Δομημένος προγραμματισμός

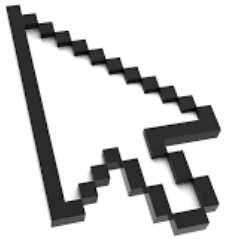
https://publicdomainvectors.org

Οι δείκτες pointers στη C

Περίγραμμα της ύλης που θα καλυφθεί

- Η έννοια του δείκτη στη C
- Μεταβλητές τύπου δείκτη
- Αριθμητική των δεικτών
- Οι πίνακες ως δείκτες
- Επεξεργασία με δείκτες
- Παραδείγματα

https://www.pexels.com



https://www.flickr.com

Η έννοια του δείκτη στη C

- Την έννοια του δείκτη τη συναντήσαμε σε προηγούμενα μαθήματα:
  - *είδαμε ότι συνάρτηση* scanf(), *καταχωρίζει τιμές σε μεταβλητές χρησιμοποιώντας τον τελεστή διεύθυνσης* &.
- Όταν δηλώνεται μια μεταβλητή, τότε δεσμεύεται μια περιοχή μνήμης η οποία ξεκινά από κάποια διεύθυνση.
- Αυτή η διεύθυνση μπορεί να είναι περιεχόμενο μιας άλλης μεταβλητής. Τότε η μεταβλητή αυτή ονομάζεται *δείκτης*.
- Ο τελεστής & μπροστά από το όνομα μιας μεταβλητής μας δίνει τη διεύθυνση της δεσμευμένης μνήμης όπου αποθηκεύεται η μεταβλητή. Πχ int A = 7;

```
printf("Value of A =  %d \nAddress of A = %u\n", A, &A);
```

- Αν λοιπόν Α είναι το όνομα μιας μεταβλητής, τότε *&A* είναι η διεύθυνση της μεταβλητής Α στη μνήμη και είναι ένας *δείκτης*.

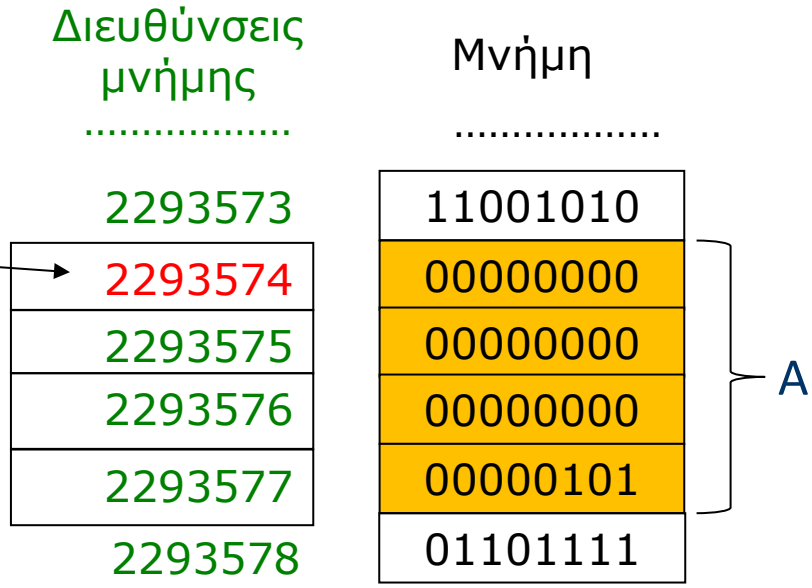
Η έννοια της διεύθυνσης μνήμης

- Η περιοχή που δεσμεύεται στη μνήμη όταν δηλώνεται μια μεταβλητή είναι μια σειρά από διαδοχικά κελιά μνήμης που έχουν χωρητικότητα *1 Byte*.
- Σε κάθε κελί της μνήμης αντιστοιχεί μια διεύθυνση.
- Η διεύθυνση του 1ου κελιού που δεσμεύεται για κάποια μεταβλητή, είναι η διεύθυνση της μεταβλητής.

Δήλωση στη C: *int A = 5 ;*

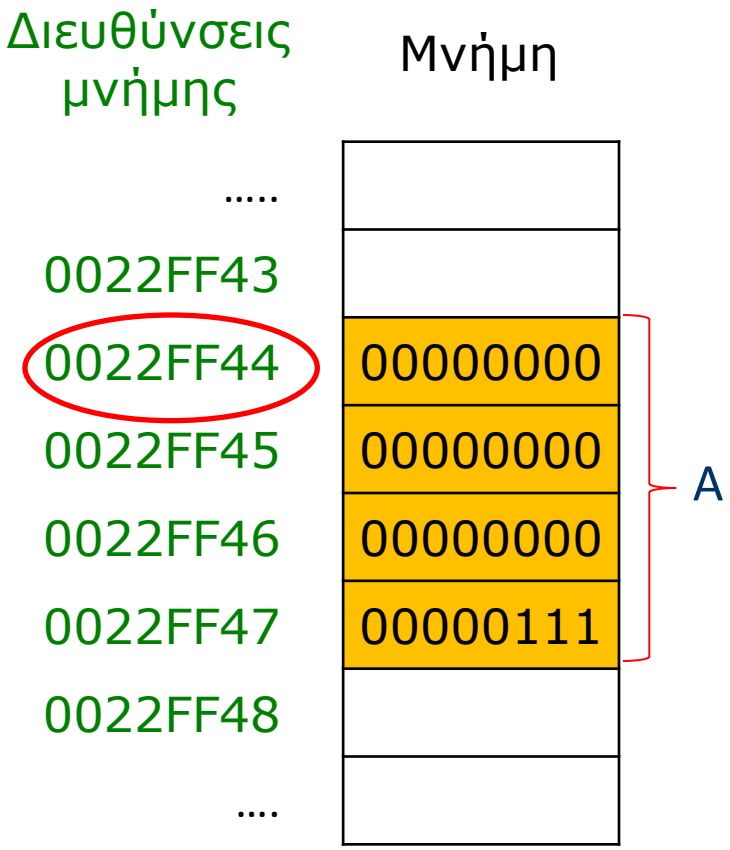
Η διεύθυνση της A (&A) δείχνει στην A (δείκτης της A)

Η A καταλαμβάνει 4 Bytes. Γιατί?



Στην πράξη…

```
#include <stdio.h>
#include <stdlib.h>
main()
{
      int A;
      A = 7;
      printf("Value is: %u\n", A);
      printf("Address is: %p\n", &A);
}
```



Αποτέλεσμα:

```
Value is: 7
Address is: 0022FF44
```

Το μήκος σε bits μιας διεύθυνσης εξαρτάται από τον υπολογιστή

Οι μεταβλητές τύπου δείκτη (δήλωση)

- Στην C υπάρχει η δυνατότητα, οι διευθύνσεις μεταβλητών να μπορούν ν' αποθηκευτούν επίσης σε μεταβλητές.

- Αυτές του είδους οι μεταβλητές που περιέχουν (δείχνουν) διευθύνσεις άλλων μεταβλητών ονομάζονται *δείκτες* (*pointers*).

- Η δήλωση μιας μεταβλητής-δείκτη γίνεται ως εξής:

*<τύπος> *<όνομα_δείκτη>;*

(δημιουργώ ένα δείκτη που μπορεί να δείξει σε διεύθυνση μεταβλητής που έχει δηλωθεί σαν <τύπος>)

- Ο *<τύπος>* μπορεί είναι ένας από τους τύπους δεδομένων της C και δηλώνει τον τύπο της μεταβλητής που δείχνει ο δείκτης.

- Ο αστερίσκος * υποδεικνύει ότι η μεταβλητή που ακολουθεί είναι ένας δείκτης.

- Το *<όνομα_δείκτη>* είναι οποιοδήποτε όνομα επιθυμούμε.

Παραδείγματα δήλωσης δεικτών

- *int *intPtr;*

  Η μεταβλητή intPtr είναι δείκτης σε ακεραία μεταβλητή (int)

  Δηλαδή στο δείκτη intPtr μπορούμε να αποθηκεύσουμε τη διεύθυνση κάποιας ακεραίας μεταβλητής:

  intPtr = &A;

- *char *charPtr;*

  (η μεταβλητή charPtr είναι δείκτης σε μεταβλητή χαρακτήρα)

- *float *floatPtr*;

  (η μεταβλητή floatPtr είναι δείκτης σε μεταβλητή float)

- Επειδή οι δείκτες δείχνουν σε διευθύνσεις μεταβλητών και όχι στο περιεχόμενο τους *το μέγεθος τους είναι σταθερό και ίδιο* για όλους τους τύπους. Εξαρτάται από την αρχιτεκτονική του υπολογιστή, τον compiler ή το λειτουργικό σύστημα. Δεν εξαρτάται από τον τύπο της μεταβλητής.

Ανάθεση τιμών σε δείκτες

- Πριν χρησιμοποιηθεί ένας δείκτης πρέπει να δείχνει σε μια διεύθυνση.

- Ο τελεστής & που δίνει την διεύθυνση μιας μεταβλητής χρησιμοποιείται για να δώσουμε τιμές σε δείκτες:

```
int A, *intPtr;

intPtr = &A;
```

Προσοχή:
Ο τύπος του δείκτη πρέπει να είναι ο ΙΔΙΟΣ με αυτόν της μεταβλητής.

- Η τιμή που δίνουμε σε έναν δείκτη πρέπει να είναι διεύθυνση κάποιας μεταβλητής και ΟΧΙ μια αριθμητική τιμή.

~~`intPtr = 2000;`~~

- Στην εντολή *printf()* για να δούμε το περιεχόμενο ενός δείκτη χρησιμοποιούμε το προσδιοριστή *%p* που εμφανίζει τη τιμή του δείκτη σε *16-δική* μορφή. Μπορούμε επίσης να κάνουμε χρήση και των προσδιοριστών *%u* και *%d*.

Η τιμή NULL

- Αν προσπαθήσουμε να χρησιμοποιήσουμε ένα δείκτη πριν αυτός πάρει κάποια τιμή διεύθυνσης, τότε το πρόγραμμα δεν θα εκτελεστεί σωστά.

- Όταν δηλώνεται ένας δείκτης δεν του αποδίδεται από τη C κάποια συγκεκριμένη τιμή.

- Μπορούμε να χρησιμοποιήσουμε τη σταθερά *NULL* για να δώσουμε αρχική τιμή σ' ένα δείκτη:

```
int *intPtr = NULL;
```

- Η σταθερά NULL βρίσκεται στη βιβλιοθήκη *<stdio.h>* και έχει τη τιμή *0.*

- Δεν δείχνει πουθενά και πρέπει να χρησιμοποιείται με προσοχή.

- Έλεγχος της τιμής ενός δείκτη:

```
if (intPrt == NULL)   // ή if (!intPrt)
```

Ο τελεστής *

- Εφαρμόζεται μόνο σε δείκτες. Δεν είναι ο ίδιος με τον τελεστή του πολλαπλασιασμού (ο οποίος χρειάζεται 2 όρους).

- Ο τελεστής * πριν το όνομα του δείκτη μας δίνει το περιεχόμενο (την τιμή) της διεύθυνσης που δείχνει ο δείκτης:

```
int  a =5, *aptr;
aptr=&a;
printf(" Value of a = %d", *aptr);
```

Value of a = 5

- Μπορούμε ν' αλλάξουμε το περιεχόμενο μιας μεταβλητής με τον δείκτη της:

```
*aptr =78;
```

a = 78

```
*aptr = *aptr+1;
```

a = 79

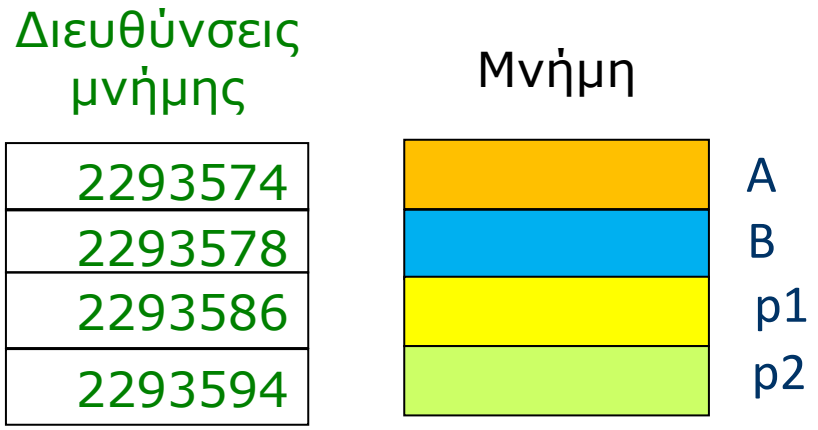
Πρόκειται για μια έμμεση αναφορά στη τιμή μιας μεταβλητής

- Μπορούμε ακόμα να δηλώσουμε και δείκτη σε δείκτη:

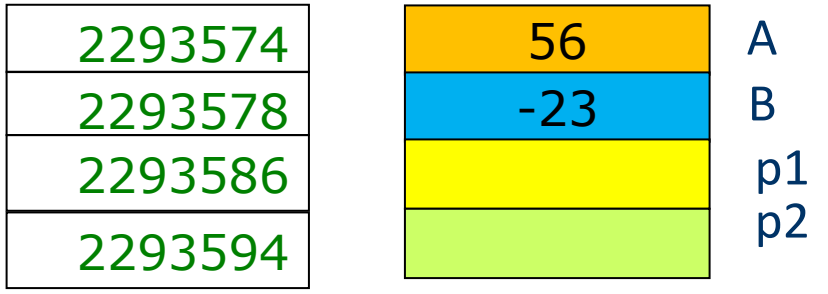
```
int  **aptr; //δείκτης σε δείκτη που δείχνει σε ακέραιο!
```

Αναφορά σε στοιχεία ενός δισδιάστατου πίνακα

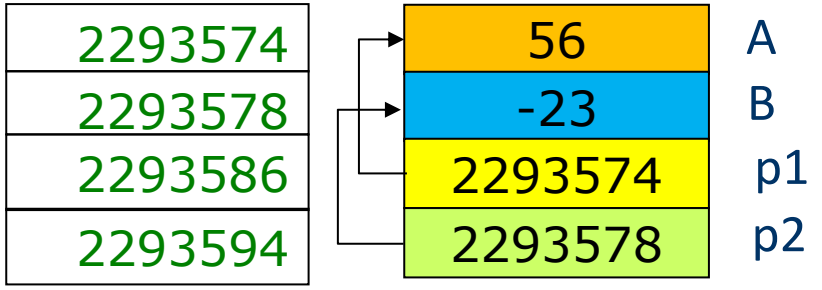
```
int A, B;
int *p1, *p2;
```




```
A = 56;
B = -23;
```



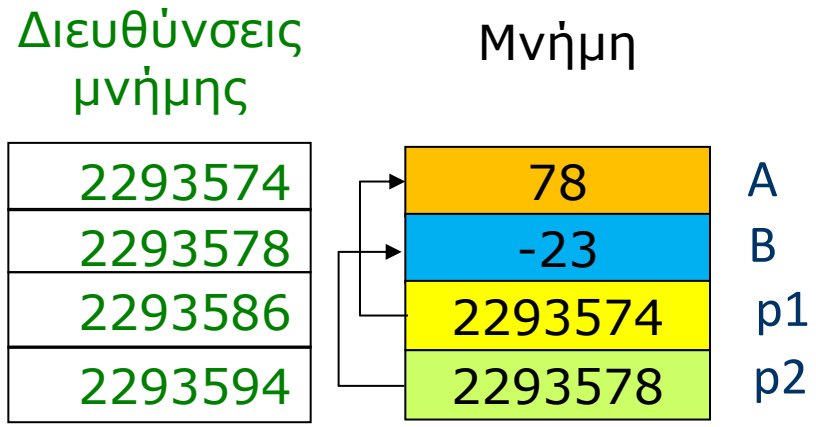

```
p1 = &A;
p2 = &B;
```



Απόδοση τιμών σε δισδιάστατους πίνακες - Ι

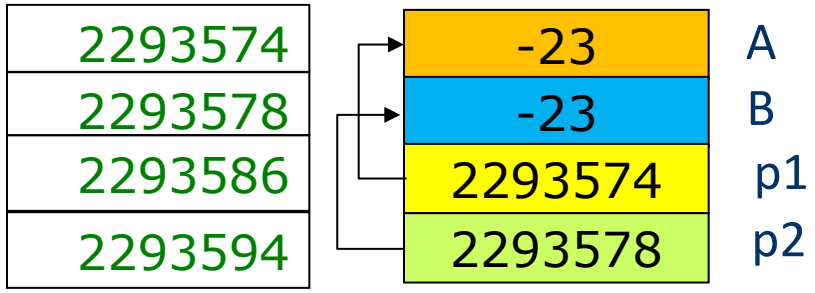
*p1 = 78;

(αλλάζω το περιεχόμενο της A)



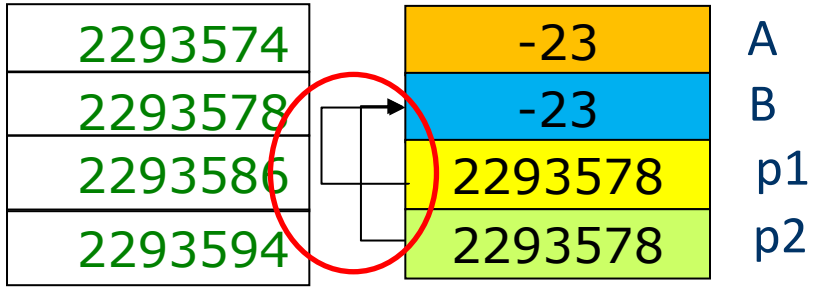
*p1 = *p2;

(αποτέλεσμα ίδιο σαν A=B)



p1 = p2;

(ο p1 δείχνει όπου και ο p2)



Πρόγραμμα με αλλαγή σε τιμή μεταβλητής με δείκτη

```
#include <stdio.h>
main()
{
      int i, *pi;
      i = 155;
      pi = &i;
      printf("%d  - %d \n",i,*pi);
      printf("%p - %p \n",pi,&i);
      printf("%u - %u \n\n",pi,&i);
      *pi=199;
      printf("%d - %d \n",i,*pi);
      printf("%p - %p \n",pi,&i);
      printf("%u - %u \n",pi,&i);
}
```

Αποτέλεσμα:

```
155  - 155
0022FF44 - 0022FF44
2293572 – 2293572

199 - 199
0022FF44 - 0022FF44
2293572 - 2293572
```

Το περιεχόμενο αλλάζει
Η διεύθυνση ΔΕΝ αλλάζει

Πράξεις με χρήση δεικτών - πρόγραμμα

```
#include <stdio.h>
main()
{
   int a, b, c;
   int *pa, *pb, *pc;
   printf("\n a:"); scanf("%d", &a);
   printf("\n b:"); scanf("%d", &b);
   printf("\n c:"); scanf("%d", &c);
   pa = &a;
   pb = &b;
   pc = &c;
   printf("\n a(%d)+ b(%d)+c(%d) = %d\n", a, b, c, *pa+*pb+*pc);
   printf("\n a(%d)* b(%d)*c(%d) = %d\n", a, b, c, *pa**pb**pc);
}
```

Αποτέλεσμα:

```
a:5
b:6
c:7

a(5) + b(6)+c(7) = 18
a(5) * b(6)*c(7) = 210
```

Πόσα Bytes καταλαμβάνει ένας δείκτης;

```
#include <stdio.h>
main()
{
  int a, *pa;
  double b, *pb;
  char c, *pc;
  pa = &a;
  pb = &b;
  pc = &c;
  printf("\n a:"); scanf("%d", pa);
  printf(" b:"); scanf("%lf", pb);
  printf(" c:"); scanf("%s", pc);
  printf("\n a=%d, size of a=%d, size of pa=%d\n", *pa, sizeof(a),sizeof(pa));
  printf(" b=%f, size of b=%d, size of pb=%d\n", *pb, sizeof(b),sizeof(pb));
  printf(" c=%c, size of c=%d, size of pc=%d\n", *pc, sizeof(c),sizeof(pc));
}
```

Αποτέλεσμα:

```
a:3
b:5.2
c:a

a=3, size of a=4, size of pa=8
b=5.200000, size of b=8, size of pb=8
c=a, size of c=1, size of pc=8
```

Όλοι οι δείκτες έχουν το ίδιο μέγεθος

Τα & και * είναι συμπληρωματικά (αντίστροφα)

Αποτέλεσμα:

```
The address of a is 0028FF44
The value of aPtr is 0028FF44
The value of a is 7
 The value of *aPtr is 7
 &*aPtr = 0028FF44
 *&aPtr = 0028FF44
```

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int a;
    int *aPtr;
    a = 7;
    aPtr = &a;    // aPtr δείχνει στη διεύθυνση του a
    printf( "The address of a is %p\n The value of aPtr is %p\n", &a, aPtr );
    printf( "The value of a is %d\n The value of *aPtr is %d", a, *aPtr );
    // Τα & και * είναι συμπληρωματικά - αλληλοαναιρούνται
    printf( "\n &*aPtr = %p\n*&aPtr = %p\n", &*aPtr, *&aPtr );
}
```

Αριθμητική των δεικτών

- Επιτρέπονται μόνο 2 αριθμητικές πράξεις με τους δείκτες:
  - *πρόσθεση και αφαίρεση*.
- Οι τελεστές που χρησιμοποιούνται είναι οι +, -, ++ και -- .
- Όταν προσθέτω το 1 σ' ένα δείκτη τότε:
  - *Αν η μεταβλητή που δείχνει είναι char, ο δείκτης αυξάνεται κατά 1 (μέγεθος του char 1 = Byte).*
  - *Αν η μεταβλητή που δείχνει είναι int, ο δείκτης αυξάνεται κατά 4 (υποθέτουμε ότι το μέγεθος του int 4 = Bytes).*
  - *Αν η μεταβλητή που δείχνει είναι double, ο δείκτης αυξάνεται κατά 8 (μέγεθος του double 8 = Bytes).*
- Κάνουμε ΜΟΝΟ *πρόσθεση ακεραίου* σε δείκτη ή *αφαίρεση ακεραίου* από δείκτη.
- Μπορούμε να κάνουμε ΜΟΝΟ αφαίρεση ή σύγκριση 2 δεικτών, όταν αυτοί δείχνουν στο ίδιο τύπο μεταβλητής.

Πρόγραμμα με αριθμητική δεικτών

```
char *chp, ch;
 int *inp, in;
 float *flp, fl;
 double *dp, d;
 chp=&ch;
 inp=&in;
 flp=&fl;
 dp=&d;
 printf("\nDeiktis char:%u, Deiktis int:%u\n", chp,inp);
 printf("Deiktis float:%u, Deiktis double:%u\n", flp, dp);
 chp++;
 inp++;
 flp++;
 dp++;
 printf("\nDeiktis char:%d, Deiktis int:%d\n", chp,inp);
 printf("Deiktis float:%d, Deiktis double:%d\n", flp, dp);
```

Αποτέλεσμα:

Deiktis char:2293571, Deiktis int:2293560
Deiktis float:2293552, Deiktis double:2293536

Deiktis char:2293572, Deiktis int:2293564
Deiktis float:2293556, Deiktis double:2293544

Οι πίνακες ως δείκτες

- Στη C όταν δηλώνεται ένας πίνακας (πχ int Pin[100];) έχουμε τα εξής:
  - *Η έκφραση Pin[0] είναι το 1ο στοιχείο του πίνακα*
  - *Η έκφραση &Pin[0] είναι η διεύθυνση του 1ου στοιχείου του πίνακα*
  - *Η έκφραση &Pin[0] είναι ισοδύναμη με την έκφραση Pin (το όνομα του πίνακα).*
- Όπως έχουμε δει, ο *Pin* είναι δείκτης στο πρώτο στοιχείο του πίνακα *Pin* (περιέχει την διεύθυνση μνήμης του 1ου στοιχείου).
- Μπορούμε λοιπόν να χρησιμοποιούμε το *Pin* ακριβώς όπως ένα δείκτη.
- Ο τύπος του *Pin* είναι *ΣΤΑΘΕΡΟΣ* δείκτης σε ακέραιο. Σταθερός σημαίνει ότι δεν μπορεί να δείξει κάπου αλλού, όπως μπορούμε να κάνουμε με τους άλλους δείκτες.

Πρόγραμμα επίδειξης χρήσης πίνακα σαν δείκτης

Αποτέλεσμα:

```
Show Pin using indexing
Pin[0]=  3, mem addr: 0022FF30
Pin[1]=  2, mem addr: 0022FF34
Pin[2]=  1, mem addr: 0022FF38
Pin[3]=  9, mem addr: 0022FF3C
Show Pin using pointers
Pin[0]=  3, mem addr: 0022FF30
Pin[1]=  2, mem addr: 0022FF34
Pin[2]=  1, mem addr: 0022FF38
Pin[3]=  9, mem addr: 0022FF3C
```

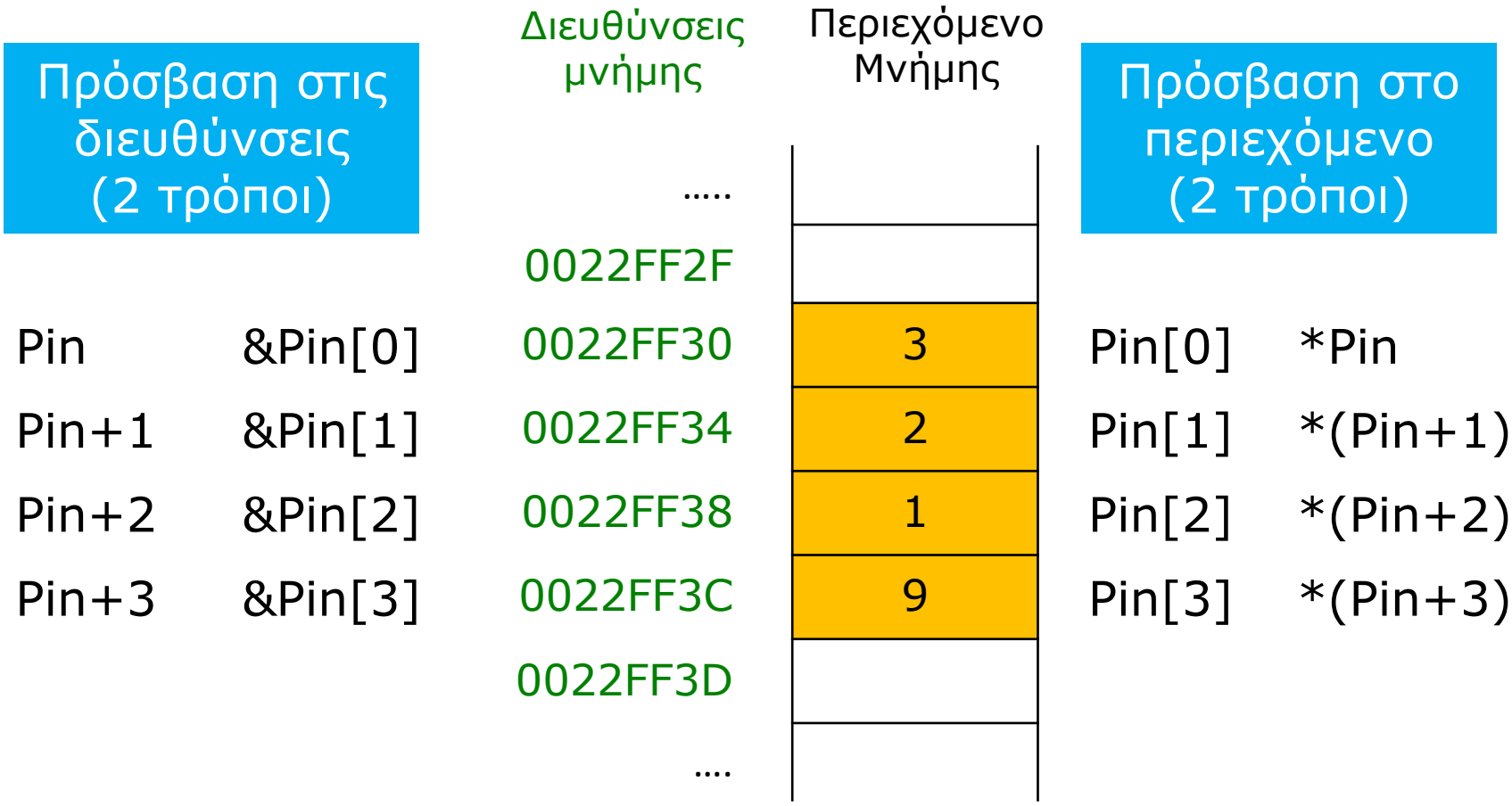
```
#include <stdio.h>
#define SIZE 4
int main()
{
    int Pin[SIZE] = {3, 2, 1, 9};
    int i;
    printf("Show Pin using indexing\n");
    for (i = 0; i < SIZE ; i++)
        printf("Pin[%1d]= %2d, mem addr: %p\n", i, Pin[i], &Pin[i]);
    printf("Show Pin using pointers\n");
    for (i = 0; i < SIZE ; i++)
        printf("Pin[%1d]= %2d, mem addr: %p\n", i, *(Pin+i), Pin+i);
}
```

Ένας υπολογισμός Μέσου Όρου

```
#define N 5
main() {
        int Arr[N];
        int i, sum=0;
        float MO;
        printf("Doste stoixeia:\n");
        for(i=0;i<N;i++)
                scanf("%d",(Arr+i)); //αντί για &Arr[i]
        sum=0;
        for(i=0;i<N;i++)
                sum=sum+*(Arr+i);  //αντί για Arr[i]
        MO=(float)sum/N;
        printf("MO: %f\n",sum);
}
```

Τι γίνεται όταν δηλώνεται ένας πίνακας

int Pin[SIZE] = {3, 2, 1, 9};



Τι διαφορά έχει ένας δείκτης από ένα πίνακα

- Σε τι διαφέρουν:

  int Pin[10];

  *int *Pin;*

int Pin[10];

- *δημιουργεί ένα δείκτη Pin,*
- *δεσμεύει 10 θέσεις ακεραίων,*
- *δίνει αρχική τιμή στο Pin τη διεύθυνση του 1ου ακεραίου και*
- *δεν μπορεί ν' αλλάξει η τιμή του όταν εκτελείται το πρόγραμμα.*

int *Pin;

- *δημιουργεί ένα δείκτη σε ακέραια μεταβλητή,*
- *ο δείκτης αυτός δεν έχει αρχική τιμή,*
- *δεν υπάρχει δέσμευση θέσεων και*
- *μπορούμε ν' αλλάξουμε τη τιμή του.*

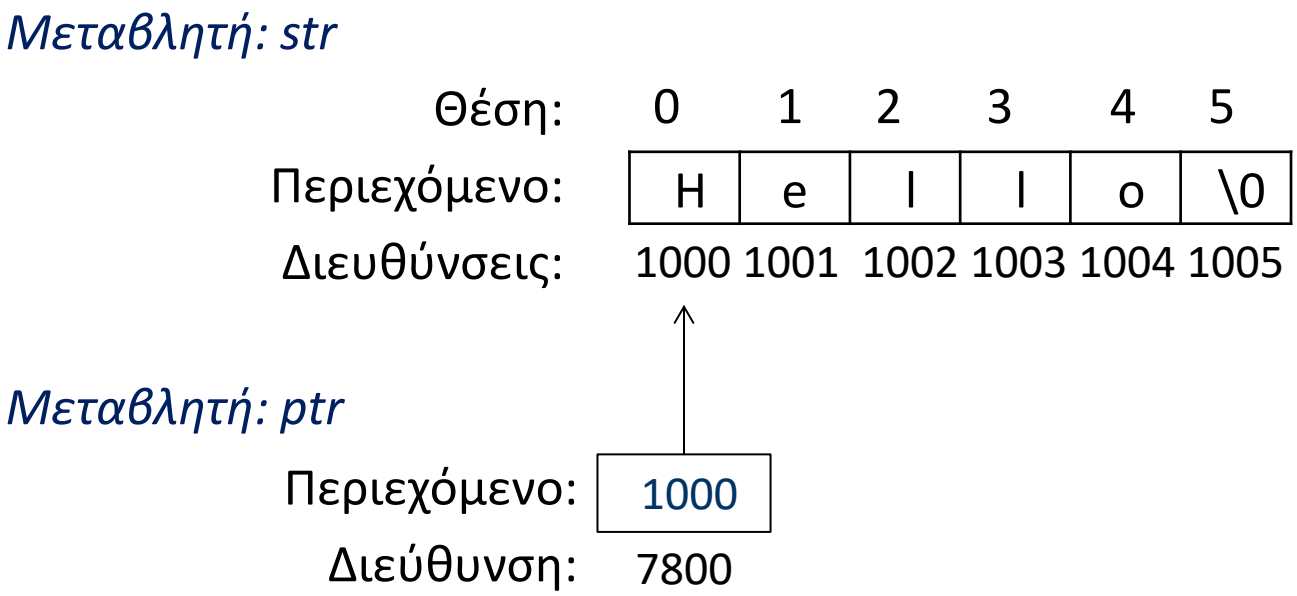
Άλλοι υπολογισμοί του Μέσου Όρου

```
#define N   5
main() {
    int Arr[N], i, sum=0;
    float MO;
    int *ptr;
    printf(" Doste stoixeia:\n");
    for(i=0;i<N;i++)
                scanf("%d",(Arr+i));
     ptr = Arr;    // Arr=&Arr[0]
     sum=0;
     for(i=0;i<N;i++){
              sum=sum+*ptr;
              ptr++;
     }
     MO=(float)sum/N;
    printf("MO: %f\n", MO);
}
```

```
#define N   5
main() {
  int Arr[N], i,sum=0;
  float MO;
  int *ptr = Arr;   // Arr=&Arr[0]
  printf(" Doste stoixeia:\n");
  for(i=0;i<N;i++)
          scanf("%d", (ptr+i));
   printf("You have entered:\n");
   for(i=0;i<N;i++)
            printf("%d\n", *(Arr+i));
    sum=0;
    for(i=0;i<N;i++)
            sum=sum+*(ptr+i);
    MO=(float)sum/N;
     printf("MO: %f\n",MO);
}
```

Δείκτες και συμβολοσειρές

- Δημιουργώντας ένα δείκτη σε συμβολοσειρά:
  - *char str[6] = "Hello"; //δήλωση συμβολοσειράς*
  - *char *ptr ; //δήλωση δείκτη τύπου χαρακτήρα*
  - *ptr = str; //σύνδεση δείκτη με τη συμβολοσειρά*



Χειρισμός συμβολοσειράς με δείκτη

- Εμφάνιση μιας συμβολοσειράς

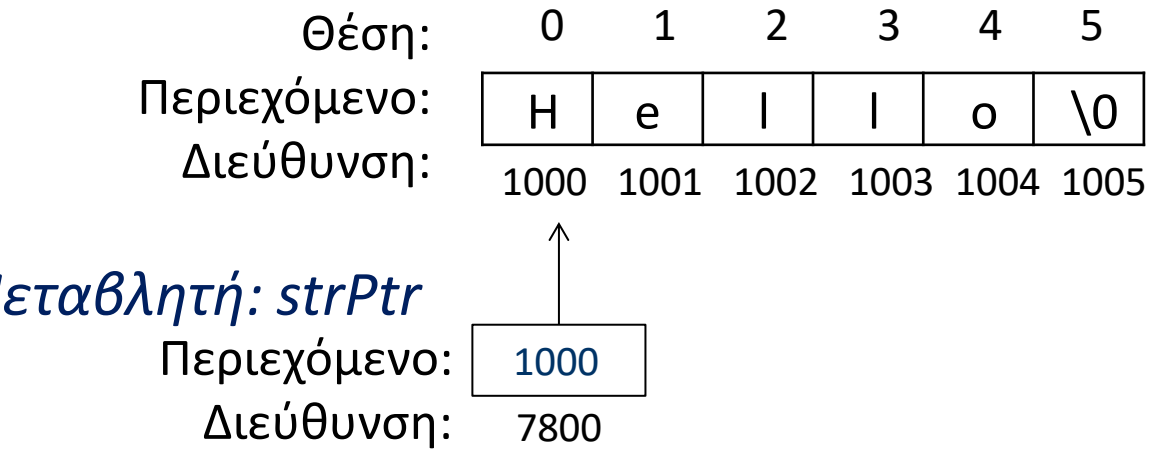
```
#include<stdio.h>
main() {
     char str[80], *ptr;
     printf("String:");
     gets(str);
     ptr=str;
//εμφάνιση της συμβολοσειράς
    puts(str); // σαν str
     puts(ptr); // σαν ptr
     printf("%s\n",str);
     printf("%s\n",ptr);
}
```

```
#include<stdio.h>
main() {
  char str[80], *ptr;
  printf("String:");
  gets(str);
  ptr = str;  //σύνδεση δείκτη με str
  /* εμφανίζω τη str χαρακτήρα-
   χαρακτήρα, με χρήση του ptr */
  while(*ptr != '\0') {
            printf("%c", *ptr);
            ptr++;
    }
}
```

Χειρισμός συμβολοσειράς με δείκτη

Δεν είναι πίνακας

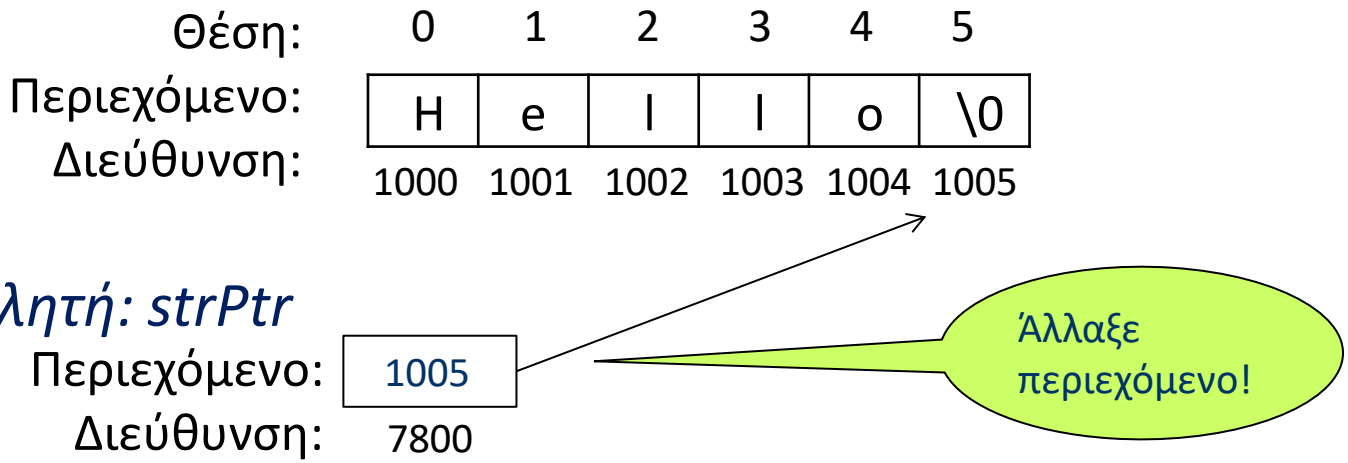
```
#include <stdio.h>
main() {
  char *strPtr = "Hello"; //αρχική τιμή
  puts(strPtr); //Εμφανίζω τη  συμβολοσειρά
 //Άλλος τρόπος χειρισμού: χαρακτήρα-χαρακτήρα
   while(*strPtr != '\0') {
            printf("%c", *strPtr);
            strPtr++;
  }
}
```



Χειρισμός συμβολοσειράς με δείκτη

Προσοχή: μετά την εμφάνιση της συμβολοσειράς, ο strPtr δείχνει στη διεύθυνση του τελευταίου στοιχείου της συμβολοσειράς

```
#include <stdio.h>
int main(void) {
   char *strPtr = "Hello"; //αρχική τιμή
  puts(strPtr); // Εμφανίζω τη  συμβολοσειρά
 //Άλλος τρόπο χειρισμού
   while(*strPtr != '\0') {
            printf("%c", *strPtr);
            strPtr++;
  }
}
```

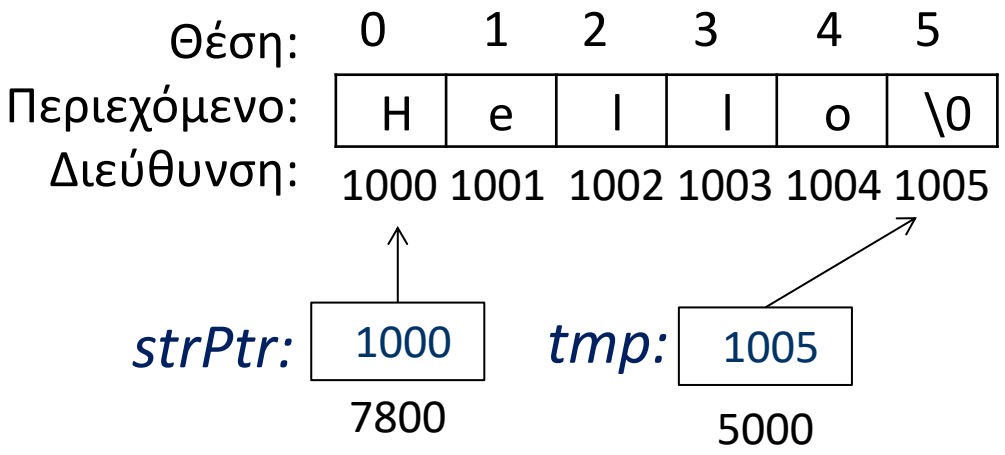


Μια λύση στο προηγούμενο πρόβλημα

- Με χρήση ενός προσωρινού δείκτη (*tmp)

Έτσι διατηρώ το περιεχόμενο του αρχικού δείκτη

```
#include <stdio.h>
int main() {
  char *strPtr = "Hello";
  // προσωρινός δείκτης. Δείχνει όπου ο strPtr
  char *tmp = strPtr;
  while(*tmp != '\0') {
        printf("%c", *tmp);
        tmp++; //αυξάνω μόνο τον προσωρινό
  }
}
```



```
char *str;
 str = "CProgramming";
 printf("\n%s", str + 0);
 printf("\n%s", str + 1);
 printf("\n%s", str + 2);
 printf("\n%s", str + 3);
 printf("\n%s", str + 4);
 printf("\n%s", str + 5);
 printf("\n%s", str + 6);
 printf("\n%s", str + 7);
 printf("\n%s", str + 8);
 printf("\n%s", str + 9);
 printf("\n%s", str + 10);
 printf("\n%s", str + 11);
```

~~`char str[20];`~~
~~`str = "CProgramming";`~~

Αποτέλεσμα:

```
CProgramming
Programming
rogramming
ogramming
gramming
ramming
amming
mming
ming
ing
ng
g
```

Πίνακες δεικτών

- Στη C μπορούμε να δηλώσουμε πίνακες τα στοιχεία των οποίων είναι δείκτες:

*int *Pptr[10] // ο Pptr περιέχει 10 δείκτες σε ακέραιο*

Pptr[5] = &A; // ο A έχει δηλωθεί σαν ακέραιος

**Pptr[5] =12; // ίδιο με A=12;*

```
int *Pptr[5];
int i, i1=2, i2=0, i3=5, i4=1, i5=0;
Pptr[0]=&i1; Pptr[1]=&i2; Pptr[2]=&i3;
Pptr[3]=&i4; Pptr[4]=&i5;
for (i = 0; i < 5 ; i++)
        printf("*Pptr[%1d] = %d\n", i, *Pptr[i]);
```

Αποτέλεσμα:

```
*Pptr[0] = 2
*Pptr[1] = 0
*Pptr[2] = 5
*Pptr[3] = 1
*Pptr[4] = 0
```

Δείκτες και συμβολοσειρές

```
int msg_no;
  char *str []= {
          "Low ink",
          "Paper Jam",
          "A4 Tray is empty",
          "A3 Tray is empty",
          "Printer needs service",
          "There are waiting jobs",
          "Printing now",
          "A4 is set on",
          "A3 is set on"
    } ;
   srand(time(NULL));
   msg_no=rand() % 9;
   printf("Message: ***%s!\n", str[msg_no]);
```

Εμφανίζει το κατάλληλο μήνυμα ανάλογα με τη κατάσταση του εκτυπωτή

Δείκτες και συμβολοσειρές

```
#include <stdio.h>
main()
{
    int *pa, *pb;
    int a=5;
    pa=&a;
    pb=pa;
    *pb=*pa+1;
    *pa=*pb-2;
    printf("%d ",a);
}
```

Ίδιο αποτέλεσμα:
4

```
#include <stdio.h>
main() {
  int *pa, **pb;
  int a=5;
  pa=&a;
  pb=&pa;
  **pb=*pa+1;
  *pa=**pb-2;
  printf("%d ",a);
}
```