Ορισμός μεταβλητών δεικτών και αρχικοποίηση

Η έννοια του δείκτη
Κάθε μεταβλητή σχετίζεται με μια θέση στην κύρια μνήμη του Η/Υ η οποία έχει τη δική της ξεχωριστή διεύθυνση
Ο δείκτης είναι μια μεταβλητή η οποία χρησιμοποιείται για την αποθήκευση μιας διεύθυνσης της κύριας μνήμης του Η/Υ

Μεταβλητές δεικτών
Οι συνηθισμένες μεταβλητές περιέχουν μια ορισμένη τιμή
Οι δείκτες περιέχουν τη διεύθυνση μιας μεταβλητής που έχει μια συγκεκριμένη τιμή

Ορισμός δείκτη
<όνομα τύπου> * <όνομα δείκτη>

```
int  *p;
```

Ορίζει ένα δείκτη σε έναν ακέραιο (δείκτης τύπου int *)

Ορισμός πολλαπλών δεικτών απαιτεί * πριν από κάθε μεταβλητή

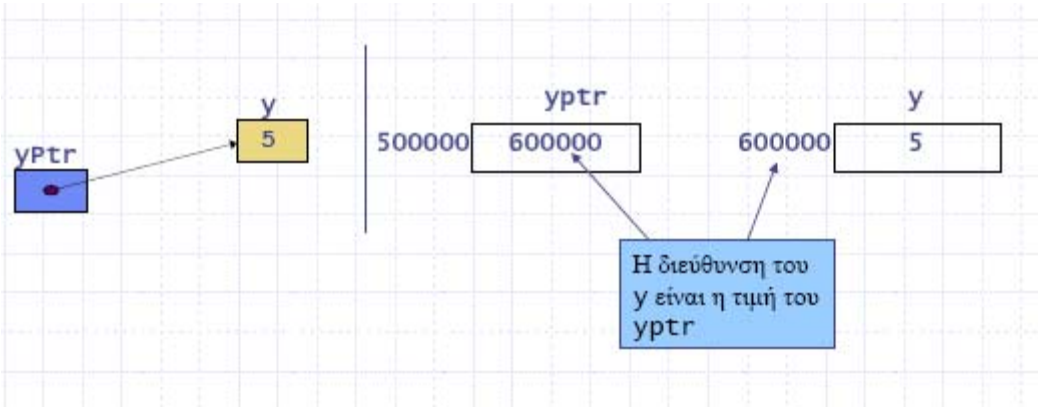
```
int *Ptr1, *Ptr2;
```

Οι δείκτες μπορούν να είναι οποιουδήποτε τύπου και η αρχικοποίηση των δεικτών γίνεται με NULL(NULL – "δείχνει" στο κενό)

Τελεστές δεικτών

Επιστρέφει τη διεύθυνση του τελεστέου πχ

```
int y = 5;
int *yptr;
yptr = &y; /*yptr παίρνει τη διεύθυνση του y */
```



Η χρήση δεικτών **<u>επιβάλλεται</u>** στις ακόλουθες περιπτώσεις:

Α περίπτωση
Όταν θέλουμε να επιστρέφονται περισσότερα από ένα αποτελέσματα από μια συνάρτηση.

Για παράδειγμα έστω ότι μας ζητείται να διαβάσουμε 2 τιμές στο main, να τις μεταβιβάσουμε σε μια συνάρτηση, αυτή να τις εναλλάξει και να τις επιστρέψει εναλλαγμένες στο main όπου και θα εκτυπωθούν.

Αν το πρόγραμμα γραφτεί χωρίς δείκτες θα έχει τη μορφή:

```
#include <stdio.h>

void swap(int x, int y)
{
int temp;

    temp= x;
    x= y;
    y= temp;
}

void main()
{
    int a=1,b=2;

    printf("%d %d \n", a, b);
    swap(&a, &b);
    printf("%d %d \n", a, b);
}
```

Στο κύριο πρόγραμμα δίνονται στο a η τιμή 1 και στο b η τιμή 2 και τυπώνονται αρχικά οι τιμές a=1 και b=2. Όταν καλείται η συνάρτηση swap, η τιμή της a δίνεται στη x και η τιμή της b δίνεται στο y. Όταν τερματιστεί η συνάρτηση swap οι τιμές των x και y έχουν μεν εναλλαχθεί αλλά δεν επιστρέφονται πίσω στο main (όπως θα θέλαμε), οπότε στην εντολή *printf*("%d %d \n", a, b); τυπώνονται πάλι η τιμή 1 για το a και η τιμή 2 για το b (η επιθυμητή εναλλαγή δεν έγινε!)

Σημείωση: Μια συνάρτηση στη C μπορεί να επιστρέψει ΜΟΝΟ ένα ΑΠΟΤΕΛΕΣΜΑ με την εντολή *return*.

Αν το πρόγραμμα γραφτεί με δείκτες θα έχει τη μορφή:

```
#include <stdio.h>

void swap(int *x, int *y)
{
    int temp;

    temp= *x;
    *x= *y;
    *y= temp;
}

void main()
```

```
{
    int a=1,b=2;

    printf("%d %d \n", a, b);
    swap(&a, &b);
    printf("%d %d \n", a, b);
}
```

Στο κύριο πρόγραμμα δίνονται πάλι η τιμή 1 στο a και η τιμή 2στο b και τυπώνονται αρχικά οι τιμές a=1 και b=2. Όταν καλείται η συνάρτηση μεταβιβάζεται η διεύθυνση της a στο δείκτη x και η διεύθυνση της b στο δείκτη y δηλαδή σχηματικά γίνεται το εξής:

x→a (ο x δείχνει στην a)
και
y→b (ο y δείχνει στην b)

Με την εντολή:

- temp= *x;

στο temp καταχωρείται το περιεχόμενο της μεταβλητής στην οποία δείχνει ο δείκτης x δηλαδή στο temp καταχωρείται το περιεχόμενο της a που είναι 1.

Με την εντολή:

- *x=*y;

στο περιεχόμενο της μεταβλητής στην οποία δείχνει ο δείκτης x δηλαδή στην a καταχωρείται το περιεχόμενο της μεταβλητής που δείχνει ο δείκτης y δηλαδή το περιεχόμενο της b, άρα η μεταβλητή a του main παίρνει την τιμή 2.

Με την εντολή:

- *y=temp;

στο περιεχόμενο της μεταβλητής στην οποία δείχνει ο δείκτης y δηλαδή στην b καταχωρείται το περιεχόμενο της temp, άρα η μεταβλητή b του main()παίρνει την τιμή 1.

Συνεπώς όταν η συνάρτηση swap τερματιστεί τυπώνονται στο main οι τιμές a=2 και b=1 (άρα η επιθυμητή εναλλαγή έγινε!)

Το συμπέρασμα είναι ότι χρησιμοποιώντας δείκτες στα ορίσματα μιας συνάρτησης υπάρχει η δυνατότητα να επεμβαίνουμε έμμεσα στο περιεχόμενο των μεταβλητών του main ή γενικά της συνάρτησης που καλεί αυτή με τους δείκτες και να το τροποποιούμε. Με τους δείκτες λέμε δηλαδή ότι υπάρχει η δυνατότητα επιστροφής περισσότερων από ένα αποτελεσμάτων από μια συνάρτηση.

Β περίπτωση

Όταν θέλουμε να κάνουμε δυναμική δέσμευση μνήμης για ένα πίνακα. Αυτό γίνεται με την εντολή *malloc* η οποία βρίσκεται στο αρχείο επικεφαλίδας (header file) stdlib.h

Παράδειγμα

```
int *x;
```

printf("Dose megethos pinaka \n");
scanf("%d", &n);

x=(*int* *)*malloc*(n * *sizeof*(*int*));

Η malloc δεσμεύει δυναμικά μνήμη όταν το πρόγραμμα εκτελείται. Συγκεκριμένα δεσμεύονται n θέσεις μνήμης και η κάθε θέση μνήμης έχει χώρο για την αποθήκευση ενός ακεραίου. Η συνάρτηση *sizeof*(*int*) υπολογίζει το μέγεθος ενός ακεραίου σε bytes (ανάλογα με το σύστημα δεσμεύει 2 ή 4 bytes) οπότε η συνολική μνήμη που δεσμεύεται είναι n*2 ή n*4 bytes. Η διεύθυνση της πρώτης θέσης από τη μνήμη που δεσμεύτηκε επιστρέφεται και καταχωρείται στο δείκτη x (αφού μετατραπεί κατάλληλα ώστε να δείχνει σε ακέραιο. Αυτό γίνεται με το (*int* *)).

Γ περίπτωση

Όταν θέλουμε να αναπαραστήσουμε τους πίνακες με τη μορφή δείκτη (ανεξάρτητα από τον τρόπο που δεσμεύσαμε μνήμη γιαυτούς δηλαδή είτε στατικά είτε δυναμικά). Αυτό γίνεται ως εξής:

Έστω ότι έχουμε δηλώσει τον πίνακα:
int x[5]; //στατική δήλωση πίνακα

Ισχύει εξορισμού σε όλους του πίνακες ότι το όνομα τους ΤΑΥΤΙΖΕΤΑΙ με τη διεύθυνση του αρχικού τους στοιχείου, δηλαδή:

Ο x ταυτίζεται με την &x[0]

Άρα

x+i ταυτίζεται με την &x[i]

και συνεπώς

*(x+i) ταυτίζεται με το x[i]

Άρα μπορούμε να συμβολίσουμε τα στοιχεία των πινάκων είτε ως x[i] είτε ως *(x+i).

Σημείωση: Για τους δισδιάστατους πίνακες ισχύει ισοδύναμα:

((x+i)+j) ταυτίζεται με το x[i][j]

Στην ερώτηση τι πλεονεκτήματα και τι μειονεκτήματα υπάρχουν από τη χρήση δεικτών και απλών ονομάτων μεταβλητών η απάντηση είναι ότι οι δείκτες χρησιμοποιούνται ΜΟΝΟ στις 3 περιπτώσεις που προαναφέραμε και σε καμία άλλη περίπτωση. Άρα η χρήση τους επιβάλλεται από την ίδια τη γλώσσα σε συγκεκριμένες περιπτώσεις και δεν μπορούμε να πούμε ότι υπάρχουν πλεονεκτήματα και μειονεκτήματα από την χρήση τους.

ΕΝΤΟΛΗ BREAK

Η εντολή *break* είναι μια εντολή που χρησιμοποιείται είτε όταν θέλουμε να κάνουμε τερματισμό μιας επανάληψης συνήθως προτού αυτή ολοκληρωθεί κανονικά είτε όταν θέλουμε να κάνουμε έξοδο από μια εντολή *switch*.

Παράδειγμα χρήσης του break σε switch

```
switch(x)
{
```

```
        case 1:printf("Monday\n");
                break;

        case 2:printf("Tuesday\n");
                break;

        case 3:printf("Wednesday\n");
                break;

        case 4:printf("Thursday\n");
                break;

        case 5:printf("Friday\n");
                break;

        case 6:printf("Saturday\n");
                break;

        case 7:printf("Sunday\n");
                break;

        default: printf("Wrong Number\n");
}
```

Παράδειγμα χρήσης του break σε εντολή επανάληψης

```
for (i=1; i<=10;i++)
{
        printf("Dose %d arithmo \n",i);
        scanf("%d", &x);

        if (x%2==0)
              break;
        else
              printf("%d \n", x);
}
```

Η επανάληψη αυτή λειτουργεί κανονικά για 10 ακεραίους και τους εμφανίζει, αλλά όταν δοθεί άρτιος τότε σταματά προτού ολοκληρωθεί.

ΕΝΤΟΛΗ CONTINUE

Η εντολή *continue* είναι μια εντολή που χρησιμοποιείται είτε όταν θέλουμε να αγνοήσουμε τις εντολές στο σώμα μιας επανάληψης και να μεταβούμε κατευθείαν στο επόμενο βήμα της επανάληψης.

Παράδειγμα χρήσης του continue σε εντολή επανάληψης

```
for (i=1; i<=10;i++)
{
        printf("Dose %d arithmo \n",i);
        scanf("%d", &x);
```

```
        if (x%2==0)
            continue;
        else
            printf("%d \n", x);
}
```

Η επανάληψη αυτή λειτουργεί κανονικά για 10 ακεραίους και τους εμφανίζει, αλλά όταν δοθεί άρτιος τότε ο αριθμός αυτός δεν εκτυπώνεται, το πρόγραμμα αγνοεί τις εντολές από το continue και μετά και μεταβαίνει κατευθείαν στο επόμενο βήμα της επανάληψης (δηλαδή αυξάνει την τιμή του i)

```
//ΕΝΑΛΛΑΓΗ 2 ΤΙΜΩΝ ΜΕ ΧΡΗΣΗ ΣΥΝΑΡΤΗΣΗΣ. ΛΑΘΟΣ ΤΡΟΠΟΣ
#include <stdio.h>

void swap(int a,int b)
{
        int temp;

        temp=a;
        a=b;
        b=temp;
}

void main()
{
        int x,y;

        printf("Dose 2 times:\n");
        scanf("%d%d",&x,&y);
        printf("Before swap a=%d b=%d\n",x,y);
        swap(x,y);
        printf("After swap a=%d b=%d\n",x,y);
}
/ΕΝΑΛΛΑΓΗ 2 ΤΙΜΩΝ ΜΕ ΧΡΗΣΗ ΣΥΝΑΡΤΗΣΗΣ. ΣΩΣΤΟΣ ΤΡΟΠΟΣ
#include <stdio.h>

void swap(int *a,int *b)//Pass by reference. Με *a δηλώνουμε ότι ο a είναι δείκτης σε μεταβλητή τύπου int
{
        int temp;

        temp=*a;
        *a=*b;
        *b=temp;
}
```

```
void main()
{
        int x,y;

        printf("Dose 2 times:\n");
        scanf("%d%d",&x,&y);
        printf("Before swap a=%d b=%d\n",x,y);
        swap(&x,&y);
        printf("After swap a=%d b=%d\n",x,y);
}
```