

Εισαγωγή στον Προγραμματισμό (Introduction to Programming)

(CEID_NY131)

Δείκτες (Pointers)

```
4 int num;  
5 int *num_ptr;  
6 int **ptr;  
7  
8 int main(int argc, char *argv[])  
9     num=12;  
10    num_ptr=&num;  
11    ptr=&num_ptr;
```

<https://sites.google.com/view/i2art-of-programming>

```
int num;  
int *ni  
int **  
  
int ma (int ar  
nu 12;  
num_ptr=&ni  
ptr=&num_ptr;
```

Κεφάλαιο **19**

Δείκτες και
σύνθετες Δηλώσεις



Πηγή:

[Μια Εισαγωγή στην
Τέχνη του
Προγραμματισμού](#)

Pointers



Συχνές ερωτήσεις

- Τι είναι **διεύθυνση**;
- Πώς και πότε μπορώ να πάρω την διεύθυνση μιας μεταβλητής;
- Που αποθηκεύω μια διεύθυνση;
- Τι είναι **δείκτης**;
- Πως δηλώνω ένα δείκτη;
- Έχει ένας δείκτης διεύθυνση;



Διεύθυνση

- Για κάθε μεταβλητή υπάρχει μια μοναδική διεύθυνση που προσδιορίζει το χώρο της μνήμης στον οποίο είναι αποθηκευμένη η τιμή της
- η διεύθυνση μιας μεταβλητής προκύπτει με την εφαρμογή πάνω στη μεταβλητή του **τελεστή άμεσης διεύθυνσης &**

π.χ. **&num,** **&ch**

Δίνει τη διεύθυνση της θέσης μνήμης που έχει δεσμευθεί για την μεταβλητή num



Δείκτης (Pointer)

- **είναι**

- **μία μεταβλητή** που χρησιμοποιείται για την αποθήκευση διεύθυνσης μιας άλλης μεταβλητής.



Δήλωση Δείκτη

- για τη δήλωση μεταβλητής δείκτη χρησιμοποιείται
 - ο μοναδιαίος **τελεστής άμεσης αναφοράς ***
- **Επιπλέον απαιτείται** η δήλωση του τύπου του αντικειμένου του οποίου τη διεύθυνση περιέχει ή θα περιέχει.

int *num_ptr;

Η **num_ptr** είναι δείκτης σε **int**.

char *nameptr;

Η **nameptr** είναι δείκτης σε **char**.

float *f_ptr;

Η **f_ptr** είναι δείκτης σε **float**.

Απαντήσεις (1/2)

Η πρόταση δεσμεύει χώρο στη μνήμη για την αποθήκευση διεύθυνσης μεταβλητής τύπου `int`.

```
int num = 12;
```

```
int *num_ptr;
```

```
num_ptr = &num
```

Μνήμη

Διευθύνσεις

1000

1002

1004

1008

100A

12

1004

Η πρόταση αποδίδει τη διεύθυνση της μεταβλητής `num` ως τιμή στη μεταβλητή δείκτη `num_ptr`.

Απαντήσεις (2/2)

Η έκφραση έχει ως τιμή την τιμή της ακέραιας μεταβλητής που είναι αποθηκευμένη στη θέση μνήμης στην οποία δείχνει ο δείκτης **num_ptr**. Δηλαδή 12.

```
int num = 12;
```

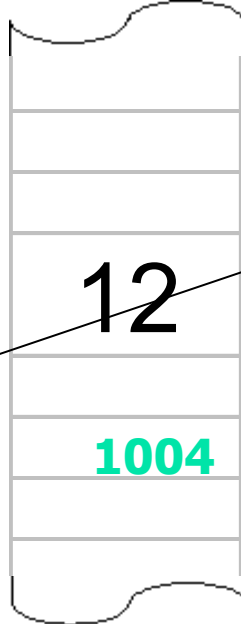
```
int *num_ptr;
```

```
num_ptr = &num;
```

***num_ptr**

&num_ptr

Μνήμη



Διευθύνσεις

1000

1002

1004

1008

100A

Η έκφραση έχει ως τιμή τη διεύθυνση της μεταβλητής δείκτη **num_ptr**.

Για την αποθήκευση της διεύθυνσης της **num_ptr** απαιτείται η δήλωση μιας μεταβλητής δείκτη σε δείκτη σε int. i.e., `int **ptr;`
`ptr = &num_ptr;`

Παράδειγμα 1 (1/3)

PointersTest1.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int num;
5  int *num_ptr;
6  int **ptr;
7
8  int main(int argc, char *argv[]) {
9      num=12;
10     num_ptr=&num;
11     ptr=&num_ptr;
12
13     printf("1.num=%d\n", num);
```

Η **num_ptr** είναι μεταβλητή δείκτη που δείχνει σε int

Η **ptr** είναι μεταβλητή δείκτη που δείχνει σε δείκτη σε int

Παράδειγμα 1 (2/3)

```
8 int main(int argc, char *argv[]) {
9     num=12;
10    num_ptr=&num;
11    ptr=&num_ptr;
12
13    printf("1.num=%d\n", num);
14    printf("2.address of num in memory is %p\n", &num);
15    printf("3.num_ptr=%p\n", num_ptr);
16    printf("4.address of num_ptr in memory is %p\n", &num_ptr);
17    printf("5.ptr=%p\n", ptr);
18    printf("6.address of num_ptr in memory is %p\n", *ptr);
19    printf("7.num=%d\n", *num_ptr);
20    printf("8.num=%d\n", **ptr);
21    printf("9.address of ptr in memory is %p\n", &ptr);
22    return 0;
23 }
```

Παράδειγμα 1 (3/3)

```
8 int main(int argc, char *argv[]) {
9     num=12;
10    num_ptr=&num;
11    ptr=&num_ptr;
12
13    printf("1.num=%d\n", num);
14    printf("2.address of num in memory is %p\n", &num);
15    printf("3.num_ptr=%p\n", num_ptr);
16    printf("4.address of num_ptr in memory is %p\n", &num_ptr);
17    printf("5.ptr=%p\n", ptr);
18    printf("6.address of num_ptr in memory is %p\n", *ptr);
19    printf("7.num=%d\n", *num_ptr);
20    printf("8.num=%d\n", **ptr);
21    printf("9.address of ptr in memory is %p\n", &ptr);
22    return 0;
23 }
```

C:\Code\courses\I2P22-23\PointersTest1\PointersTest1.exe

```
1.num=12
2.address of num in memory is 00405434
3.num_ptr=00405434
4.address of num_ptr in memory is 00405438
5.ptr=00405438
6.address of num_ptr in memory is 00405434
7.num=12
8.num=12
9.address of ptr in memory is 00405430
```

Process exited after 0.07675 seconds with return value 0
Press any key to continue . . .

Δείκτης σε πίνακα - Αριθμητική Δεικτών

```
4  int ar[10]={1,2,3,4,5,6,7,8,9,10};  
5  int *arPtr=ar;
```

- Το όνομα του πίνακα (**ar**) είναι δείκτης στο πρώτο στοιχείο του πίνακα.
 - Αυτός είναι ο λόγος που οι πίνακες ως ορίσματα συναρτήσεων περνάνε by reference.
- Αν αυξήσουμε τον **arPtr** κατά ένα θα δείχνει στο δεύτερο στοιχείο του πίνακα, δηλαδή στο **ar[1]**.
 - Αυτό μας επιτρέπει να αναφερθούμε στο δεύτερο στοιχείο του πίνακα με την έκφραση ***(arPtr+1)** ή την ***(++arPtr)** ή την ***(ar+1)**. Προσέξτε! η έκφραση ***(++ar)** δεν είναι αποδεκτή καθώς η ar είναι σταθερά.

Έτσι αν η τιμή του arPtr είναι 1000 η τιμή του arPtr+1 δεν θα είναι 1001 αλλά η διεύθυνση του επόμενου ακέραιου στην μνήμη (**αριθμητική δεικτών**).

Παράδειγμα 2 (1/3)

PointersAndArrays.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int ar[10]={1,2,3,4,5,6,7,8,9,10};
5  int *arPtr=ar;
6
7  void printArray(int ar[], int size);
8
9  int main(int argc, char *argv[]) {
10     int i;
11
12     for(i=0;i<10;i++)
13         // printArray(&ar[i],10-i); //1
14         // printArray(ar+i,10-i); //2
15         printArray(arPtr++,10-i); //3
```

Η **arPtr** είναι μεταβλητή δείκτη σε ακέραιο και την βάλαμε να δείχνει το πρώτο στοιχείο του πίνακα **ar**, καθώς το ar είναι δείκτης στο πρώτο στοιχείο του πίνακα.

Η **printArray** είναι συνάρτηση που δέχεται ως 1^ο όρισμα ένα πίνακα ακεραίων και ως 2^ο τον αριθμό των στοιχείων του και τυπώνει τα στοιχεία του.

Η πρόταση for καλεί την printArray για να τυπώσει τα στοιχεία του πίνακα από το i στοιχείο μέχρι το τελευταίο.

- (1) Περνάει ως όρισμα την διεύθυνση του i στοιχείου του πίνακα
- (2) Περνάει ως όρισμα τον δείκτη στο πρώτο στοιχείο του πίνακα αυξημένο κατά i.
- (3) Περνάει ως όρισμα τον δείκτη arPtr ο οποίο έχει αυξηθεί i φορές κατά 1 και άρα δείχνει στο i στοιχείο του πίνακα ar.

Παράδειγμα 2 (2/3)

Η **arPtr** είναι τοπική μεταβλητή δείκτη σε ακέραιο και την βάζουμε να δείχνει το πρώτο στοιχείο του πίνακα **ar**, που δέχεται ως όρισμα η `printArray()`.

```
12     for(i=0;i<10;i++)
13         //      printArray(&ar[i],10-i);    //1
14         //      printArray(ar+i,10-i);      //2
15         //      printArray(arPtr++,10-i);    //3
16     return 0;
17 }
18
19 void printArray(int ar[], int size){
20     int i;
21     int *arPtr=ar;
22     for(i=0;i<size;i++)
23         //      printf("%d\t",ar[i]);        //1
24         //      printf("%d\t",*ar+i);        //2
25         //      printf("%d\t",*arPtr++);     //3
26     printf("\n");
27 }
```

Η πρόταση `for` καλεί την `printf` για να τυπώσει τα στοιχεία του πίνακα που η `printArray` δέχεται ως όρισμα από το πρώτο στοιχείο του μέχρι το τελευταίο.

- (1) Περνάει ως όρισμα το `i` στοιχείο του πίνακα `ar`.
- (2) Περνάει ως όρισμα το περιεχόμενο που δείχνει ο δείκτης που προκύπτει από την πρόσθεση στον δείκτη `ar` του `i`, δηλαδή το στοιχείο `ar[i]`.
- (3) Περνάει ως όρισμα το περιεχόμενο του δείκτη `arPtr` ο οποίος έχει αυξηθεί `i` φορές κατά 1 και άρα δείχνει στο `i` στοιχείο του πίνακα `ar`.

Παράδειγμα 2 (3/3)

C:\Code\courses\I2P22-23\PointersAndArrays\PointersAndArrays.exe

1	2	3	4	5	6	7	8	9	10
2	3	4	5	6	7	8	9	10	
3	4	5	6	7	8	9	10		
4	5	6	7	8	9	10			
5	6	7	8	9	10				
6	7	8	9	10					
7	8	9	10						
8	9	10							
9	10								
10									

Process exited after 0.1801 seconds with return value 0
Press any key to continue . . .

```
12     for(i=0;i<10;i++)
13         //     printArray(&ar[i],10-i); //1
14         //     printArray(ar+i,10-i); //2
15         //     printArray(arPtr++,10-i); //3
16     return 0;
17 }
18
19 void printArray(int ar[], int size){
20     int i;
21     int *arPtr=ar;
22     for(i=0;i<size;i++)
23         //     printf("%d\t",ar[i]); //1
24         //     printf("%d\t",*ar+i); //2
25         //     printf("%d\t",*arPtr++); //3
26     printf("\n");
27 }
```

Παράδειγμα 3 (1/2)

```
DisplayCommandLineArg.c HelloWorld.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char *argv[]) {
5      int i;
6
7      for(i=1; i<argc; i++)
8          printf("%s ", argv[i]);
9      printf("\n");
10     printf("argc=%d\n", argc);
11     for(i=0; i<argc; i++)
12         printf("argv[%d]=%s\n", i, argv[i]);
13     return 0;
14 }
```

Η **argv** είναι πίνακας δεικτών σε **char**.

Η **argc** είναι μεταβλητή με τιμή τον αριθμό των παραμέτρων της γραμμής διαταγών για την κλίση του προγράμματος, συμπεριλαμβανομένου του ονόματος του προγράμματος.

Η πρόταση τυπώνει μόνο τις παραμέτρους με τις οποίες κλήθηκε το πρόγραμμα.

Η πρόταση τυπώνει όλα τα στοιχεία του πίνακα **argv**.

Παράδειγμα 3 (2/2)

Έξοδος Προγράμματος

```
C:\Code\courses\I2P22-23\DisplayCommandLineArg>DisplayCommandLineArg Hello World
```

```
Hello World
```

```
argc=3  
argv[0]=DisplayCommandLineArg  
argv[1]=Hello  
argv[2]=World
```

Command line για την κλήση του προγράμματος από την γραμμή διαταγών με δύο παραμέτρους, τις Hello και World.

```
DisplayCommandLineArg.c  HelloWorld.c  
1  #include <stdio.h>  
2  #include <stdlib.h>  
3  
4  int main(int argc, char *argv[]) {  
5      int i;  
6  
7      for(i=1;i<argc;i++)  
8          printf("%s ", argv[i]);  
9      printf("\n");  
10     printf("argc=%d\n", argc);  
11     for(i=0;i<argc;i++)  
12         printf("argv[%d]=%s\n", i,argv[i]);  
13     return 0;  
14 }
```