Εισαγωγή στον δομημένο προγραμματισμό

Ενότητα 7η: Δείκτες

Αν. καθηγητής Στεργίου Κώστας
e-mail: kstergiou@uowm.gr

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Στόχοι της διάλεξης

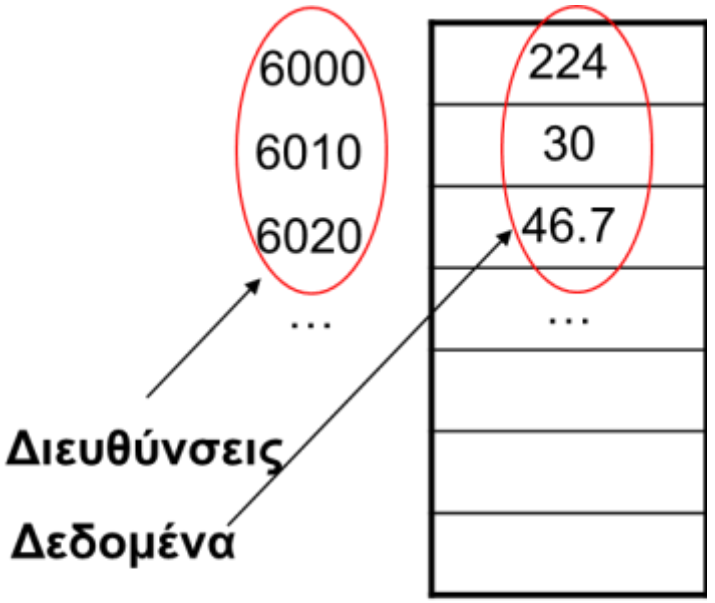
- Κατανόηση της έννοιας του δείκτη στη C.
- Περιγραφή των βασικών χρήσεων των δεικτών για χειρισμό πινάκων.
- Ανάλυση των προβλημάτων που μπορούν να δημιουργηθούν από την λανθασμένη χρήση δεικτών.
- Εισαγωγή σε προχωρημένα θέματα δεικτών.

Περιεχόμενα

- Δείκτες.
  - Ο τύπος δείκτη.
  - Δήλωση δείκτη.
  - Τελεστές για δείκτες.
- Δείκτες και πίνακες.
  - Παραδείγματα .
- Αριθμητική δεικτών.
- Δείκτες και αλφαριθμητικά.
  - Παραδείγματα .
- Πίνακες δεικτών.

Τι είναι ένας δείκτης (pointer);

- Όταν δηλώνουμε μια μεταβλητή σε ένα πρόγραμμα C, ο compiler κρατάει μια θέση μνήμης με μια μοναδική διεύθυνση για την αποθήκευση της.
- Ο compiler αντιστοιχεί αυτή τη διεύθυνση με το όνομα της μεταβλητής.
- Όταν το πρόγραμμά χρησιμοποιεί τη μεταβλητή, βρίσκει αυτόματα στη σωστή θέση μνήμης.
- Η διεύθυνση της θέσης είναι άγνωστη στο χρήστη.
- Ο δείκτης μας δίνει τη δυνατότητα να έχουμε πρόσβαση σε αυτή τη διεύθυνση.




```
int var1 = 224;
int x = 30;
float y = 46.7;
…
```

Τύπος Δείκτη (pointer) (1/2)

- Δείκτης είναι μία μεταβλητή στην οποία αποθηκεύεται μία διεύθυνση της κύριας μνήμης του υπολογιστή.
- Συνήθως στις γλώσσες υψηλού επιπέδου δεν υπάρχει αυτή η δυνατότητα.
- Η χρήση δεικτών δίνει τη δυνατότητα δημιουργίας πολύ αποδοτικού (γρήγορου) κώδικα.
  - Αλλά εγκυμονεί και κινδύνους…

- Υπάρχει άμεση αντιστοίχηση μεταξύ πινάκων και δεικτών.
  - Ότι λειτουργία γίνεται με πίνακες μπορεί να γίνει και με δείκτες.
- Από τα ισχυρά πλεονεκτήματα της C.
- **ΠΡΟΣΟΧΗ:**
  - Δείκτης πίνακα (index).
  - Τύπος δείκτη (pointer).

Τύπος Δείκτη (pointer) (2/2)

- Ο δείκτης είναι μια μεταβλητή που περιέχει τη διεύθυνση μιας άλλης μεταβλητής.
- Ένας δείκτης είναι μια αριθμητική μεταβλητή και, όπως όλες οι μεταβλητές, πρέπει να δηλώνεται πριν μπορέσει να χρησιμοποιηθεί.
- Τα ονόματα των δεικτών ακολουθούν τους ίδιους κανόνες με άλλες μεταβλητές.
- Μια δήλωση δείκτη παίρνει τη μορφή:

```
type *ptrname;
```

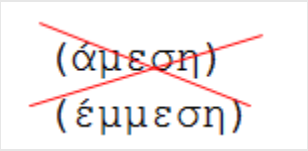
- Υποδεικνύει τον τύπο της μεταβλητής.
- Ο αστερίσκος (*) είναι δείχνει ότι *ptrname* είναι ένας δείκτης για τον τύπο *type* και όχι μεταβλητή τύπου *type*.

Δήλωση Δείκτη

- <τύπος> *<όνομα δείκτη>;
  - `int *num_ptr;`
  - Η μεταβλητή num_ptr είναι δείκτης σε int.
  - Το περιεχόμενο της θέσης μνήμης που δείχνει ο num_ptr είναι int.
- **Αρχικοποίηση δείκτη.**
- <τύπος> *<όνομα δείκτη> = <διεύθυνση>;
  - `int *num_ptr = 1000;` (άμεση)
  - `int *num_ptr = &num;` (έμμεση)
    - Ο δείκτης num_ptr δείχνει στη μεταβλητή num.
    - *Όταν ο τελεστής & τοποθετείται πριν από το όνομα μιας μεταβλητής, επιστρέφει τη διεύθυνση της μεταβλητής.*

Ανάθεση Τιμή σε Δείκτη

- Αν ένας δείκτης δεν έχει αρχικοποιηθεί πρέπει οπωσδήποτε να του ανατεθεί μία τιμή (διεύθυνση) πριν χρησιμοποιηθεί.
  - Για σιγουριά φροντίστε να αρχικοποιείτε τους δείκτες.
  - Δείκτες χωρίς τιμή έχουν καταστροφικά αποτελέσματα στα προγράμματα.
- <όνομα δείκτη> = <διεύθυνση>;
- `num_ptr = 1000;`
- `num_ptr = &num;`



- Οι άμεσες αρχικοποιήσεις και αναθέσεις τιμών σε δείκτες πρέπει να αποφεύγονται.
- Όταν δεν θέλουμε συγκεκριμένη αρχικοποίηση για κάποιον δείκτη, τον αρχικοποιούμε σε NULL.

```
int *num_ptr = NULL;
```

Τελεστές που σχετίζονται με Δείκτες

*** και &**

*****

- Τελεστής περιεχομένου.
- Εφαρμόζεται μόνο σε δείκτες.
- Δίνει το περιεχόμενο (τιμή) της διεύθυνσης που δείχνει ο δείκτης.
- Διαφέρει από τον πολλαπλασιασμό στο ότι έχει ένα μόνο τελεστέο.

&

- Τελεστής διεύθυνσης
- Δίνει τη διεύθυνση της μεταβλητής στην οποία εφαρμόζεται.
- Το λογικό ΚΑΙ είναι `&&` .

```
int *num_ptr, num;

num = 20;
num_ptr = &num;
printf("%d\n",*num_ptr);
```

Τι θα εκτυπωθεί αν παραλείψουμε το *;

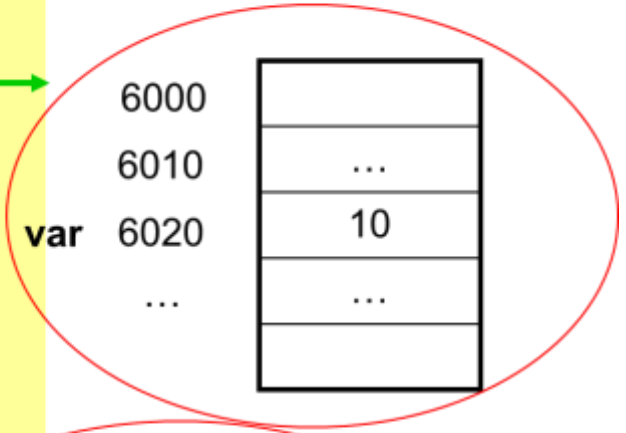
Παράδειγμα 1ο

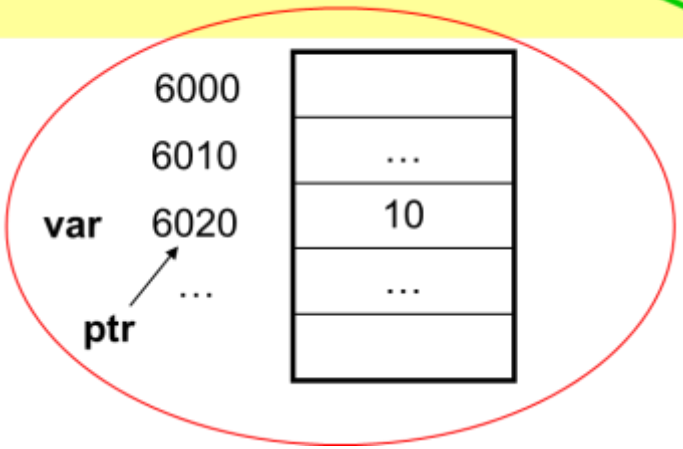
```
#include <stdio.h>
int var = 1;
int *ptr;
int main() {
     ptr = &var;
     printf("\nDirect access, var = %d", var);
     printf("\nIndirect access, var = %d", *ptr);
     printf("\n\nThe address of var = %d", &var);
     printf("\nThe address of var = %d\n", ptr);
     return 0;
}
```

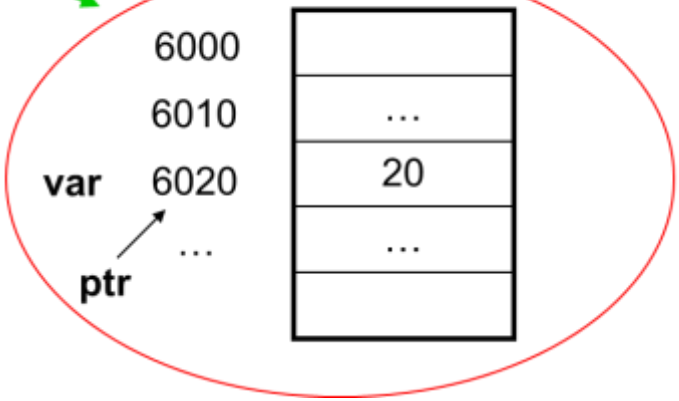
Direct access, var = 1
Indirect access, var = 1
The address of var = 4264228
The address of var = 4264228

Προσοχή στη χρήση δεικτών

```
#include <stdio.h>
int main() {
int var;
int *ptr;
   var = 10;
   ptr = &var;
   printf("The value of var is %d\n", var);
   *ptr = 20;
   printf("The value of var is %d\n", var);
   return 0;
}
```







Παράδειγμα 2ο

```
int x=1;          /* Αρχικοποίηση x */
int y=2;          /* Αρχικοποίηση y */
int *p,*q;        /* Δήλωση p, q */
p = &x;           /* Ο p δείχνει στην x */
y = *p;           /* y=x → y=1 */
*p = 0;           /* x=0 */
*p+=1;            /* x=x+1 → x=1 */
++*p;             /* x=x+1 → x=2 */
(*p)++;           /* x=x+1 → x=3 */
*p++;             /* Το p θα δείχνει στο
                     επόμενο στοιχείο */
q=p;              /* Τα περιεχόμενα του p
                     αντιγράφονται στον q */
```

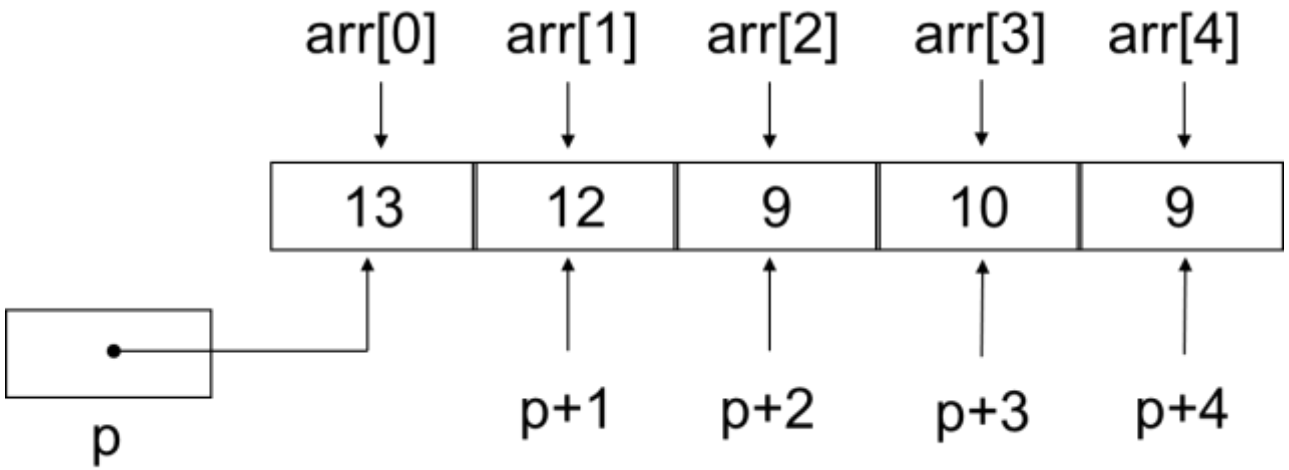
Σχέση Δεικτών με Πίνακες

- Το όνομα ενός πίνακα είναι «σταθερά τύπου δείκτη» με τιμή τη διεύθυνση του πρώτου στοιχείου του πίνακα.
  - `int arr[5];`
  - `arr`: η διεύθυνση του `arr[0]`.
    - Ισοδύναμα: `&arr[0]`.
- Αλλά πρέπει να θυμόμαστε ότι αυτό είναι σταθερά, δεν μπορεί να αλλάξει τιμή κατά τη διάρκεια εκτέλεσης του προγράμματος.
- Μπορούμε να ορίσουμε ένα δείκτη στον πίνακα και να χειριστούμε τον πίνακα μέσω αυτού του δείκτη.

```
int array[100], *p_array;
/* …. */
p_array = array;
```

Παράδειγμα: Δείκτες-Πίνακες

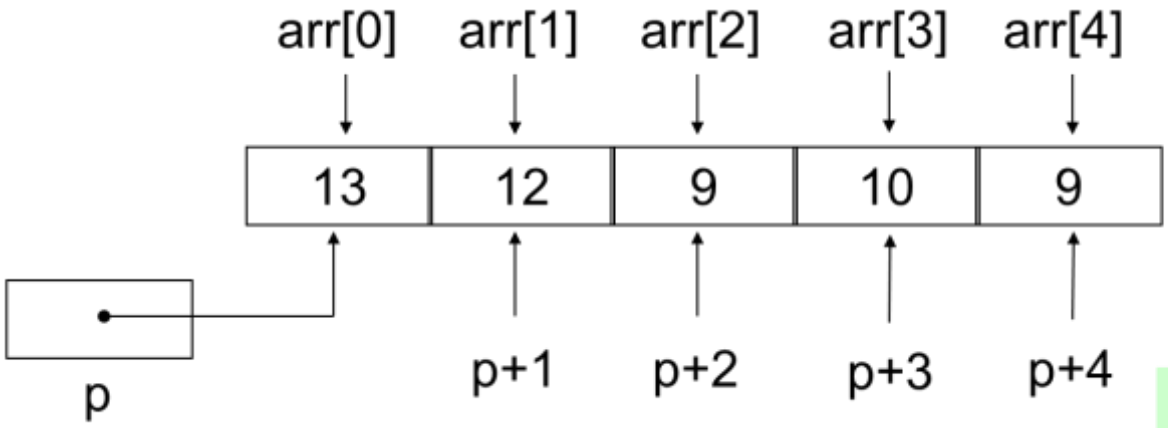
```
int arr[5] = {13, 12, 9, 10, 9};
int *p;
p = &arr[0]; ή p = arr;
```



Αριθμητική δεικτών

- Η αριθμητική των δεικτών είναι απλή.
- Η C διαθέτει μόνο δύο πράξεις για δείκτες: **προσαύξηση** και **μείωση**.
  - Π.χ. `p++, p += 2, p--`
- Όταν *προσαυξάνουμε* ένα δείκτη κατά 1, ο δείκτης αυξάνεται αυτόματα ώστε να παραπέμπει στο επόμενο στοιχείο.
  - Έχει νόημα μόνο όταν ο δείκτης δείχνει σε πίνακα!
- Ο compiler γνωρίζει το είδος των δεδομένων (από τη δήλωση pointer) και αυξάνει τη διεύθυνση του δείκτη σύμφωνα με το μέγεθος του τύπου δεδομένων.
- Η ίδια λογική ισχύει και για τη μείωση pointer.

Παράδειγμα: Δείκτες-Πίνακες



`arr+1` δείχνει στο `arr[1]`

`*(arr+1)` η τιμή του `arr[1]`

Ισοδυναμίες:

`arr+n` ↔ `&arr[n]`

`*(arr+n)` ↔ `arr[n]`

ΛΑΘΟΣ:

```
arr++;
arr = p;
```

ΣΩΣΤΟ:

```
p++;
p = arr;
```

Παράδειγμα 3ο

```
#include <stdio.h>
#define MAX 10
int i_array[MAX] = { 0,1,2,3,4,5,6,7,8,9 };
int *i_ptr, count;
float f_array[MAX] = { 0.0, 0.1, 0.2, 0.3, 0.4, 0.5,
  0.6, 0.7, 0.8, 0.9 };
float *f_ptr;
main() {
     i_ptr = i_array;
     f_ptr = f_array;
     for (count = 0; count < MAX; count++)
         printf("%d\t%f\n", *i_ptr++, *f_ptr++);
     return 0;
}
```

Συμβουλές!

- Μην προσπαθήστε να εκτελέσετε μαθηματικές πράξεις, όπως διαίρεση, πολλαπλασιασμός, κλπ. με δείκτες.
- Με την αφαίρεση ή την προσθήκη σε ένα δείκτη μεταβάλλεται με βάση το μέγεθος του τύπου δεδομένων που δείχνει.
- Αρχικοποιείτε τους δείκτες. Πρέπει να το κάνετε οι ίδιοι. Ο compiler δεν θα το κάνει αυτό για εσάς!
  - Αν δεν θέλουμε να αρχικοποιήσουμε σε συγκεκριμένη διεύθυνση αρχικοποιούμε σε NULL:

```
int *ptr=NULL;
```

Δείκτες και Αλφαριθμητικά

- **Δήλωση δείκτη αλφαριθμητικού:**
  - char *<όνομα δείκτη>;
  - `char *str_ptr;`
- **Ανάθεση τιμής σε δείκτη αλφαριθμητικού:**
  - <όνομα δείκτη> = <αλφαριθμητικό>;
  - `str_ptr = “Hello”;`
- Μην ξεχνάτε ότι το αλφαριθμητικό είναι ένας πίνακας χαρακτήρων που τερματίζει με το χαρακτήρα ‘\0’.

Παράδειγμα: Αντιγραφή ενός αλφαριθμητικού σε ένα άλλο

Χωρίς δείκτες

```
int i=0;
while ((s[i] = t[i]) !='\0')
      i++;
```

Με δείκτες
1η έκδοση

```
char *sp=s;
char *tp=t;
while ((*sp = *tp) !='\0') {
      sp++;
      tp++; }
```

Με δείκτες
2η έκδοση

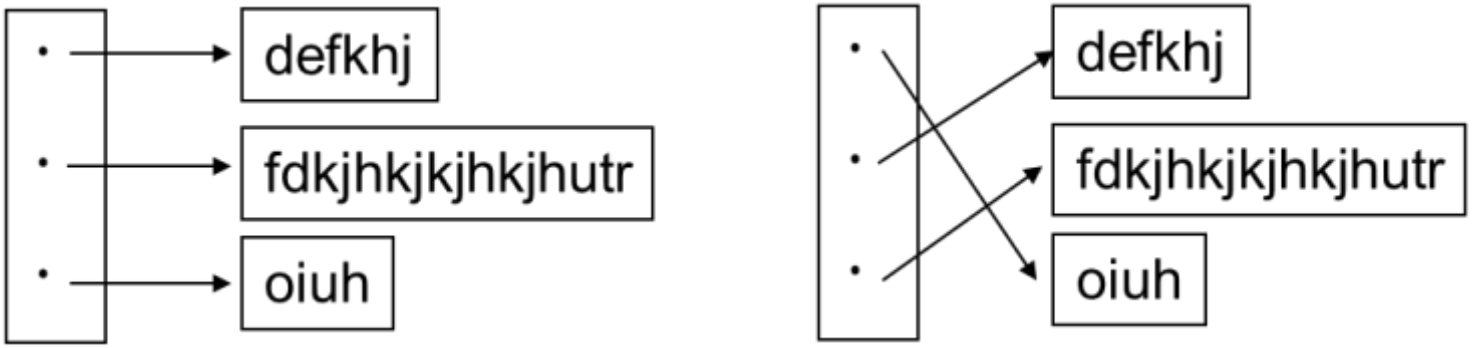
```
char *sp=s;
char *tp=t;
while ((*sp++ = *tp++) !='\0')
      ;
```

Πίνακες Δεικτών

- Αφού οι δείκτες είναι μεταβλητές, μπορούν και αυτοί να αποθηκεύονται σε πίνακες.
  - **char** `str[5]`
    (Αλφαριθμητικό μήκους 5 χαρακτήρων)
  - **char** `*ptr[5];`
    (Πίνακας 5 δεικτών σε αλφαριθμητικά)
- Ευκολία στον χειρισμό αλφαριθμητικών διαφορετικού μεγέθους.

Παράδειγμα: Πίνακες Δεικτών

- Τα αλφαριθμητικά στα οποία δείχνουν οι δείκτες μπορεί να είναι διαφορετικού μήκους.
- Αν θέλουμε να αλλάξουμε τη θέση των αλφαριθμητικών απλά αλλάζουμε τη θέση των δεικτών.



Αρχικοποίηση Πίνακα Δεικτών

- Πίνακας δεικτών:
  - **char** *name[] = {“Λάθος μήνας”,
    “Ιαν”,
    “Φεβ”,
    “Μάρ” };
  - Δέσμευση όσων θέσεων μνήμης χρειάζονται + τον αριθμό των δεικτών.
- Πίνακας αλφαριθμητικών:
  - **char** aname[][15] = {“Λάθος μήνας”,
    “Ιαν”,
    “Φεβ”,
    “Μάρ” };
  - Δέσμευση 15*4 θέσεων μνήμης για char.

Δείκτες και Πολυδιάστατοι Πίνακες

- Δεδομένων των δηλώσεων:
  - `char a[10][20]; /* δισδιάστατος πίνακας */`
  - `char *b[10];  /* πίνακας δεικτών */`
- Τα παρακάτω είναι συντακτικά αποδεκτά:
  - `x=a[3][4];`
  - `x=b[3][4];`

Παράδειγμα: Μέτρηση μήκους του αλφαριθμητικού str

Χωρίς Δείκτες

```
int i=0;
while (str[i++] != '\0');
printf("%d", i);
```

Με Δείκτες

```
int i=0;
char *s=str;
while (*s++ != '\0')
   i++;
printf("%d", i);
```

Αποθήκευση τιμών σε μονοδιάστατο πίνακα

```
/* Δεδομένου πίνακα a[N] */
int *p;
p = a;
for (i=0; i<N; i++)
{
  scanf("%d", p);
  p++;
}
```

Άθροιση τιμών σε μονοδιάστατο πίνακα

```
/* Δεδομένου πίνακα a[N] */

int *p;
int sum=0;
p=a;
for (i=0; i<N; i++)
  sum += *(p+i);
```

Εμφάνιση τιμών μονοδιάστατου πίνακα

```
/* Δεδομένου πίνακα a[N] */

int *p=a;
for (i=0; i<N; i++)
  printf("ΘΕΣΗ %d - ΤΙΜΗ %d\n", i,
  *(p+i));
```

Υπολογισμός Μέσου Όρου Στοιχείων Πίνακα

```
#include <stdio.h>
#define N 40
main()
{
    float temper[N], sum=0.0, *ptr;
    int i;
    ptr=temper;
    for (i=0; i<N; i++)
     {
          printf("ΕΠΟΜΕΝΟ ΣΤΟΙΧΕΙΟ: ");
          scanf("%f",ptr+i);
    }
    for (i=0; i<N; i++)
          sum += *(ptr+i);
    printf("Ο ΜΕΣΟΣ ΟΡΟΣ ΕΙΝΑΙ %f",sum/N);
}
```

Τέλος Ενότητας

Σημείωμα Αναφοράς

- Copyright Πανεπιστήμιο Δυτικής Μακεδονίας, Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών, Στεργίου Κωνσταντίνος. «Εισαγωγή στον Δομημένο Προγραμματισμό». Έκδοση: 1.0. Κοζάνη 2015. Διαθέσιμο από τη δικτυακή διεύθυνση:

  https: //eclass.uowm.gr/courses/ICTE258/

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.