Προγραμματισμός Ι (ECE115)

#9 δείκτες (pointers)

Χρήστος Αντωνόπουλος
cda@uth.gr

Δείκτες

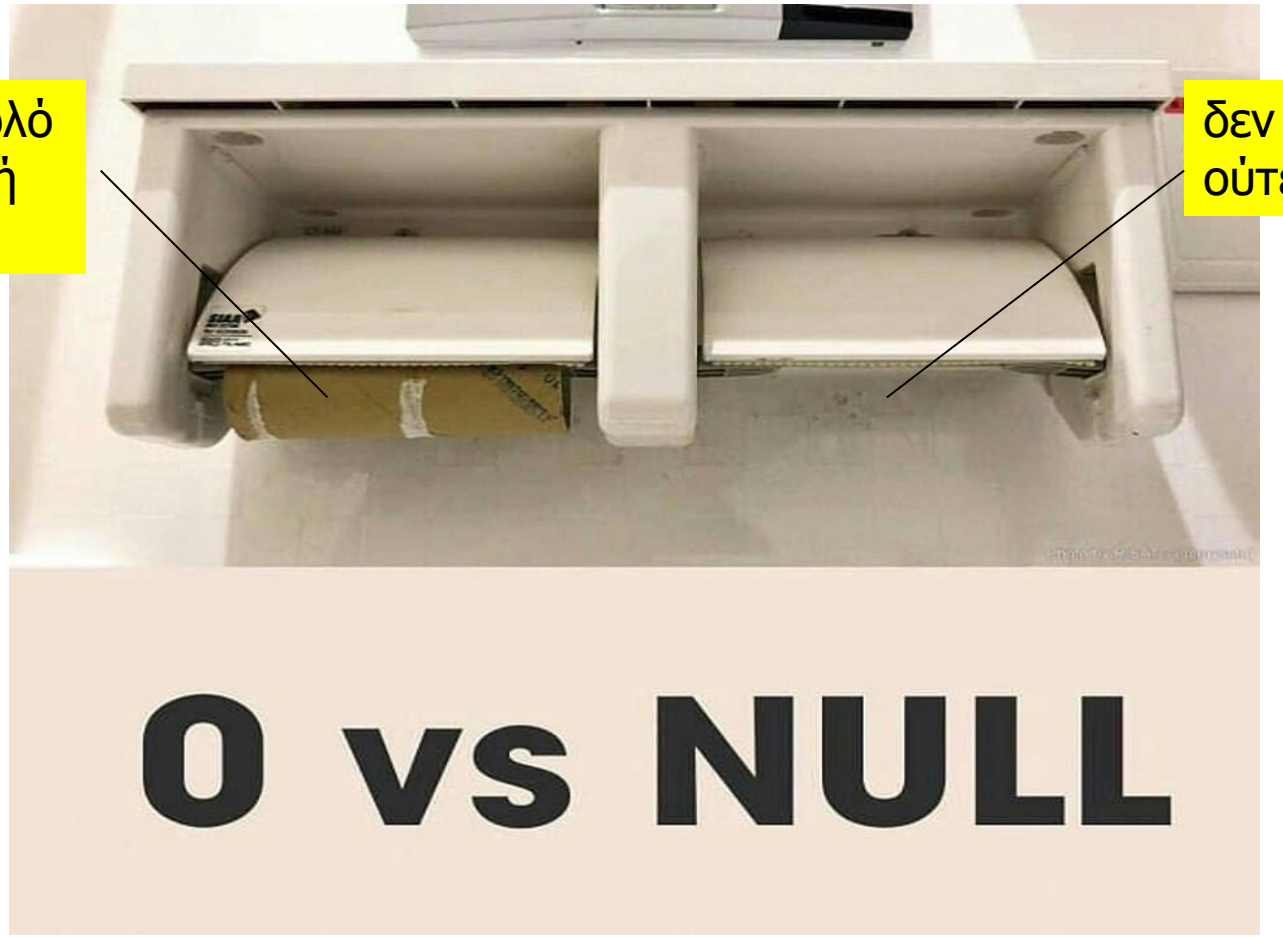
- Υπάρχουν περιπτώσεις που δεν ενδιαφέρει το περιεχόμενο αλλά η **διεύθυνση** μιας μεταβλητής
  - θέση αποθήκευσης στην μνήμη του προγράμματος

- Υπάρχουν μεταβλητές ειδικού τύπου: **δείκτες**
- Στην ουσία, μια **ακέραια τιμή** που **ερμηνεύεται** σαν μια **διεύθυνση** στην μνήμη

- Μέσω ενός δείκτη μπορούμε να **διαβάσουμε** ή/και να **αλλάξουμε** την τιμή που βρίσκεται αποθηκευμένη στην αντίστοιχη διεύθυνση
  - που μπορεί να αντιστοιχεί στην διεύθυνση μιας μεταβλητής!

Δείκτες και τύποι δεδομένων

- Η έννοια του δείκτη έχει ισχύ για **κάθε** τύπο `T`
- Ουσιαστικά ορίζει τον τύπο `pointer-to-T`
- Υπάρχει **συντακτική** συμβατότητα ανάμεσα στην τιμή μιας μεταβλητής `pointer-to-T` και την **διεύθυνση** μιας μεταβλητής τύπου `T`
- Υπάρχει **συντακτική** συμβατότητα ανάμεσα στην τιμή που υπάρχει στην μνήμη όπου **δείχνει** μια μεταβλητή `pointer-to-T` και την τιμή τύπου `T`
- Η ειδική τιμή `NULL` σημαίνει «καμία διεύθυνση»
  - συνηθισμένη τιμή αρχικοποίησης μιας μεταβλητής δείκτη
  - η προσπάθεια προσπέλασης της διεύθυνσης `NULL` οδηγεί στον **τερματισμό** του προγράμματος

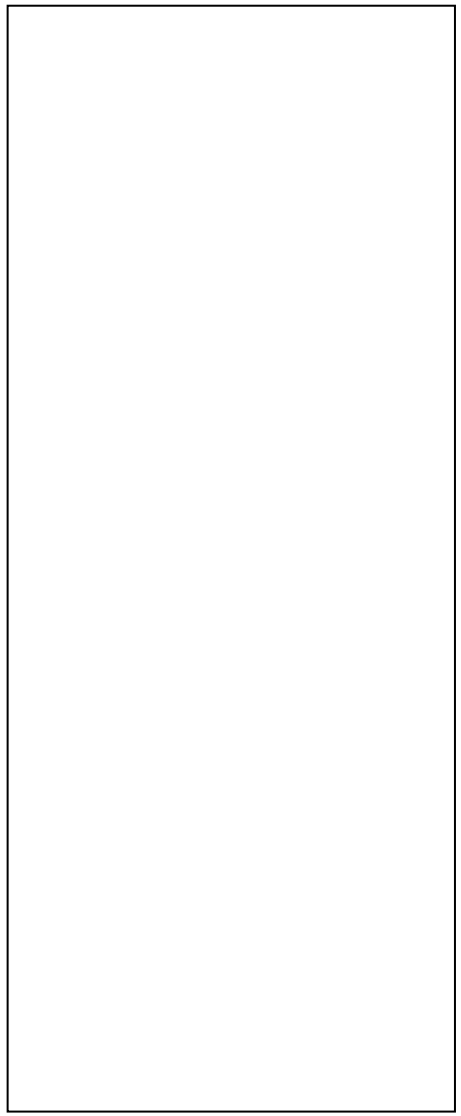

υπάρχει ρολό αλλά η τιμή του είναι 0
δεν υπάρχει ούτε καν ρολό
0 vs NULL

Τελεστές δεικτών

- Ο τελεστής **&** (reference) εφαρμόζεται σε μια μεταβλητή τύπου `T` και επιστρέφει την **διεύθυνση**, που είναι μια τιμή τύπου `pointer-to-T`
- Ο τελεστής ***** (de-reference) εφαρμόζεται σε μια μεταβλητή τύπου `pointer-to-T` έτσι ώστε να **προσπελαστεί** η μνήμη στην εν λόγω διεύθυνση
  - μπορεί να χρησιμοποιηθεί σε **οποιαδήποτε** έκφραση αποτίμησης ή/και ανάθεσης όπου αναμένεται ο τύπος `T`
- Μέσω ενός δείκτη μπορούμε να προσπελάσουμε **έμμεσα** τις θέσεις μνήμης μιας μεταβλητής
  - **χωρίς** να αναφερθούμε ρητά στο όνομα της
- Αυτό είναι μια μορφή **παρενέργειας!**
  - η κακή χρήση δεικτών οδηγεί σε δυσνόητο κώδικα

Το παράδειγμα που ακολουθεί, υποθέτει

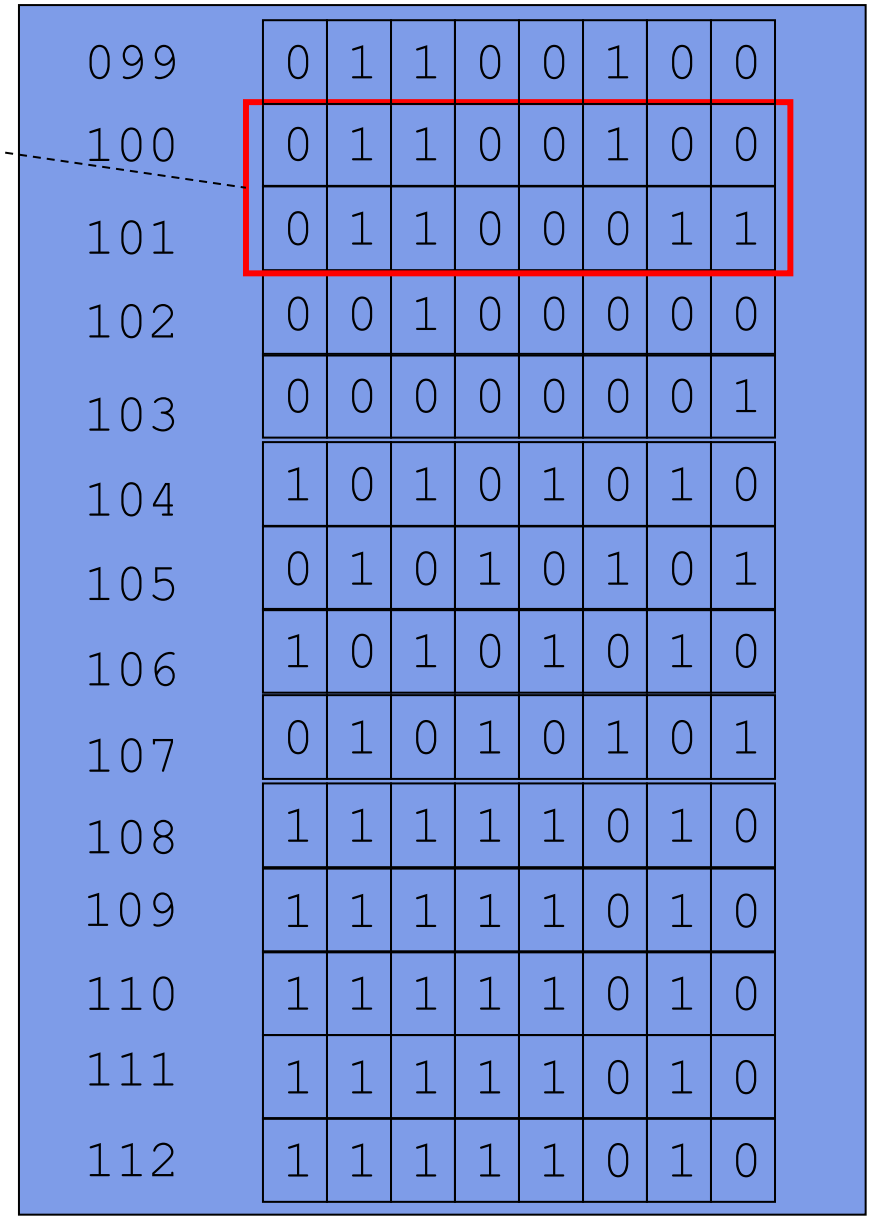
- μέγεθος `short int` 2 bytes
- μέγεθος δεικτών/pointers 4 bytes
- αποθήκευση αριθμητικών τιμών στην μνήμη με την σύμβαση little endian
  - τα λιγότερο σημαντικά bytes ενός multi-byte ακεραίου αποθηκεύονται στις μικρότερες διευθύνσεις μνήμης



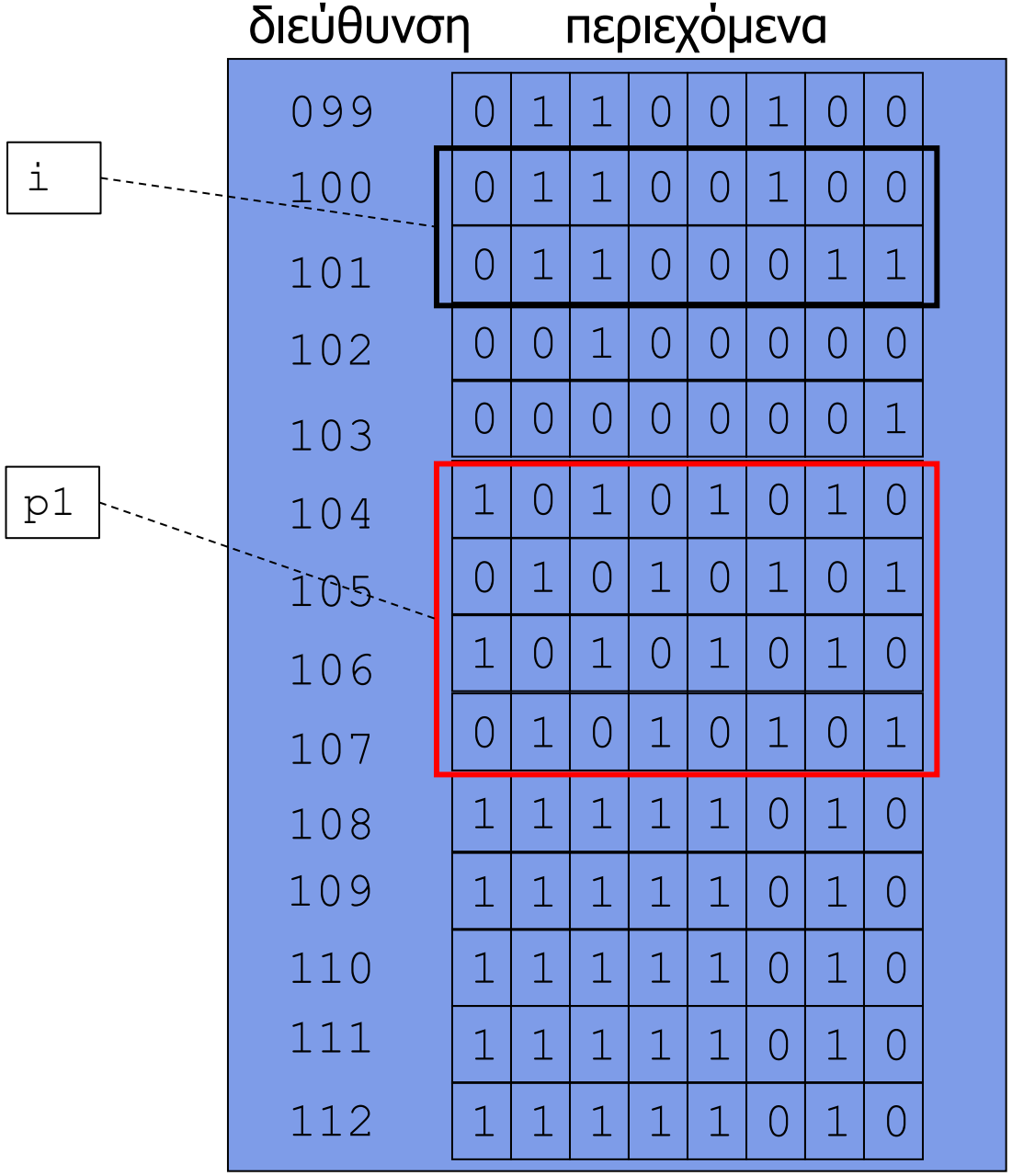
διεύθυνση	περιεχόμενα							
099	0	1	1	0	0	1	0	0
100	0	1	1	0	0	1	0	0
101	0	1	1	0	0	0	1	1
102	0	0	1	0	0	0	0	0
103	0	0	0	0	0	0	0	1
104	1	0	1	0	1	0	1	0
105	0	1	0	1	0	1	0	1
106	1	0	1	0	1	0	1	0
107	0	1	0	1	0	1	0	1
108	1	1	1	1	1	0	1	0
109	1	1	1	1	1	0	1	0
110	1	1	1	1	1	0	1	0
111	1	1	1	1	1	0	1	0
112	1	1	1	1	1	0	1	0

```
short int i;
```

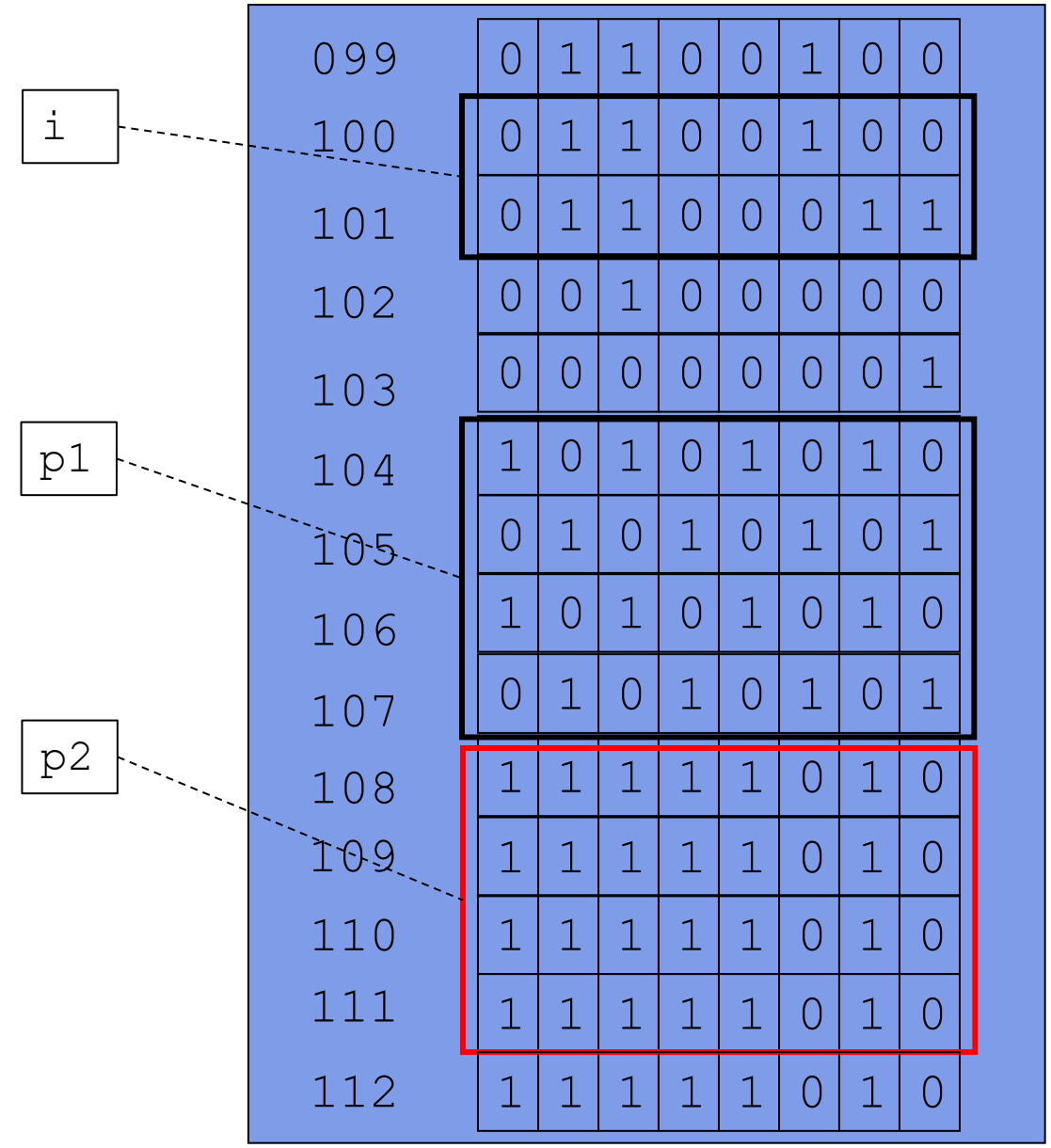
i


διεύθυνση
περιεχόμενα
099 0 1 1 0 0 1 0 0
100 0 1 1 0 0 1 0 0
101 0 1 1 0 0 0 1 1
102 0 0 1 0 0 0 0 0
103 0 0 0 0 0 0 0 1
104 1 0 1 0 1 0 1 0
105 0 1 0 1 0 1 0 1
106 1 0 1 0 1 0 1 0
107 0 1 0 1 0 1 0 1
108 1 1 1 1 1 0 1 0
109 1 1 1 1 1 0 1 0
110 1 1 1 1 1 0 1 0
111 1 1 1 1 1 0 1 0
112 1 1 1 1 1 0 1 0

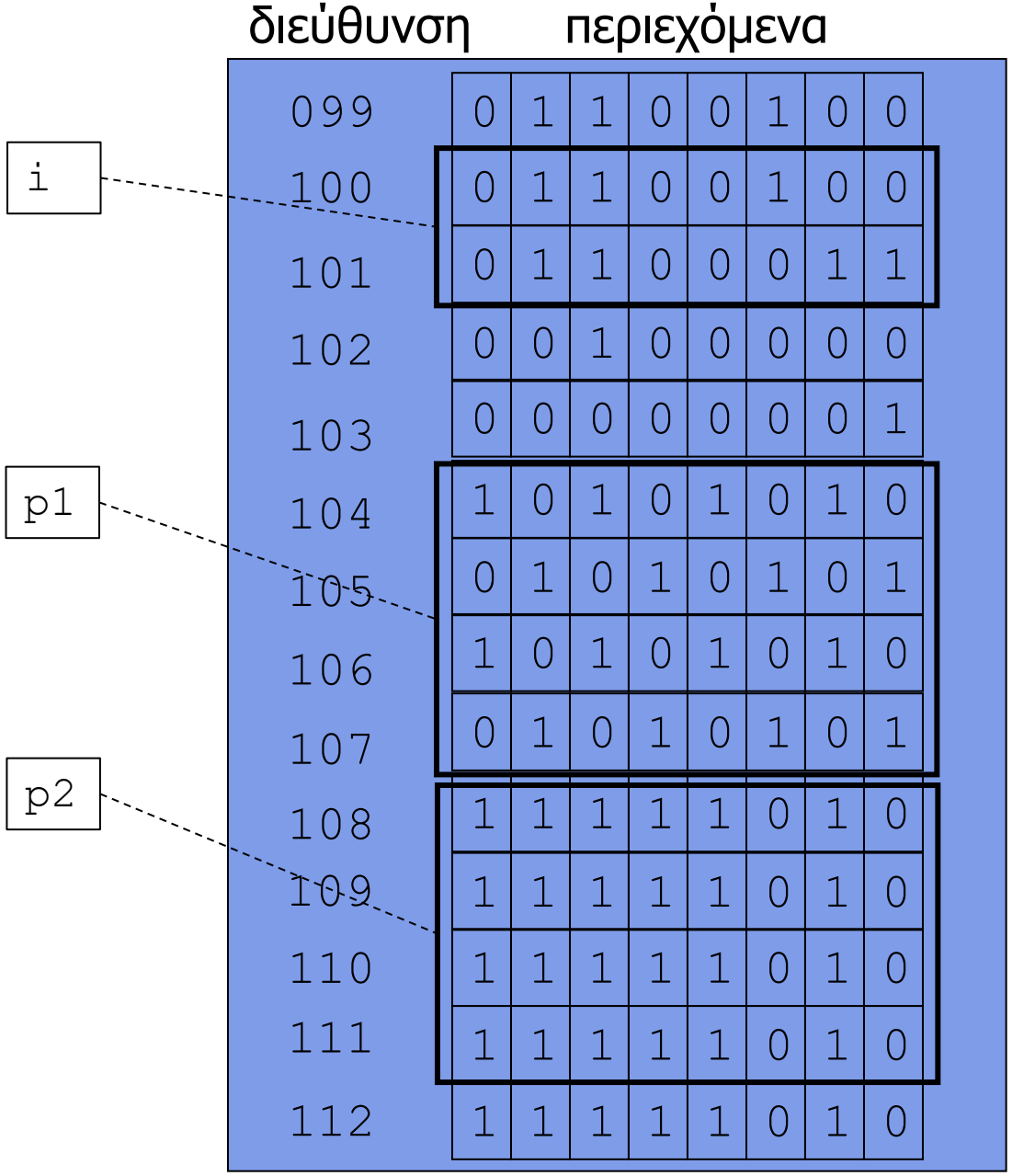
```
short int i;
…
short int *p1;
```



```
short int i;
…
short int *p1;
short int *p2;
```

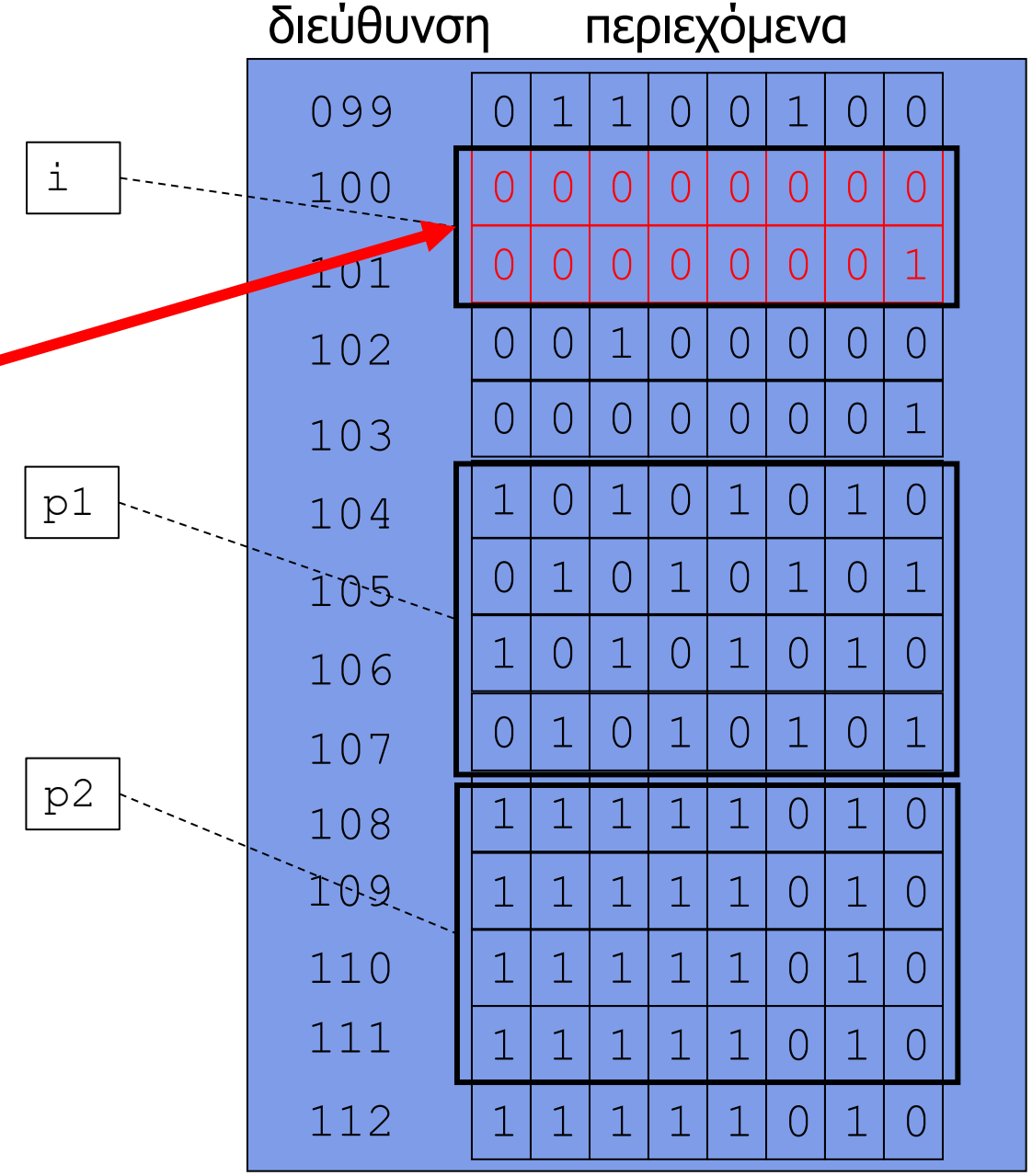


```
short int i;
…
short int *p1;
short int *p2;
```



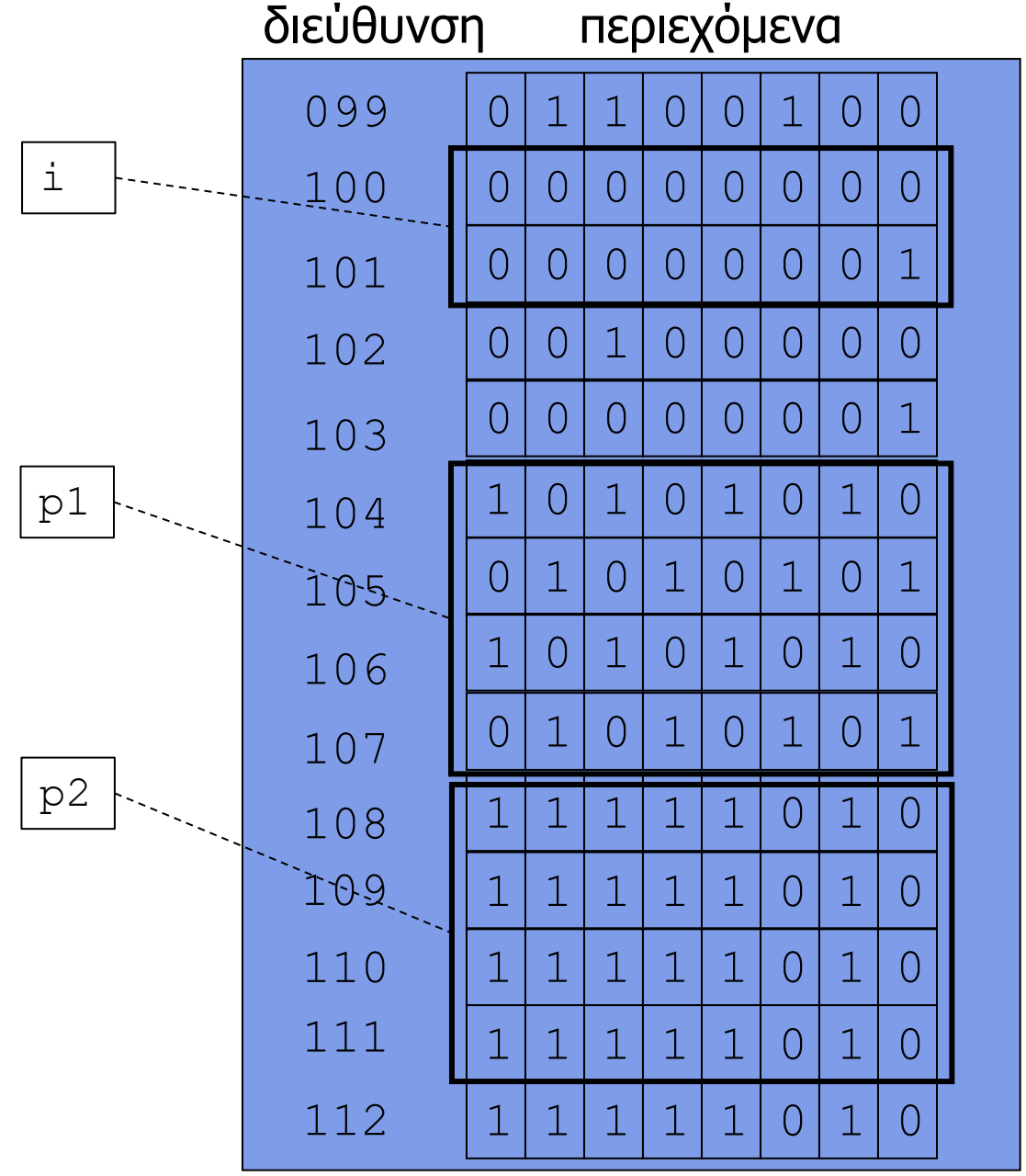
```
short int i;
…
short int *p1;
short int *p2;

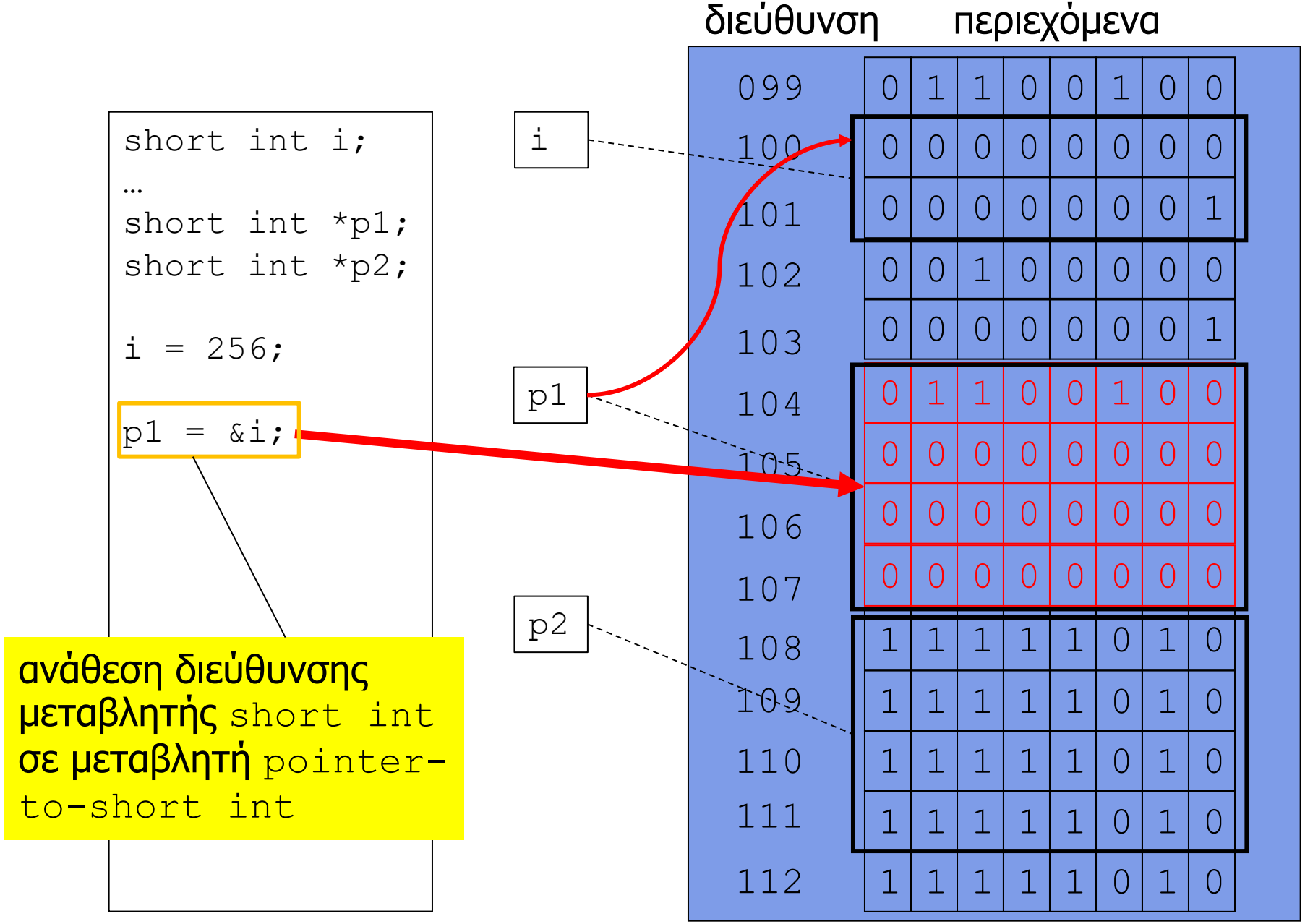
i = 256;
```



```
short int i;
…
short int *p1;
short int *p2;

i = 256;
```

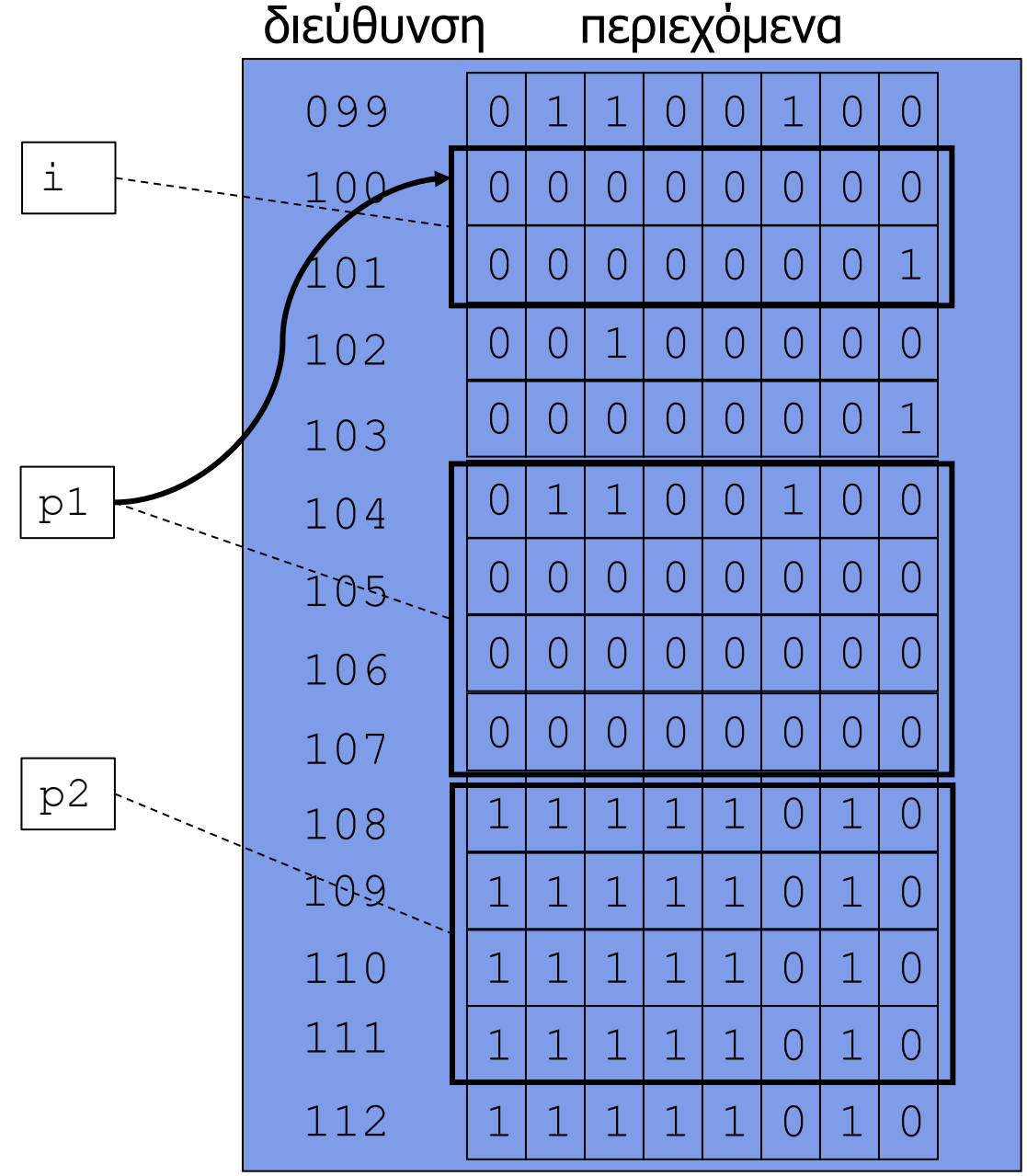


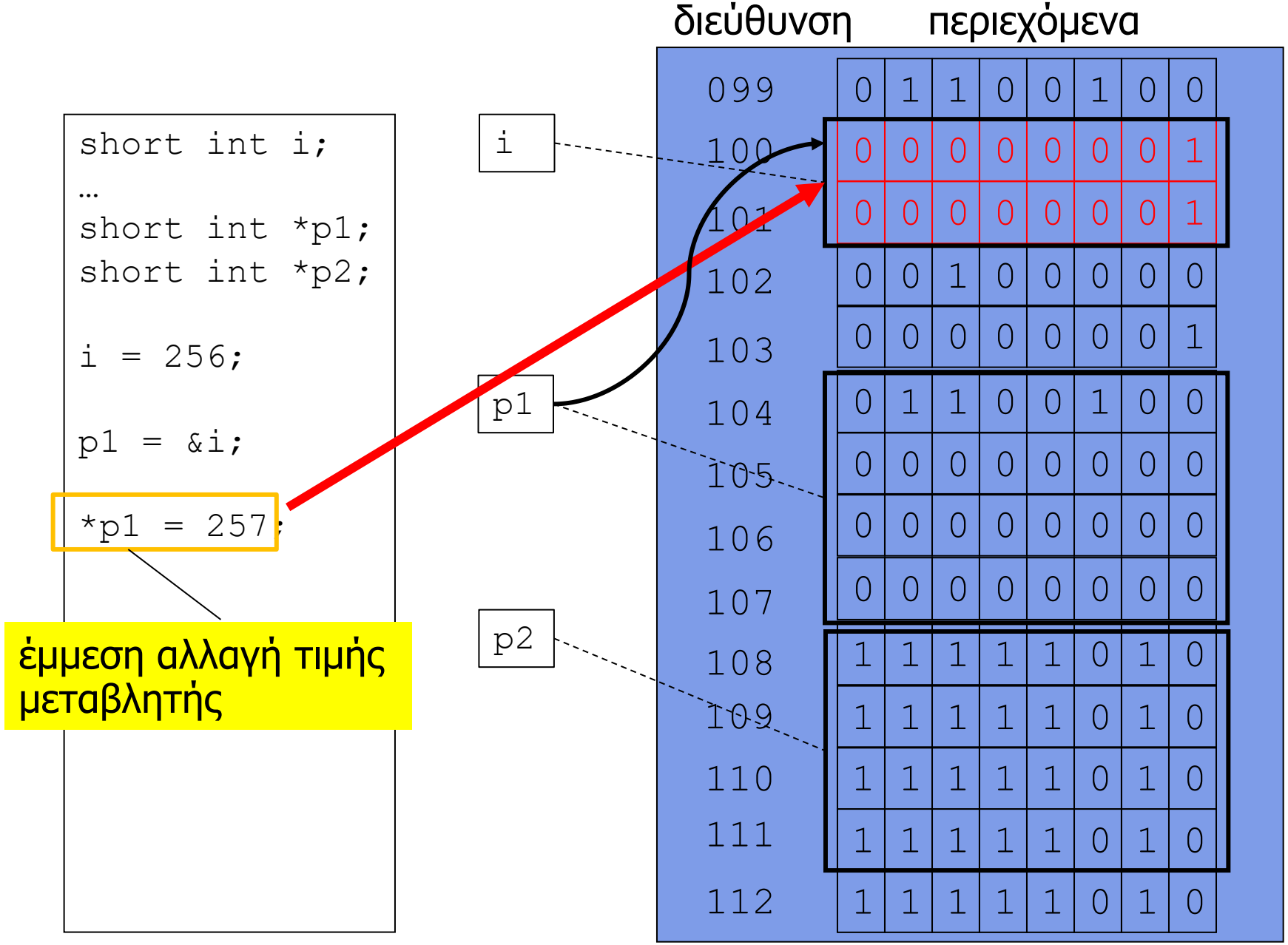

διεύθυνση
περιεχόμενα
short int i;
…
short int *p1;
short int *p2;
i = 256;
p1 = &i;
ανάθεση διεύθυνσης μεταβλητής short int σε μεταβλητή pointer-to-short int
i
p1
p2
099
100
101
102
103
104
105
106
107
108
109
110
111
112
0 1 1 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1
0 1 1 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 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
1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0

```
short int i;
…
short int *p1;
short int *p2;

i = 256;

p1 = &i;
```



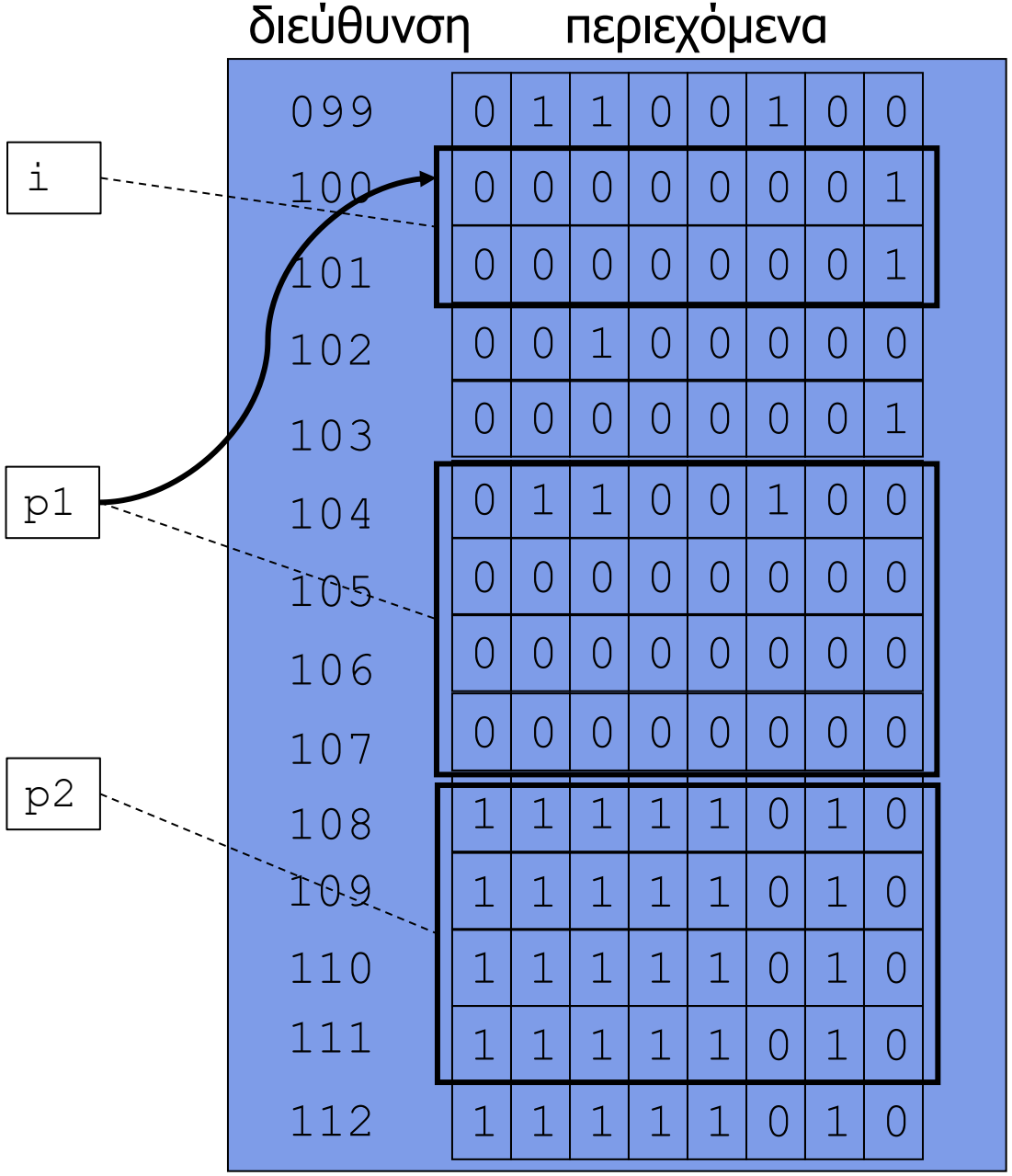

διεύθυνση
περιεχόμενα
short int i;
…
short int *p1;
short int *p2;
i = 256;
p1 = &i;
*p1 = 257;
έμμεση αλλαγή τιμής μεταβλητής
i
p1
p2
099
100
101
102
103
104
105
106
107
108
109
110
111
112
0 1 1 0 0 1 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1
0 1 1 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 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
1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0

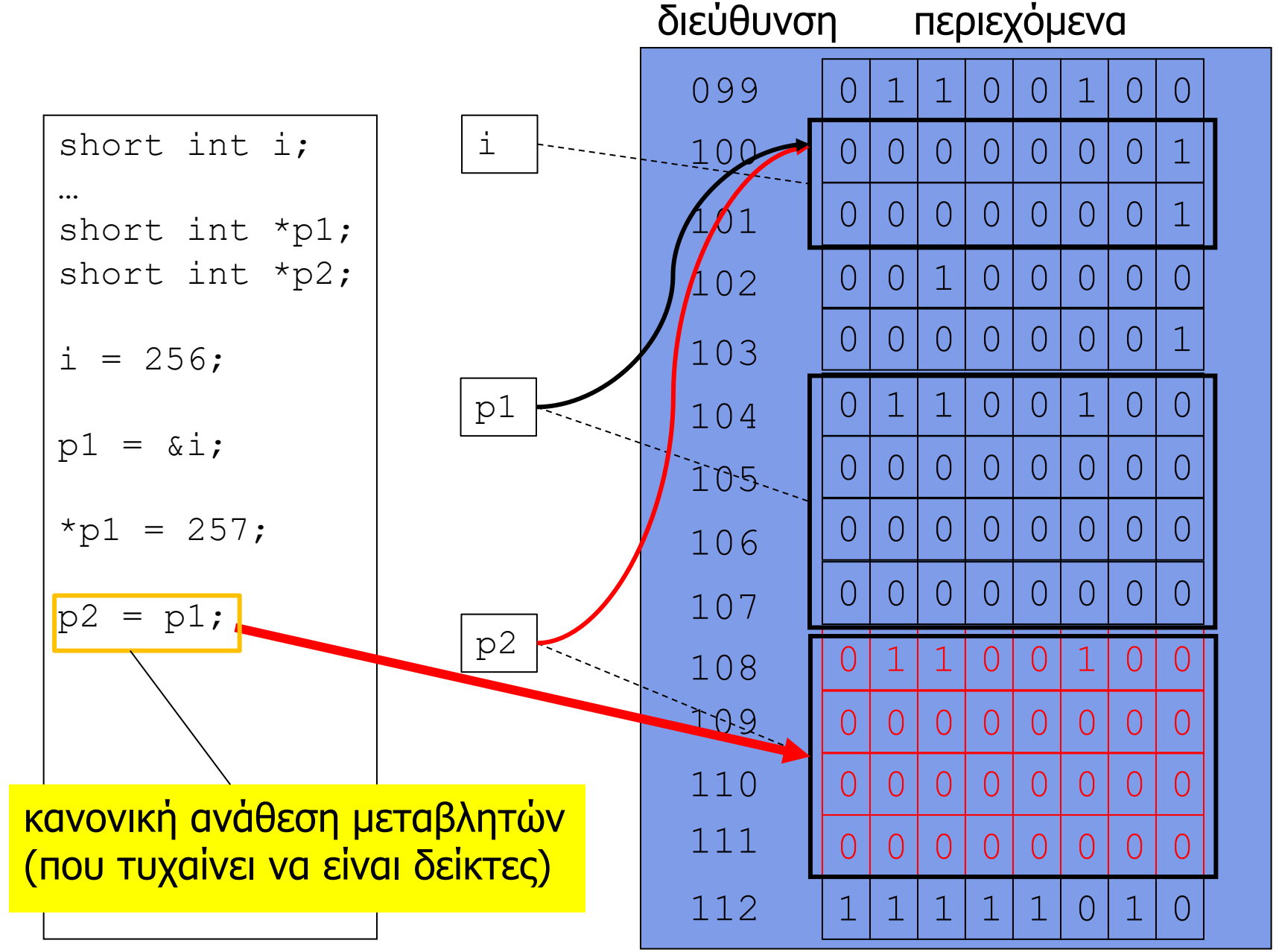
```
short int i;
…
short int *p1;
short int *p2;

i = 256;

p1 = &i;

*p1 = 257;
```





κανονική ανάθεση μεταβλητών
(που τυχαίνει να είναι δείκτες)

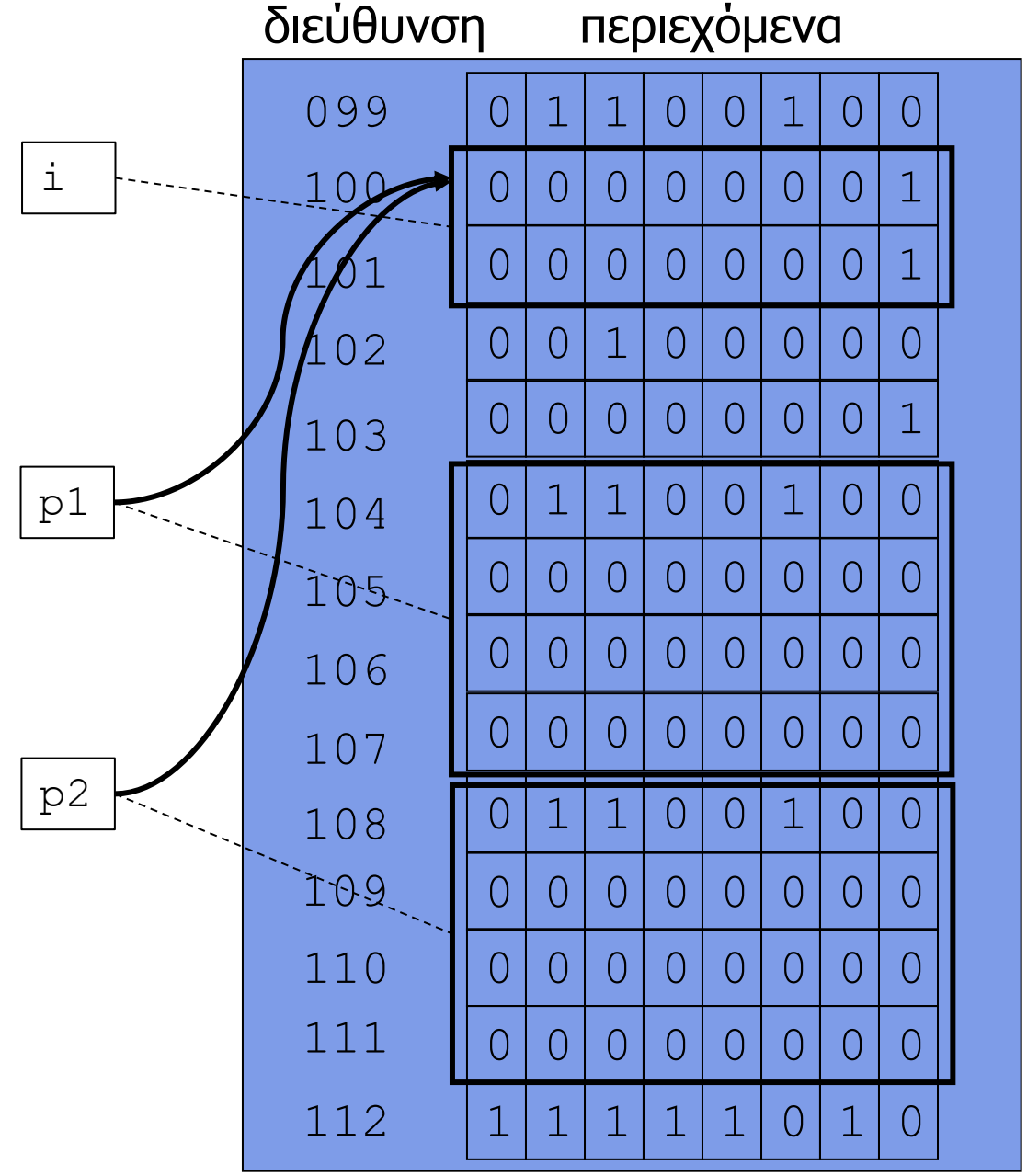
```
short int i;
…
short int *p1;
short int *p2;

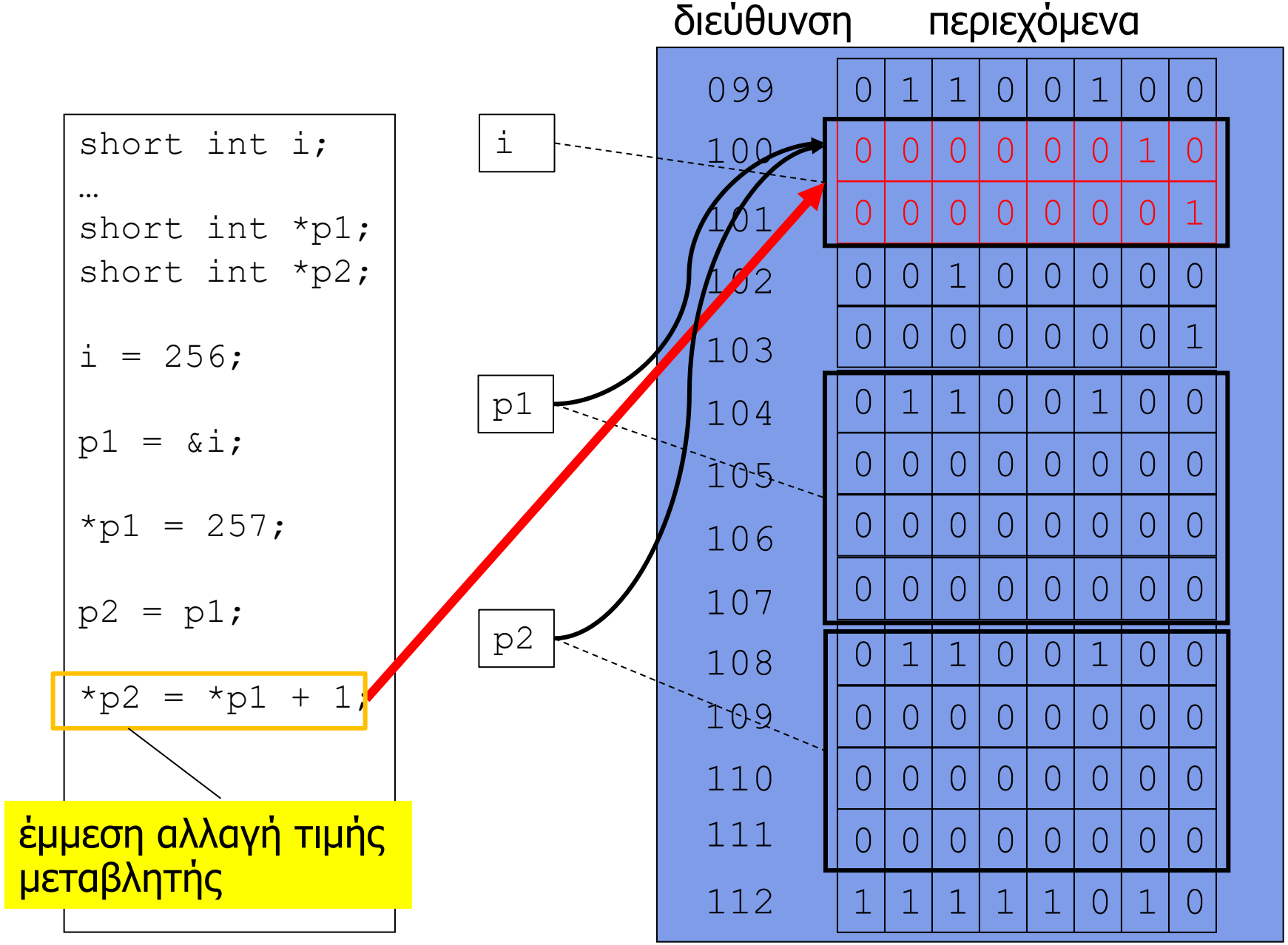
i = 256;

p1 = &i;

*p1 = 257;

p2 = p1;
```





έμμεση αλλαγή τιμής μεταβλητής

```
short int i;
…
short int *p1;
short int *p2;

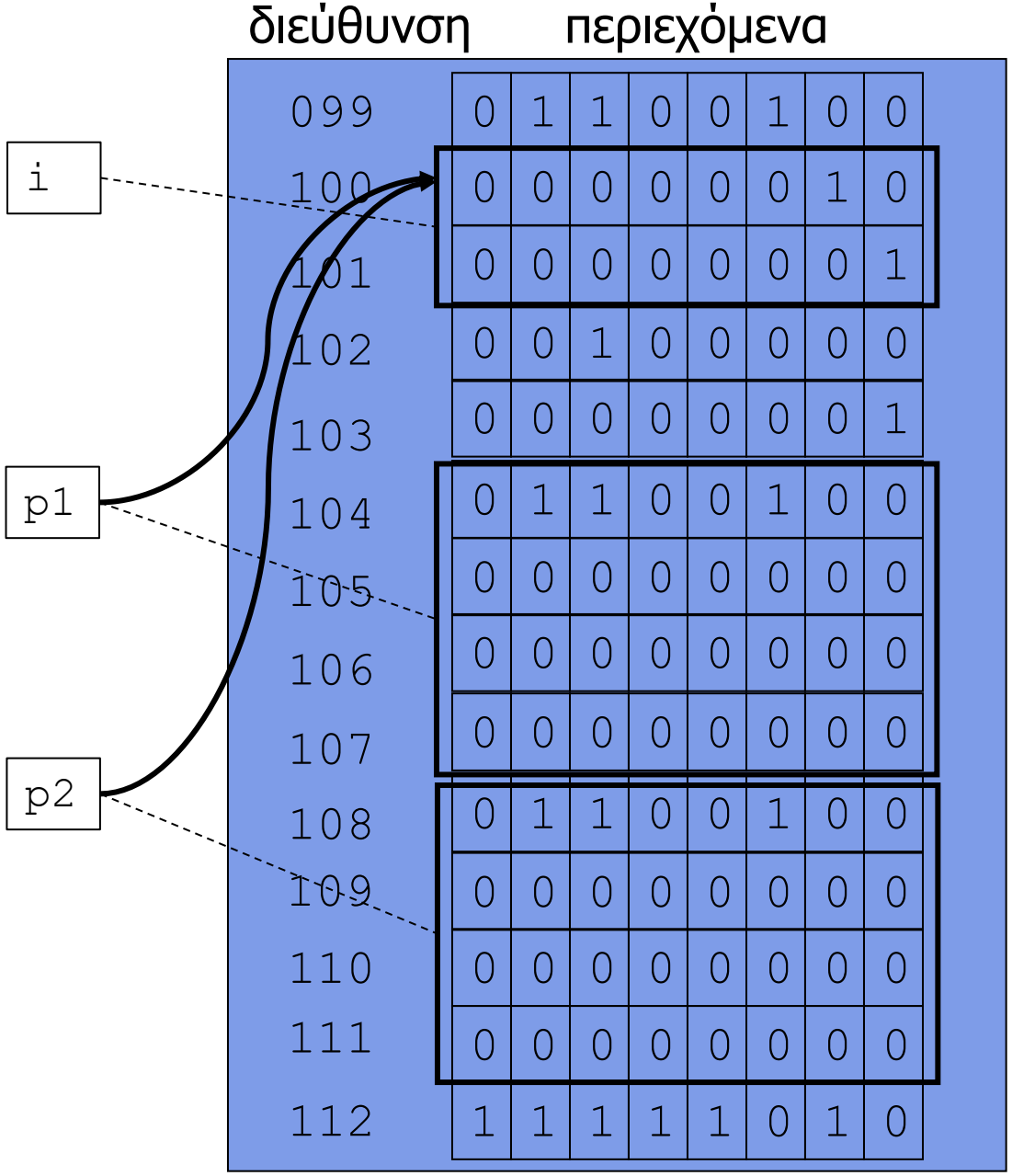
i = 256;

p1 = &i;

*p1 = 257;

p2 = p1;

*p2 = *p1 + 1;
```



```
short int i;
…
short int *p1;
short int *p2;

i = 256;

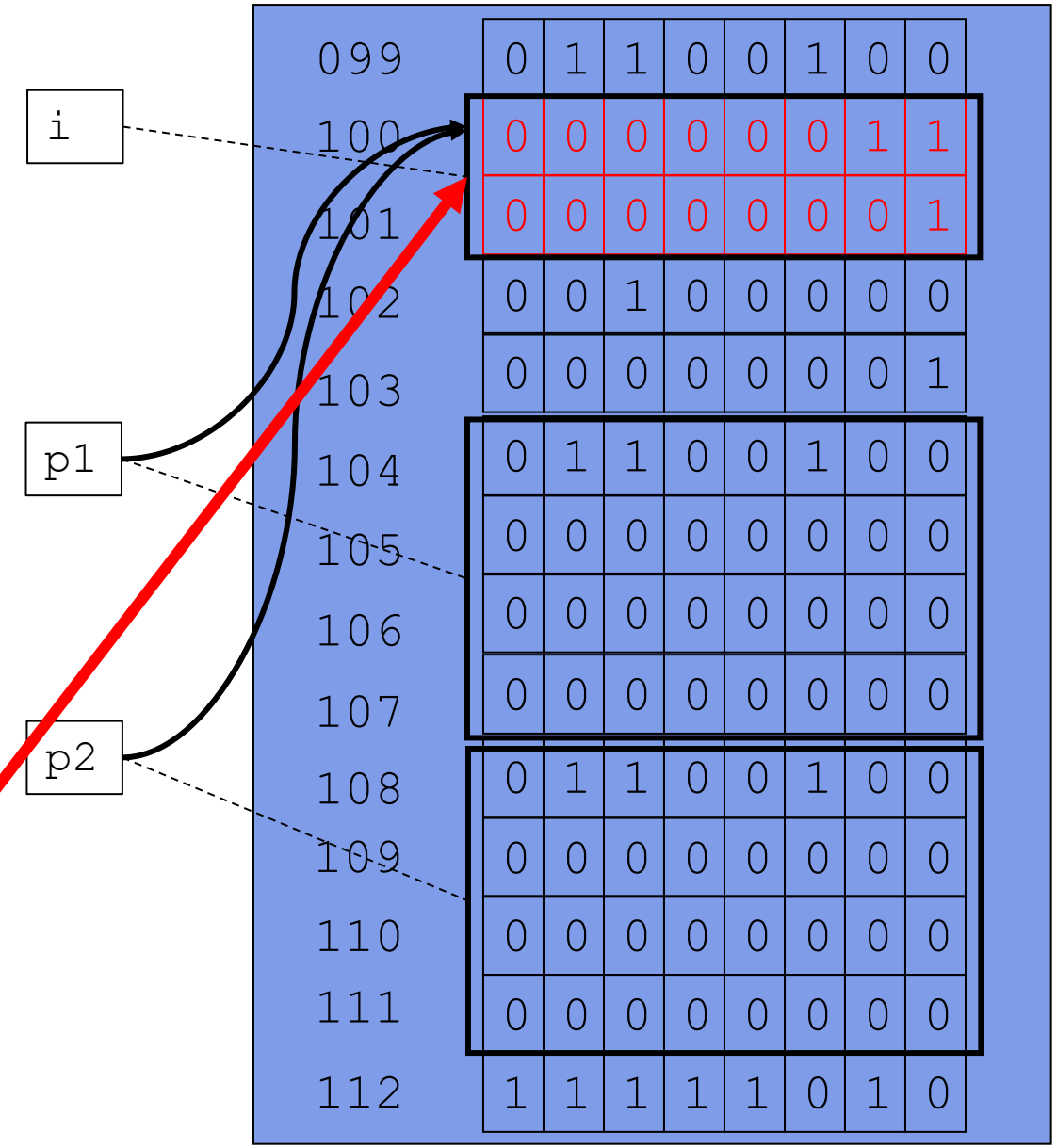
p1 = &i;

*p1 = 257;

p2 = p1;

*p2 = *p1 + 1;

*p1 = *p2 + 1;
```



έμμεση αλλαγή τιμής μεταβλητής

```
short int i;
…
short int *p1;
short int *p2;

i = 256;

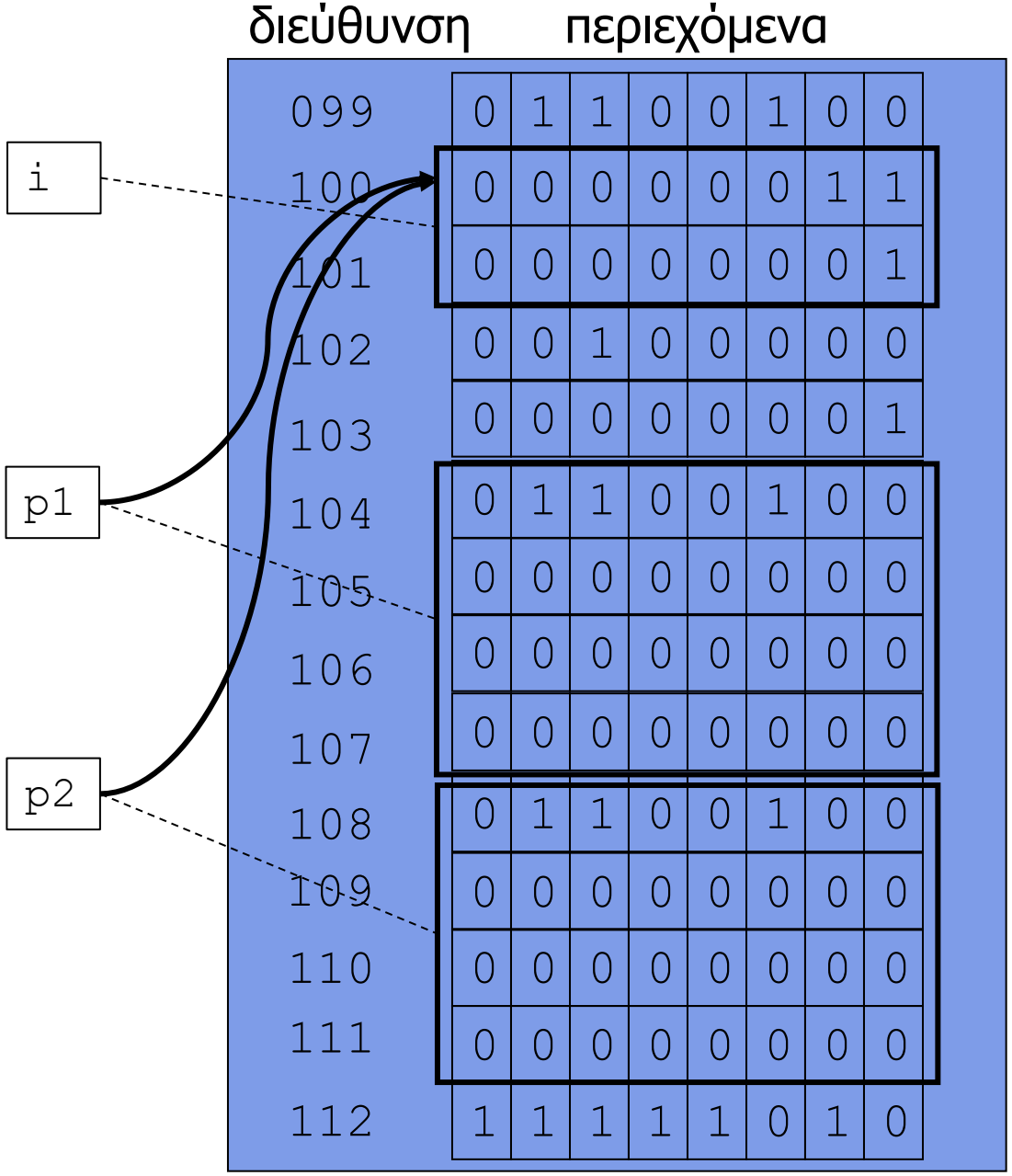
p1 = &i;

*p1 = 257;

p2 = p1;

*p2 = *p1 + 1;

*p1 = *p2 + 1;
```



```
short int i;
…
short int *p1;
short int *p2;

i = 256;

p1 = &i;

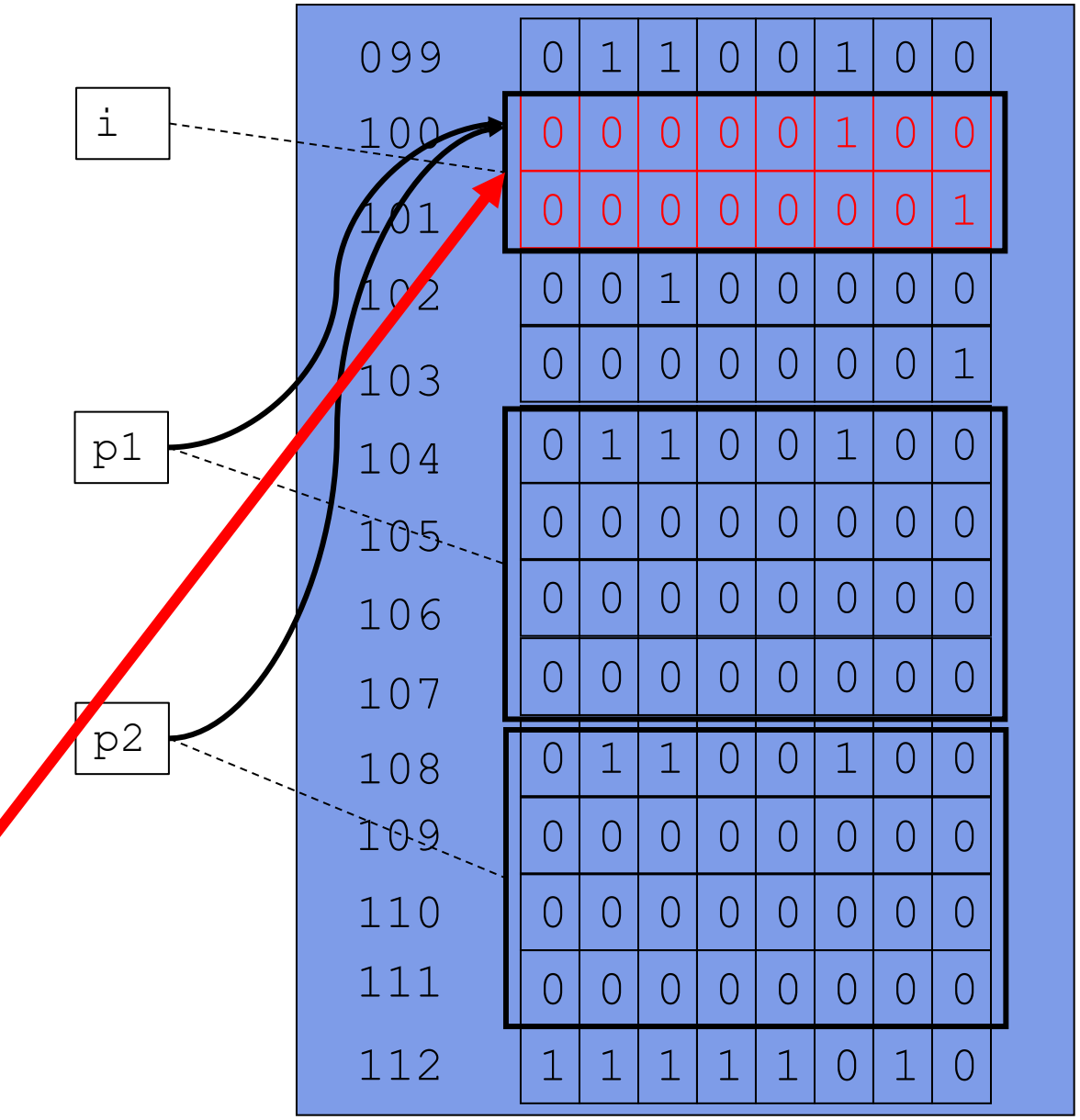
*p1 = 257;

p2 = p1;

*p2 = *p1 + 1;

*p1 = *p2 + 1;

i = i + 1;
```



```
short int i;
…
short int *p1;
short int *p2;

i = 256;

p1 = &i;

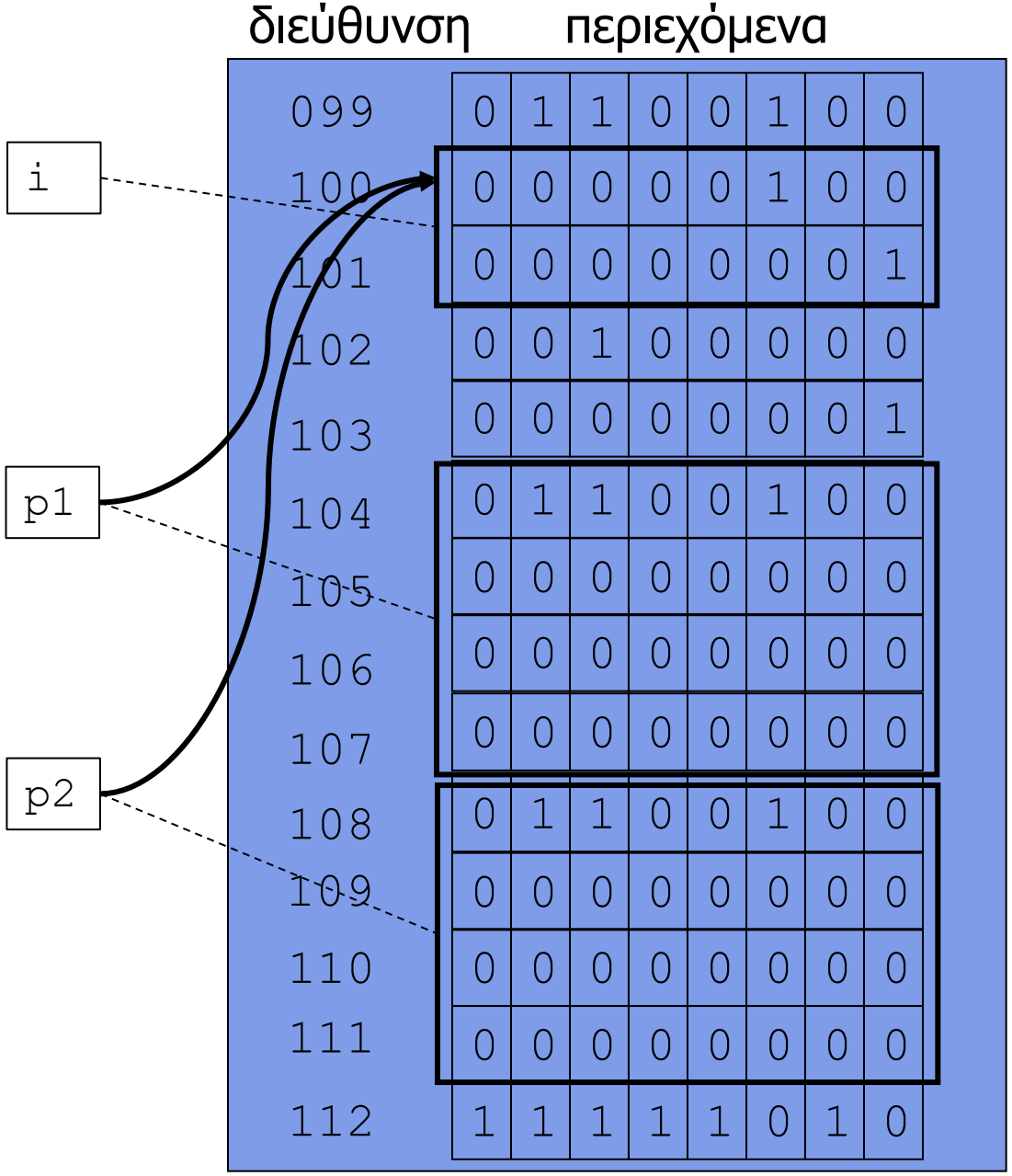
*p1 = 257;

p2 = p1;

*p2 = *p1 + 1;

*p1 = *p2 + 1;

i = i + 1;
```



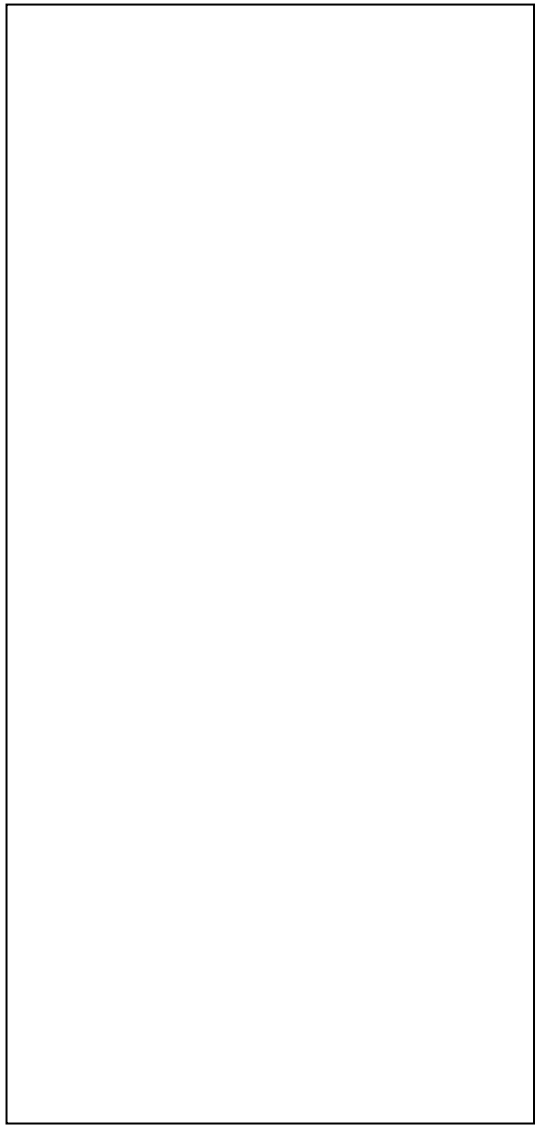
Παράκαμψη συντακτικού ελέγχου

- Με type casting, μια **οποιαδήποτε** ακέραια τιμή μπορεί να ανατεθεί σε μια μεταβλητή `pointer-to-T`
- Επίσης, επιτρέπεται **αριθμητική με δείκτες**
  - αν `p` είναι μεταβλητή `pointer-to-T`, τότε η έκφραση `p+i` ερμηνεύεται (αυτόματα) σε `p+i*sizeof(T)`

- Ο μεταφραστής **δεν** ελέγχει αν μια τιμή δείκτη `pointer-to-T` που «προκύπτει» με type casting ή με αριθμητική δεικτών, **πραγματικά** αντιστοιχεί στη διεύθυνση ενός αντικειμένου `T`
- Ένας δείκτης μπορεί (τελικά) να δείχνει **οπουδήποτε**
- Την ευθύνη την έχει **ο ίδιος** ο προγραμματιστής!

Το παράδειγμα που ακολουθεί, υποθέτει

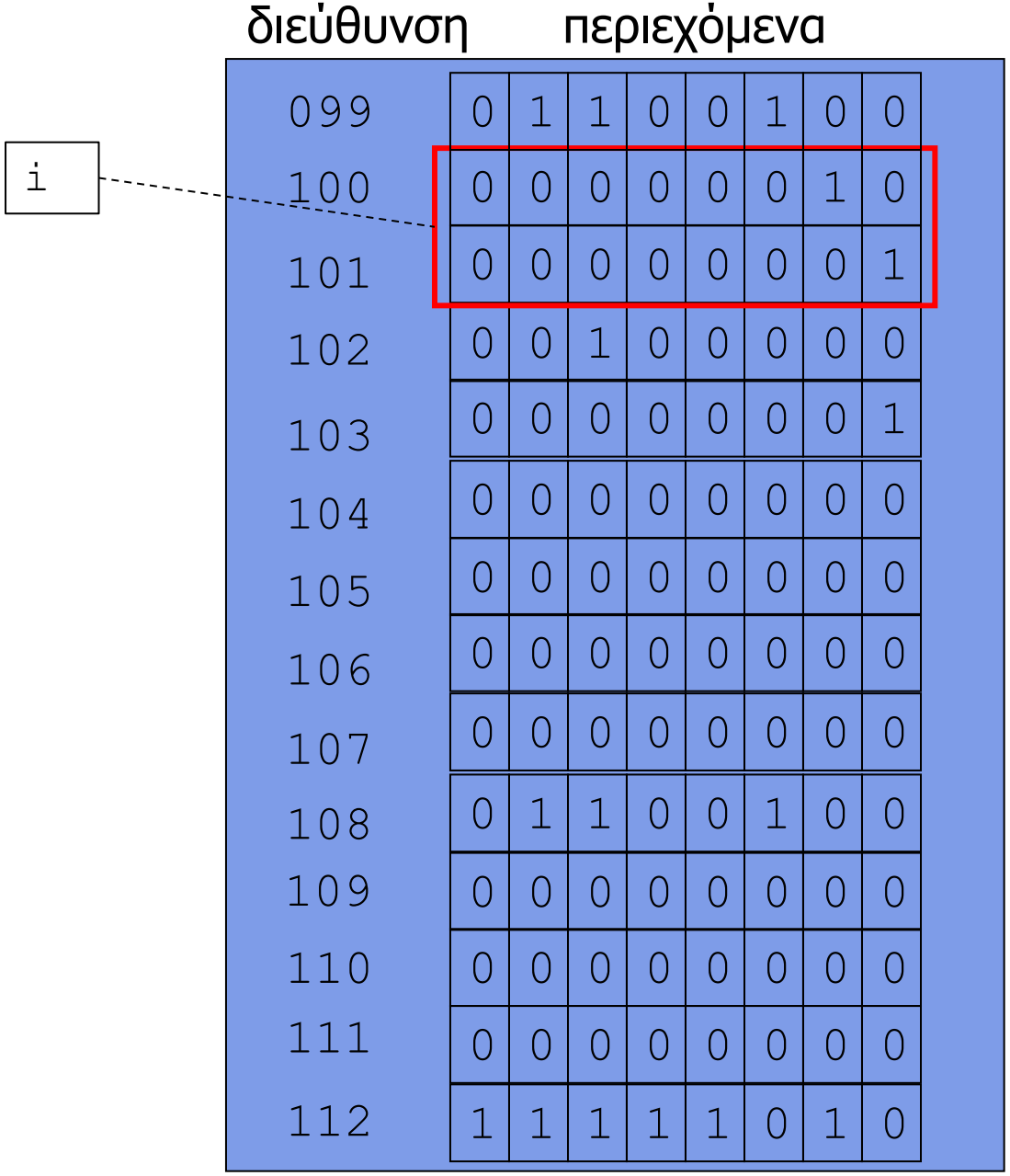
- μέγεθος `short int` 2 bytes
- μέγεθος δεικτών/pointers 4 bytes
- αποθήκευση αριθμητικών τιμών στην μνήμη με την σύμβαση little endian
  - τα λιγότερο σημαντικά bytes ενός multi-byte ακεραίου αποθηκεύονται στις μικρότερες διευθύνσεις μνήμης



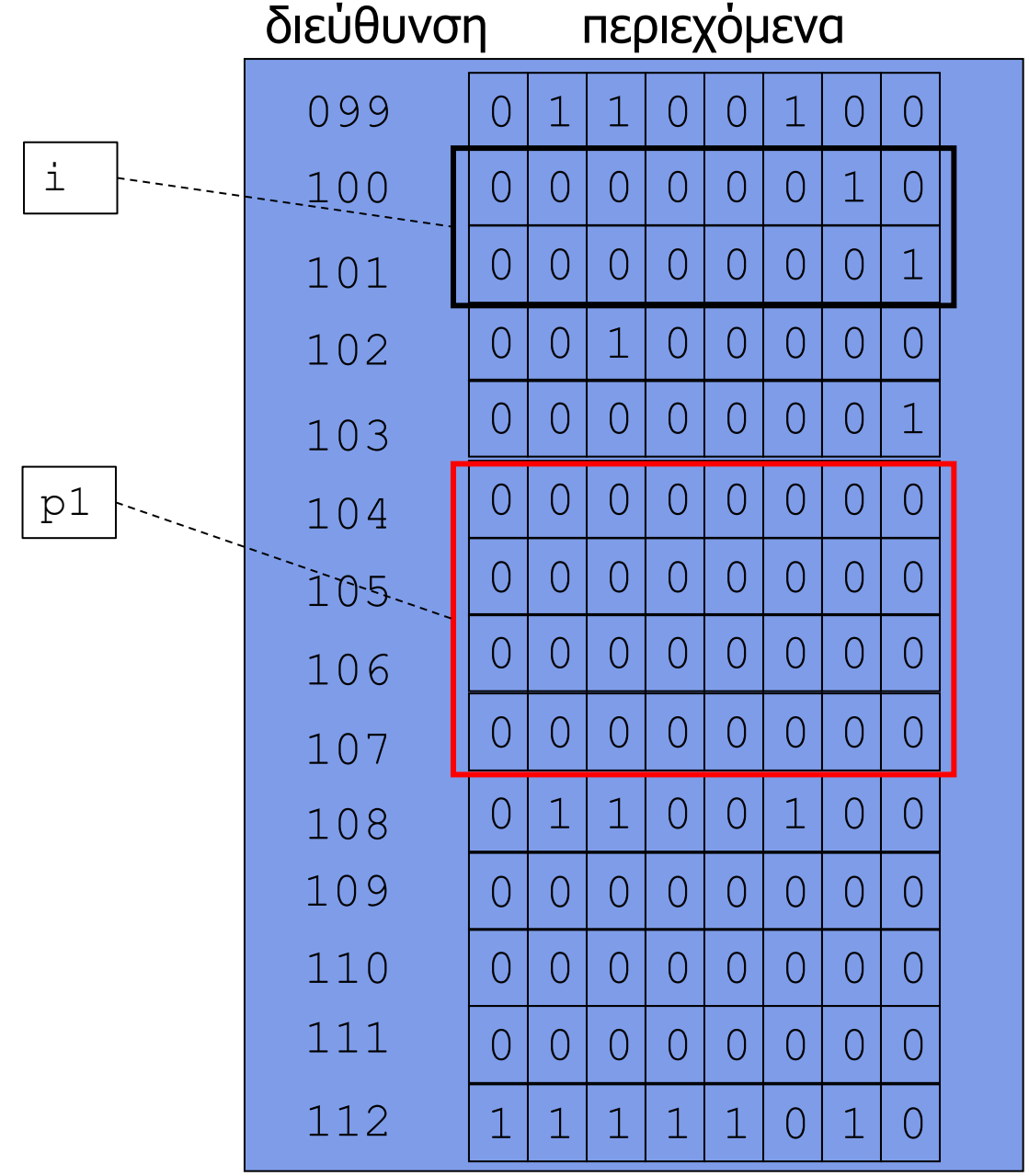
διεύθυνση περιεχόμενα

099	0	1	1	0	0	1	0	0
100	0	0	0	0	0	0	1	0
101	0	0	0	0	0	0	0	1
102	0	0	1	0	0	0	0	0
103	0	0	0	0	0	0	0	1
104	0	0	0	0	0	0	0	0
105	0	0	0	0	0	0	0	0
106	0	0	0	0	0	0	0	0
107	0	0	0	0	0	0	0	0
108	0	1	1	0	0	1	0	0
109	0	0	0	0	0	0	0	0
110	0	0	0	0	0	0	0	0
111	0	0	0	0	0	0	0	0
112	1	1	1	1	1	0	1	0

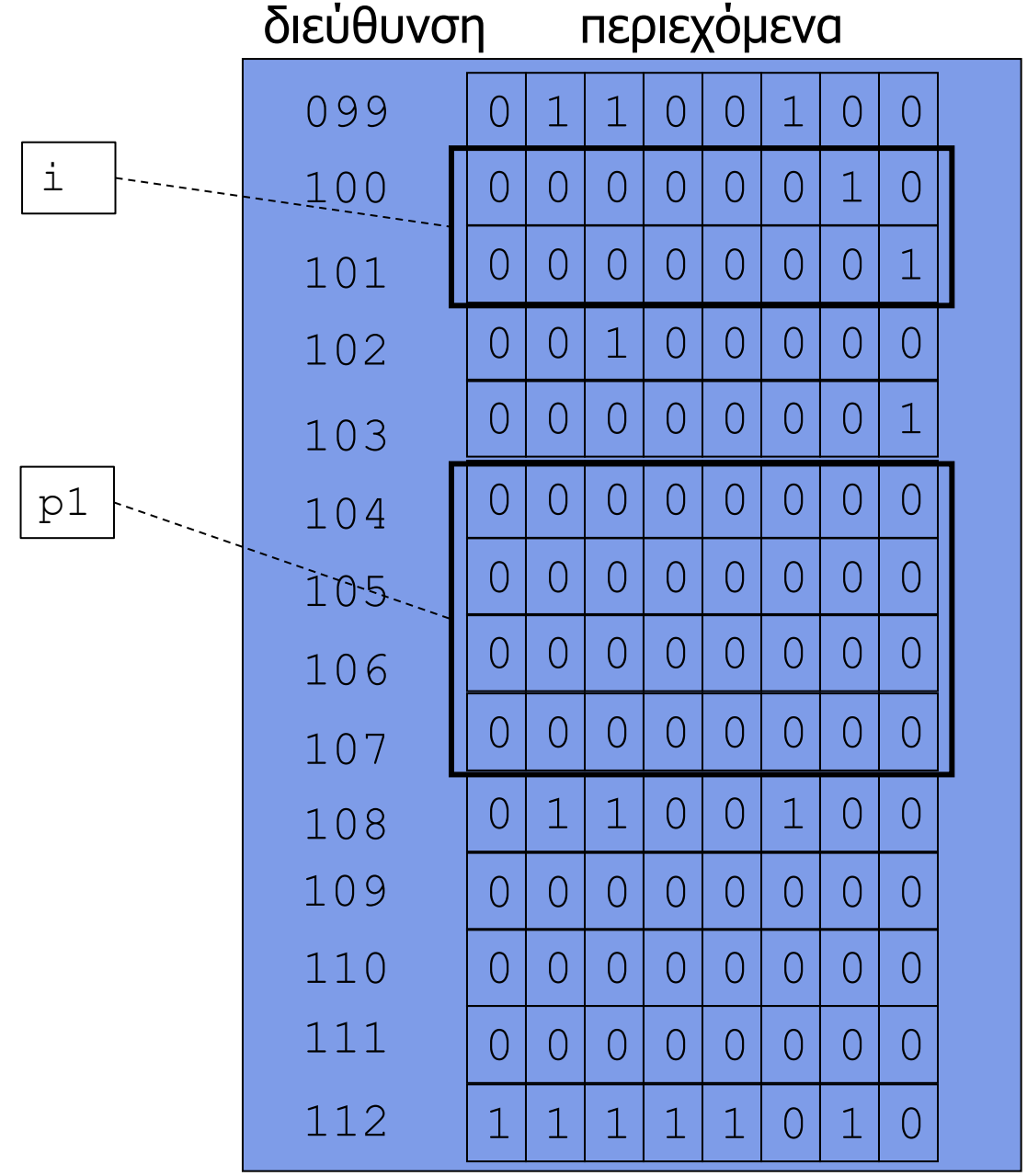
```
short int i;
```


διεύθυνση
περιεχόμενα
i
099 0 1 1 0 0 1 0 0
100 0 0 0 0 0 0 1 0
101 0 0 0 0 0 0 0 1
102 0 0 1 0 0 0 0 0
103 0 0 0 0 0 0 0 1
104 0 0 0 0 0 0 0 0
105 0 0 0 0 0 0 0 0
106 0 0 0 0 0 0 0 0
107 0 0 0 0 0 0 0 0
108 0 1 1 0 0 1 0 0
109 0 0 0 0 0 0 0 0
110 0 0 0 0 0 0 0 0
111 0 0 0 0 0 0 0 0
112 1 1 1 1 1 0 1 0

```
short int i;
…
short int *p1;
```

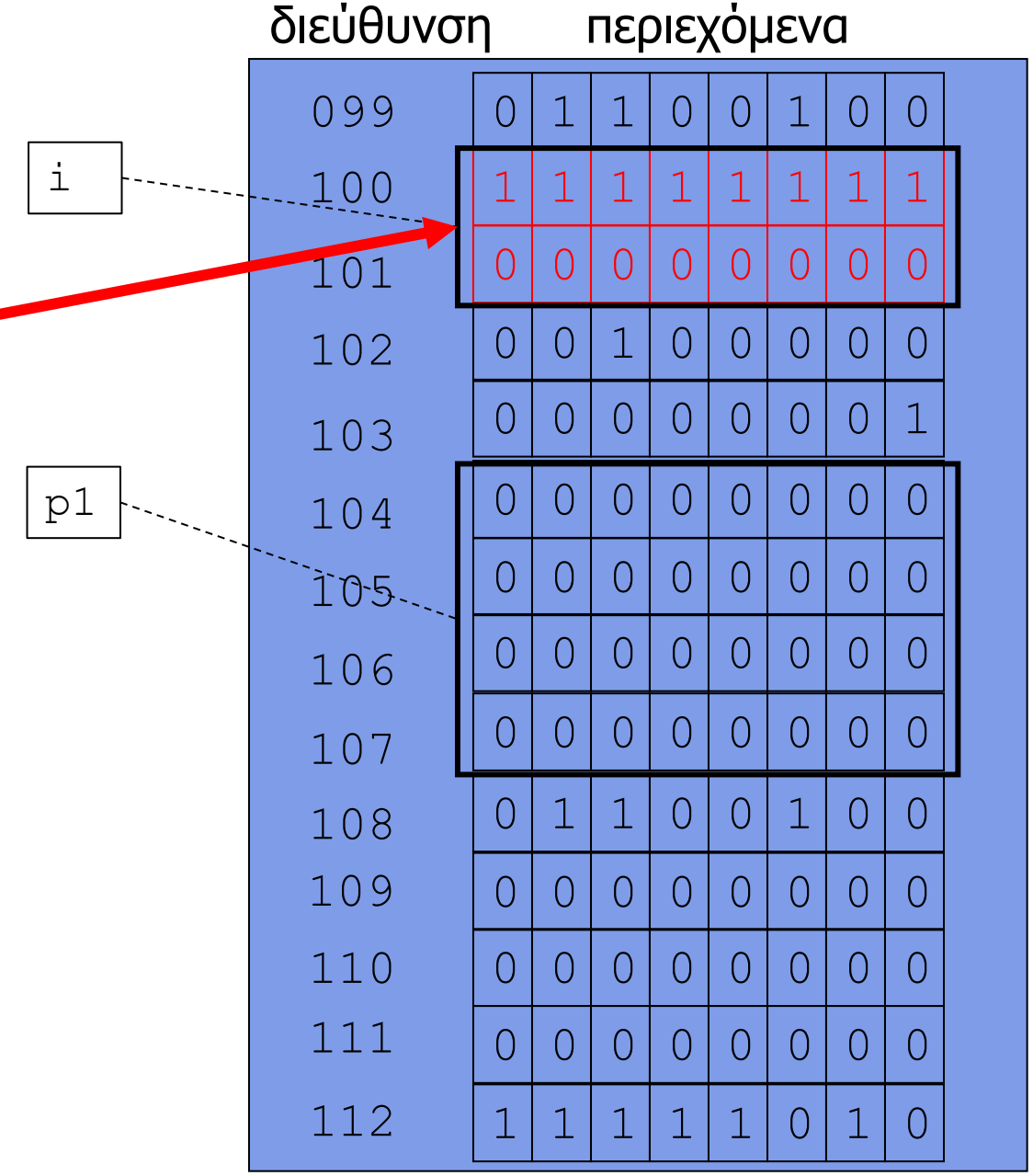


```
short int i;
…
short int *p1;
```



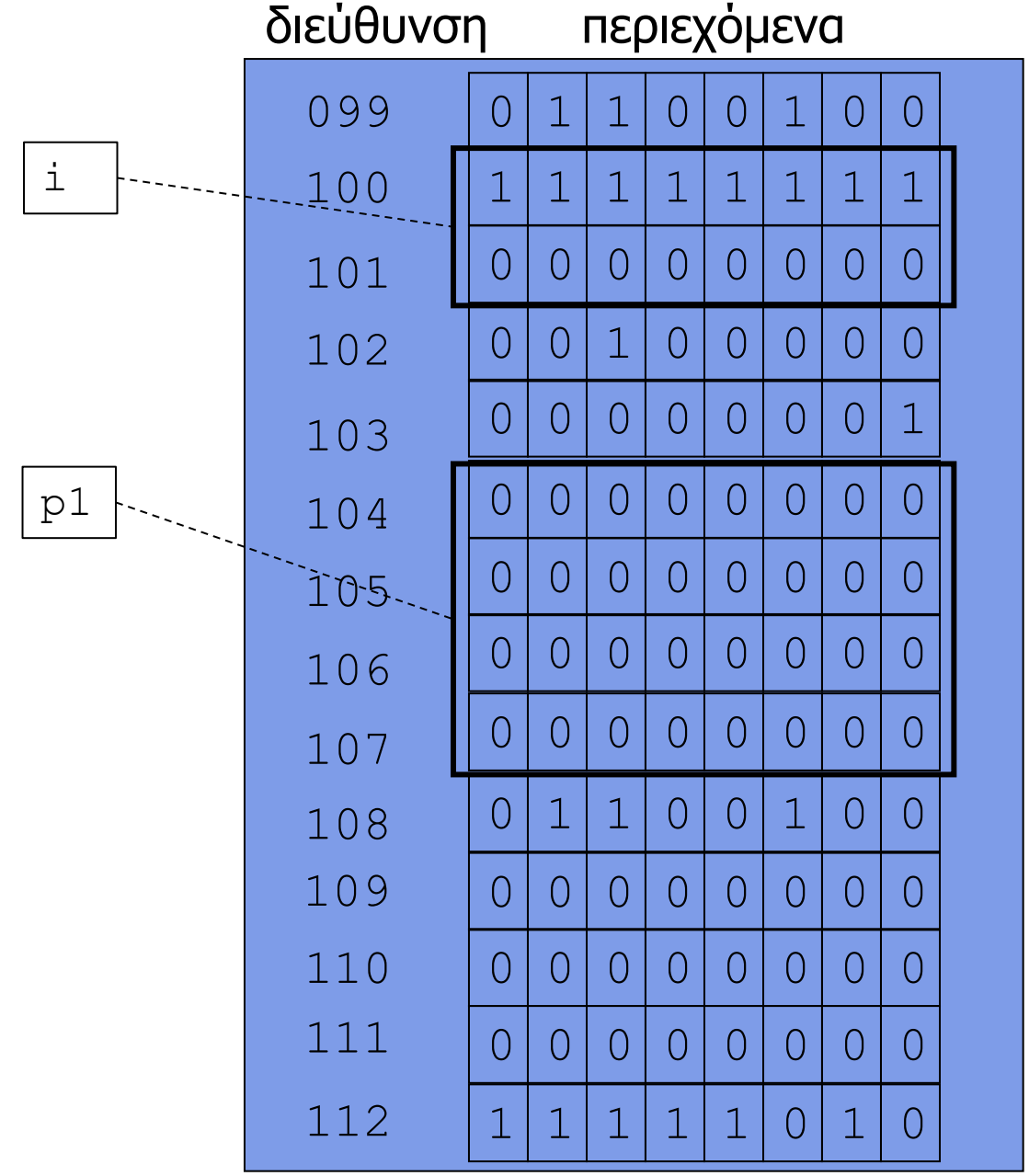
```
short int i;
…
short int *p1;

i = 255;
```



```
short int i;
…
short int *p1;

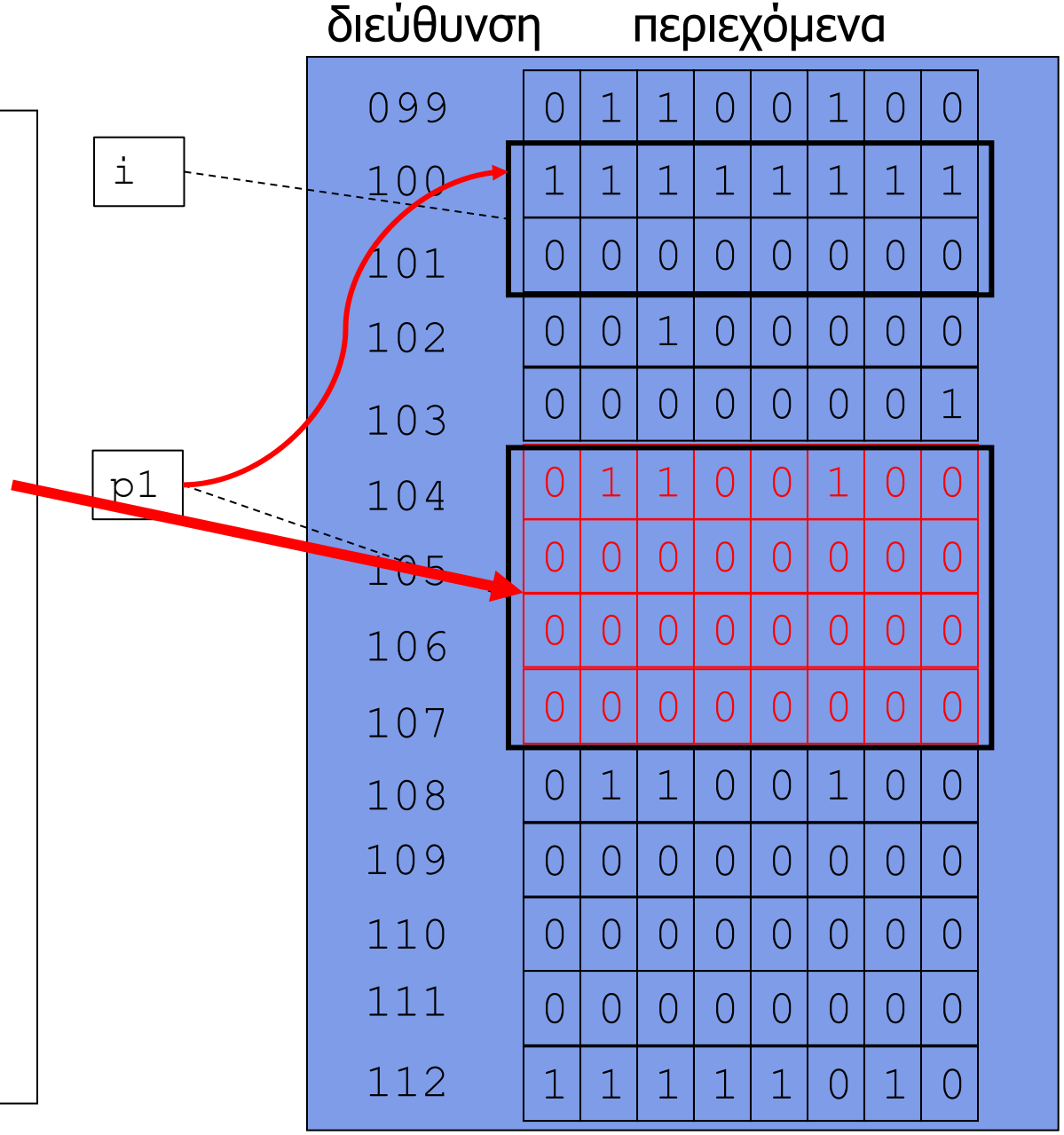
i = 255;
```



```
short int i;
…
short int *p1;

i = 255;

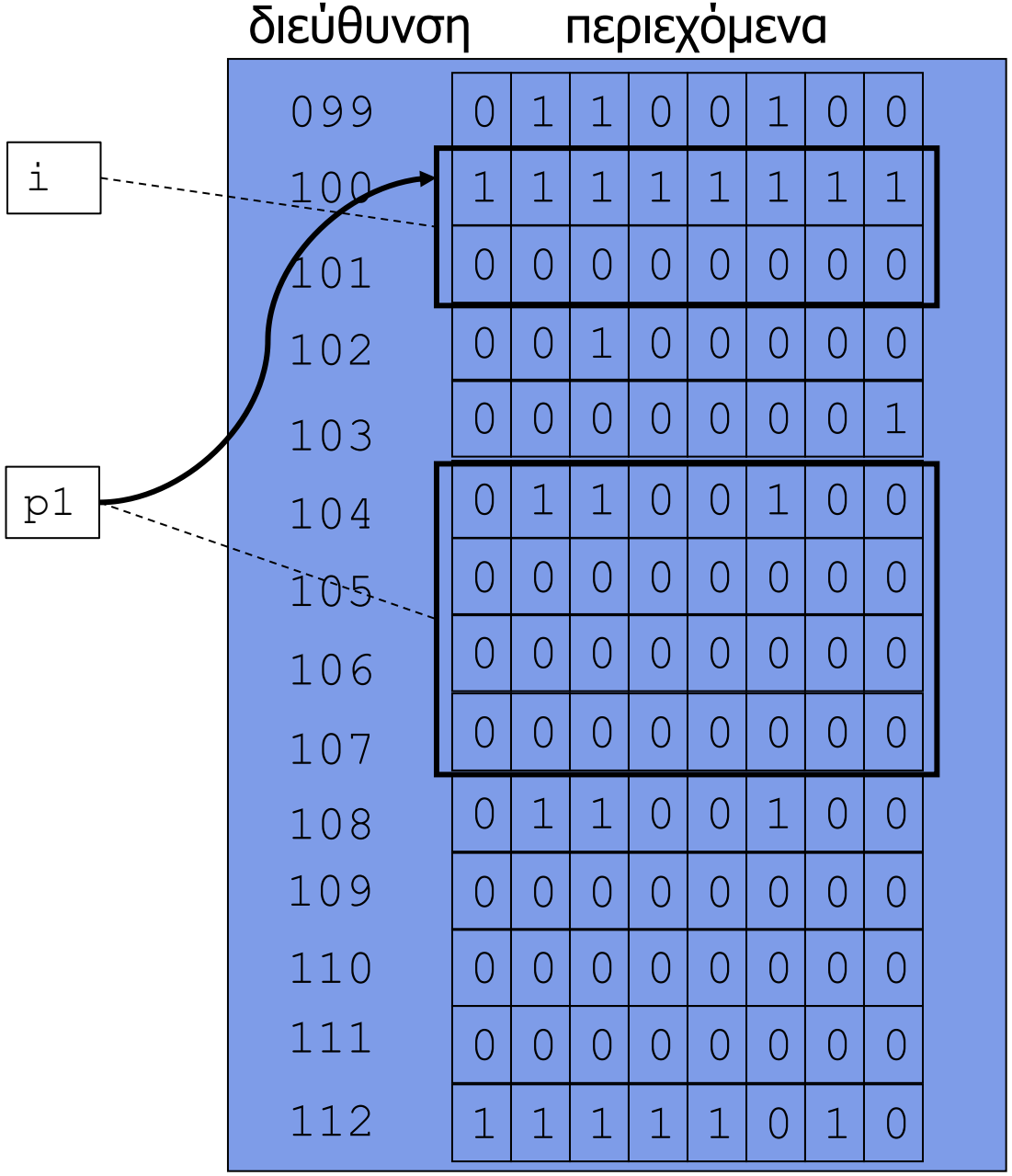
p1 = &i;
```



```
short int i;
…
short int *p1;

i = 255;

p1 = &i;
```



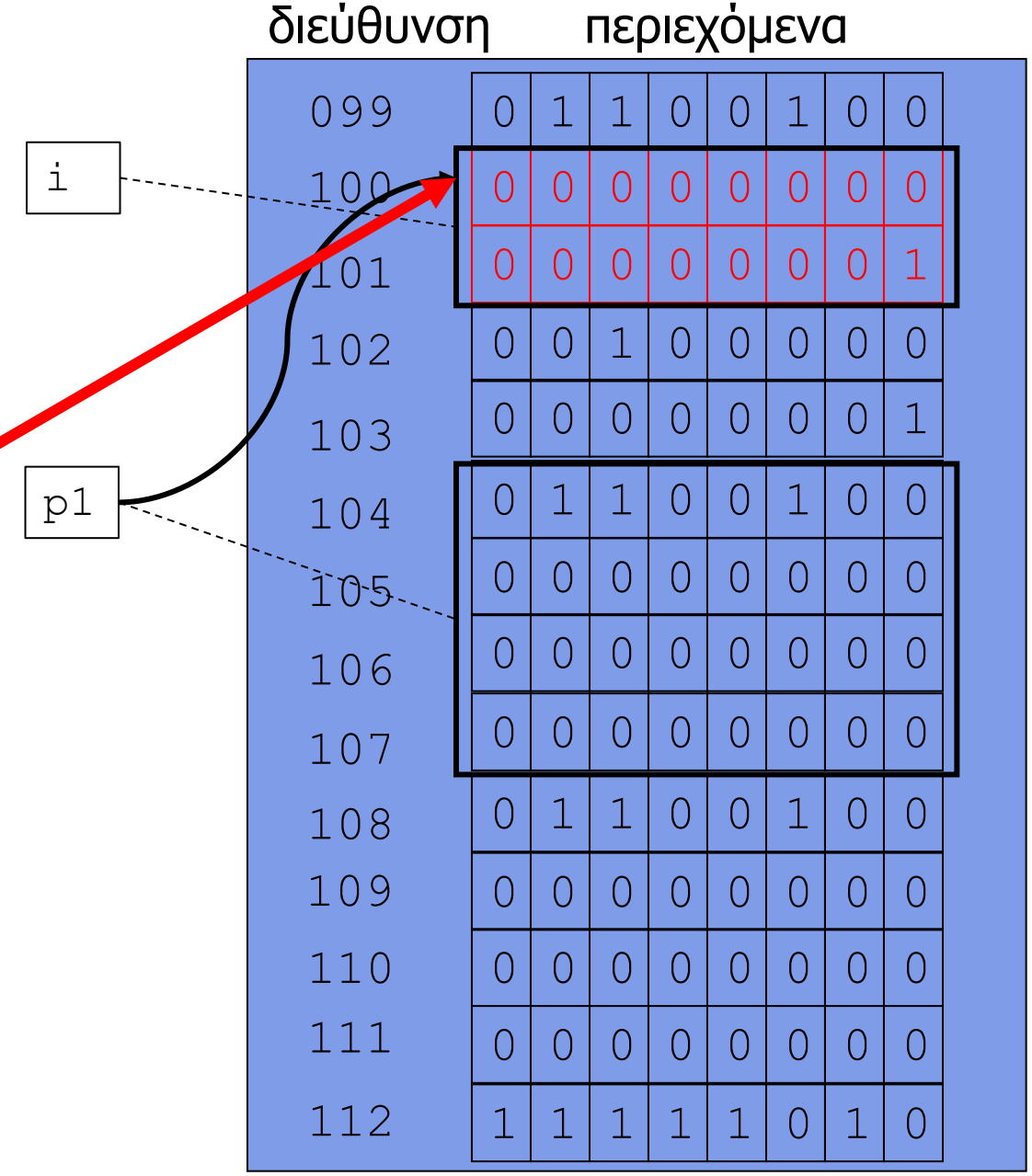
```
short int i;
…
short int *p1;

i = 255;

p1 = &i;

*p1 = 256;
```

έμμεση αλλαγή τιμής μεταβλητής

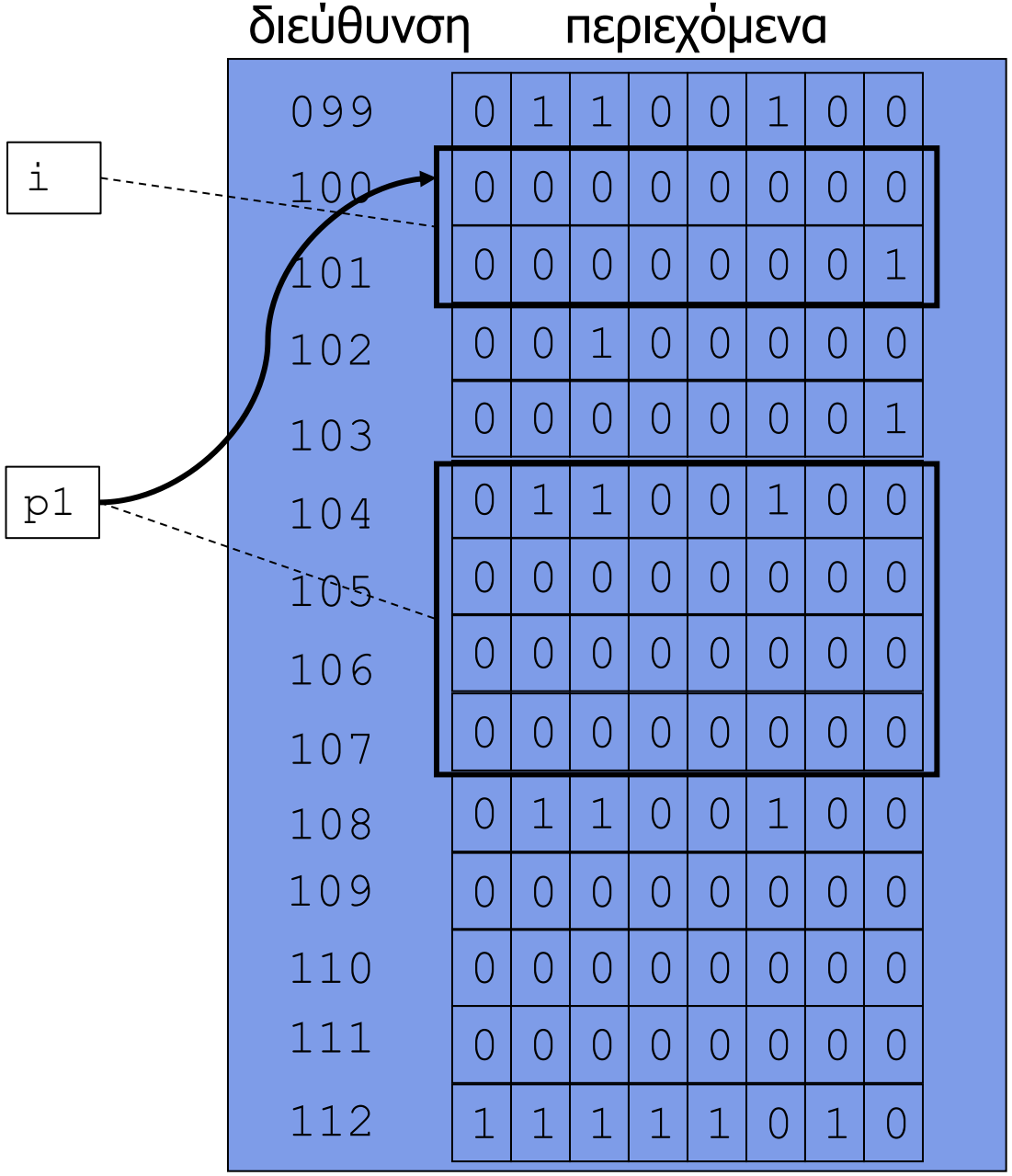


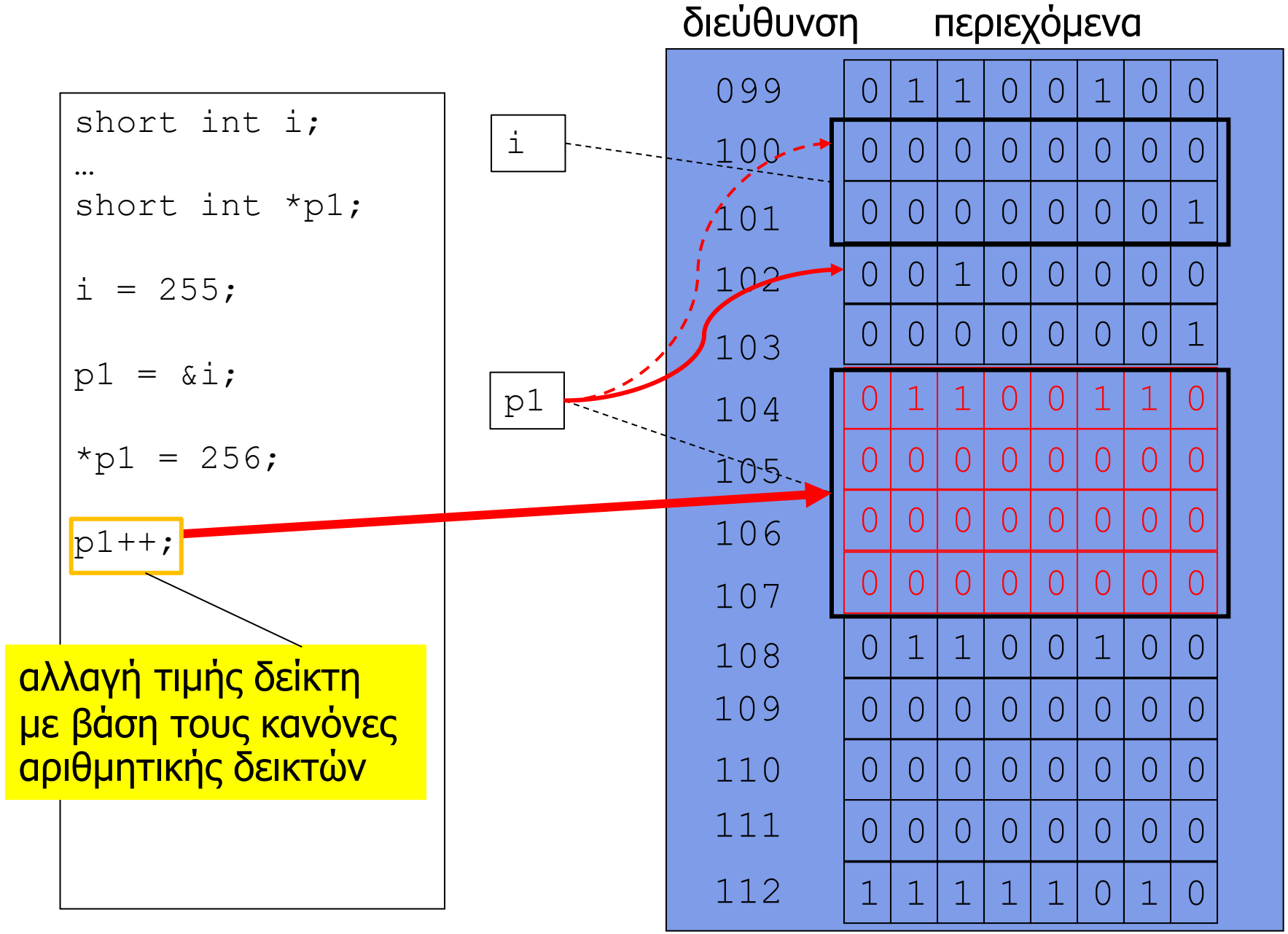
```
short int i;
…
short int *p1;

i = 255;

p1 = &i;

*p1 = 256;
```




διεύθυνση
περιεχόμενα
short int i;
…
short int *p1;
i = 255;
p1 = &i;
*p1 = 256;
p1++;
αλλαγή τιμής δείκτη
με βάση τους κανόνες
αριθμητικής δεικτών
i
p1
099 0 1 1 0 0 1 0 0
100 0 0 0 0 0 0 0 0
101 0 0 0 0 0 0 0 1
102 0 0 1 0 0 0 0 0
103 0 0 0 0 0 0 0 1
104 0 1 1 0 0 1 1 0
105 0 0 0 0 0 0 0 0
106 0 0 0 0 0 0 0 0
107 0 0 0 0 0 0 0 0
108 0 1 1 0 0 1 0 0
109 0 0 0 0 0 0 0 0
110 0 0 0 0 0 0 0 0
111 0 0 0 0 0 0 0 0
112 1 1 1 1 1 0 1 0

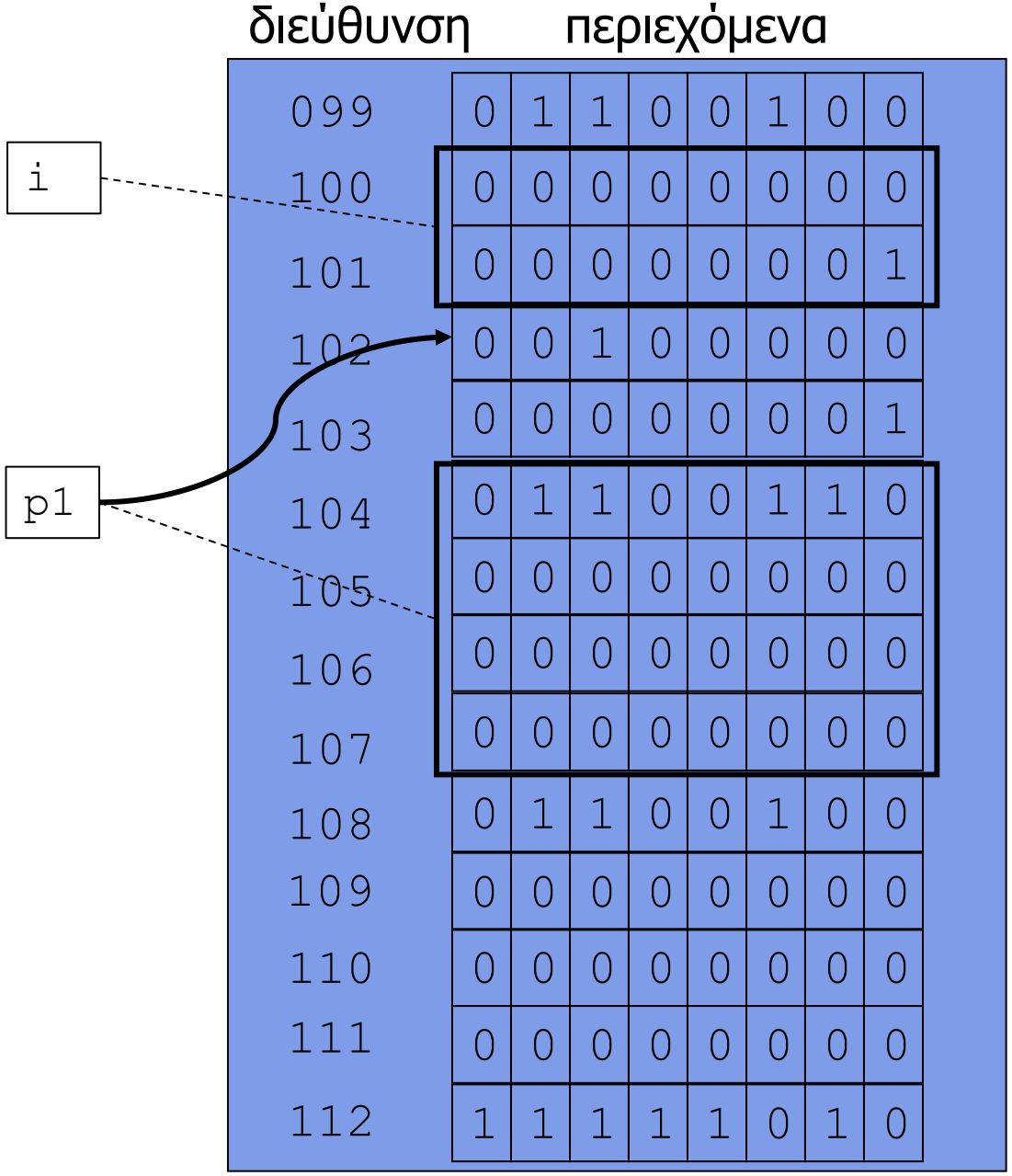
```
short int i;
…
short int *p1;

i = 255;

p1 = &i;

*p1 = 256;

p1++;
```



```
short int i;
…
short int *p1;

i = 255;

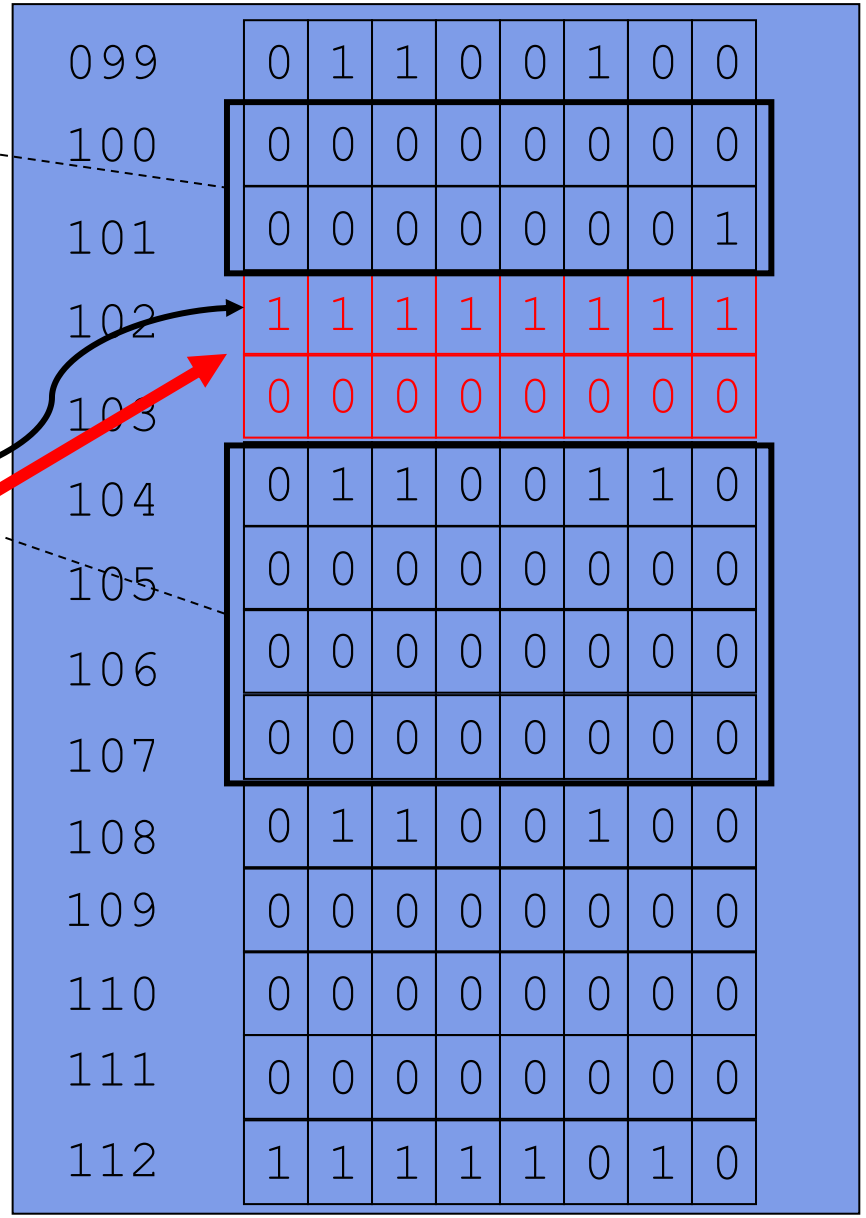
p1 = &i;

*p1 = 256;

p1++;

*p1 = 255;
```

αλλαγή μνήμης **εκτός** της μεταβλητής `i`



```
short int i;
…
short int *p1;

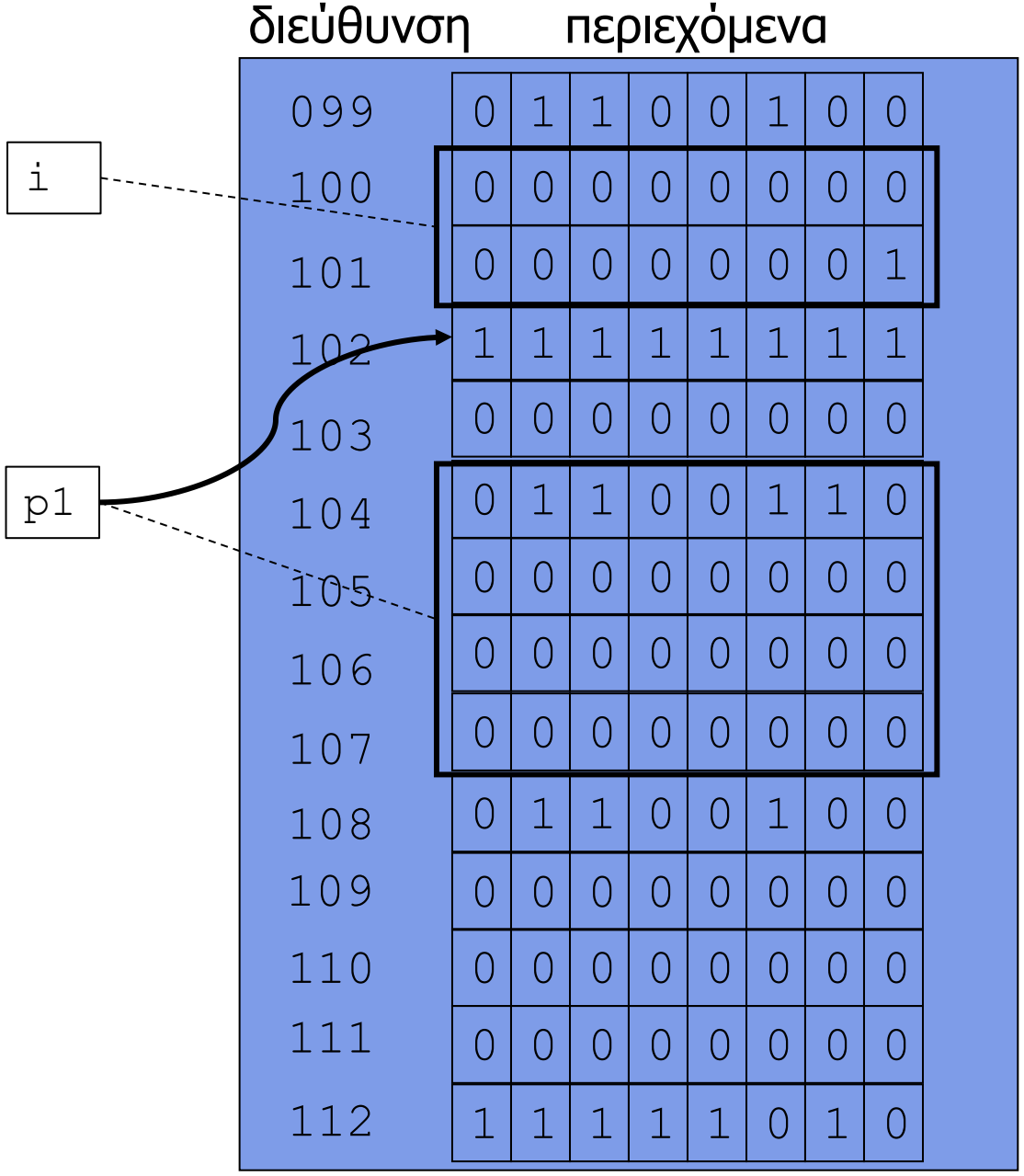
i = 255;

p1 = &i;

*p1 = 256;

p1++;

*p1 = 255;
```



```
short int i;
…
short int *p1;

i = 255;

p1 = &i;

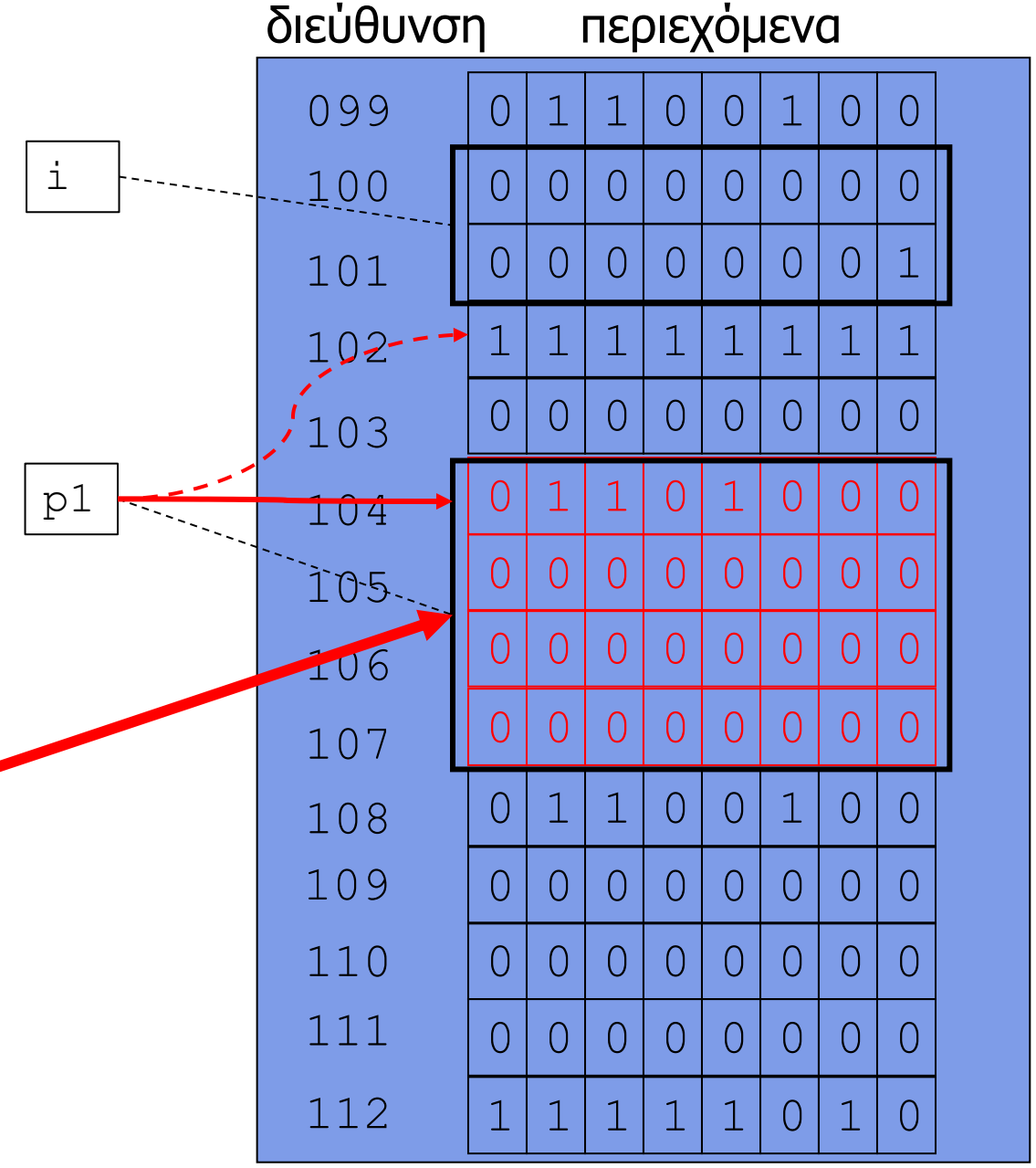
*p1 = 256;

p1++;

*p1 = 255;

p1++;
```

αλλαγή τιμής δείκτη
με βάση τους κανόνες
αριθμητικής δεικτών



```
short int i;
…
short int *p1;

i = 255;

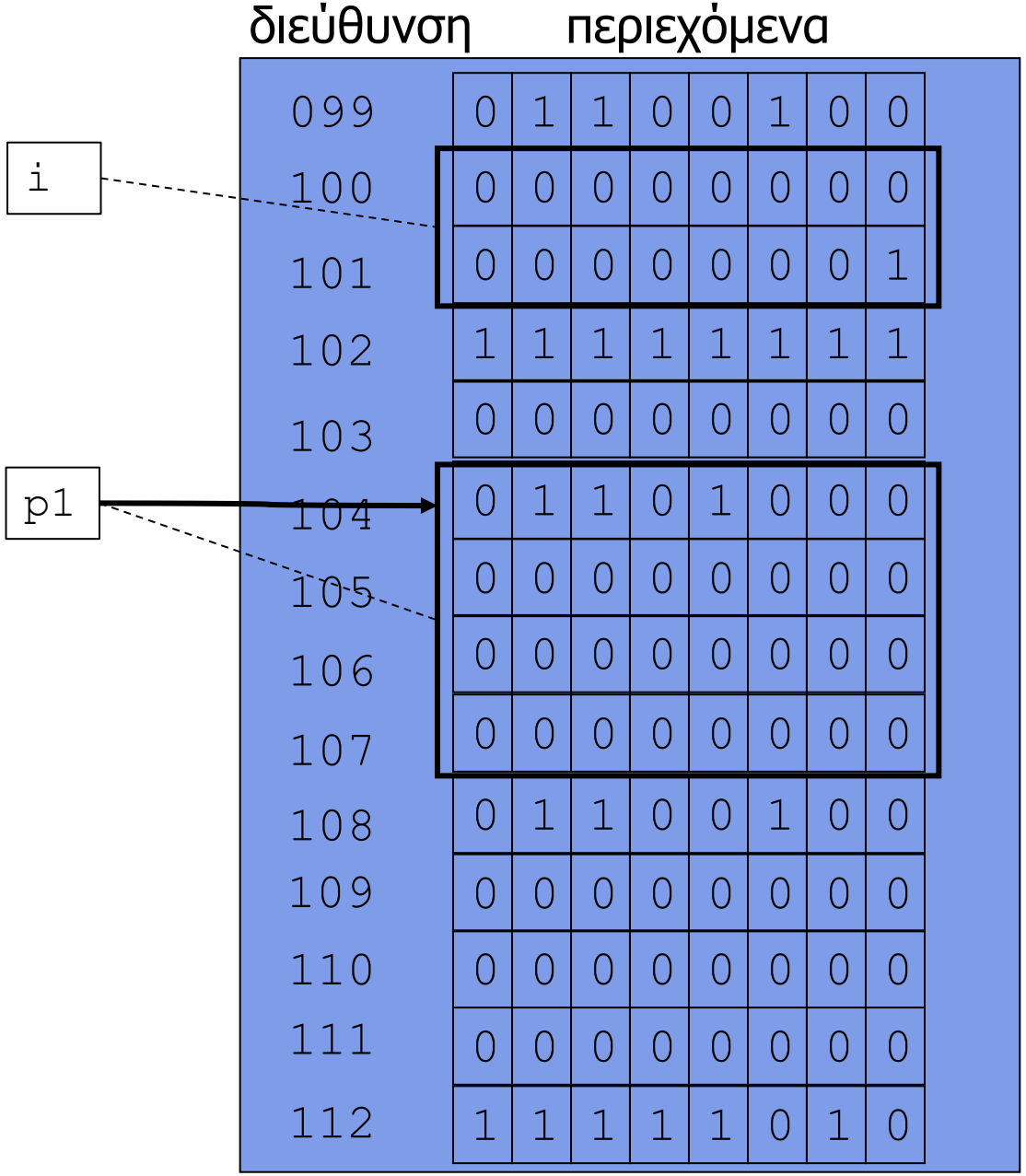
p1 = &i;

*p1 = 256;

p1++;

*p1 = 255;

p1++;
```



```
short int i;
…
short int *p1;

i = 255;

p1 = &i;

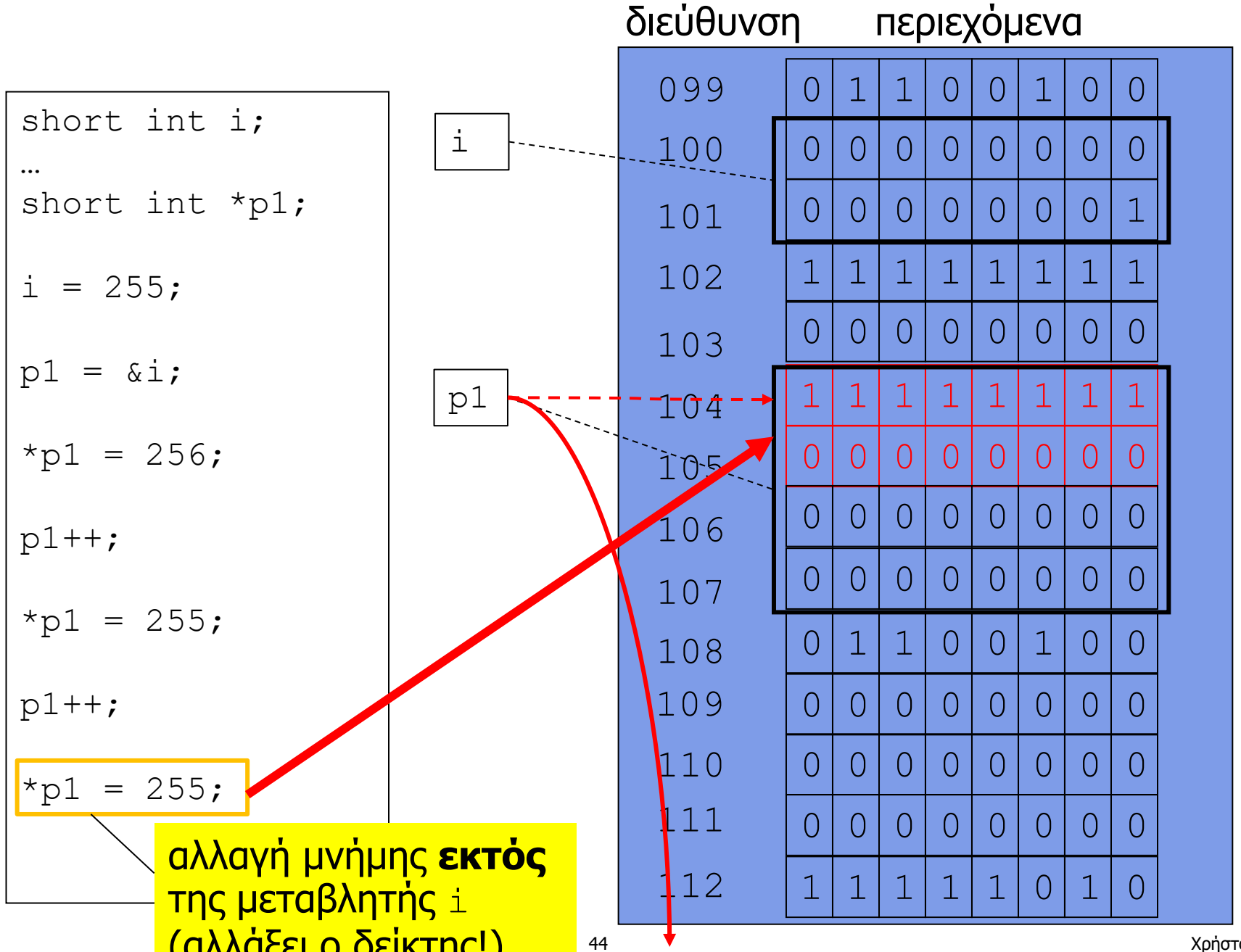
*p1 = 256;

p1++;

*p1 = 255;

p1++;

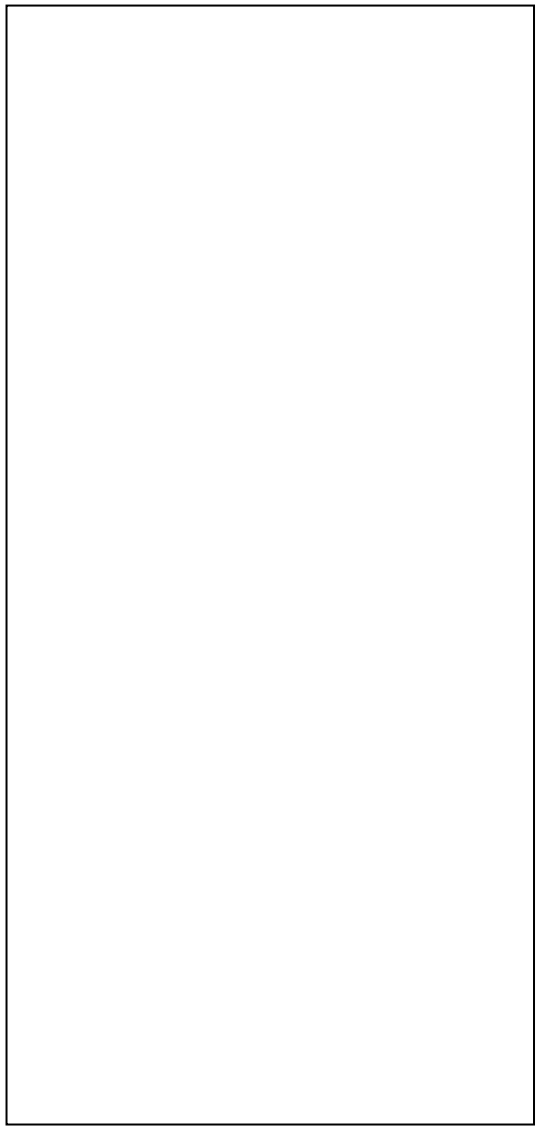
*p1 = 255;
```



αλλαγή μνήμης **εκτός** της μεταβλητής `i` (αλλάξει ο δείκτης!)

Το παράδειγμα που ακολουθεί, υποθέτει

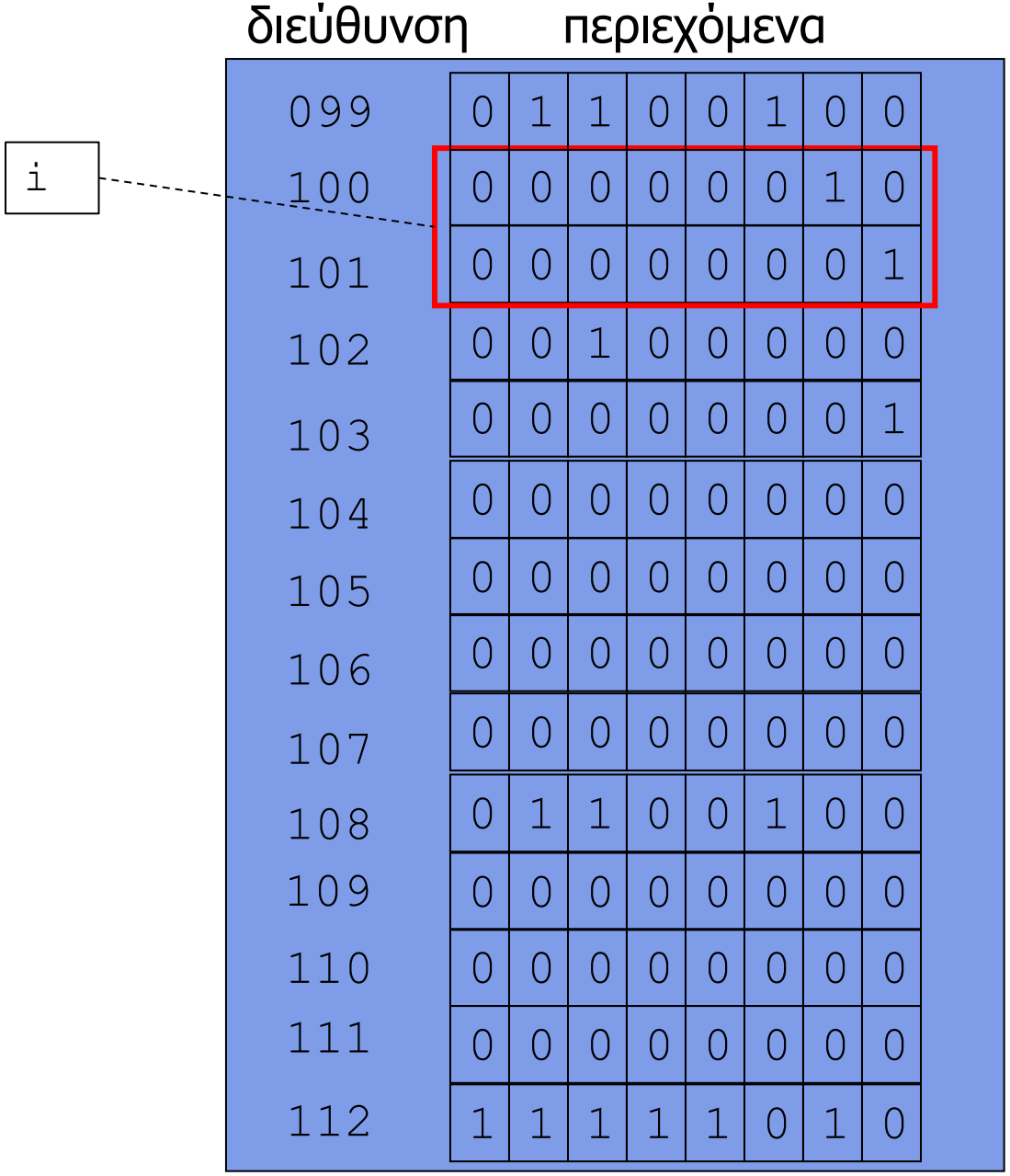
- μέγεθος `short int` 2 bytes
- μέγεθος δεικτών/pointers 4 bytes
- αποθήκευση αριθμητικών τιμών στην μνήμη με την σύμβαση little endian
  - τα λιγότερο σημαντικά bytes ενός multi-byte ακεραίου αποθηκεύονται στις μικρότερες διευθύνσεις μνήμης



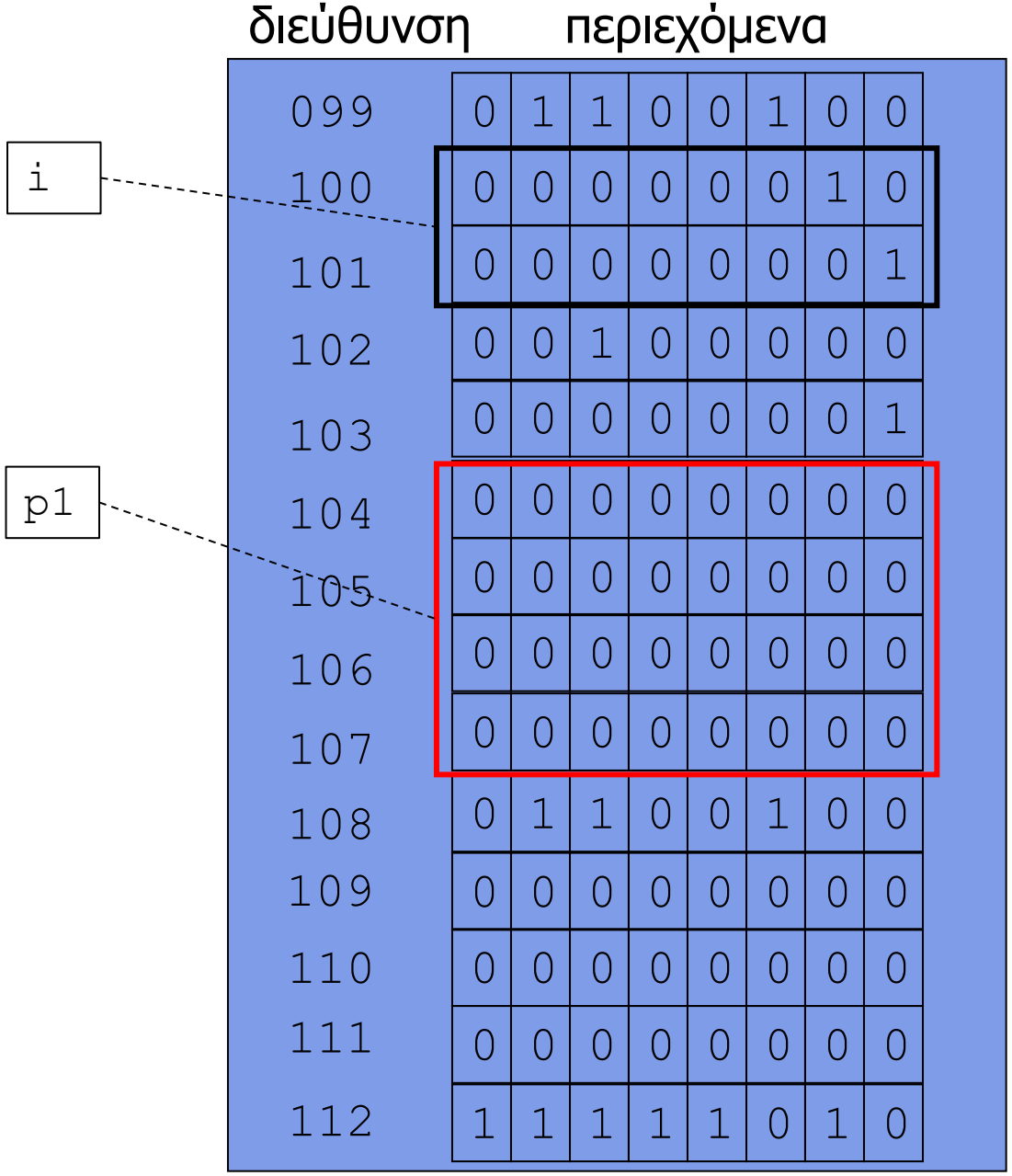
διεύθυνση περιεχόμενα

099	0	1	1	0	0	1	0	0
100	0	0	0	0	0	0	1	0
101	0	0	0	0	0	0	0	1
102	0	0	1	0	0	0	0	0
103	0	0	0	0	0	0	0	1
104	0	0	0	0	0	0	0	0
105	0	0	0	0	0	0	0	0
106	0	0	0	0	0	0	0	0
107	0	0	0	0	0	0	0	0
108	0	1	1	0	0	1	0	0
109	0	0	0	0	0	0	0	0
110	0	0	0	0	0	0	0	0
111	0	0	0	0	0	0	0	0
112	1	1	1	1	1	0	1	0

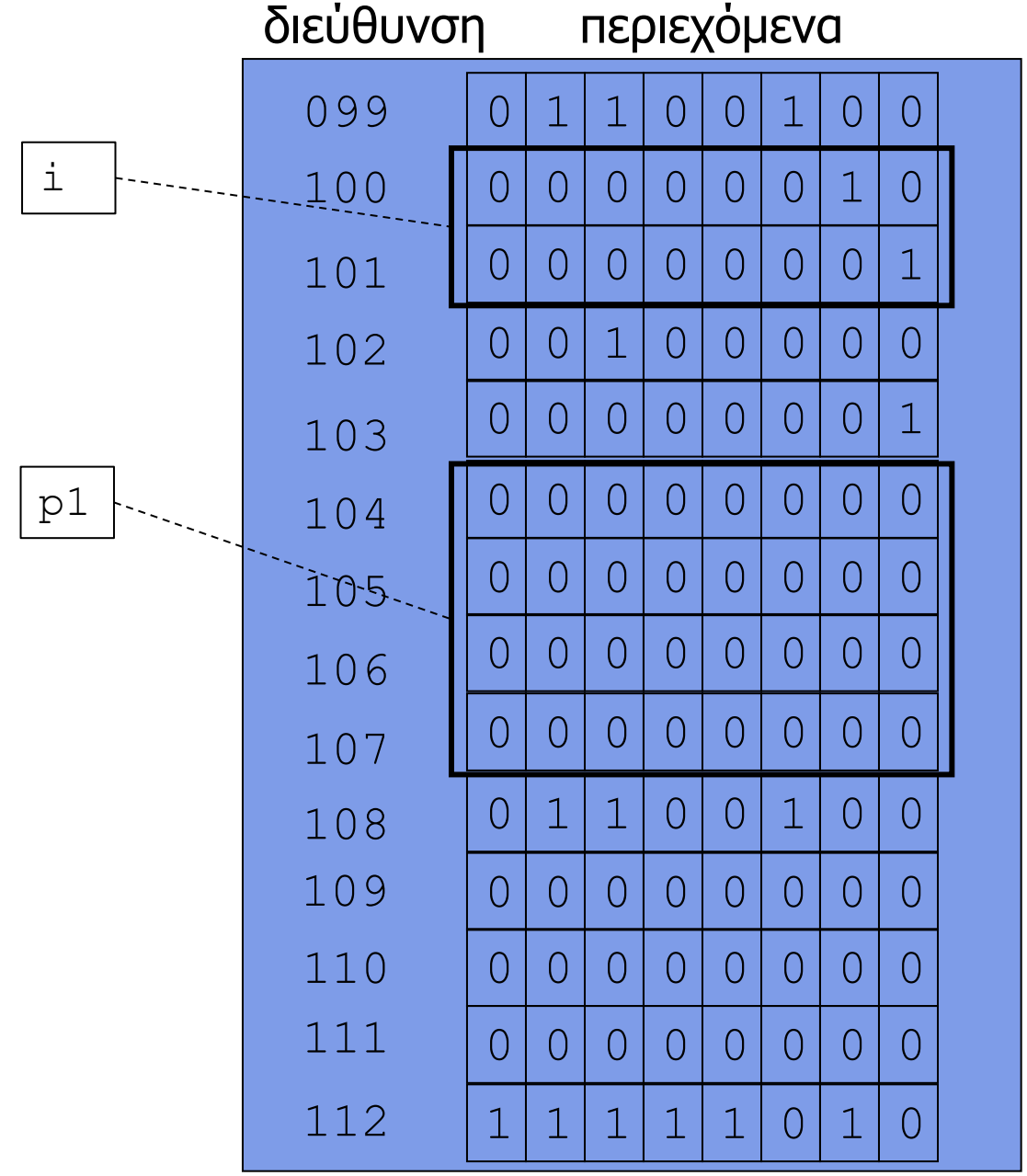
```
short int i;
```


διεύθυνση
περιεχόμενα
i
099 0 1 1 0 0 1 0 0
100 0 0 0 0 0 0 1 0
101 0 0 0 0 0 0 0 1
102 0 0 1 0 0 0 0 0
103 0 0 0 0 0 0 0 1
104 0 0 0 0 0 0 0 0
105 0 0 0 0 0 0 0 0
106 0 0 0 0 0 0 0 0
107 0 0 0 0 0 0 0 0
108 0 1 1 0 0 1 0 0
109 0 0 0 0 0 0 0 0
110 0 0 0 0 0 0 0 0
111 0 0 0 0 0 0 0 0
112 1 1 1 1 1 0 1 0

```
short int i;
…
char *p1;
```

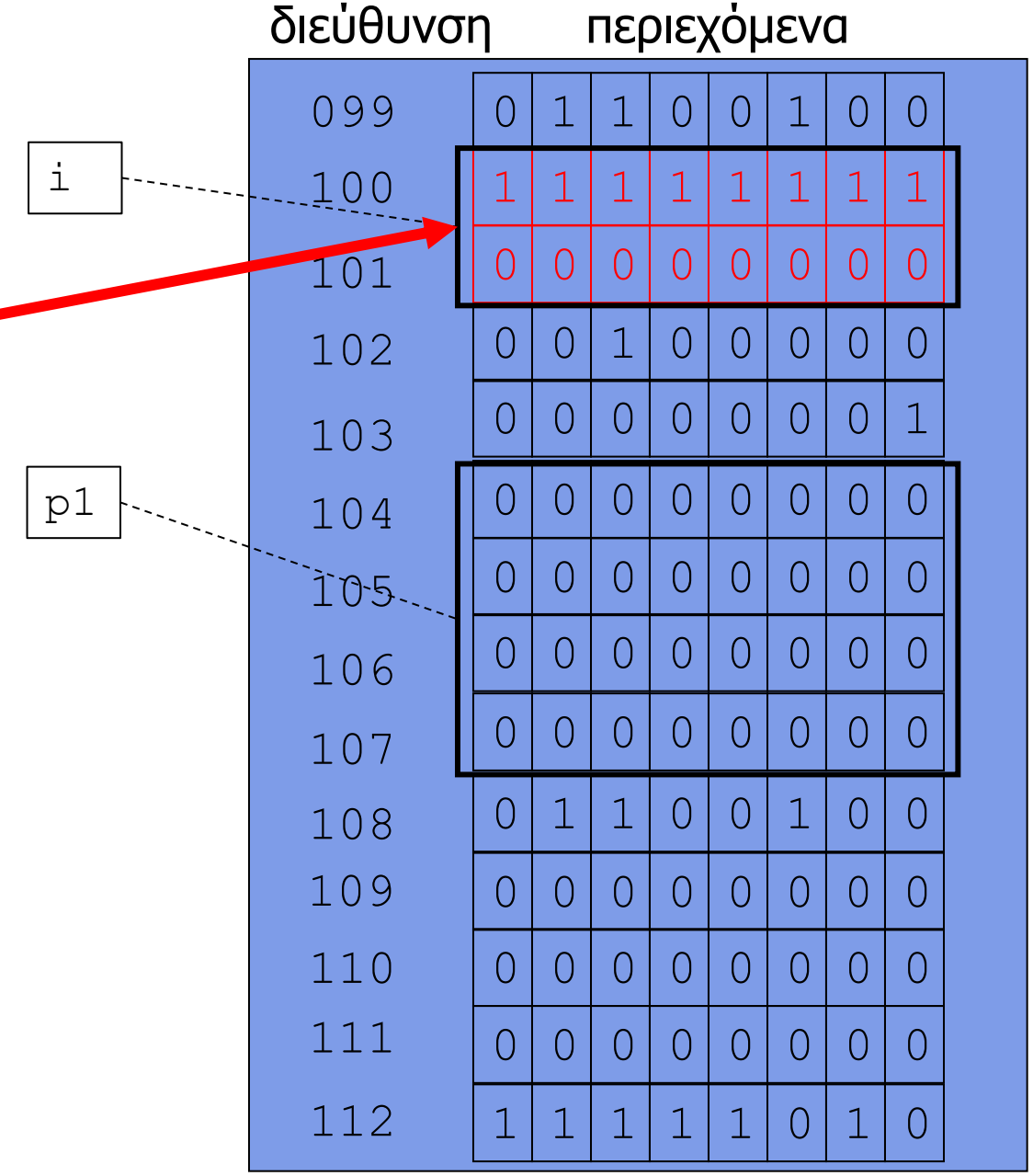


```
short int i;
…
char *p1;
```



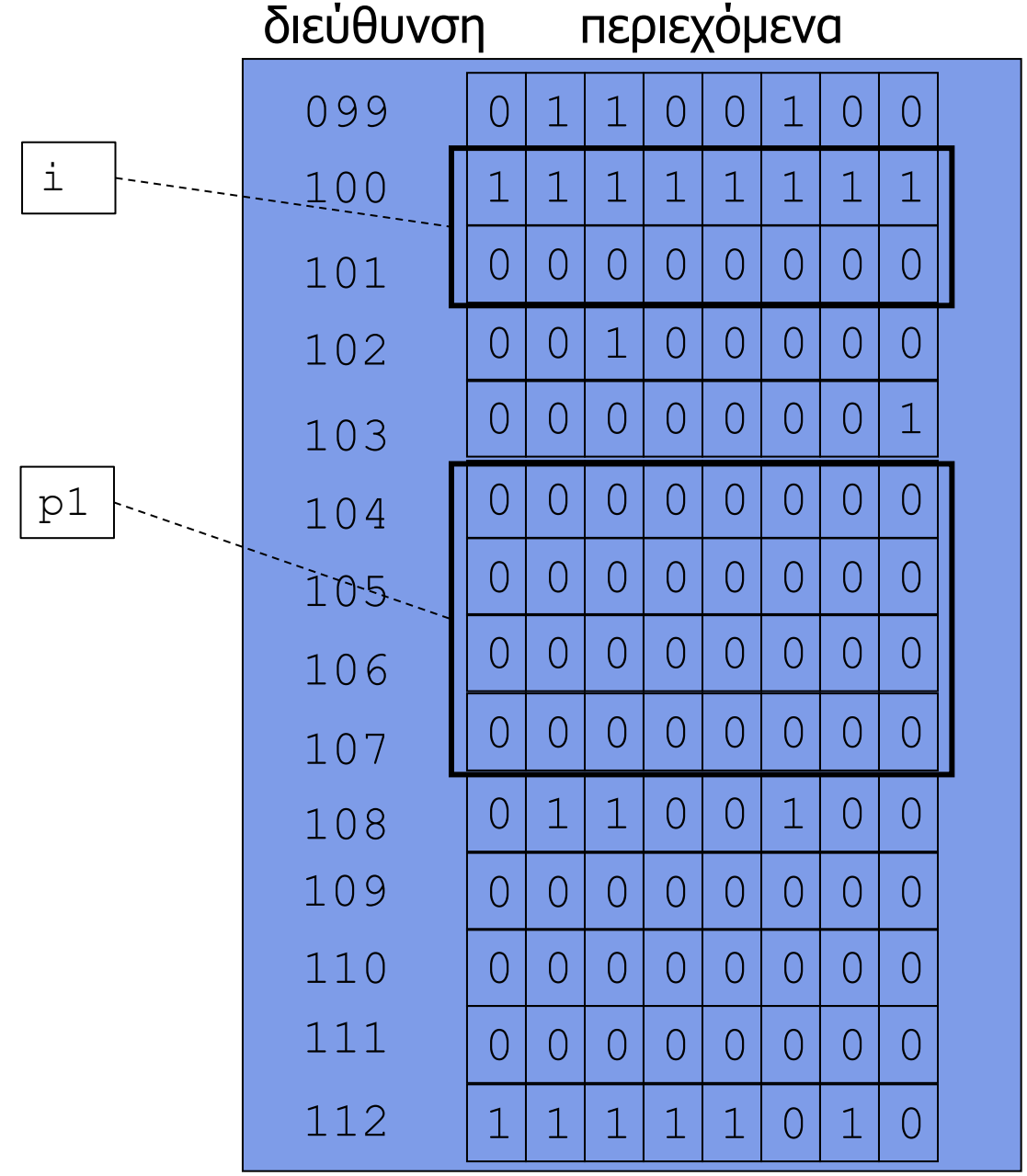
```
short int i;
…
char *p1;

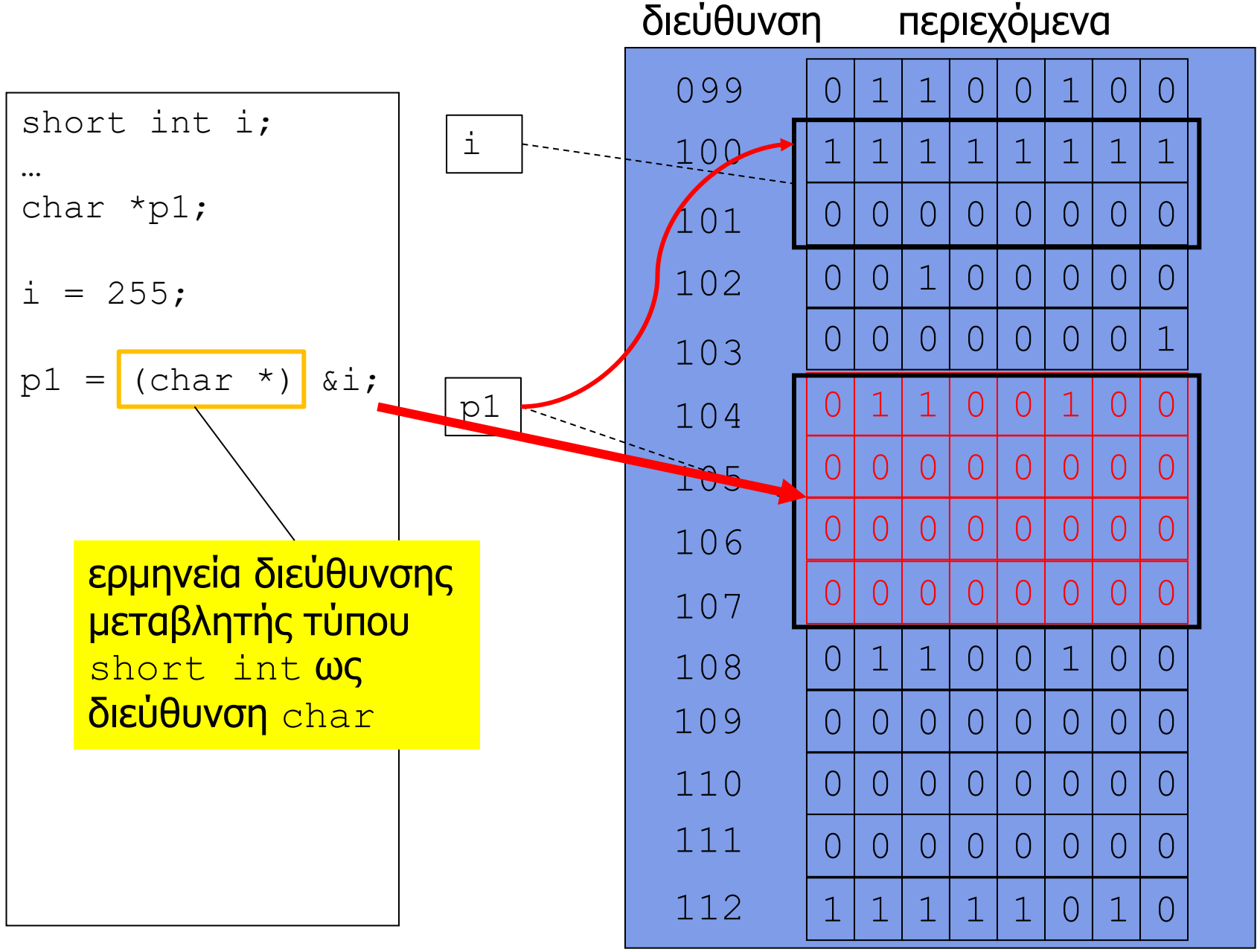
i = 255;
```



```
short int i;
…
char *p1;

i = 255;
```

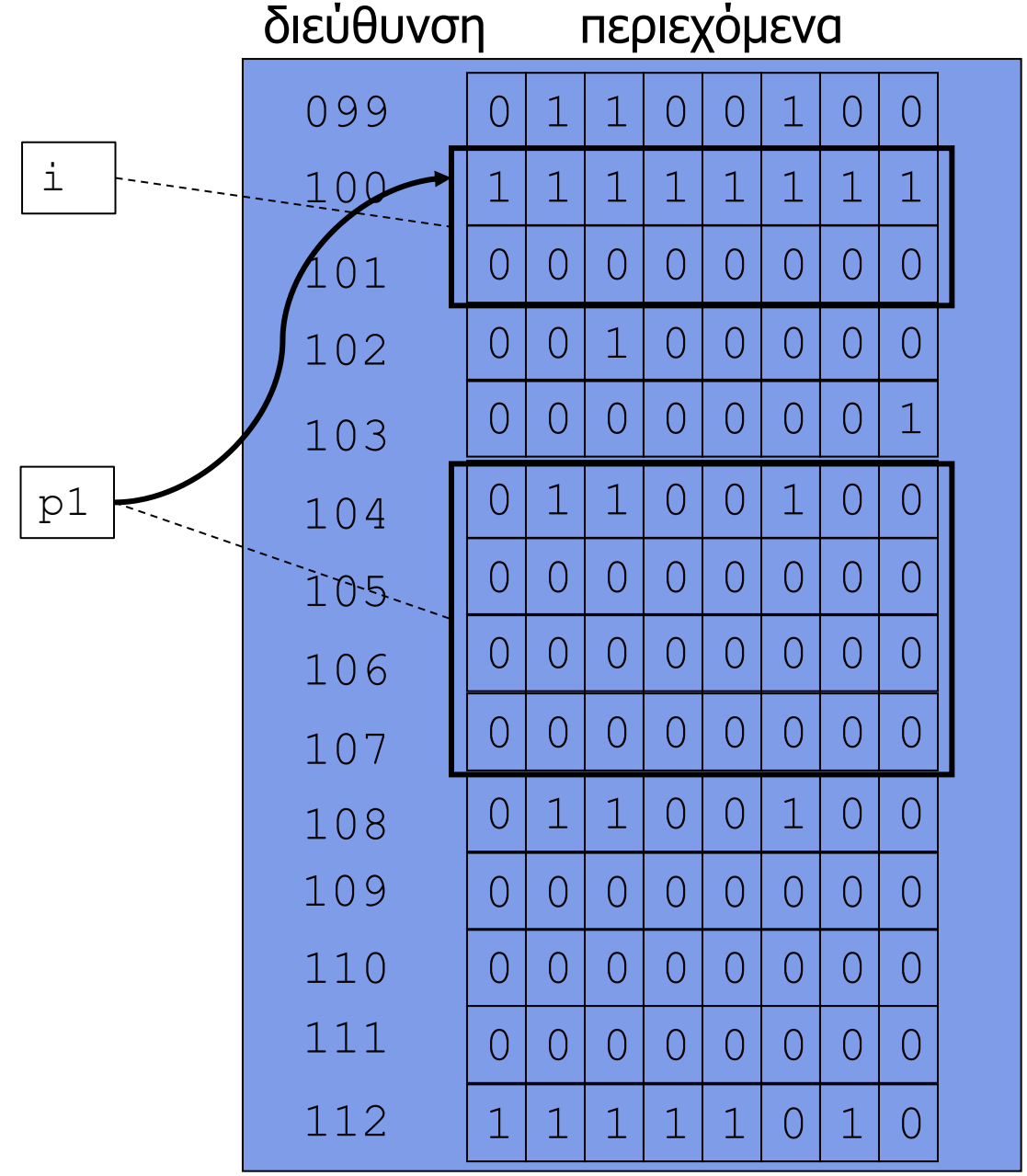



διεύθυνση
περιεχόμενα
short int i;
…
char *p1;
i = 255;
p1 = (char *) &i;
i
p1
099
100
101
102
103
104
105
106
107
108
109
110
111
112
0 1 1 0 0 1 0 0
1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1
0 1 1 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 1 0 0 1 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
1 1 1 1 1 0 1 0
ερμηνεία διεύθυνσης μεταβλητής τύπου short int ως διεύθυνση char

```
short int i;
…
char *p1;

i = 255;

p1 = (char *) &i;
```



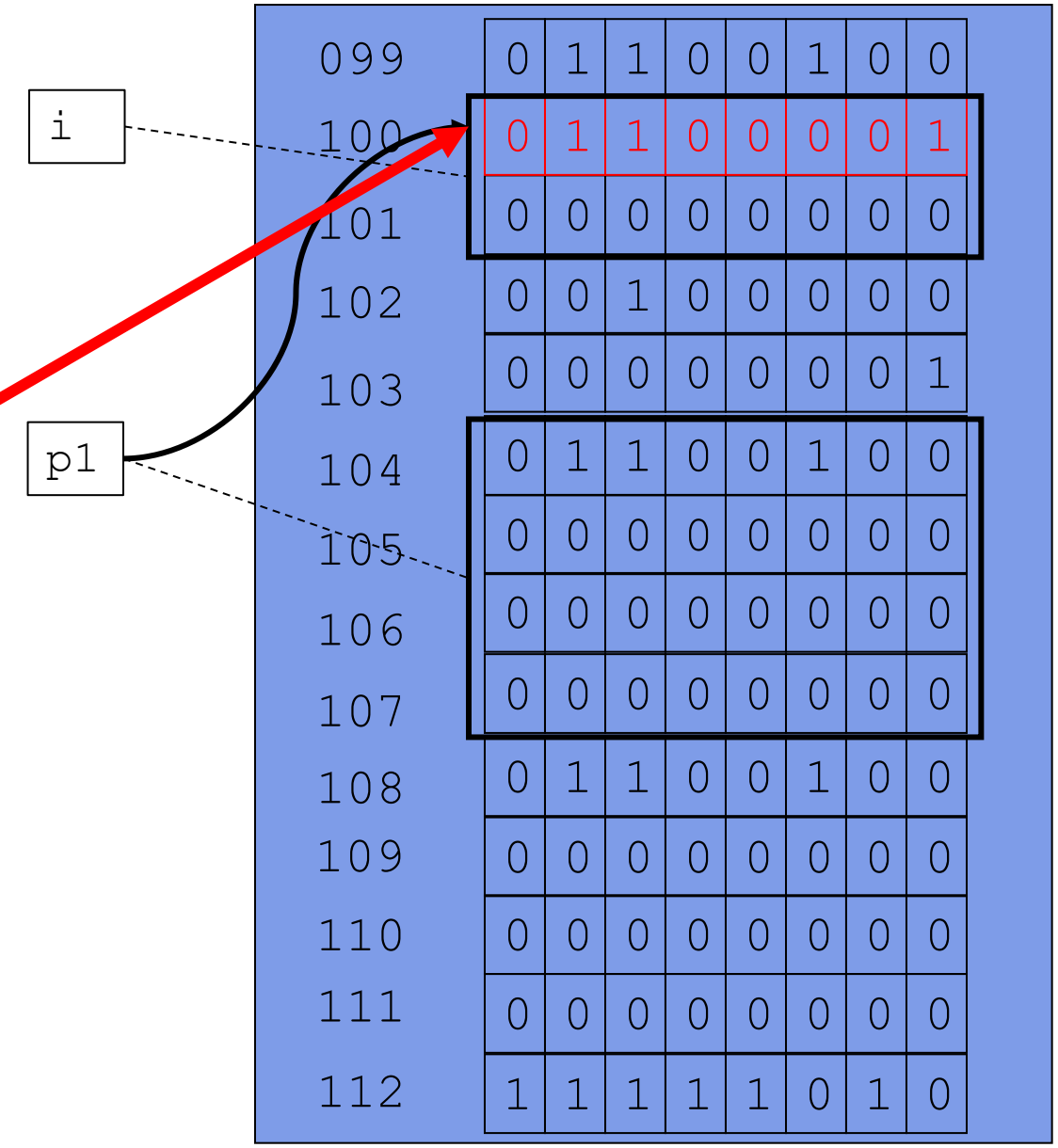
```
short int i;
…
char *p1;

i = 255;

p1 = (char *) &i;

*p1 = 'a';
```

έμμεση αλλαγή τιμής μεταβλητής

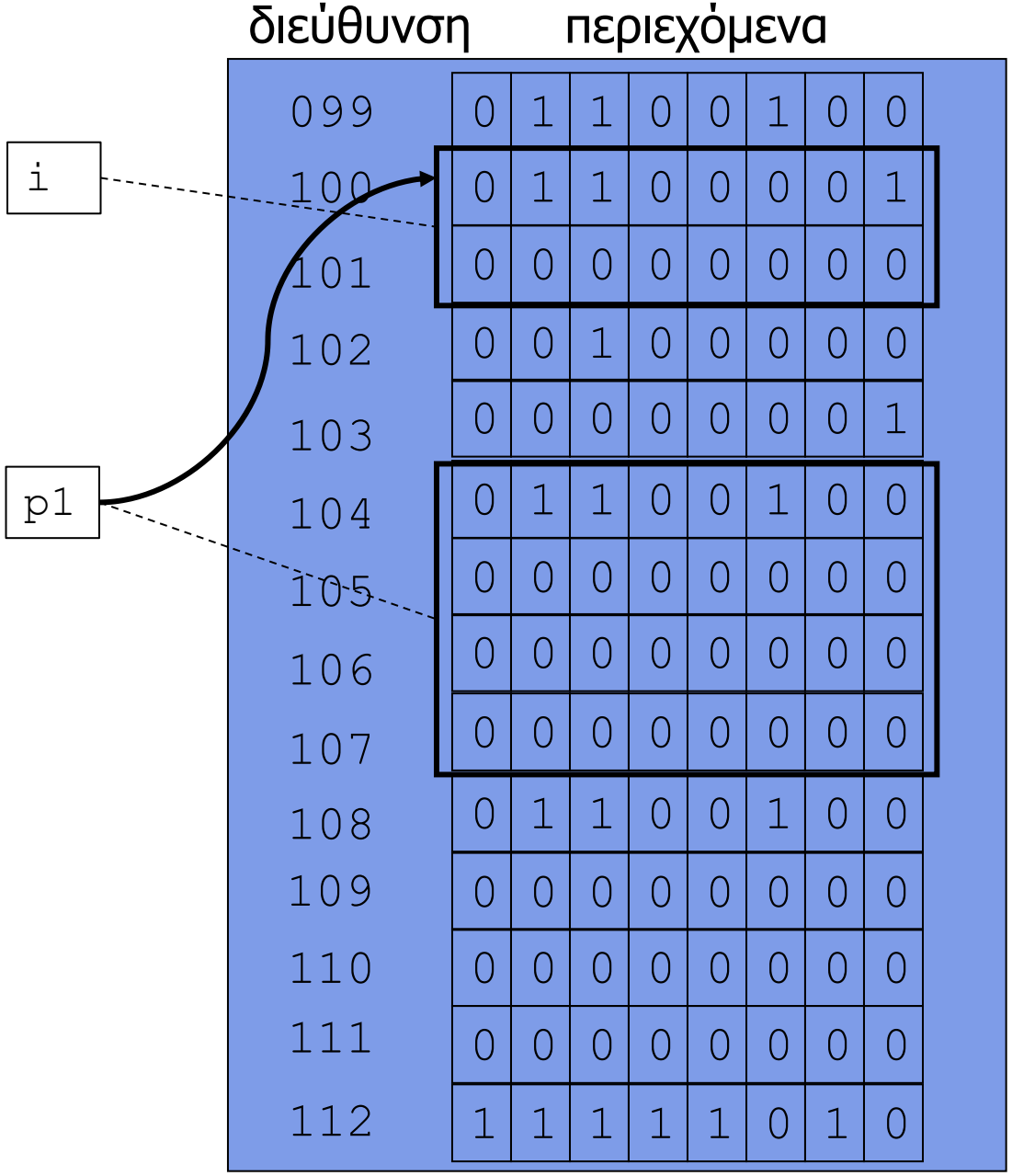


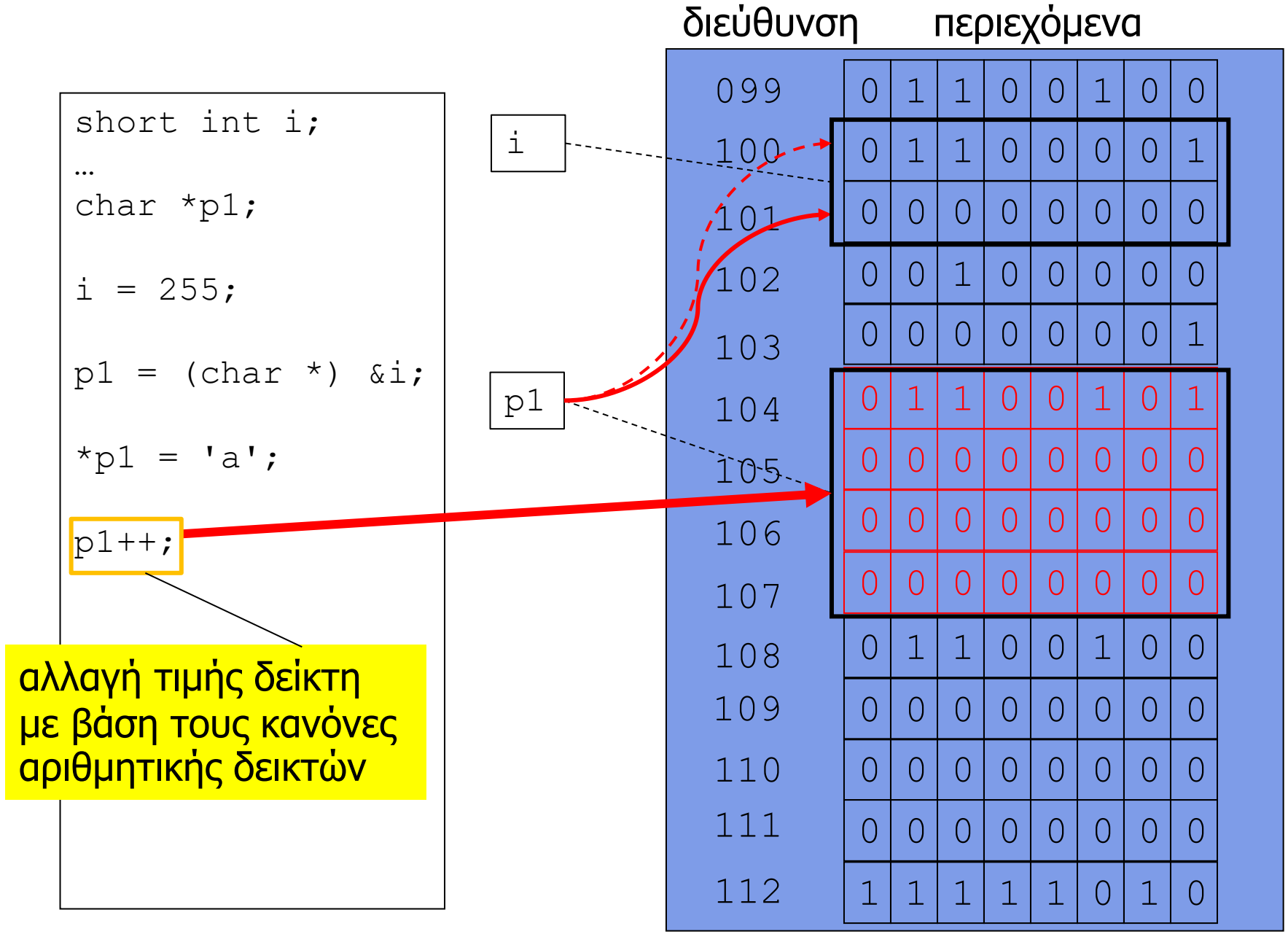
```
short int i;
…
char *p1;

i = 255;

p1 = (char *) &i;

*p1 = 'a';
```





αλλαγή τιμής δείκτη
με βάση τους κανόνες
αριθμητικής δεικτών

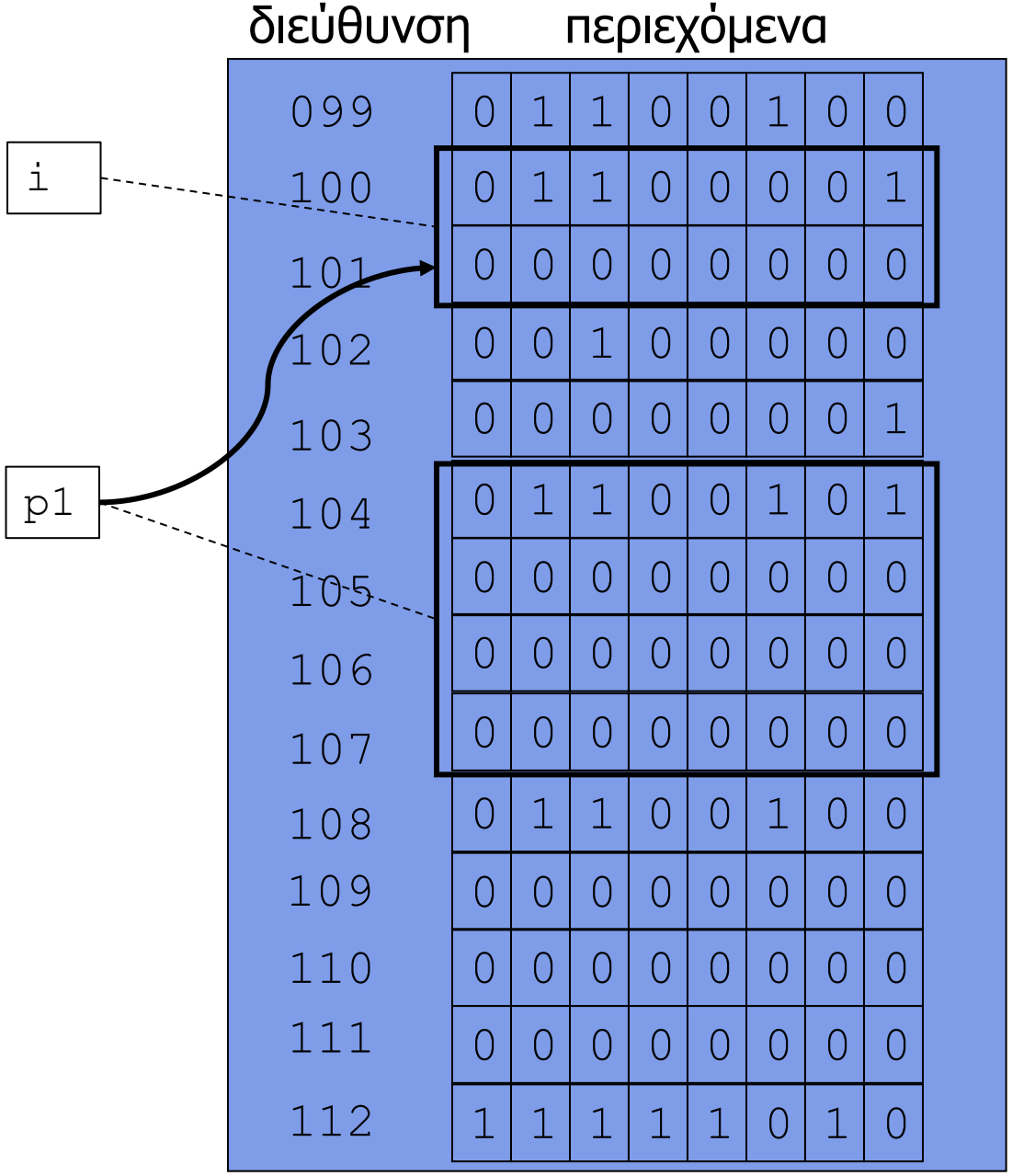
```
short int i;
…
char *p1;

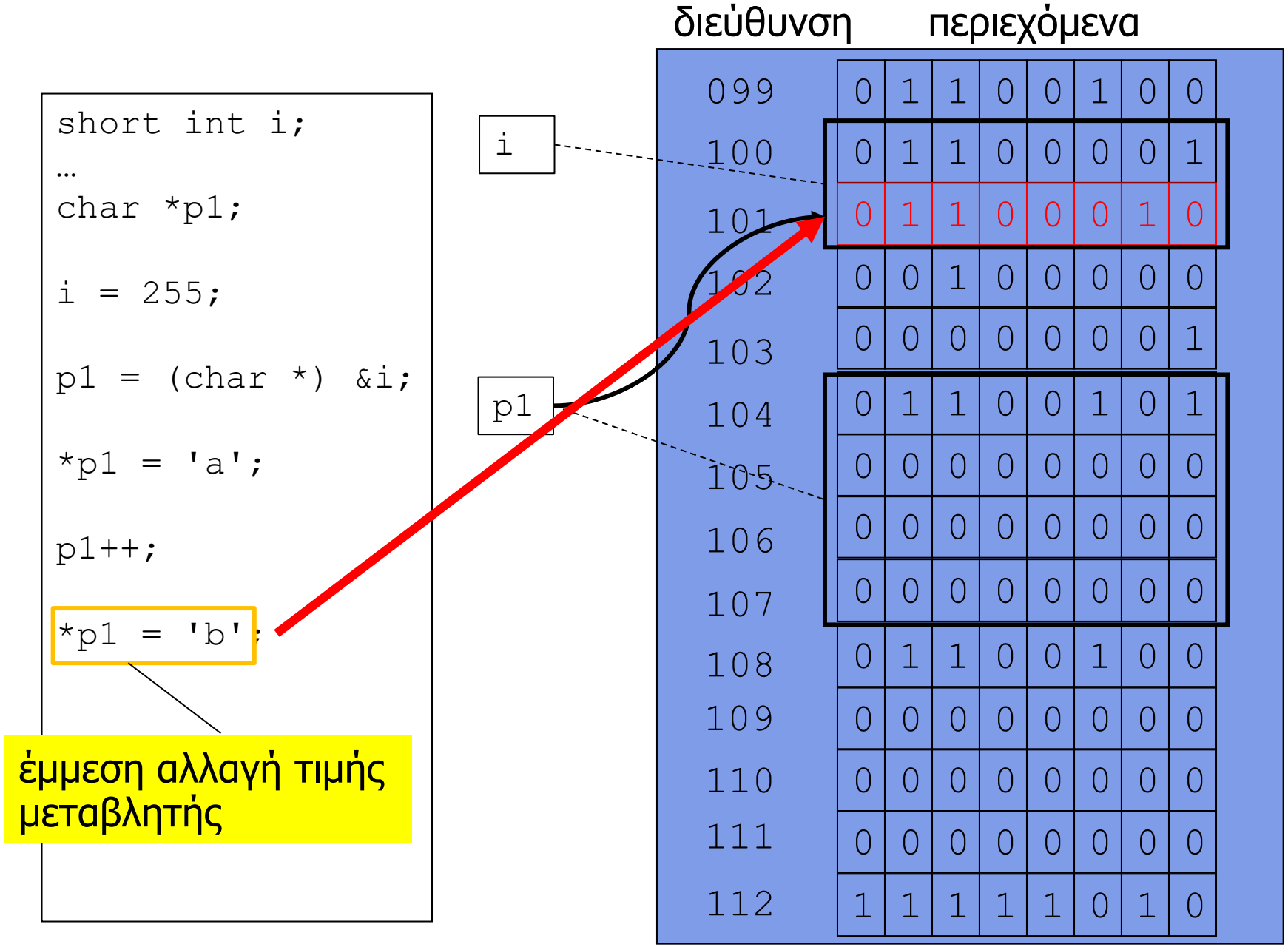
i = 255;

p1 = (char *) &i;

*p1 = 'a';

p1++;
```




διεύθυνση
περιεχόμενα
short int i;
…
char *p1;
i = 255;
p1 = (char *) &i;
*p1 = 'a';
p1++;
*p1 = 'b';
i
p1
099 0 1 1 0 0 1 0 0
100 0 1 1 0 0 0 0 1
101 0 1 1 0 0 0 1 0
102 0 0 1 0 0 0 0 0
103 0 0 0 0 0 0 0 1
104 0 1 1 0 0 1 0 1
105 0 0 0 0 0 0 0 0
106 0 0 0 0 0 0 0 0
107 0 0 0 0 0 0 0 0
108 0 1 1 0 0 1 0 0
109 0 0 0 0 0 0 0 0
110 0 0 0 0 0 0 0 0
111 0 0 0 0 0 0 0 0
112 1 1 1 1 1 0 1 0
έμμεση αλλαγή τιμής μεταβλητής

```
short int i;
…
char *p1;

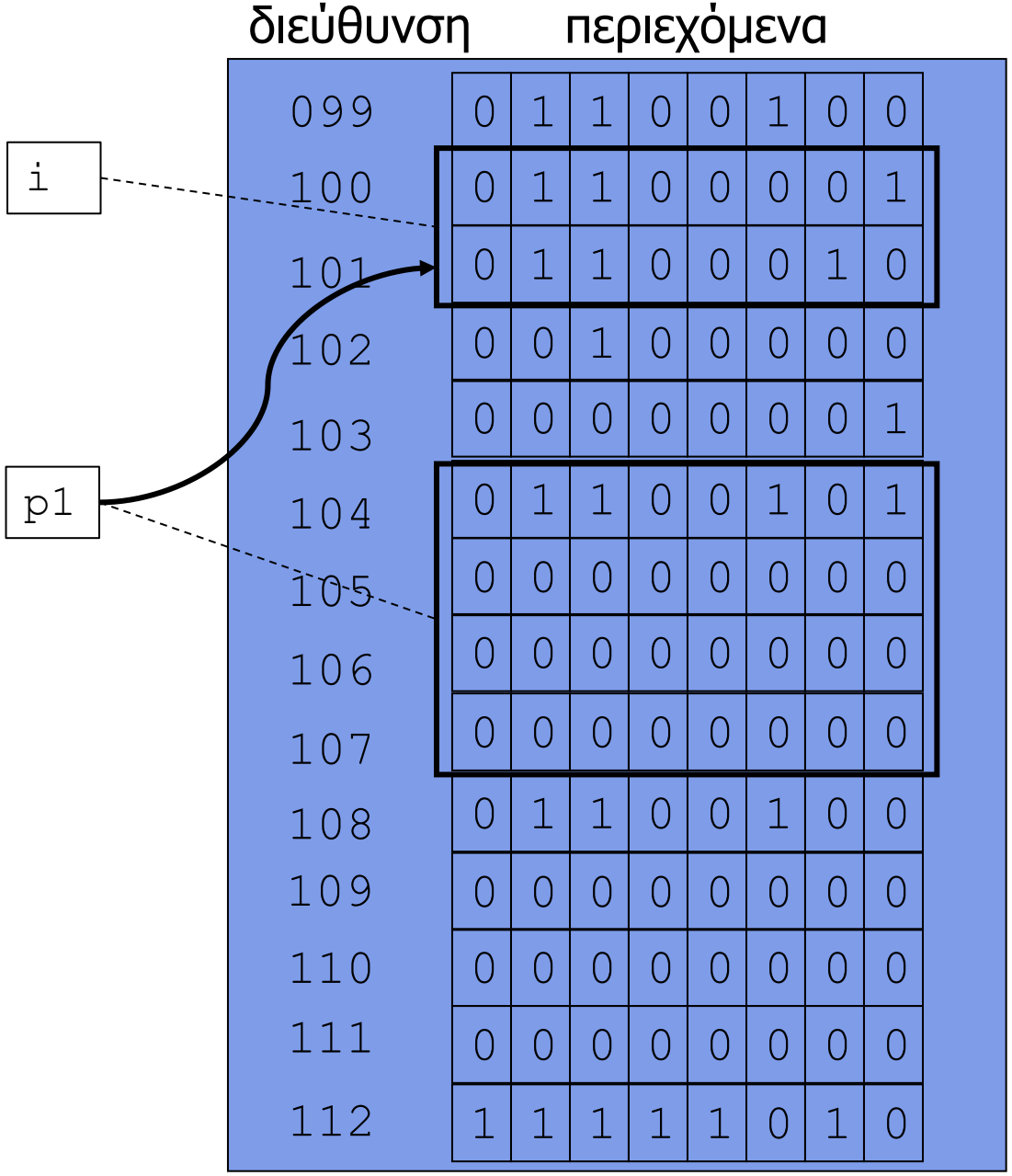
i = 255;

p1 = (char *) &i;

*p1 = 'a';

p1++;

*p1 = 'b';
```



```
short int i;
…
char *p1;

i = 255;

p1 = (char *) &i;

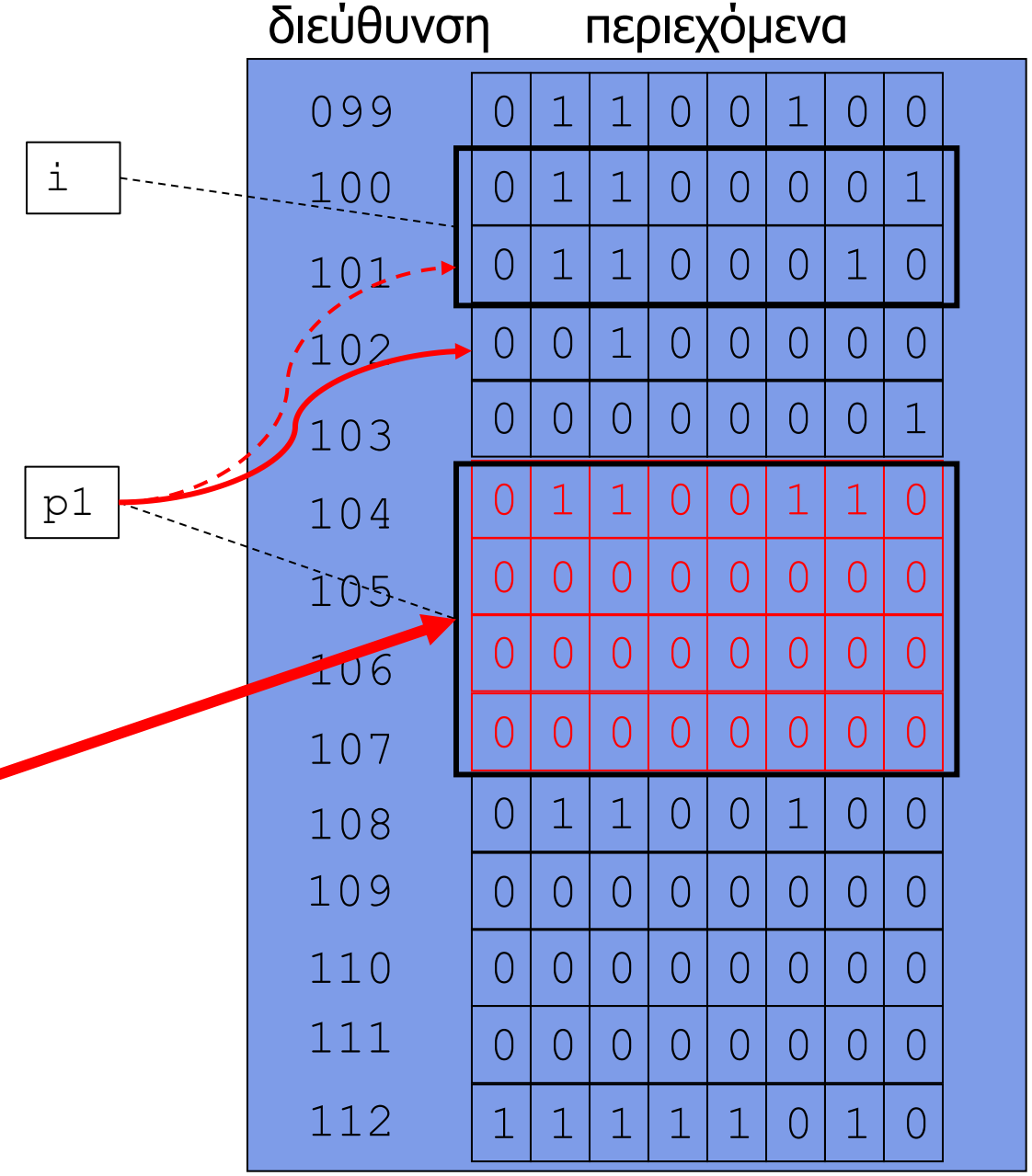
*p1 = 'a';

p1++;

*p1 = 'b';

p1++;
```

αλλαγή τιμής δείκτη
με βάση τους κανόνες
αριθμητικής δεικτών



```
short int i;
…
char *p1;

i = 255;

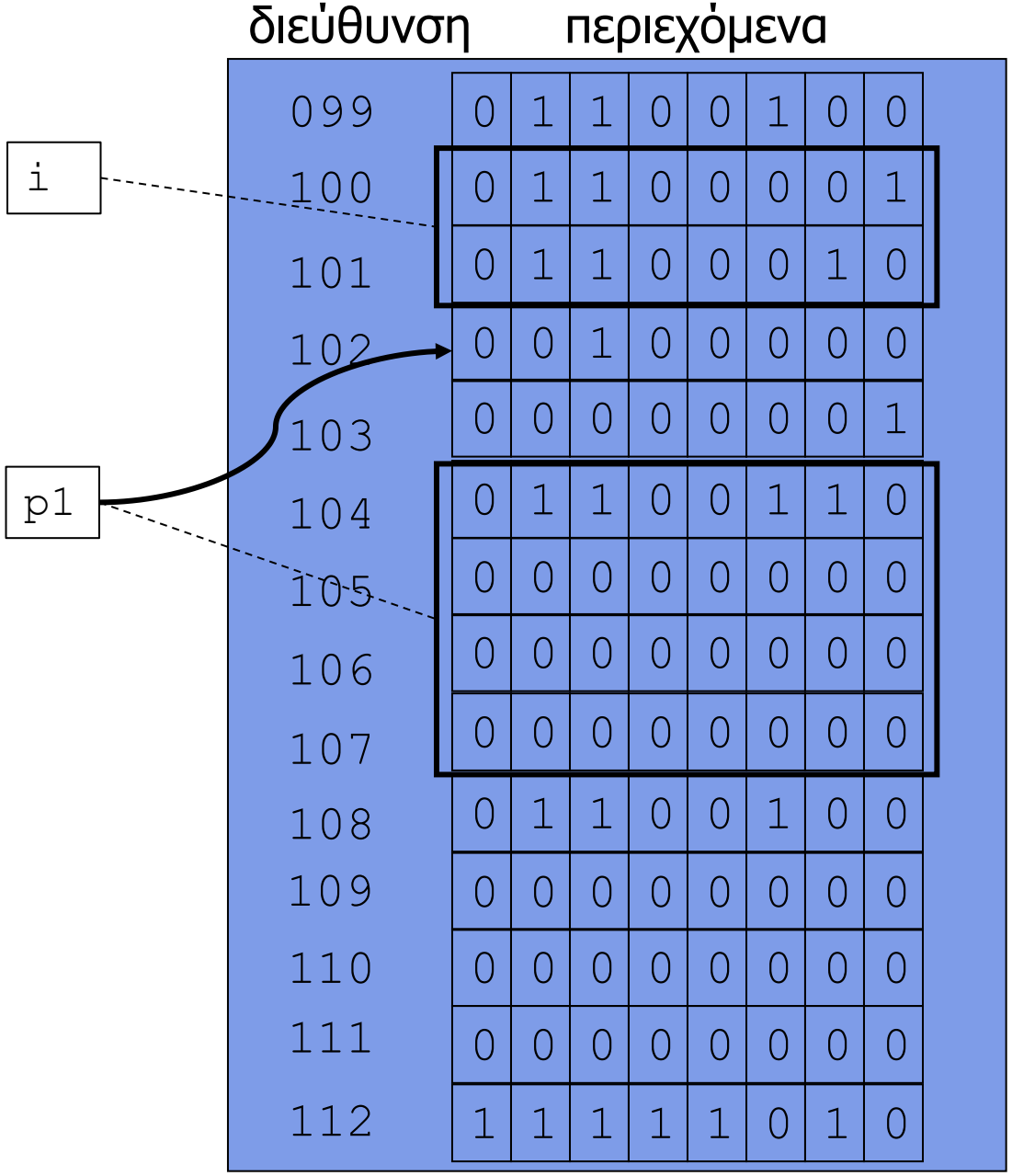
p1 = (char *) &i;

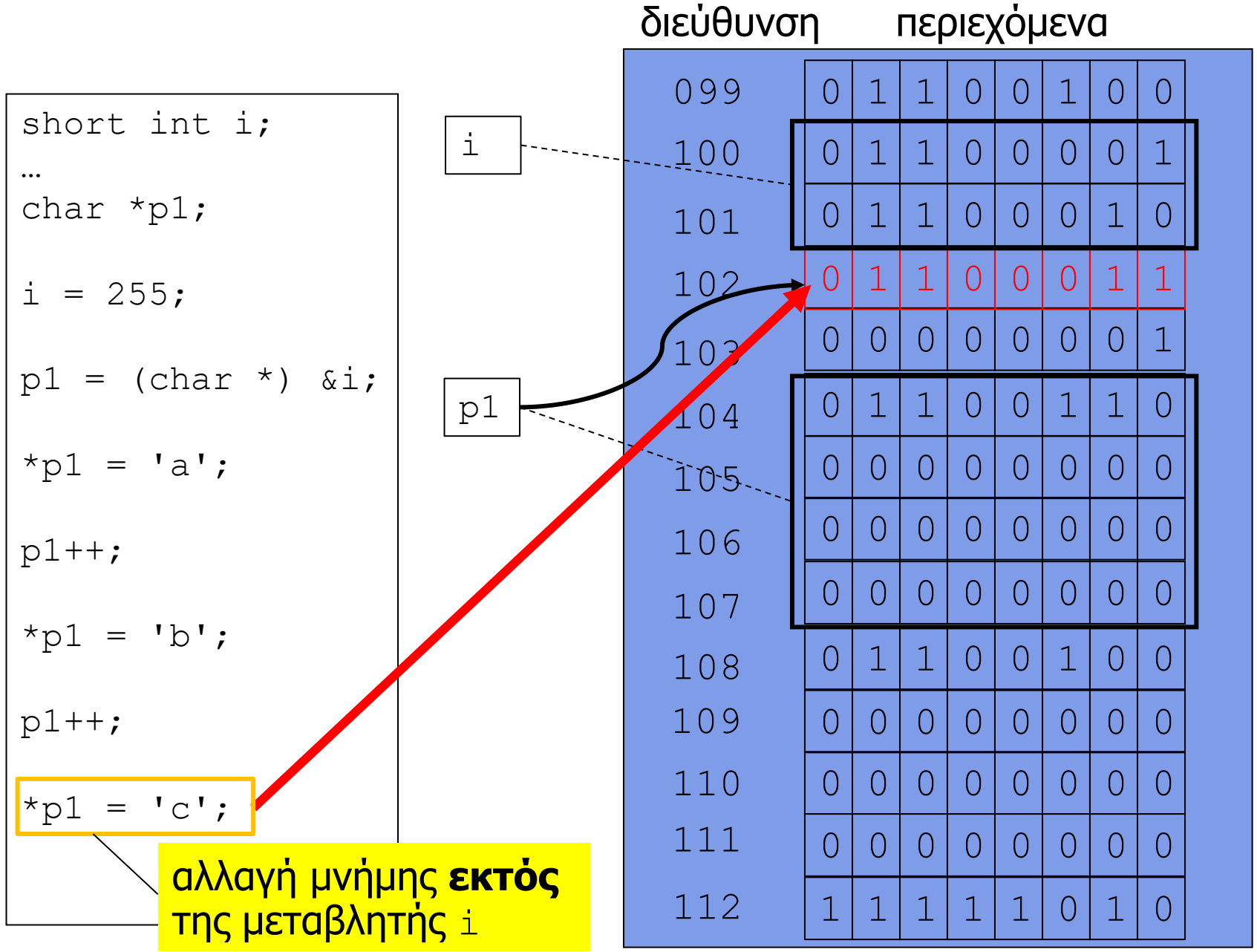
*p1 = 'a';

p1++;

*p1 = 'b';

p1++;
```




διεύθυνση
περιεχόμενα
short int i;
…
char *p1;
i = 255;
p1 = (char *) &i;
*p1 = 'a';
p1++;
*p1 = 'b';
p1++;
*p1 = 'c';
i
p1
099 0 1 1 0 0 1 0 0
100 0 1 1 0 0 0 0 1
101 0 1 1 0 0 0 1 0
102 0 1 1 0 0 0 1 1
103 0 0 0 0 0 0 0 1
104 0 1 1 0 0 1 1 0
105 0 0 0 0 0 0 0 0
106 0 0 0 0 0 0 0 0
107 0 0 0 0 0 0 0 0
108 0 1 1 0 0 1 0 0
109 0 0 0 0 0 0 0 0
110 0 0 0 0 0 0 0 0
111 0 0 0 0 0 0 0 0
112 1 1 1 1 1 0 1 0
αλλαγή μνήμης **εκτός** της μεταβλητής i

```
short int i;
…
char *p1;

i = 255;

p1 = (char *) &i;

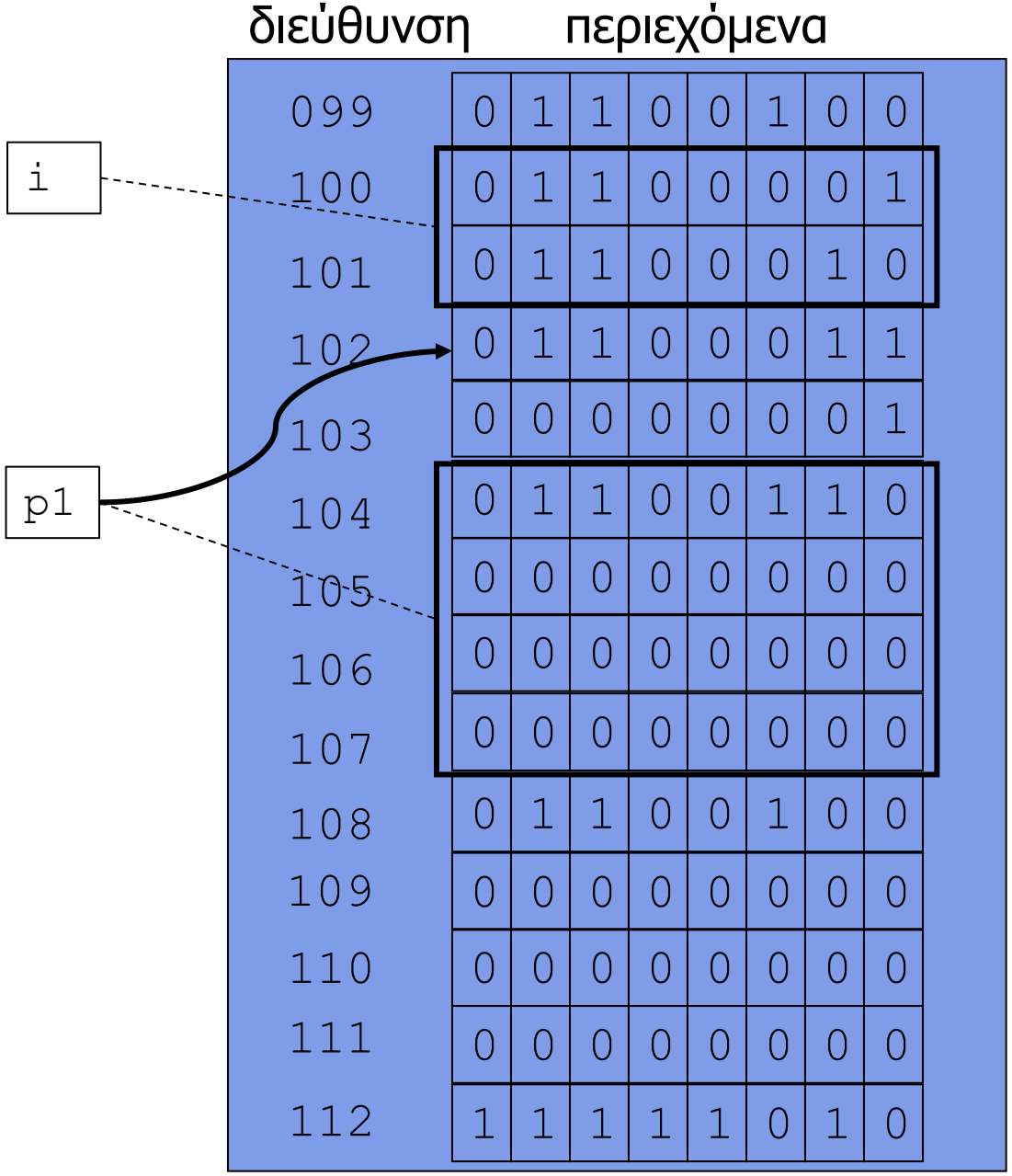
*p1 = 'a';

p1++;

*p1 = 'b';

p1++;

*p1 = 'c';
```



Προσοχή, προσοχή, προσοχή …

- Η ανάθεση τιμών σε μεταβλητές δείκτες γίνεται με **αποκλειστική ευθύνη** του προγραμματιστή
  - δεν γίνεται (ολοκληρωμένος) έλεγχος από τον μεταφραστή
- Το παραμικρό λάθος όταν χρησιμοποιούμε δείκτες, μπορεί να οδηγήσει σε **απρόβλεπτα αποτελέσματα**
  - μπορεί κατά λάθος να αλλάξουμε/καταστρέψουμε τιμές δεδομένων / μεταβλητών (και δεικτών) του προγράμματος
- Τέτοια σφάλματα εντοπίζονται **πολύ** δύσκολα
  - αν είμαστε «τυχεροί», οδηγούν σε προσπέλαση άκυρης διεύθυνσης μνήμης ή άμεση καταστροφή της στοίβας
  - τότε το πρόγραμμα θα τερματιστεί απότομα, και έτσι θα καταλάβουμε ότι κάτι δεν πάει καλά