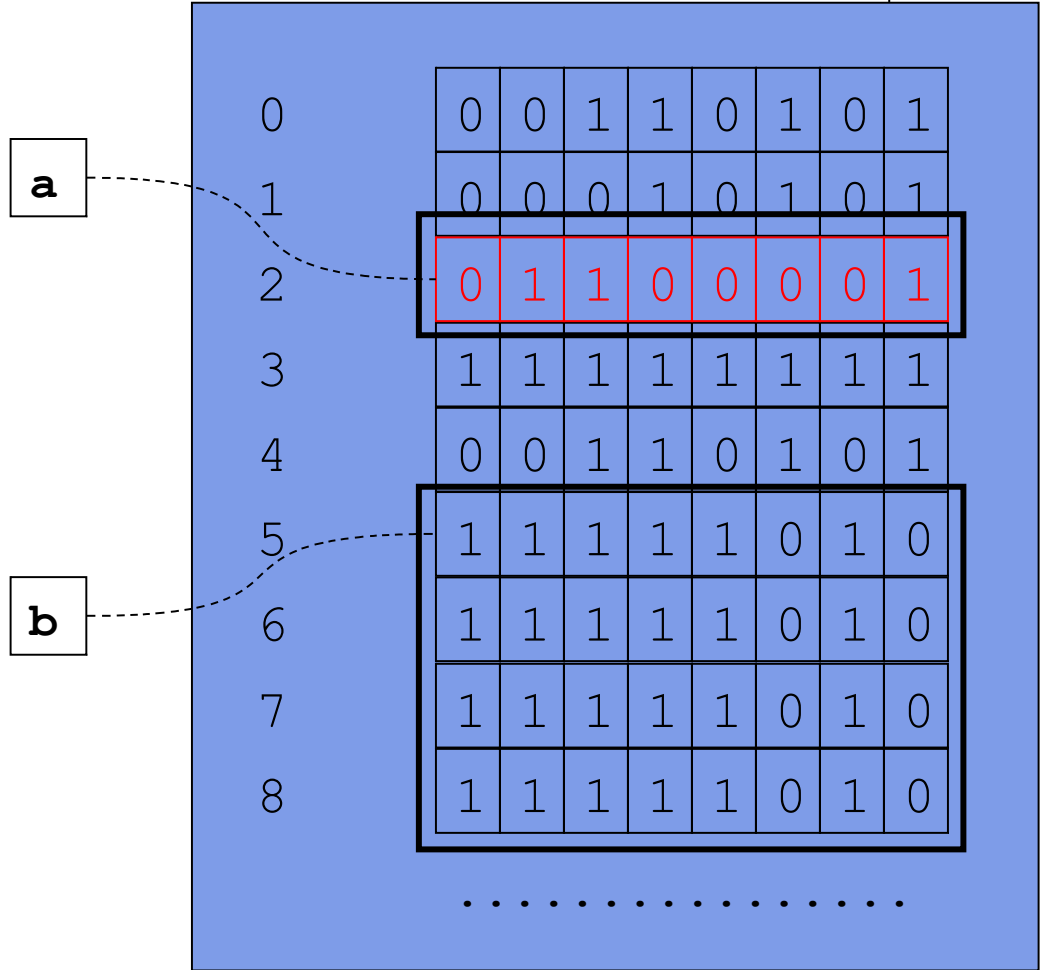
διεύθυνση περιεχόμενα





διεύθυνση περιεχόμενα

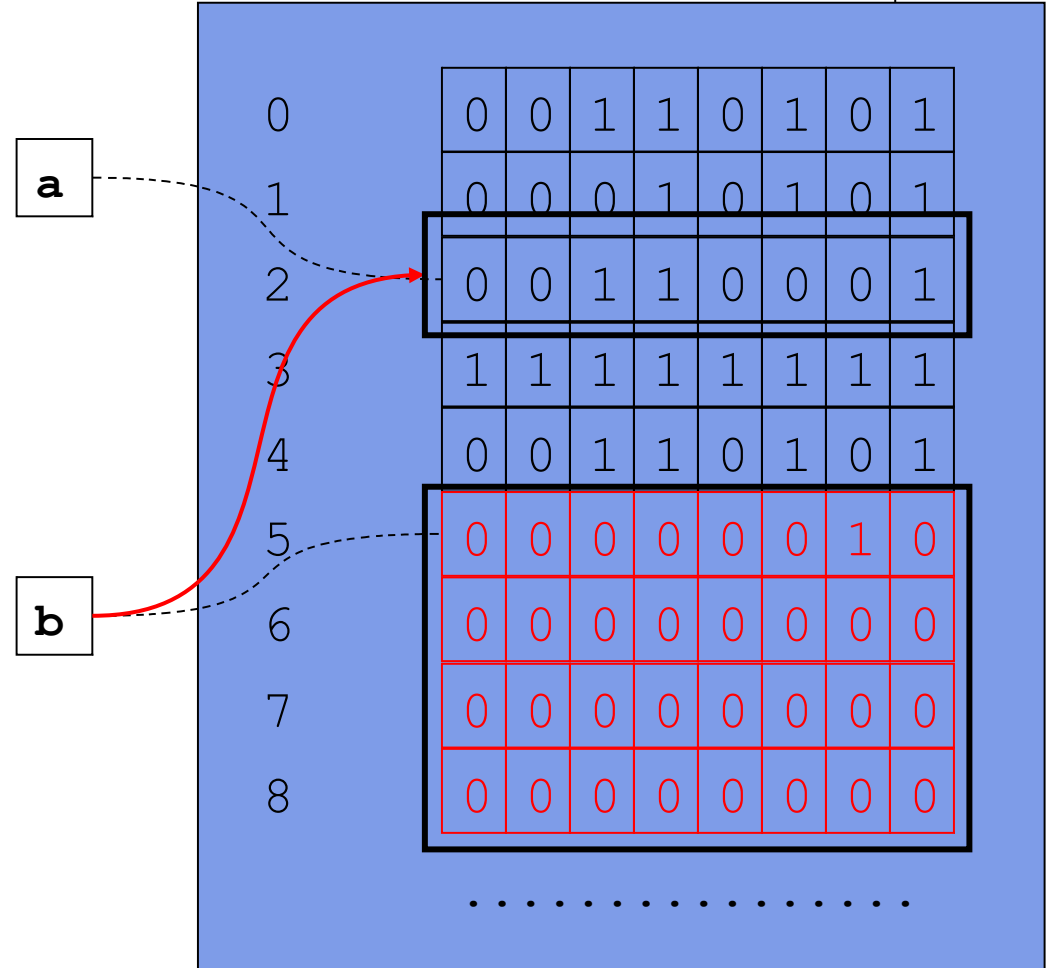
```
...  
char a;  
...  
char *b;  
a='a';
```





διεύθυνση περιεχόμενα

```
...  
char a;  
...  
char *b;  
  
a='a';  
  
b=&a;
```



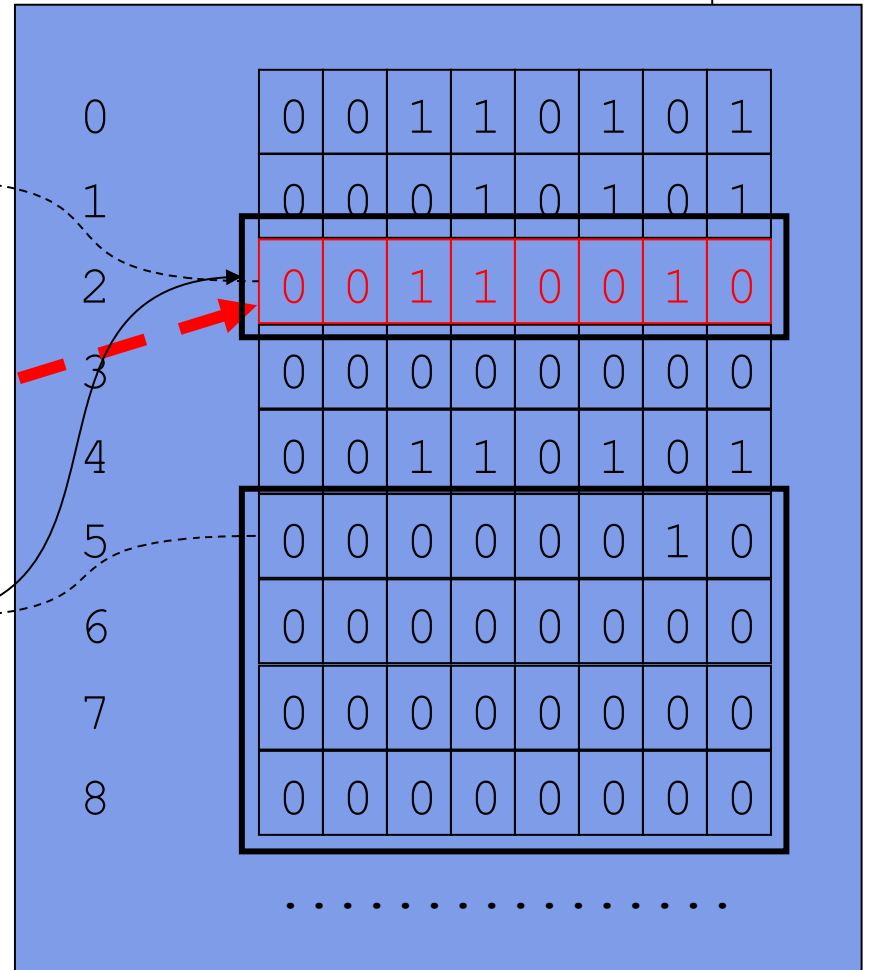


διεύθυνση περιεχόμενα

```
...  
char a;  
...  
char *b;  
  
a='a';  
  
b=&a;  
  
*b='b';
```

a

b



Παράκαμψη συντακτικού ελέγχου



11

- 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-`T`, 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-`T` (μέσω `type cast` ή αριθμητικής) αντιστοιχεί πραγματικά στη διεύθυνση ενός αντικειμένου `T`.
 - Την ευθύνη ανάθεσης κατάλληλων τιμών σε μεταβλητές δείκτη την έχει ο προγραμματιστής.



```
short int a;
```

```
/* ακέραιος a (έστω 2 bytes) */
```

12

```
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 */
```



13

```
short int i;  
...  
char *c;
```

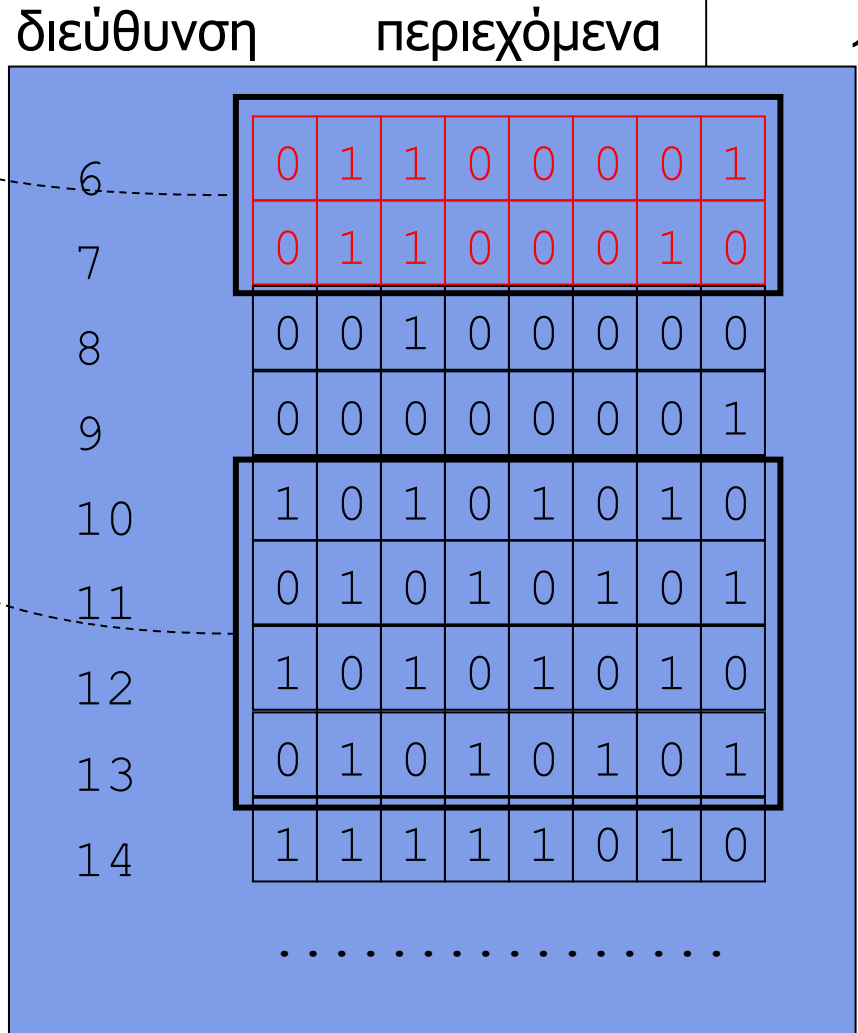
διεύθυνση περιεχόμενα

6	0	1	1	0	0	1	0	0
7	0	1	1	0	0	0	1	1
8	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	1
10	1	0	1	0	1	0	1	0
11	0	1	0	1	0	1	0	1
12	1	0	1	0	1	0	1	0
13	0	1	0	1	0	1	0	1
14	1	1	1	1	1	0	1	0
							

υποθέτουμε μέγεθος short 2 bytes



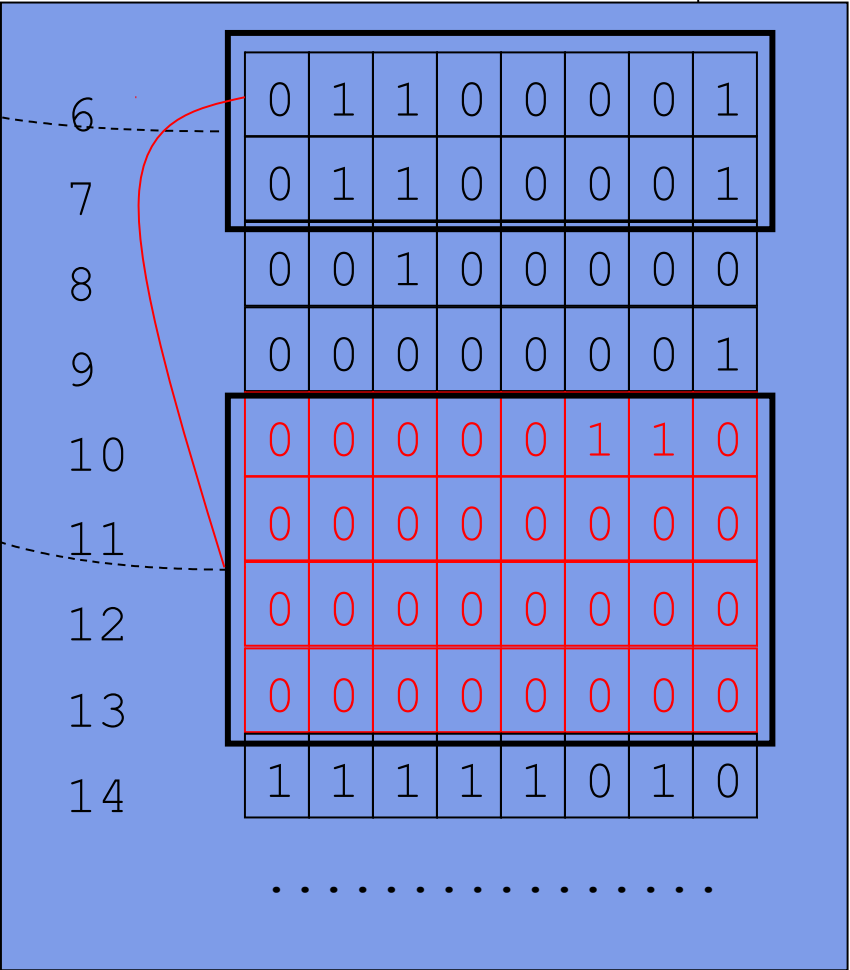
```
short int i;  
...  
char *c;  
  
i=0x6261;
```





```
short int i;  
...  
char *c;  
i=0x6261;  
c = (char *) &i;
```

διεύθυνση περιεχόμενα





```
short int i;
```

```
...
```

```
char *c;
```

```
i=0x6261;
```

```
c = (char *) &i;
```

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

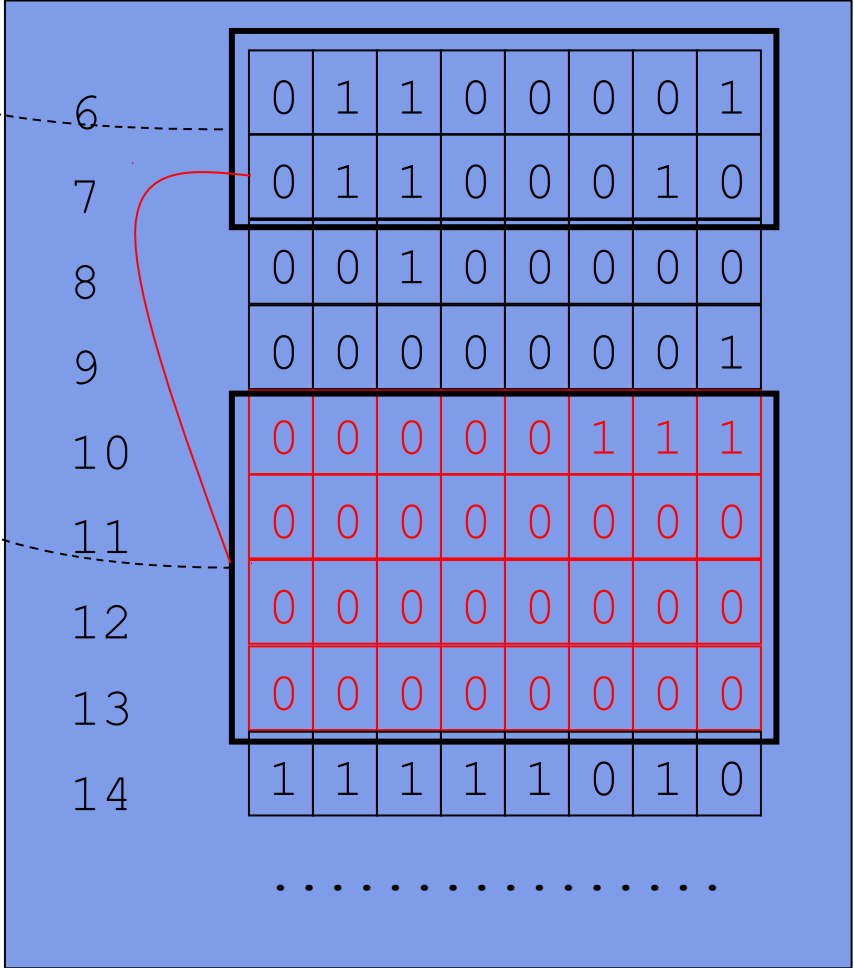
διεύθυνση περιεχόμενα

6	0	1	1	0	0	0	0	1
7	0	1	1	0	0	0	1	0
8	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	1
10	0	0	0	0	0	1	1	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	1	1	1	1	1	0	1	0
							



```
short int i;  
...  
char *c;  
  
i=0x6261;  
  
c = (char *) &i;  
  
putchar(*c); /* -> 'a' */  
c++;
```

διεύθυνση περιεχόμενα





```
short int i;  
...  
char *c;  
  
i=0x6261;  
  
c = (char *) &i;  
  
putchar(*c); /* -> 'a' */  
c++;  
putchar(*c); /* -> 'b' */
```

διεύθυνση περιεχόμενα

6	0	1	1	0	0	0	0	1
7	0	1	1	0	0	0	1	0
8	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	1
10	0	0	0	0	0	1	1	1
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	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;
```

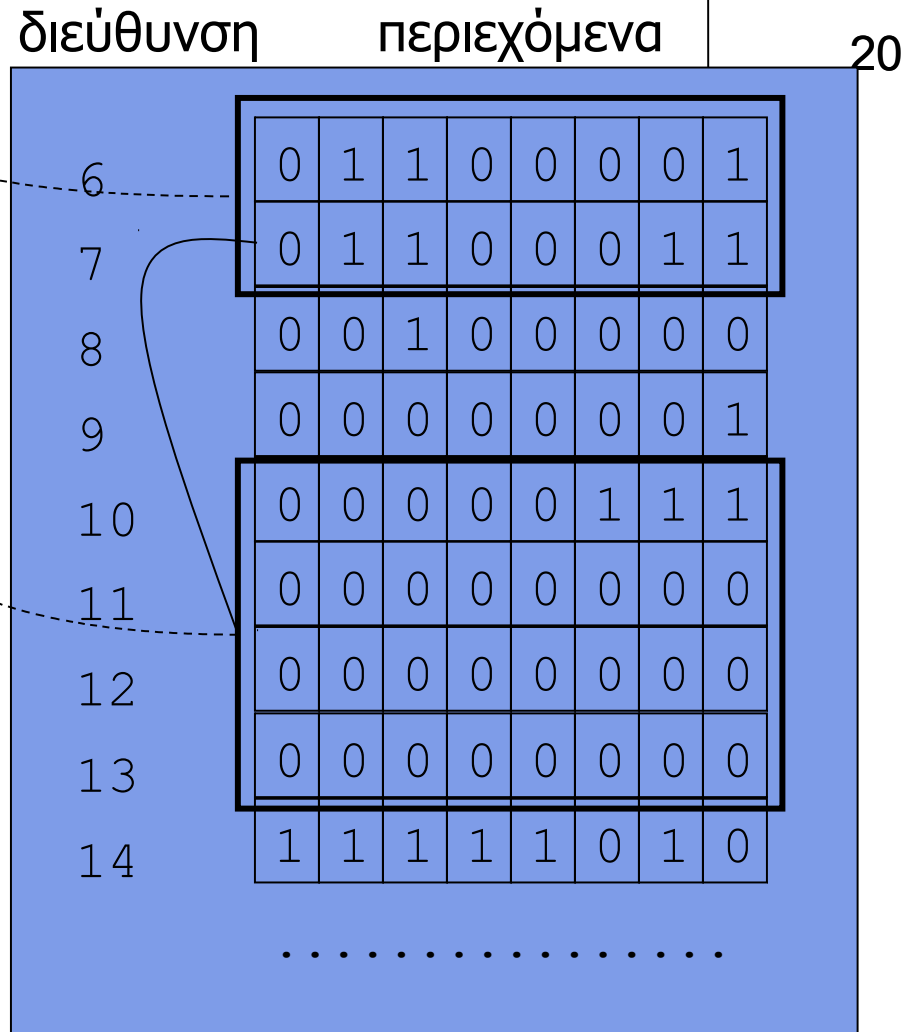
διεύθυνση

περιεχόμενα

6	0	1	1	0	0	0	0	1
7	0	1	1	0	0	0	1	1
8	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	1
10	0	0	0	0	0	1	1	1
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	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 */
```

διεύθυνση περιεχόμενα

6	0	1	1	0	0	0	0	1
7	0	1	1	0	0	0	1	1
8	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	1
10	0	0	0	0	0	1	1	1
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	1	1	1	1	1	0	1	0
.....								



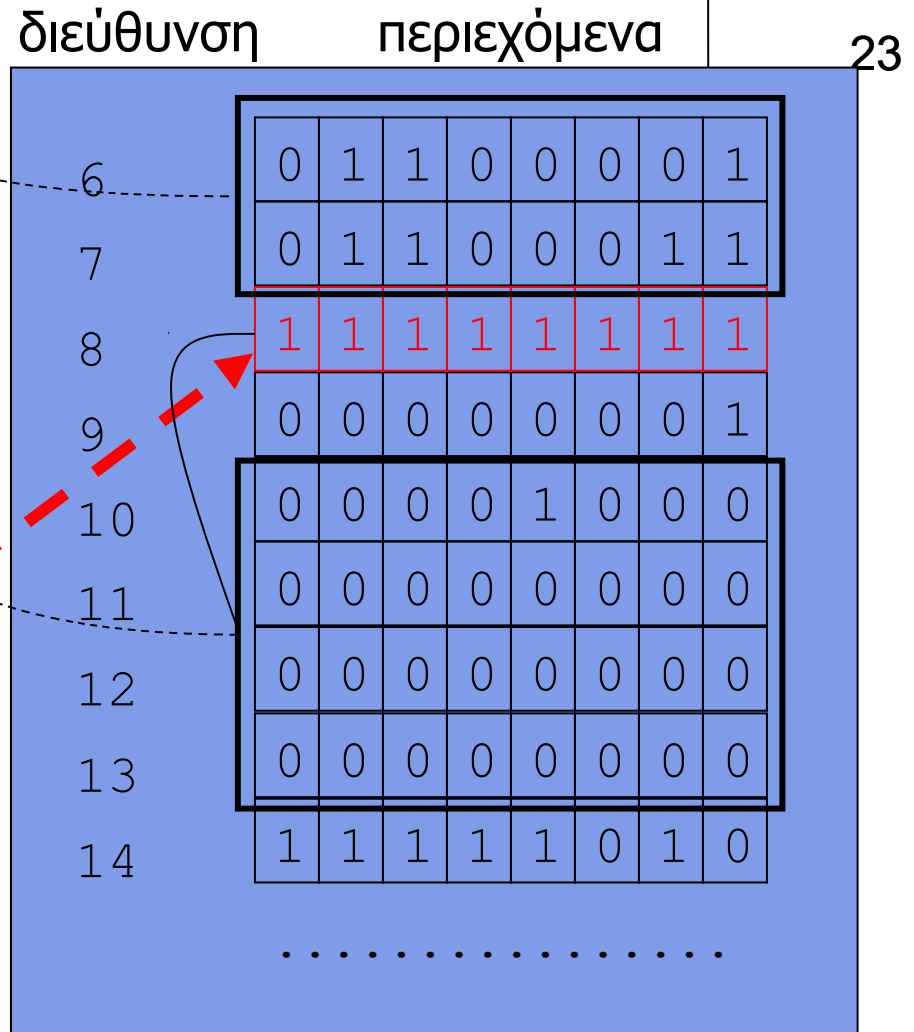
```
short int i;  
...  
char *c;  
  
i=0x6261;  
  
c = (char *) &i;  
  
putchar(*c); /* -> 'a' */  
c++;  
putchar(*c); /* -> 'b' */  
*c = *c + 1;  
putchar(*c); /* -> 'c' */  
printf("%x", i); /* 6361 */  
c++;
```

διεύθυνση περιεχόμενα

6	0	1	1	0	0	0	0	1
7	0	1	1	0	0	0	1	1
8	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	1
10	0	0	0	0	1	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	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' */  
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.