Δείκτες

Τι είναι οι δείκτες

Οι ***δείκτες*** είναι μεταβλητές που δεν έχουν δικές τους τιμές, αλλά αναφέρονται (*δείχνουν*) σε άλλες μεταβλητές και χρησιμοποιούν τις τιμές τους.

Η αναφορά σε μία άλλη μεταβλητή γίνεται με βάση την ***διεύθυνση*** που έχει στην μνήμη η μεταβλητή αυτή.

Ένας δείκτης λοιπόν έχει ως περιεχόμενο την διεύθυνση της μεταβλητής, άρα έχει ***έμμεση πρόσβαση*** στην τιμή της μεταβλητής αφού αναφέρεται στην διεύθυνσή της.

Άρα μια *μεταβλητή* αναφέρεται **άμεσα** σε μία τιμή ενώ ένας *δείκτης* αναφέρεται **έμμεσα** σε μια τιμή.

Που τους χρησιμοποιούμε

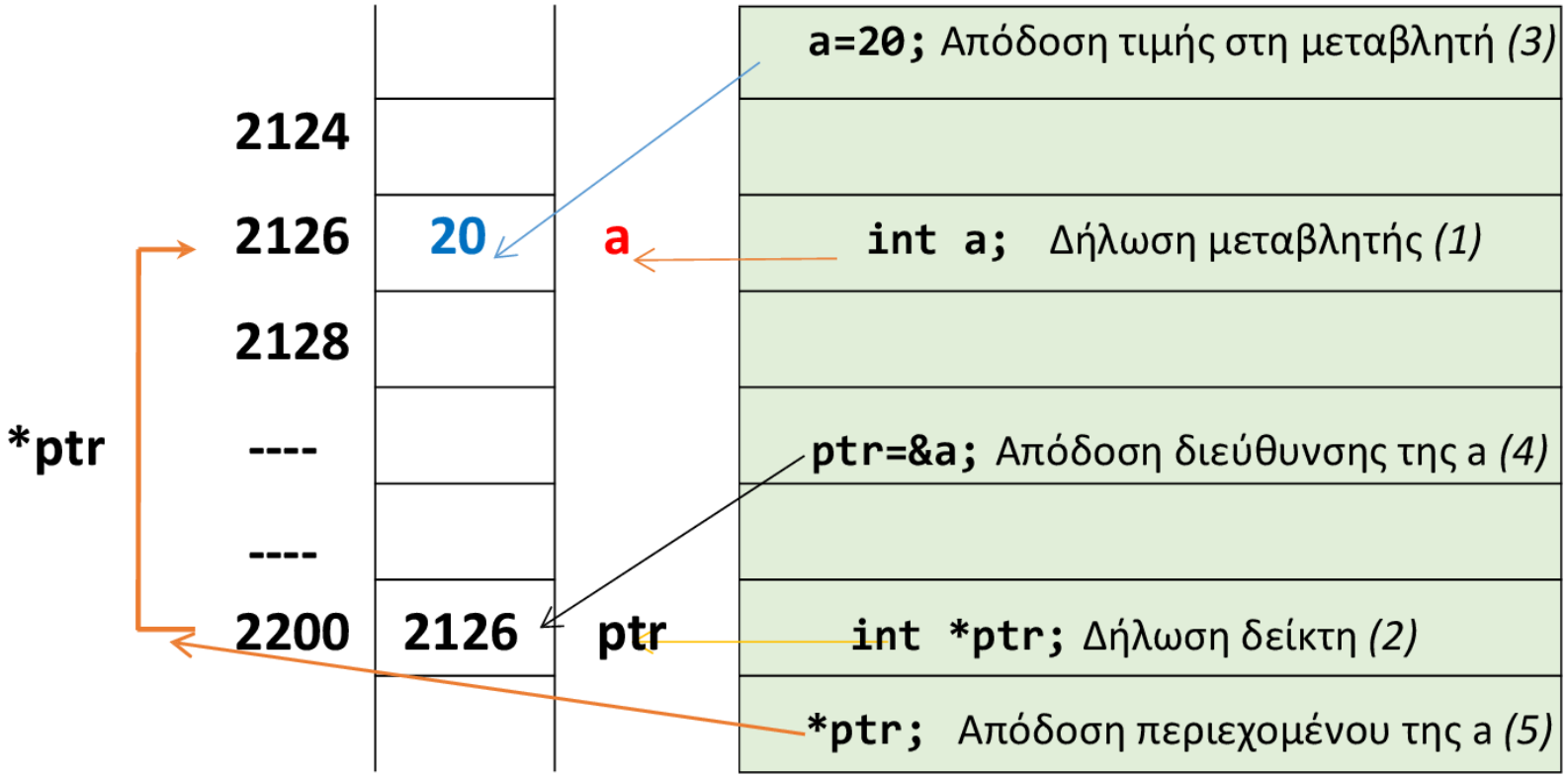
Οι δείκτες είναι πολύ χρήσιμοι για την κατασκευή μιας δομής δεδομένων. Δηλαδή τους χρησιμοποιούμε:

- στις Λίστες
- στις Στοίβες
- στις Ουρές
- στους Γράφους

Σε πολλές γλώσσες προγραμματισμού συναντάμε τους δείκτες, ορισμένες μάλιστα χρησιμοποιούν δείκτες κρυφά από τους χρήστες για να μην τους μπερδεύουν. Οι δείκτες στην ουσία μας επιτρέπουν να κάνουμε καλύτερη διαχείριση μνήμης στα προγράμματά μας.

Δείκτες χρησιμοποιούμε επίσης και σαν **παραμέτρους σε συναρτήσεις**.

Σχηματική παράσταση



Κώδικας του προηγούμενου παραδείγματος

```
include <iostream.h>
void main()
{
    int a; // δήλωση μεταβλητής a
    int *ptr; // δήλωση δείκτη ptr
    a=20; // η τιμή της a (a=20)
    cout << &a; // η διεύθυνση της a (a=0xfff4)
    ptr=&a; // η διεύθυνση του prt (ptr=0xfff4)
    cout << *ptr; // η τιμή του prt (ptr=20)
}
```

Εκφράσεις και Πράξεις με δείκτες

- Οι δείκτες μπορούν να χρησιμοποιηθούν σε αριθμητικές πράξεις μέσα σε εκφράσεις αλλά και σε συγκρίσεις.
- Οι τιμές που χρησιμοποιούν δεν είναι δικές τους αλλά ανήκουν στις μεταβλητές στις οποίες αναφέρονται (που δείχνουν).

Στην επόμενη διαφάνεια φαίνεται ένα παράδειγμα πράξεων μεταβλητών αλλά και δεικτών.

Εκφράσεις και Πράξεις με δείκτες

```
int a, b, c, d *ptr1, *ptr2;
a=5;
b=10;
//έκφραση με πράξεις μεταξύ μεταβλητών
c=(a*b+a)/(b-a); // η c έχει τιμή 11
ptr1=&a; // η διεύθυνση της a πάει στον ptr1
ptr2=&b; // η διεύθυνση της b πάει στον ptr2
//έκφραση με πράξεις μεταξύ δεικτών
d=((*ptr1)*(*ptr2)+(*ptr1))/((*ptr2)-(*ptr1));
// η d έχει τιμή 11
```

Εκφράσεις και Πράξεις με δείκτες

- Μπορούμε να **συγκρίνουμε** δείκτες με τελεστές ισότητας ή συγκριτικούς τελεστές μόνο εάν οι δείκτες δείχνουν σε μεταβλητές ίδιου τύπου.
- Χρησιμοποιούμε αριθμητικές πράξεις δεικτών που βρίσκονται μέσα σε πίνακες αλλά και σε δείκτες που δείχνουν αντικείμενα.

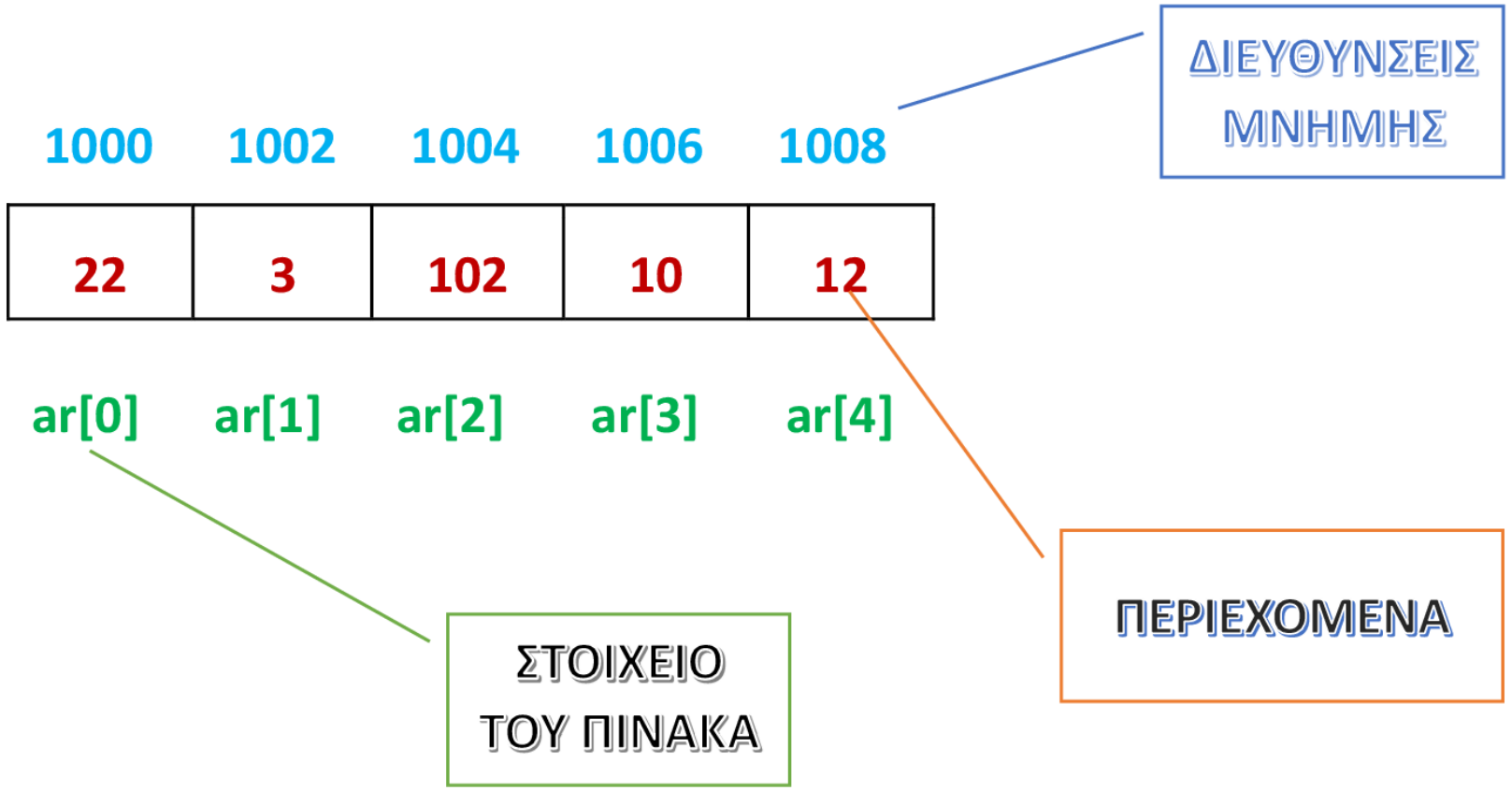
Ο δείκτης αλλάζει την τιμή της μεταβλητής

Ένας δείκτης μπορεί να αλλάξει την τιμή της μεταβλητής στην οποία αναφέρεται αν στο ***prt** δώσουμε νέα τιμή. Δηλ.:

```
int a, *ptr;
a=10;  //η a έχει τιμή 10
ptr=&a;  // αναφορά του ptr στην a
*ptr // το *ptr έχει την τιμή της a.
*ptr=12; /* αλλάζει η τιμή του ptr αλλά και της μεταβλητής a που δείχνει. */
```

Δείκτες και πίνακες

Έστω ο πιο κάτω πίνακας ακεραίων **ar[5]**



Δείκτες και πίνακες

Αν γράψουμε:

```
ptr=&ar[0];
```

ο δείκτης ***prt*** παίρνει την διεύθυνση του πρώτου στοιχείου του πίνακα `ar[].`

και αν γράψουμε:

```
*ptr;
```

Τότε ο δείκτης ***prt*** έχει το περιεχόμενο 22, δηλ. την τιμή του στοιχείου `ar[0].`

Δείκτες και πίνακες

Εάν θέλουμε να δείξουμε στον επόμενο στοιχείο του πίνακα απλά γράφουμε **`ptr++;`**

Εάν θέλουμε να δείξουμε στο τρίτο στοιχείο του πίνακα γράφουμε

`ptr+=2;`

Βλέπουμε ότι ένας δείκτης μπορεί να λειτουργεί και σαν μετρητής.

Για την διεύθυνση του prt γράφουμε:

`ptr=&ar[0]+i` *[όπου i=0,1,2,3,4]*

Ο ***`ptr`*** θα πάρει διαδοχικά τις τιμές: 1000, 1002, 1004, 1006, 1008 που είναι και οι διευθύνσεις των στοιχείων του πίνακα *`ar[ ]`*.

Αυτό θα γίνει με την χρήση ενός βρόχου for.

Δείκτες και συμβολοσειρές

Γνωρίζουμε ότι οι συμβολοσειρές είναι ουσιαστικά μονοδιάστατοι πίνακες τύπου *char*. Η συμβολοσειρά λοιπόν “My Pointer” μπορεί να δηλωθεί ως εξής:

```
char ca[15]=“My Pointer”;
```

Ας ορίσουμε τώρα και έναν δείκτη τύπου *char* δηλ.:

```
char *ptrch;
```

Με την δήλωση: **`ptrch=&ca[0];`** ή **`ptrch=&ca;`** ο δείκτης δείχνει στον πρώτο χαρακτήρα της συμβολοσειράς δηλαδή στο “M”.

Εάν αυξήσουμε κατάλληλα την τιμή του *ptrch* αυτός θα μας δείξει τον επόμενο χαρακτήρα ή οποιονδήποτε χαρακτήρα ή μια ομάδα χαρακτήρων από την συμβολοσειρά.

Δείκτες και συναρτήσεις

Δείκτες χρησιμοποιούμε επίσης και σαν παραμέτρους σε συναρτήσεις.

Το κάνουμε αυτό γιατί αν σε μια συνάρτηση περάσουν οι τιμές των μεταβλητών σαν παράμετροι, η συνάρτηση φτιάχνει αντίγραφα των μεταβλητών και όταν τελειώσει η συνάρτηση τη δουλειά της τα αντίγραφα αυτά σταματούν να υπάρχουν (καταστρέφονται).

Αυτό έχει ως αποτέλεσμα αν υπάρχουν οποιεσδήποτε αλλαγές στις τιμές των παραμέτρων που ενδεχομένως να ήταν επιθυμητές μετά το τέλος της συνάρτησης καθίσταται άκυρες.

Με την χρήση δεικτών πετυχαίνουμε να αλλάζουμε τις τιμές των μεταβλητών στην διεύθυνση της μεταβλητής.

Δείκτες και συναρτήσεις

ΚΛΗΣΗ ΜΕ ΤΙΜΗ

Μάθαμε ότι όταν χρησιμοποιούμε συναρτήσεις για υπολογισμούς, μέσα στις παρενθέσεις της συνάρτησης βάζουμε παραμέτρους ή αλλιώς τοπικές μεταβλητές, που σκοπό έχουν να πάρουν αντίγραφα των τιμών των αρχικών μεταβλητών του προγράμματος και να εκτελέσουν τους υπολογισμούς.

Αυτή η διαδικασία ονομάζεται ***κλήση με τιμή.***

Οι τιμές των αρχικών μεταβλητών δεν επηρεάζονται από την συνάρτηση.

Δείκτες και συναρτήσεις

ΚΛΗΣΗ ΜΕΣΩ ΑΝΑΦΟΡΑΣ ΜΕ ΔΕΙΚΤΗ

Με την χρήση δεικτών στην θέση των παραμέτρων μίας συνάρτησης, μπορούμε να δούμε εκτός από τις τιμές και την διεύθυνση των μεταβλητών, καθώς οι δείκτες δείχνουν απευθείας στο περιεχόμενο της αρχικής μεταβλητής.

Ως ετούτου μπορούμε να αλλάξουμε τις τιμές στις αρχικές μεταβλητές ή να επιστρέφουμε με το *return* παραπάνω από μία τιμές.

Αυτή η διαδικασία ονομάζεται ***κλήση μέσω αναφοράς με δείκτη***.

Διαχείριση μνήμης

Όταν θέλουμε να αποθηκεύσουμε πολλά στοιχεία, χρησιμοποιούμε πίνακες που είμαστε υποχρεωμένοι να γνωρίζουμε το μέγεθός τους (πόση μνήμη θα δεσμεύσουμε) και να το καθορίσουμε πριν από την εκτέλεση του προγράμματος.

Όταν δεν γνωρίζουμε εκ των προτέρων πόση μνήμη θα χρειαστούμε, χρησιμοποιούμε τους τελεστές **new** και **delete** για να δεσμεύσουμε και να αποδεσμεύσουμε τμήματα μνήμης κατά την εκτέλεση του προγράμματος.

Αυτό σημαίνει ότι παρόλο που έχουμε δηλώσει στο πρόγραμμα τους τελεστές αυτούς, η μνήμη θα δεσμευθεί όταν το πρόγραμμα εκτελεστεί και όχι από την αρχή όπως συμβαίνει με τους πίνακες.

Οι τελεστές new-delete

Η λειτουργία των τελεστών new και delete περιγράφεται παρακάτω.

- Ο τελεστής ***new*** επιστρέφει έναν δείκτη για τον κατάλληλο τύπο δεδομένων δεσμεύοντας ταυτόχρονη την μνήμη που θα χρειαστεί.
- Ο τελεστής ***delete*** αποδεσμεύει τον χώρο στην μνήμη που έχει δεσμεύσει ο *new* ‘’επιστρέφοντας’’ την μνήμη στο σύστημα.

Ο τρόπος χρήσης της μνήμης στα πλαίσια του οποίου λειτουργούν αυτοί οι τελεστές λέγεται ***δυναμική ανάθεση μνήμης***.

Παράδειγμα με τελεστές new-delete

Ας υποθέσουμε ότι έχουμε τις πιο κάτω γραμμές κώδικα.

```
int size;
float *array; //dhlwsh deikth
cin>>size; //mege8os pinaka
//o deikths deixnei ston pinaka
array=new float[size];
```

Σε αυτό το παράδειγμα δηλώνουμε έναν δείκτη ***array*** ως τύπο float και μετά βάζουμε τον δείκτη να δείχνει στον πίνακα τύπου float με πλήθος στοιχείων ***size***.

Παράδειγμα με τελεστές new-delete

Εάν οι εντολές αυτές δεν εκτελούνται εξ αρχής διότι μπορεί να βρίσκονται μέσα σε μία *if* που να τους επιτρέπει να εκτελεστούν με την επαλήθευση κάποιας συνθήκης, τότε δεν δεσμεύεται μνήμη για τον πίνακα.

Ο τελεστής *delete* για την αποδέσμευση της μνήμης μπαίνει στο τέλος της χρήσης του πίνακα, με τις γωνιακές αγκύλες να τοποθετούνται μετά από το *delete* (η αλλιώς πριν τον πίνακα).

```
Delete [] array;
```

Δείκτες και αντικείμενα

Οι δείκτες μπορούν να δείχνουν και σε αντικείμενα.

Αν δεν γνωρίζουμε εξ αρχής πόσα αντικείμενα θα δημιουργήσουμε, με την χρήση του τελεστή new επιστρέφουμε έναν δείκτη σε ένα ανώνυμο αντικείμενο.

Π.χ.

```
cat *c; //dhlwsh deikth thw klashs cat

// o deikths deixnei se ena anwnymo
// antikeimeno ths klashs cat
c=new cat;
```

Δείκτες και αντικείμενα

Κατά την εκτέλεση του προγράμματος μπορούμε να δημιουργήσουμε όσα αντικείμενα θέλουμε.

Για να καλέσει το αντικείμενο μέσω του δείκτη τις συναρτήσεις μέλη της κλάσης, χρησιμοποιούμε τον τελεστή προσπέλασης μέλους **(->)**

Δηλ.

```
c->readData();
```

Πίνακας δεικτών

Μπορούμε όταν έχουμε να δημιουργήσουμε πολλά αντικείμενα σε ένα πρόγραμμα, αντί να βάλουμε τα ίδια τα αντικείμενα σε έναν πίνακα (πίνακας αντικειμένων), να φτιάξουμε έναν πίνακα δεικτών που να δείχνουν τα αντικείμενα.

Π.χ.

```
cat *c_arr[10]; //pinakas deiktwn
//o pinakas deixnei sthn klash cat
c_arr[i]= new cat;
//klhsh synarthshs apo ta antikeimena ths klashs
c_arr[i]->printData();
```