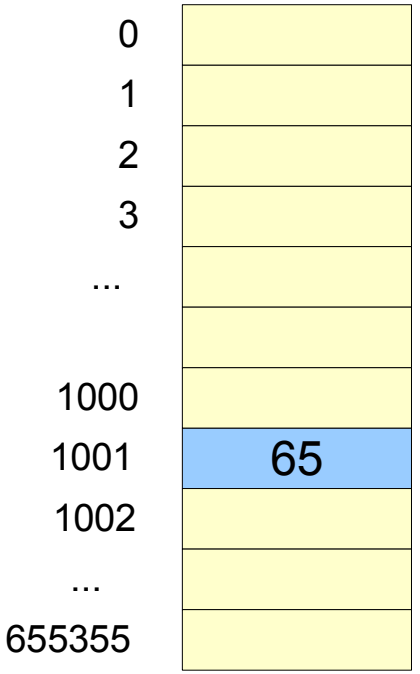
Δείκτες

Μεταβλητές

- Έστω η μεταβλητή

```
char ch ='A';
```

- Η παραπάνω δήλωση δεσμεύει χώρο στη μνήμη για τη μεταβλητή **ch**.
- Η τιμή της ch αντιστοιχεί στο περιεχόμενο μιας θέσης μνήμης, π.χ. της 1001.



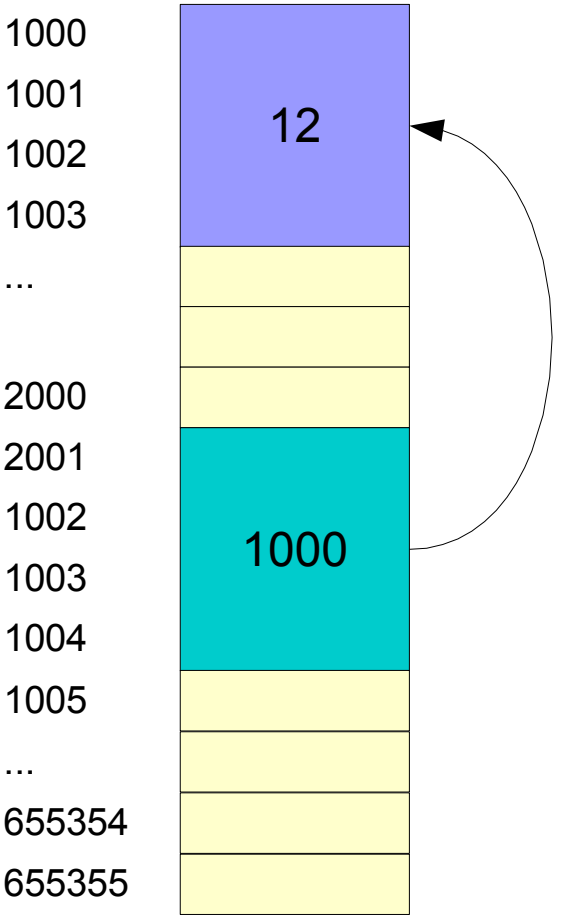
Δείκτες

- Στη C επιτρέπονται μεταβλητές που περιέχουν τη διεύθυνση μιας άλλης μεταβλητής.

```
int i = 12;
```

- Η δήλωση ενός δείκτη γίνεται ως εξής στη θέση μνήμης της μεταβλητής i γίνεται ως εξής:

```
int *p;
p = &i;
```



Δήλωση δεικτών

τύπος_βάσης * *μεταβλητή_δείκτη*;

όπου

τύπος_βάσης ο τύπος της μεταβλητής που δείχνει ο δείκτης

μεταβλητή_δείκτη είναι το όνομα της μεταβλητής που δηλώνεται

```
int *p;

char *cptr;
```

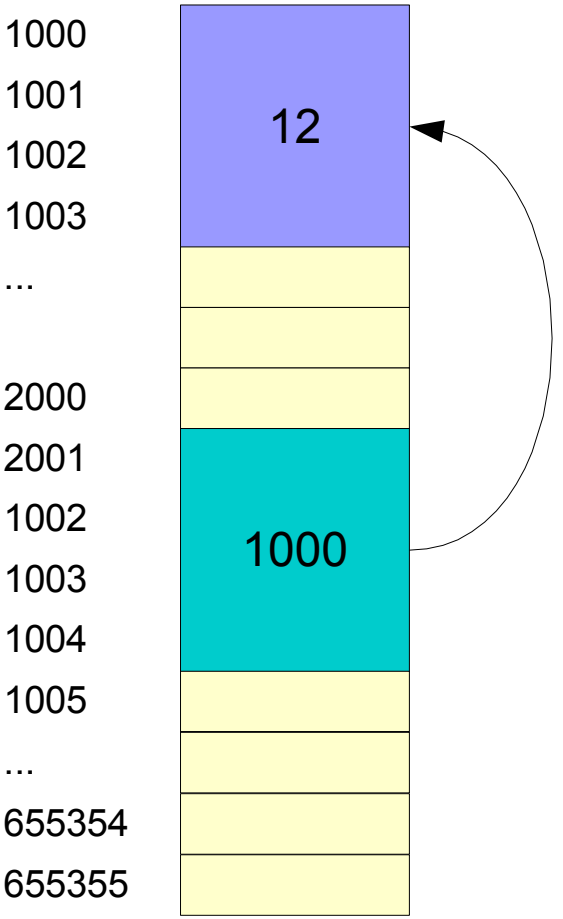
Τελεστές για το χειρισμό δεικτών

& διεύθυνση

* τιμή στην οποία δείχνει

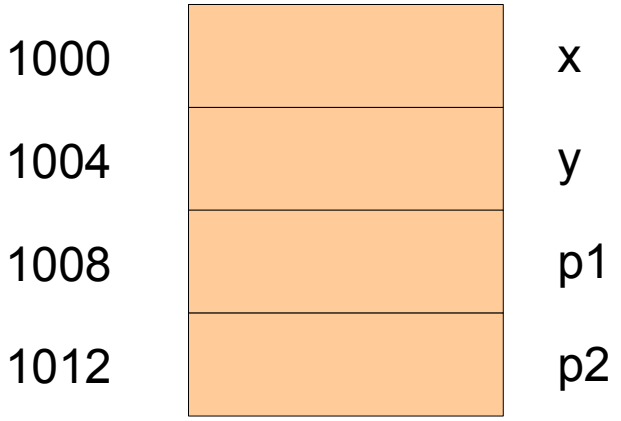
Παράδειγμα

```
int *p;
p = &i;

Η p έχει την τιμή 1000
Η *p "    "   "    12
```



΄Ενα ακόμη παράδειγμα

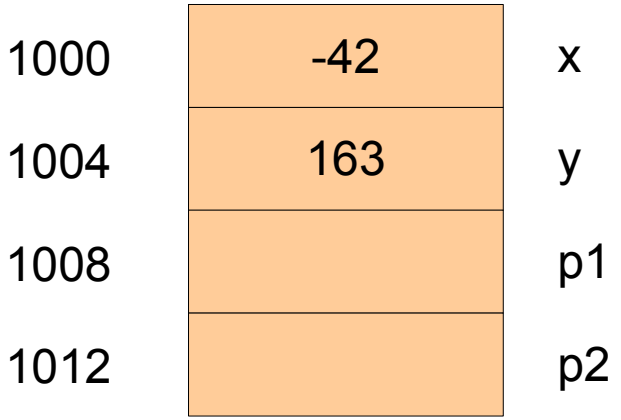
```
int x,y;
int *p1, *p2;
```



Ένα ακόμη παράδειγμα

x= - 42;

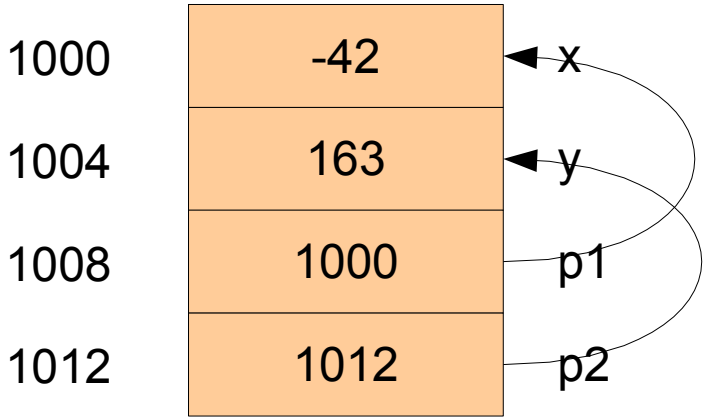
y = 163;



Ένα ακόμη παράδειγμα

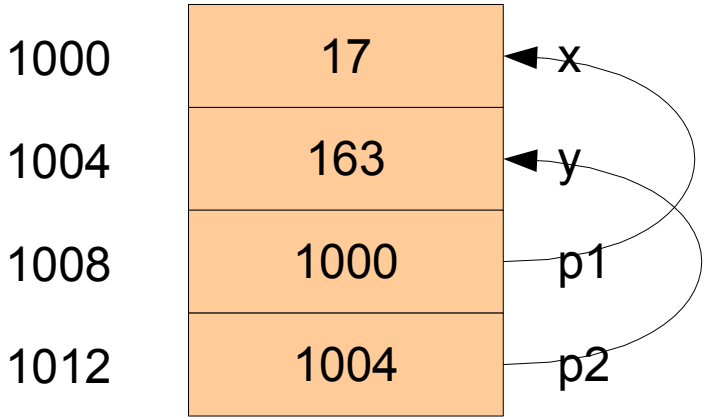
p1 = &x;

p2 = &y;



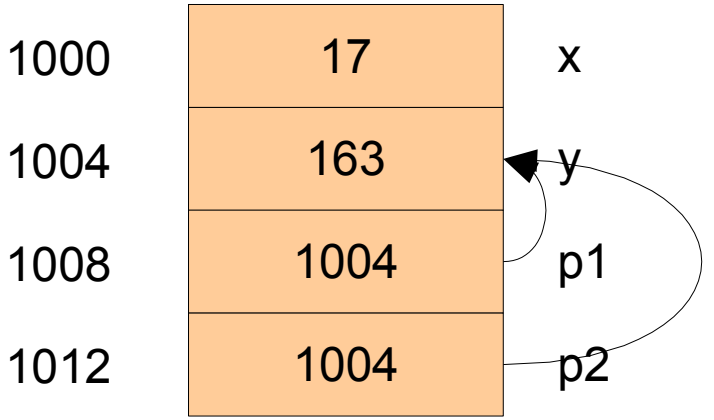
Ένα ακόμη παράδειγμα

*p1 = 17;



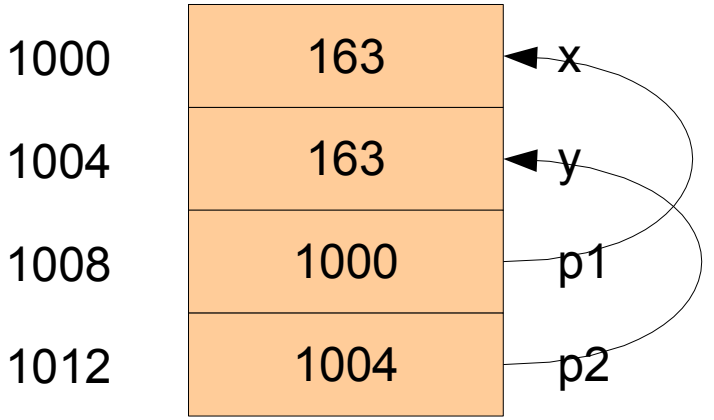
Ένα ακόμη παράδειγμα

p1 = p2;



Εναλλακτικά

```
*p1 = *p2;
```



Ο ειδικός δείκτης NULL

Ειδική τιμή που δηλώνει ότι ένας δείκτης “δείχνει” σε μια μη έγκυρη διεύθυνση.

Δεν επιτρέπεται η πρόσβαση στην τιμή ενός δείκτη που έχει τιμή NULL. Μια τέτοια κλήση έχει ως αποτέλεσμα, συνήθως, την κατάρρευση του προγράμματος.

```
p = NULL;

*p = 1; /* Το πρόγραμμα θα καταρρεύσει */
```

Μεταβίβαση παραμέτρων με αναφορά

- Η κλήση μιας συνάρτησης στη C δεν αλλάζει τις τιμές των παραμέτρων της συνάρτησης (κλήση με τιμή)
- Παράδειγμα

```
void SetToZero(int var)
{
   var = 0;
}
```

- Η κλήση της παραπάνω συνάρτησης δεν έχει κανένα αποτέλεσμα πάνω στο όρισμά της

- Η κλήση

```
int x=10;
SetToZero(x);
/* το x εξακολουθεί να έχει την τιμή
  10... */
```

- δεν έχει κανένα αποτέλεσμα

Κλήση με αναφορά

- Για την επίλυση του παραπάνω προβλήματος περνάμε ως παράμετρο όχι τη μεταβλητή αλλά τη διεύθυνσή της.

```
void SetToZero(int *ip)
{
   *ip=0;
}
```

Στην περίπτωση αυτή η κλήση της συνάρτησης αλλάζει την τιμή της παραμέτρου

- ΠΡΟΣΟΧΗ στον τρόπο περάσματος της διεύθυνσης της μεταβλητής.

Κλήση με αναφορά (2)

```
int x = 10;
SetToZero(&x);
/* Τώρα είναι x = 0 */
```

Νέα συνάρτηση ανταλλαγής ακεραίων

```
void SwapInteger(int *p1, int *p2)
{
   int tmp;
   tmp = *p1;
   *p1 = *p2;
   *p2 = tmp;
}
```

Μια συνάρτηση που επιστρέφει πολλές τιμές

- Είσοδος: Ένα χρονικό διάστημα σε λεπτά της ώρας
- Έξοδος: το χρονικό διάστημα σε ώρες και λεπτά
- Η επικεφαλίδα της αντίστοιχης συνάρτησης:

```
void ConvertTimeToHM(int time, int
  *pHours, int *pMinutes);
```

Παράδειγμα

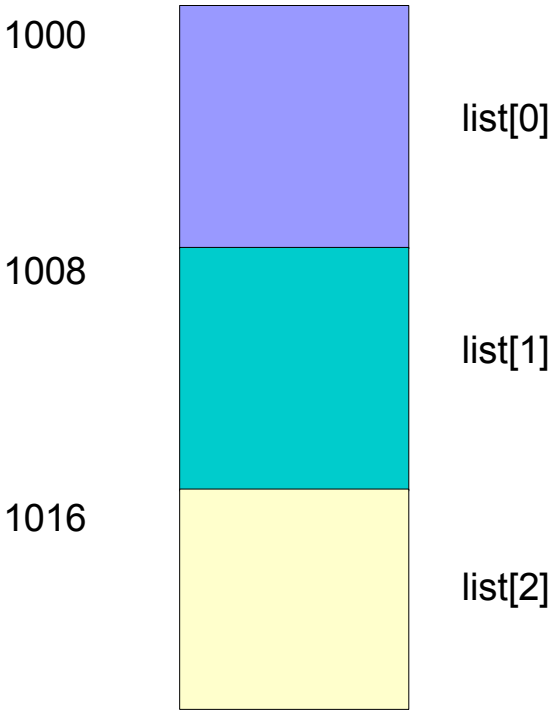
```
#define MinutesPerHour 60

/* Function prototypes */

static void ConvertTimeToHM(int time, int *pHours, int *pMinutes);

main()
{
    int time, hours, minutes;

    printf("Test program to convert time values\n");
    printf("Enter a time duration in minutes: ");
    time = GetInteger();
    ConvertTimeToHM(time, &hours, &minutes);
    printf("HH:MM format: %d:%02d\n", hours, minutes);
}

static void ConvertTimeToHM(int time, int *pHours, int *pMinutes)
{
    *pHours = time / MinutesPerHour;
    *pMinutes = time % MinutesPerHour;
}
```

Δείκτες και πίνακες

- Η παράσταση

double list[3];

ορίζει έναν πίνακα 3 στοιχείων



- Το **`&list[0]`** είναι η διεύθυνση του πρώτου στοιχείου του πίνακα
- Η έκφραση &list[0] είναι ισοδύναμη με την **`list`**.

Πίνακες ως ορίσματα συναρτήσεων

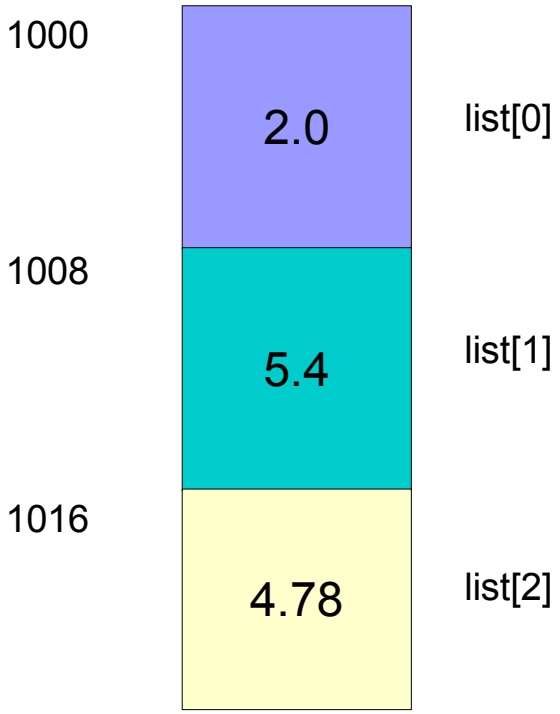
- Η δήλωση

```
void f(int a[]);
```

- είναι ισοδύναμη με την

```
void f(int *a);
```

Πίνακες ως δείκτες



- Η διεύθυνση του στοιχείου list[1] είναι η list + 1 *sizeof (double)
- Η παράσταση list + 1 αντιστοιχεί στην παραπάνω διεύθυνση (ο υπολογισμός γίνεται αυτόματα)

*(list + 1) = 5.4;

Αριθμητική δεικτών

- Η έκφραση

`p + n`

- όπου `p` δείκτης και n ένας ακέραιος
- εννοεί τη διεύθυνση του n-οστού αντικειμένου μετά από αυτό που δείχνει ο p.
- Έχουν νόημα οι εξής παραστάσεις
- **`p - k`**
- **`p1 – p2`**

Οι τελεστές ++ και --

- Μεταθεματική μορφή

  i++

  Αποτιμάται η έκφραση και και μετά αυξάνει κατά 1

  i--

- Προθεματική μορφή

  ++i

  Πρώτα αυξάνει κατά 1 και μετά αποτιμάται η έκφραση

  --i

Παραδείγματα

```
int x,y;
x = 5;
y = ++x;
```

Αλλά

```
int x,y;
x = 5;
y = x++;
```

Αύξηση και μείωση δεικτών

- Η έκφραση
- *p++
- ισοδυναμεί με την
- *(p++)
-
- Η έκφραση
- p++;
- για ένα δείκτη σε ακαίρεο αυξάνει τον δείκτη ώστε να δείχνει στον επόμενο ακέραιο.

Δυναμική κατανομή μνήμης: malloc και free

- Η συνάρτηση malloc

int *p = (int *) malloc(sizeof int);

- Δυναμικοί πίνακες

char * cp = (char *) malloc(10 * sizeof (char));

- Η συνάρτηση
  - free(void * p);
- αποδεσμεύει την περιοχή μνήμης στην οποία δείχνει ο p και η οποία είχε προηγουμένως δεσμευτεί.