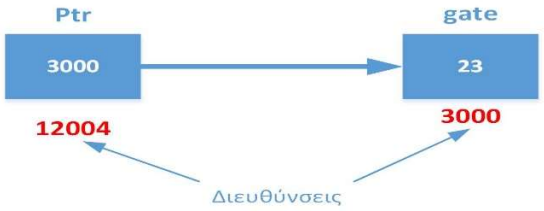
ΔΕΙΚΤΕΣ

1. Εισαγωγικά στοιχεία για τους Δείκτες

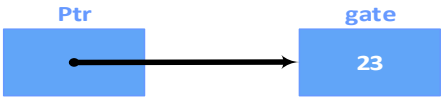
Η RAM ενός υπολογιστή αποτελείται από πολλές χιλιάδες διαδοχικές θέσεις αποθήκευσης και κάθε θέση προσδιορίζεται από μία μοναδική διεύθυνση. Όταν χρησιμοποιείτε τον υπολογιστή σας, το λειτουργικό σύστημα χρησιμοποιεί κάποιο κομμάτι της μνήμης του συστήματος για τις εκάστοτε λειτουργίες που εκτελούνται. Όταν δηλώνετε μία μεταβλητή σε ένα πρόγραμμα στη C, ο μεταγλωττιστής δεσμεύει μια θέση μνήμης με μία μοναδική διεύθυνση για να αποθηκεύσει αυτή τη μεταβλητή. Όταν το πρόγραμμα χρησιμοποιεί το όνομα της μεταβλητής προσπελαύνει αυτόματα την κατάλληλη θέση μνήμης. Ένα στιγμιότυπο αυτής της διαδικασίας παρουσιάζεται στο **Σχήμα 1**. Μια μεταβλητή με το όνομα *test* έχει δηλωθεί, στην οποία έχει αποθηκεύει η τιμή 100. Ο μεταγλωττιστής έχει δεσμεύσει χώρο στην διεύθυνση 1004 για την μεταβλητή και έχει συνδέσει το όνομα *test* με αυτήν.

Σχήμα 1: Μεταβλητή Προγράμματος σε μία συγκεκριμένη θέση μνήμης.

Θα πρέπει να σημειώσετε ότι η διεύθυνση της μεταβλητής test στη '**C**' είναι ένας ακέραιος και μπορεί να αντιμετωπιστεί όπως ένας οποιοσδήποτε άλλος ακέραιος στην '**C**'. Εάν γνωρίζετε τη διεύθυνση μιας μεταβλητής, μπορείτε να δημιουργήσετε μία δεύτερη μεταβλητή στην οποία να αποθηκεύσετε την διεύθυνση της πρώτης. Αυτό φαίνεται και στην γραφική αναπαράσταση του **Σχήματος 2 (και πιο απλοϊκά στο Σχήμα 3)**. Η μεταβλητή *Ptr* αντιστοιχεί στην θέση μνήμης 12004 και η τιμή που περιέχεται σε αυτή τη θέση μνήμης (3000) ισούται με τη διεύθυνση μνήμης που αντιστοιχεί η μεταβλητής *gate* η οποία περιέχει την τιμή 23. Σε ποια διεύθυνση δείχνει η Ptr δεν μας αφορά άμεσα και δεν αξιοποιείται σε αυτή τη λειτουργία.



Σχήμα 2: Αναπαράσταση ενός δείκτη που δείχνει σε μια ακέραια μεταβλητή στην μνήμη.



Σχήμα 3: Πιο απλοϊκή αναπαράσταση.

1.1 Δήλωση Δεικτών

Ένας δείκτης είναι μία ακέραια μεταβλητή και όπως όλες οι μεταβλητές πρέπει να δηλωθούν στην αρχή του προγράμματος πριν χρησιμοποιηθεί στην συνέχεια. Τα ονόματα των μεταβλητών-δεικτών ακολουθούν τους ίδιους κανόνες με τις άλλες μεταβλητές και πρέπει να είναι μοναδικά. Μία δήλωση δείκτη παίρνει την ακόλουθη μορφή:

Typos_Metablitis *onoma_deikti;

Ο *Typos_Metablitis* είναι οποιοσδήποτε από τους τύπους μεταβλητών της '**C**' (π.χ. int, char, double κτλ) και δηλώνει τον τύπο της μεταβλητής στην οποία δείχνει ο δείκτης. Ο αστερίσκος (*) είναι ο τελεστής έμμεσης διευθυνσιοδότησης και δηλώνει ότι το *onoma_deikti* είναι ένας δείκτης στον τύπο *Typos_Metablitis*. Οι δείκτες μπορούν να δηλώνονται μαζί με άλλες μεταβλητές.

Προσοχή στο γεγονός πως το μέγεθος που καταλαμβάνεται από μία μεταβλητή δείκτη είναι πάντα το ίδιο ανεξάρτητα από τον τύπο της μεταβλητής στον οποίο αυτή δείχνει. Υπάρχει συγκεκριμένο μέγεθος μνήμης που απαιτείται για όλες τις μεταβλητές τύπου δείκτη, και αυτό διαφέρει ανάλογα με την έκδοση της **C**, το λειτουργικό αλλά και την αρχιτεκτονική του υπολογιστή. Με την εντολή **sizeof** μπορείτε να γνωρίζετε πόσο είναι αυτό το μέγεθος σε bytes ανάλογα με τον υπολογιστή εκτέλεσης του προγράμματος σας.

1.2 Απόδοση τιμών σε δείκτες

Όπως και οι συνήθεις μεταβλητές, έτσι και **οι δείκτες χωρίς αρχική τιμή μπορούν να χρησιμοποιηθούν, αλλά τα αποτελέσματα είναι απρόβλεπτα!!** Έως ότου αντιστοιχηθεί σε μια έγκυρη διεύθυνση μεταβλητής ένας δείκτης δεν είναι χρήσιμος. Η διεύθυνση δεν αποθηκεύεται στον δείκτη με «μαγικό τρόπο» και ούτε μπορούμε να το κάνουμε αυθαίρετα. Ο προγραμματιστής θα πρέπει να τοποθετήσει μία κατάλληλη τιμή εκεί, χρησιμοποιώντας τον τελεστή διεύθυνσης (**&**). Όταν τοποθετηθεί πριν το όνομα μιας μεταβλητής ο τελεστής διεύθυνσης, τότε επιστρέφει την διεύθυνση της μεταβλητής. Έτσι μπορεί κάποιος να δώσει αρχική τιμή σε ένα δείκτη με μια πρόταση της παρακάτω μορφής:

Onoma_Deikti = &Onoma_Metablitis;

Ακολουθούν ενδεικτικά παραδείγματα για τη χρήση των τελεστών * και &:

ΠΑΡΑΔΕΙΓΜΑ_1

```
#include <stdio.h>

main()
{
    int a;
    int *a_ptr;
    a = 23;
    a_ptr = &a;

    printf("The address of a is      %p \n" , &a);
    printf("The address of a_ptr is  %p \n" , &a_ptr);
    printf("The  value  of a_ptr is  %p \n" , a_ptr);
    printf("The  value  of a is      %d \n" , a);
    printf("The  value  of *a_ptr is %d \n" , *a_ptr);

    *a_ptr = 66;
    printf("\n");
    printf("After the execution of *a_ptr = 66; : \n");
    printf("The  value  of a is      %d \n" , a);

    return 0;
}
```

Με τα εξής αποτελέσματα:

```
The address of a is      000000000062FE4C
The address of a_ptr is  000000000062FE40
The  value  of a_ptr is  000000000062FE4C
The  value  of a is      23
The  value  of *a_ptr is 23

After the execution of *a_ptr = 66; :
The  value  of a is      66
```

Όταν εκτελέσετε εσείς τον κώδικα δεν θα έχετε ακριβώς τα ίδια αποτελέσματα. Εξηγήστε γιατί συμβαίνει αυτό.

ΠΑΡΑΔΕΙΓΜΑ_2

```
#include <stdio.h>

main()
{
    int a, *b, size_b, size_d;
    float c, *d;
    b = &a;
    d = &c;
    scanf("%d",b);
    printf("value of a is equal to %d while its address is %p \n" , a, b);
    scanf("%d",&a);
    printf("value of a is equal to %d while its address is %p \n" , a, b);
    size_b = sizeof(b);
    size_d = sizeof(d);
    if (size_b == size_d)
       printf("size of pointers for int and float variables is the same and equals with %d bytes
\n" , size_b);
    else
       printf("different size of pointers for int and float variables \n");
    return 0;
}
```

Με τα εξής αποτελέσματα:

```
6
value of a is equal to 6 while its address is 000000000062FE34
17
value of a is equal to 17 while its address is 000000000062FE34
size of pointers for int and float variables is the same and equals with 8 bytes
```

Όταν εκτελέσετε εσείς τον κώδικα δεν θα έχετε ακριβώς τα ίδια αποτελέσματα. Εξηγήστε γιατί συμβαίνει αυτό.

1.3 Δείκτες και πίνακες

Εφόσον οι πίνακες αποτελούνται από συνεχόμενες θέσης μνήμης, αρκεί να βρούμε την πρώτη κάθε φορά θέση και να πλοηγηθούμε αντιστοίχως στις θέσεις των άλλων στοιχείων. Η σύνδεση των δεικτών με τους πίνακες γίνεται ως εξής: «Ένα όνομα πίνακα είναι ένας δείκτης στο πρώτο στοιχείο του πίνακα». Έτσι, εάν έχετε ένα πίνακα *data[]*, η δήλωση σε οποιαδήποτε επόμενη γραμμή του *data* αντιπροσωπεύει τη διεύθυνση του πρώτου στοιχείου της διάταξης (ισοδύναμο με το *&data[0]*). Επομένως, μπορείτε να δηλώσετε ένα μεταβλητό δείκτη και να του ορίσετε για αρχική τιμή να δείχνει τον πίνακα.

Όπως θα θυμάστε, τα στοιχεία ενός πίνακα αποθηκεύονται σε διαδοχικές θέσεις μνήμης, με το πρώτο στοιχείο στη χαμηλότερη διεύθυνση. Τα διαδοχικά στοιχεία διάταξης (εκείνα με δείκτη μεγαλύτερο από 0) αποθηκεύονται σε υψηλότερες διευθύνσεις. Το πόσο υψηλότερα εξαρτάται από τον τύπο δεδομένων της διάταξης (π.χ. char, int, float κ.οκ.).

Ας υποθέσουμε μια διάταξη τύπου **int**. Μιά απλή μεταβλητή τύπου int καταλαμβάνει 4 bytes μνήμης. Κάθε στοιχείο της διάταξης θα βρίσκεται δύο bytes πάνω από το προηγούμενο στοιχείο και η διεύθυνση κάθε στοιχείου της διάταξης θα είναι υψηλότερη κατά δύο από τη διεύθυνση του προηγούμενου στοιχείου. Έχετε τη δυνατότητα να προσπελάσετε τα στοιχεία ενός πίνακα χρησιμοποιώντας ένα πίνακα. 'Ενας δείκτης πρέπει να αυξηθεί κατά 4 για να προσπελάσει επιτυχώς ένα πίνακα τύπου **int** και κατά 8 για να προσπελάσει επιτυχώς ένα πίνακα τύπου **double**.

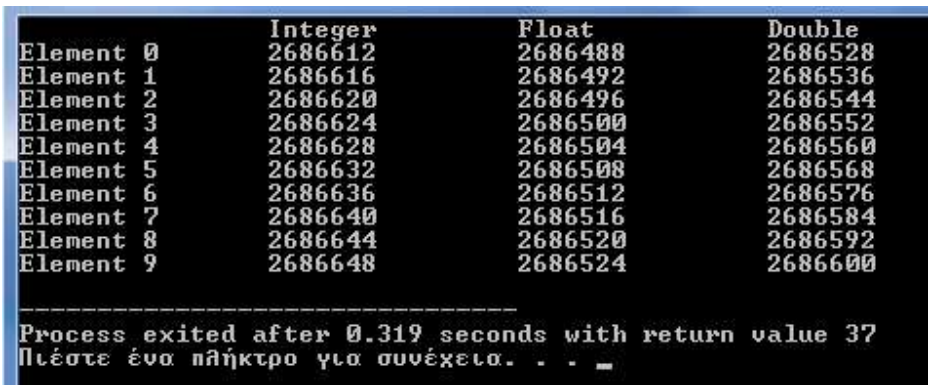
ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

void main ()
{
int      x;
int    a[10];     /* Pinakas typoy int    */
double b[10];     /* Pinakas typoy double*/
float  c[10];     /* Pinakas typoy float */

printf("\t\tInteger \tFloat \t\tDouble\n");
for(x=0;x<10;x++)
    printf("Element %d \t%ld \t%ld \t%ld\n",x,&a[x],&c[x],&b[x]);
}
```

Με τα εξής αποτελέσματα:



Οι ακριβείς διευθύνσεις που προβάλλει το σύστημα σας πιθανότατα θα διαφέρουν από αυτές, αλλά οι αποστάσεις του κάθε στοιχείου από το επόμενο θα είναι οι ίδιες. Στο συγκεκριμένο πρόγραμμα υπάρχουν 4 bytes στα στοιχεία int, 4 bytes ανάμεσα στα στοιχεία float και 8 bytes ανάμεσα στα στοιχεία double.

1.4 Αριθμητική Δεικτών

Όταν αυξάνετε ένα δείκτη αυξάνετε την τιμή του. Για παράδειγμα, όταν αυξάνετε ένα δείκτη κατά 1, η αριθμητική δεικτών αυξάνει αυτόματα την τιμή του δείκτη έτσι ώστε να δείχνει στο επόμενο στοιχείο του πίνακα. Με άλλα λόγια, η '**C**' γνωρίζει τον τύπο δεδομένων που δείχνει ο δείκτης και αυξάνει την διεύθυνση που είναι αποθηκευμένη στον δείκτη κατά το μέγεθος του τύπου δεδομένων.

Οι ίδιες έννοιες ισχύουν και για την μείωση δεικτών. Η μείωση ενός δείκτη είναι στην πραγματικότητα μία ειδική περίπτωση αύξησης με την πρόσθεση ενός αρνητικού αριθμού. Εάν μειώσετε έναν με τους τελεστές – ή -=, η αριθμητική δεικτών προσαρμόζει τον δείκτη για το μέγεθος των στοιχείων του πίνακα.

Προσοχή στο γεγονός πως η προτεραιότητα του τελεστή * είναι μικρότερη από αυτή των ++ και --. Συνεπώς, η εκτέλεση της εντολής *ptr++ διαφέρει από αυτό της (*ptr)++. Πιο συγκεκριμένα, η πρώτη έκφραση μεταφράζεται ως «αύξησε την τιμή της μεταβλητής δείκτη ptr» σύμφωνα πάντα με την αριθμητική που υπηρετεί ανάλογα με το πρόγραμμα στο οποίο καλείται, ενώ η δεύτερη ως «αύξησε κατά 1 την τιμή της μεταβλητής που δείχνει ο δείκτης ptr».

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

void main ()
{
int i_array[10]={4,8,12,27,89,-67,19,90,35,-65};
float f_array[10]={.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9};
float *f_ptr;
int *i_ptr,count;

f_ptr=f_array;
i_ptr=i_array;
printf("count\t i_array\t f_array\n");

for (count=0; count<10; count++, i_ptr++, f_ptr++)
  printf("%d\t %d\t\t %.2f\n", count, *i_ptr, *f_ptr);
}
```

Με τα εξής αποτελέσματα:

```
count    i_array         f_array
0        4               0.00
1        8               0.10
2        12              0.20
3        27              0.30
4        89              0.40
5        -67             0.50
6        19              0.60
7        90              0.70
8        35              0.80
9        -65             0.90

--------------------------------
Process exited after 0.3455 seconds with return value 14
Πιέστε ένα πλήκτρο για συνέχεια. . .
```