ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

3η Εβδομάδα: Πίνακες

Αναφορές

Οι διαφάνειες της διάλεξης στηρίζονται, εν μέρει, σε υλικό παραδόσεων παλαιότερων ετών του **Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογία Υπολογιστών του Πανεπιστημίου Πατρών** καθώς και του **Τμήματος Πληροφορικής του Πανεπιστήμιο Κύπρου.**

Θέματα Διάλεξης

- Τι είναι οι Πίνακες;
- Γιατί χρειάζονται;
- Σημασία και Σύνταξη Πινάκων
- Αρχικοποίηση και Διαχείριση Πινάκων
- Πολυδιάστατοι Πίνακες

Δομές Δεδομένων – Πίνακες

- Πίνακες αποτελούν ένα σημαντικό **δομημένο τύπο δεδομένων** (structured data type) ή απλά μια **δομή δεδομένων** (data structure).
- Μία δομή δεδομένων είναι ένα **σύνολο συγγενικών δεδομένων** τα οποία **αποθηκεύονται κάτω από το ίδιο όνομα.**
- Καταλαμβάνει **συνεχόμενες θέσεις** μνήμης για διατήρηση Συγκεκριμένου αριθμού στοιχείων **ΙΔΙΟΥ τύπου**

Δήλωση Πίνακα

```
int pinakas1[4];
int pinakas2[4]={};
int pinakas3[4]={1,4,3,2};
```

0	0	0	0
1	4	3	2

1. Ένας πίνακας με όνομα pinakas1 με 4 θέσεις ακεραίων
2. Ένας πίνακας με όνομα pinakas2 με 4 θέσεις ακεραίων
3. Ένας πίνακας με όνομα pinakas3 με 4 θέσεις ακεραίων

```
int md[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

τύπος

όνομα

Μέγεθος/ πλήθος στοιχείων

md[0]	31
md[1]	28
md[2]	31
…	
md[11]	31

τιμές

Αρχικοποίηση Πινάκων

- **Ταυτόχρονος Ορισμός και Αρχικοποίηση (Χρήση { } για αρχικοποίηση πίνακα)**

```
int values[5] = {1, 2, 3, 4, 5};
```

  - Όταν οι αρχικές τιμές είναι λιγότερες από ότι οι θέσεις του πίνακα, τότε οι θέσεις που δεν έχουν τιμές είναι οι πιο δεξιές θέσεις. Η εξ' ορισμού τιμή τους είναι το 0

```
int values[5] = {}; /* ορισμός ενός πίνακα 5 ακεραίων
με αρχικές τιμές το 0  για ΟΛΕΣ του τις θέσεις */
```

  - Όταν οι αρχικές τιμές είναι περισσότερες από τις θέσεις του πίνακα, τότε προκύπτει συντακτικό λάθος!!!

- Όταν το μέγεθος του πίνακα δεν ορίζεται, τότε αυτό καθορίζεται από το πλήθος των τιμών που θέτουμε κατά την αρχικοποίηση του πίνακα. Για παράδειγμα:

```
int values[] = {1, 2, 3, 4, 5}; // ορισμός πίνακα 5
ακεραίων με αρχικές τιμές τους αριθμούς από το 1-5
```

```
char letters[]={'a', 'b', 'c'}; //ορισμός πίνακα 3
χαρακτήρων με αρχικές τιμές τους χαρακτήρες a, b, c
```

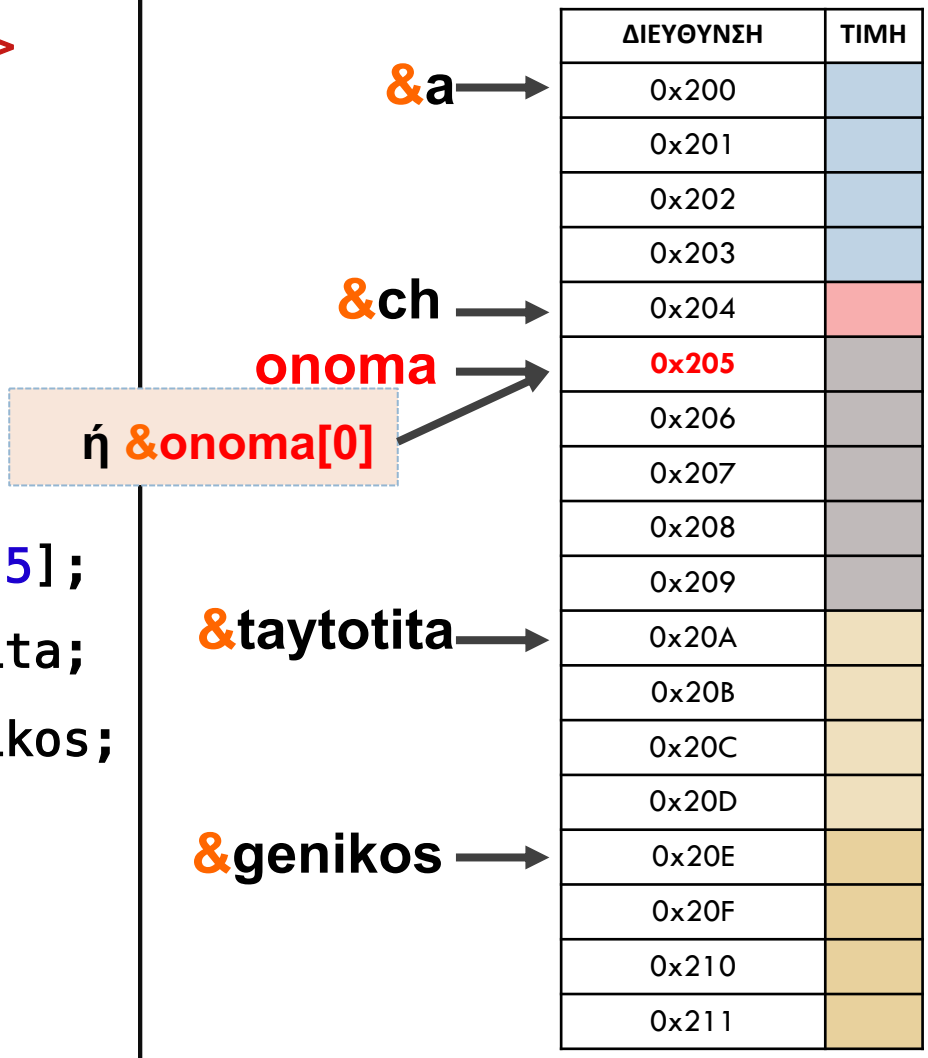
Επεξεργασία σε στοιχεία Πίνακα

- Μπορούμε να θεωρήσουμε **καθένα από τα στοιχεία ενός πίνακα σαν απλή μεταβλητή**. Δηλαδή μπορούμε να του **αναθέσουμε τιμή**, να **διαβάσουμε την τιμή** αυτή και να **αλλάξουμε την τιμή του.** Για παράδειγμα:

```
int values[]={1,2,3,5,7,11,13,17,19,23,29,31};
printf("%d", values[0]);
values[3] = 8258;
sum = values[0] + values[1];
values[3] += 1;
values[2] = values[0] + values[1];
values[2] = values[3] + 1;
```

Πίνακες

```
#include <stdio.h>



int main() {
        int a;
        char ch;
        char onoma[5];
        int taytotita;
         float genikos;
}
```

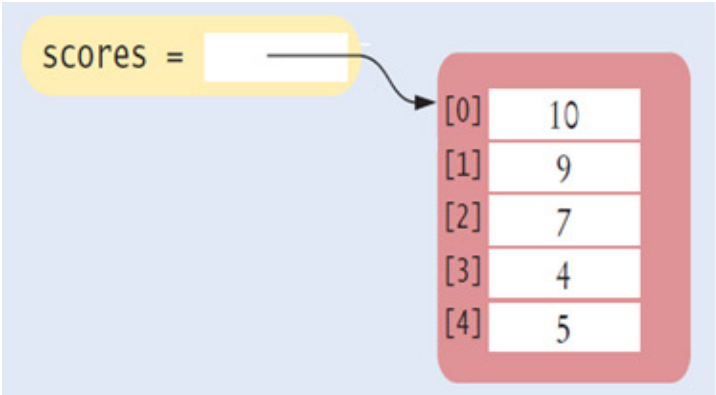


Πίνακες

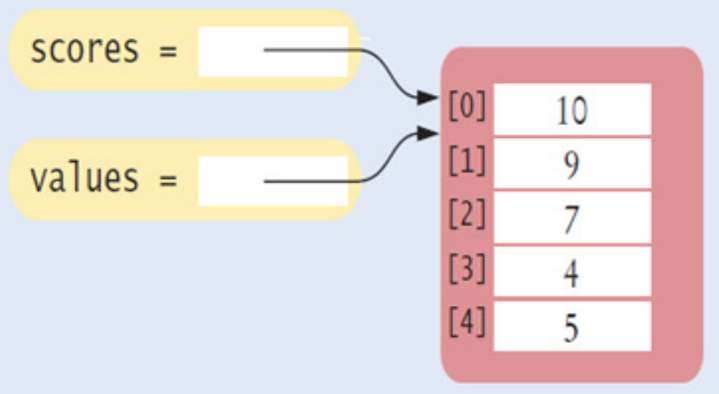
- Ένας πίνακας **ΔΕΝ ΑΝΤΙΓΡΑΦΕΤΑΙ** με τον **τελεστή ανάθεσης**. Με τον τελεστή ανάθεσης απλά **συσχετίζεται ένα δεύτερο όνομα με τον ίδιο ΠΙΝΑΚΑ**.

Το όνομα του πίνακα είναι ουσιαστικά απλά μια σταθερά διεύθυνσης στην RAM

```
int scores[5] = {10, 9, 7, 4, 5};
int values[5];
```




```
values = scores
```

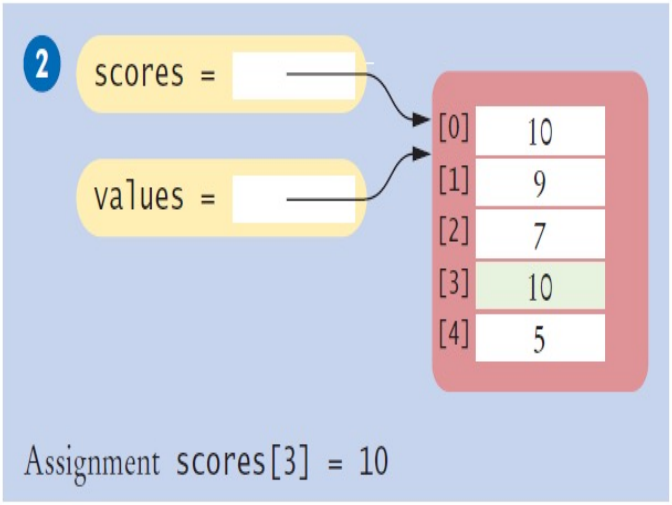


Πίνακες

- Ότι αλλαγή κάνω στην μεταβλητή μέσω του ονόματος **scores** θα φαίνεται και μέσω του ονόματος **values.**

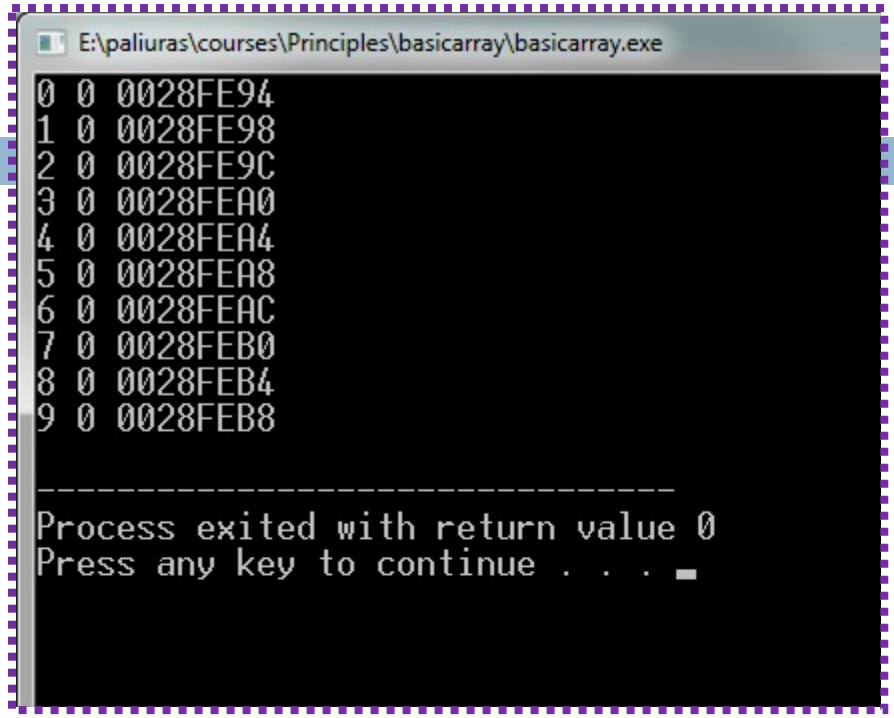
Το όνομα του πίνακα είναι ουσιαστικά απλά μια σταθερά διεύθυνσης στην RAM

```
scores[3] = 10;
printf("%d", values[3]);
```



Πίνακες

```
#include <stdio.h>
#define N 10

int main() {

        int i ;
        int data[N] = {0};

        for (i=0; i< N; i++)
                printf("%d %d %p\n", i, data[i], &data[i]);

        return 0;
}
```



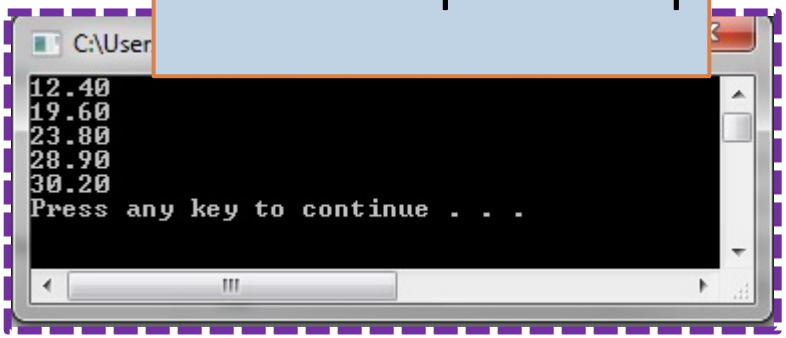
0	1	2	3	4	5	6	7	8	9

Πίνακες και βρόχοι επανάληψης


```c
#include <stdio.h>

void main(void){

    float values[5] = {12.4, 19.6, 23.8, 28.9, 30.2};


    for (int index = 0; index < 5; index++) {
        printf("%.2f\n", values[index]);
    }
}
```

Λίστα αρχικοποίησης όλων των στοιχείων του πίνακα

Ενδεικτική Επίλυση



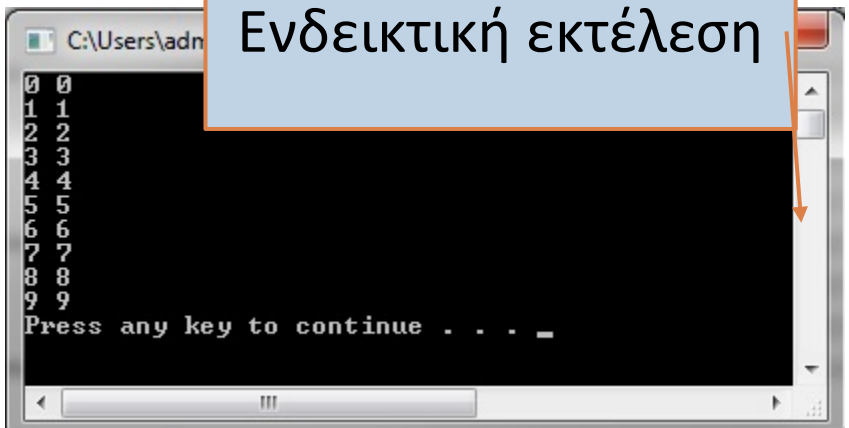
Πίνακες και βρόχοι επανάληψης


```c
#include <stdio.h>

void main(void){

    int values[10];

for (int index = 0; index < 10; index++) {
        values[index] = index;
    }

for (int index = 0; index < 10; index++) {
        printf("%d %d\n", index, values[index]);
    }
}
```

Αρχικοποίηση των τιμών του πίνακα μέσα από ένα for βρόχο

Εκτύπωση των θέσεων και των τιμών των στοιχείων του πίνακα

Ενδεικτική εκτέλεση



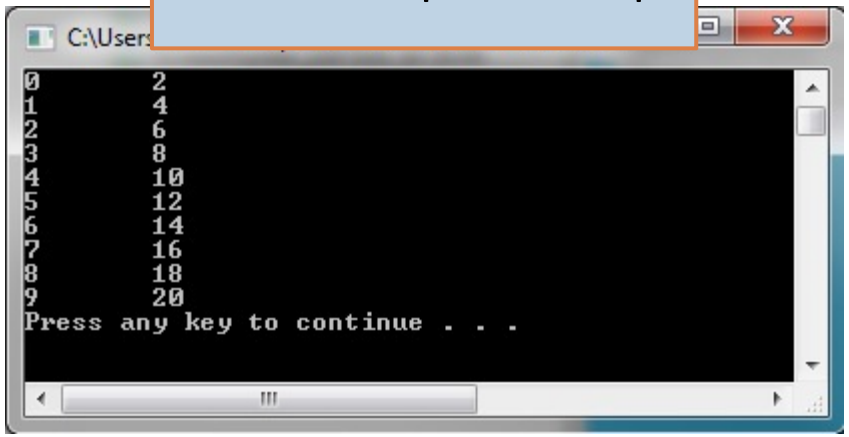
Πίνακες και βρόχοι επανάληψης


```
#include <stdio.h>
#define SIZE 10

void main(void){
     int values[SIZE];


   for (int index = 0; index < SIZE; index++) {
        values[index] = 2 + 2 * index;
    }

   for (int index = 0; index < SIZE; index++) {
        printf("%d\t%d\n", index, values[index]);
    }
}
```

Η οδηγία #define χρησιμοποιείται για να ορίσουμε την τιμή της ΣΤΑΘΕΡΑΣ SIZE σε 10 κατά το στάδιο της προεπεξεργασίας πριν την μεταγλώττιση.

Η τιμή του SIZE αντικαθιστάται με 10 από τον μεταγλωττιστή. Συνεπώς, ο πίνακας values έχει 10 στοιχεία

Ενδεικτική Εκτέλεση



Πίνακες

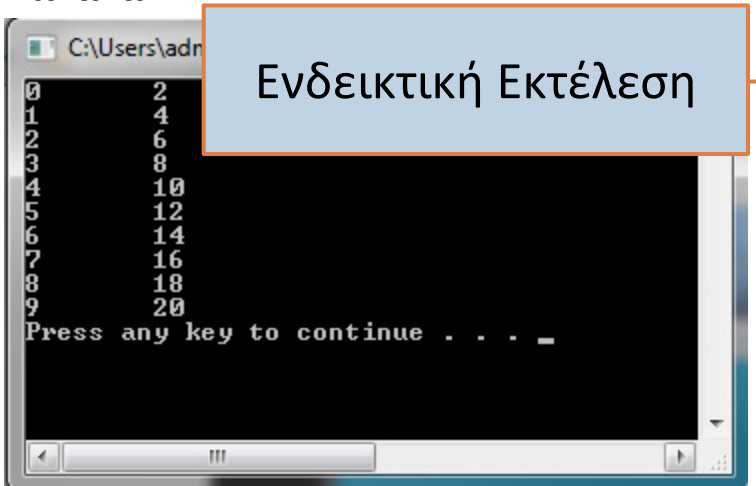

```c
#include <stdio.h>
#define SIZE 10

void main(void){

    int array1[SIZE];
    int array2[SIZE];

   for (int index = 0; index < SIZE; index++) {
        array1[index] = 2 + 2 * index;
    }

    for (int index = 0; index < SIZE; index++) {
        array2[index] = array1[index];
        printf("%d\t%d\n", i, array2[index]);
    }
}
```

Σε περίπτωση που θέλουμε να αντιγράψουμε ένα πίνακα σε ένα άλλο θα πρέπει να μεταφέρουμε τα στοιχεία ένα προς ένα!

Οι πίνακες ΔΕΝ αντιγράφονται σαν ένα στοιχείο αφού το όνομα του πίνακα είναι ουσιαστικά απλά μια σταθερά διεύθυνσης

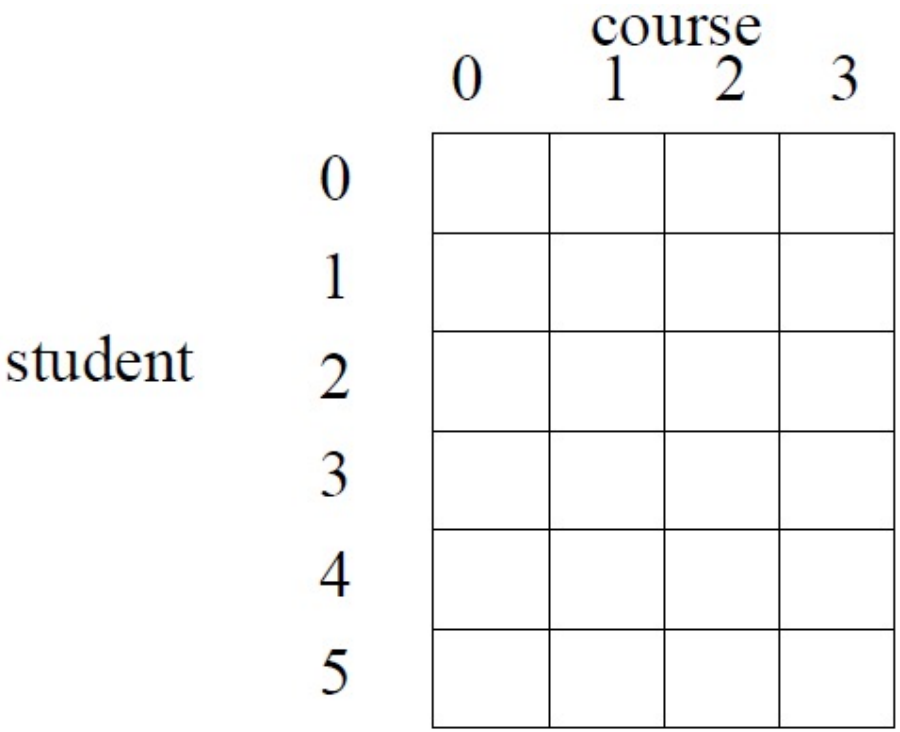
Για να το πετύχουμε αυτό χρησιμοποιούμε ένα for βρόχο που θα διαβάζει ένα στοιχείο σε κάθε επανάληψη και θα το μεταφέρει στο νέο πίνακα

Ενδεικτική Εκτέλεση



Πολυδιάστατοι Πίνακες (Multidimensional arrays)

- Πίνακας με δύο ή περισσότερες διαστάσεις
  - π.χ. **float student_grades**[6][4];



Δυσδιάστατος πίνακας
NUM_STUDENTS γραμμές
NUM_COURSES στήλες

Σημασιολογία

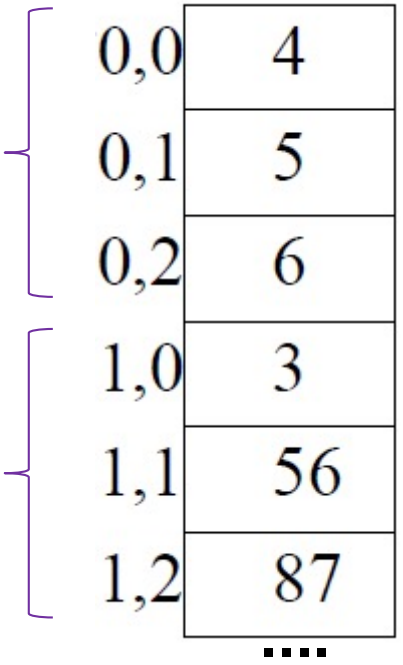
- Ένας άλλος τρόπος να δούμε έναν δυσδιάστατο πίνακα:
  - **Ένας πίνακας του οποίου κάθε στοιχείο είναι πίνακας**

- Με αυτό τον τρόπο εξηγούνται πιο εύκολα διάφορα χαρακτηριστικά των πολυδιάστατων πινάκων:
  - Ο τρόπος αρχικοποίησης στην δήλωση
  - Ο τρόπος παράθεσης στην μνήμη
  - κ.α.

Αποθήκευση Πολυδιάστατων Πινάκων

- Τα δεδομένα ενός πολυδιάστατου πίνακα αποθηκεύονται στη μνήμη με *προτεραιότητα σειράς: 1η γραμμή, 2η γραμμή, 3η γραμμή, κτλ.*

```
int count[4][3]=
   { {4,5,6},
     {3,56,87},
     {456,412,846},
     {0,0,0} };
```



Πίνακες δύο (ή περισσότερων) διαστάσεων

- `int a[3][3] ;`
- `int a[3][3] = {{1,2,3}, {3,2,1}, {1,1,1}};`

1	2	3
3	2	1
1	1	1

Επεξεργασία Πολυδιάστατων Πινάκων

- Αρχικοποίηση
  - Στον ορισμό
    - int student_grades[4][3]={{4, 5, 6}, {7, 7, 9}, {5, 8, 7}, {6, 7, 8}};
      - Πινάκας ο οποίος περιέχει τους βαθμούς **τεσσάρων φοιτητών (οι τέσσερις πίνακες που περιέχονται)** στα **τρία μαθήματα (τα στοιχεία που περιέχονται σε κάθε πίνακα)** που είχαν πάρει αυτό το εξάμηνο.
    - Πίνακας **4 (γραμμές)** x **3(στήλες)**

```c
#include <stdio.h>
#define N 3

int main ( ) {
int i, j;

int a[N][N] = {{1,2,3}, {3,2,1}, {1,1,1}};

for (i=0; i<N; i++) {
    for (j=0; j<N; j++) {
          printf("%d ",a[i][j]);
          }
    printf("\n");
}

return 0;
}
```

Κλασσικά Παραδείγματα

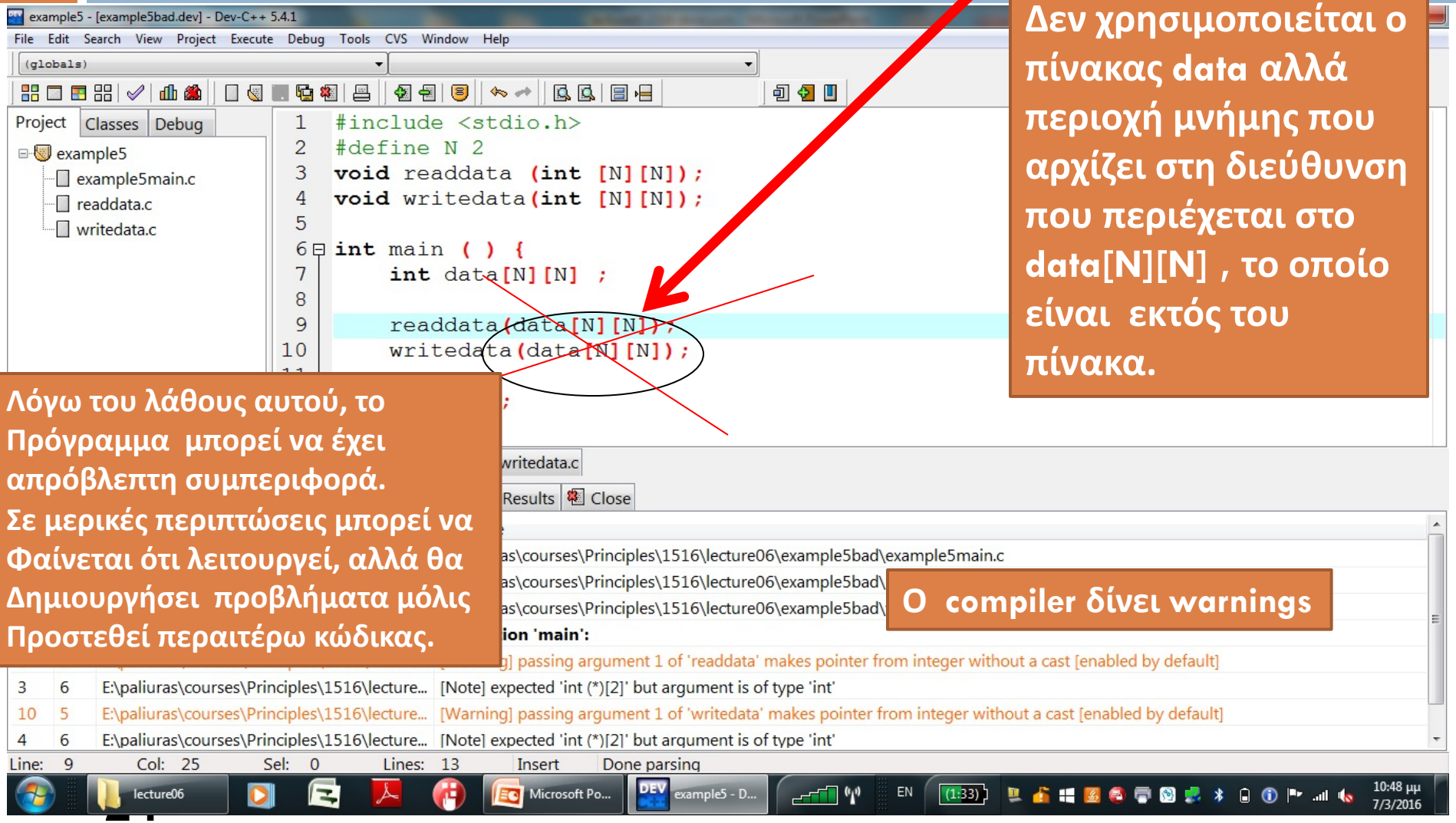
- Πρόσθεση πινάκων n x m
  - Πρόσθεση *αντίστοιχων* στοιχείων
- Πολλαπλασιασμός πινάκων *n x k, k x m*
  - Πράξεις μεταξύ γραμμών του πρώτου και στήλες του δεύτερου

```c
#define N 2
#include <stdio.h>
int main ( ) {
   int data[N][N] ;
   int i, j ;

   for (i =0 ; i < N ; i++)
    for ( j = 0 ; j < N ; j ++) {
      printf ("element (%d,%d)?\t", i, j);
      scanf("%d", &data[i][j]);
      }

   for (i =0 ; i < N ; i++) {
    for ( j = 0 ; j < N ; j ++) {
        printf ("%d\t", data[i][j]);
        }
     printf("\n");
   }
 return 0;
}
```

```c
#include <stdio.h>
#define N 2
void readdata(int [N][N]);
void writedata(int [N][N]);

int main ( ) {
   int data[N][N] ;

   readdata(data) ;
   writedata(data);

   return 0;
}
```

```c
void readdata(int a[N][N]) {
int i,j;
for (i =0 ; i < N ; i++)
   for ( j = 0 ; j < N ; j ++) {
     printf ("element (%d,%d)?\t", i, j);
     scanf("%d", &a[i][j]);
     }
}

void writedata(int b[N][N]) {
int i,j;
for (i =0 ; i < N ; i++) {
   for ( j = 0 ; j < N ; j ++)
       printf ("%d\t", b[i][j]);
   printf("\n");
 }
}
```

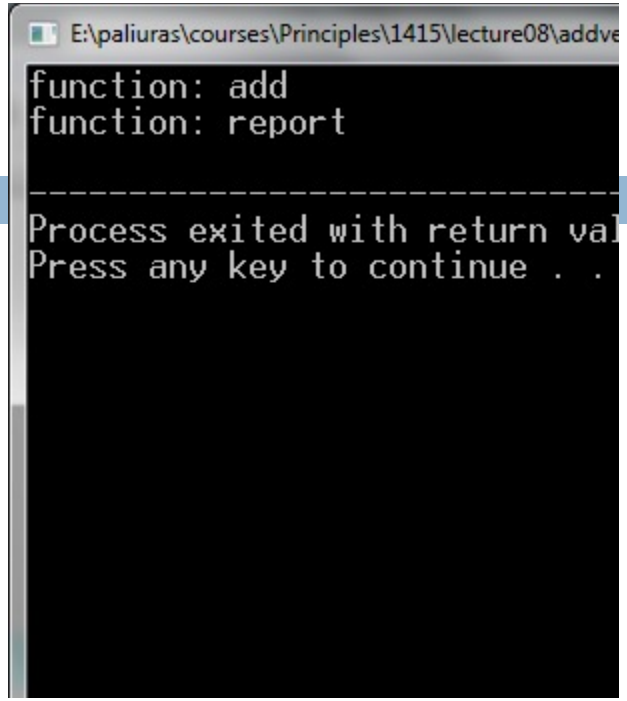
ΠΡΟΣΟΧΗ


example5 - [example5bad.dev] - Dev-C++ 5.4.1
File Edit Search View Project Execute Debug Tools CVS Window Help
Project Classes Debug
example5
example5main.c
readdata.c
writedata.c
#include <stdio.h>
#define N 2
void readdata (int [N][N]);
void writedata(int [N][N]);
int main ( ) {
int data[N][N] ;
readdata(data[N][N]);
writedata(data[N][N]);
Δεν χρησιμοποιείται ο πίνακας data αλλά περιοχή μνήμης που αρχίζει στη διεύθυνση που περιέχεται στο data[N][N] , το οποίο είναι εκτός του πίνακα.
Λόγω του λάθους αυτού, το Πρόγραμμα μπορεί να έχει απρόβλεπτη συμπεριφορά.
Σε μερικές περιπτώσεις μπορεί να Φαίνεται ότι λειτουργεί, αλλά θα Δημιουργήσει προβλήματα μόλις Προστεθεί περαιτέρω κώδικας.
writedata.c
Results
Close
Ο compiler δίνει warnings
[Note] expected 'int (*)[2]' but argument is of type 'int'
[Warning] passing argument 1 of 'writedata' makes pointer from integer without a cast [enabled by default]
[Note] expected 'int (*)[2]' but argument is of type 'int'
Line: 9 Col: 25 Sel: 0 Lines: 13 Insert Done parsing
lecture06
Microsoft Po...
example5 - D...
EN
10:48 μμ
7/3/2016

```c
#include <stdio.h>
#define N 2
void readdata(int a[N][N]);
void writedata(int b[N][N]);
int sumdata(int x[N][N]);

int main ( ) {
   int data[N][N] ;

   readdata(data) ;
   writedata(data);

   printf("The sum is: %d\n",
   sumdata(data));

return 0;
}
```

```c
int sumdata(const int x[N][N])
   {
   int i, j;
   int sum = 0;

   for (i=0; i<N; i++)
        for (j=0; j<N; j++)
                sum += x[i][j];

return sum;
}
```

```c
#include <stdio.h>
#define N 5
void add (const int [], const int [], int []);
void report (const int []);

int main() {
        int a[N] = {1, 2, 3, 4, 5};
        int b[N] = {6, 7, 8, 9, 0};
        int c[N];

        add(a, b, c);
        report (c);

        return 0;
}
```



- Πλήρης main
- Κενές add, report

```c
void add(const int a[N], const int b[N], int c[N] ) {
        printf("add vectors\n");
        return ;
}

void report (const int c[N]) {
        printf("report\n");
        return ;
}
```

```
void add(const int a[N], const int b[N], int c[N] )
{
        int i;

        for (i=0; i<N; i++)
              c[i] = b[i] + a[i];

        return ;
}

void report (const int c[N]) {
        int i;

        for (i=0; i<N; i++)
              printf("%d ", c[i]);
        printf("\n");
        return ;
}
```

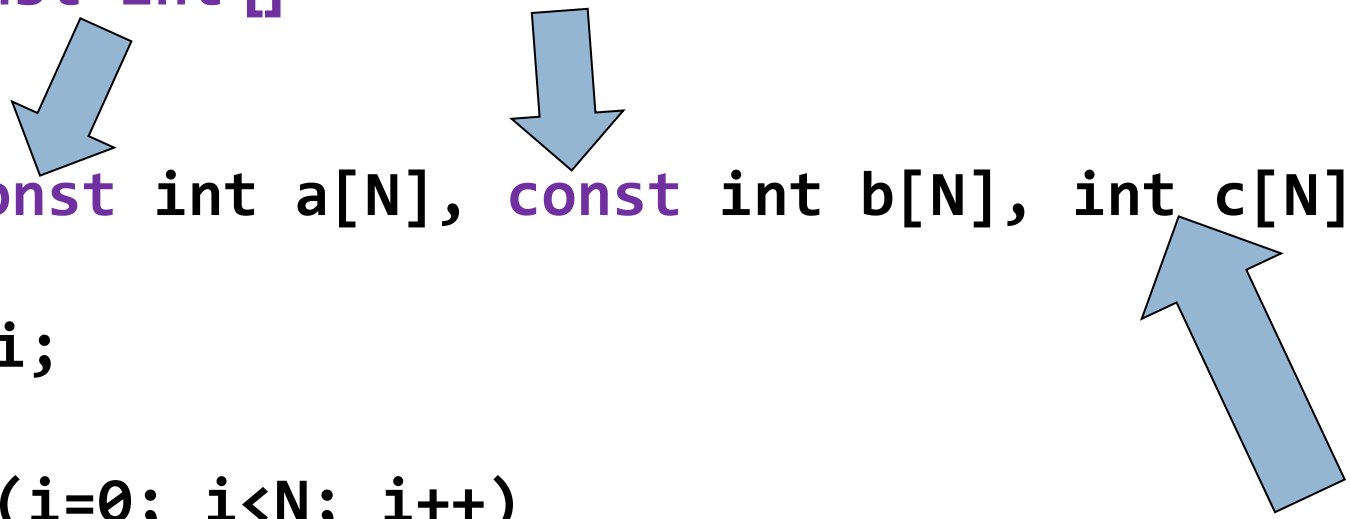
Πλήρης υλοποίηση συναρτήσεων add, report

Μια προγραμματιστική τεχνική

- ***Εξασφαλίζουμε*** ότι μια συνάρτηση μπορεί να αλλάξει τιμές πίνακα ***μόνο αν*** αναλυτικά το επιτρέψουμε.
- Εφαρμογή της αρχής ***ελαχίστου δικαιώματος*** *(principle of least privilege).*
- Χρήση τύπου **`const int [ ]`**

Πίνακες ως είσοδοι και έξοδοι

***Είσοδοι*: δεν επιτρέπεται στη συνάρτηση να αλλάξει τις τιμές πινάκων const int []**



```
void add(const int a[N], const int b[N], int c[N] )
{
        int i;

        for (i=0; i<N; i++)
              c[i] = b[i] + a[i];

        return ;
}
```

Έξοδος : **η συνάρτηση έχει δικαίωμα να Αλλάξει τα στοιχεία του πίνακα c**

```
void report (const int c[N]) {
        int i;

        for (i=0; i<N; printf("%d ", c[i++]));

        printf("\n");
        return ;
}
```

- Άλλη υλοποίηση της report:
  - Η τρίτη έκφραση στο (κενό) **for** τυπώνει και αυξάνει το μετρητή με postfix increment

```
int main() {
        int a[N] = {1, 2, 3, 4, 5};
        int b[N] = {6, 7, 8, 9, 0};
        int c[N];

        add(a, b, c);
        report (c);

        return 0;
}
```

Σωστό

ΣΥΝΗΘΗ ΛΑΘΗ

```
c = add(a, b); /* Λάθος: Το c δεν μπορεί να αλλάξει, είναι η
                  διεύθυνση του πρώτου στοιχείου του πίνακα! */

add(a[], b[], c[]); /* Λάθος: Εδώ είναι  syntax error. Μόνο σε
        δήλωση μπορεί να παραληφθεί μια (και μόνο μία) διάσταση (η
        τελευταία). */

add(a[N], b[N], c[N]); /* Λάθος: Η τιμή ενός ακεραίου (έξω από τους
πίνακες) μεταφράζεται σε διεύθυνση!!!  Warning: pointer from integer
without a cast */
```

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/

ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

3η Εβδομάδα: Πίνακες

Βασικός τύπος **char**

```
#include <stdio.h>

int main() {

	char a;

	a = 'g';

	printf("this is a char: %c\n", a);

	return 0;
}
```

Ανάγνωση και εκτύπωση αλφαριθμητικού

- `char str[N_MAX + 1];`
- `scanf ("%s", str );`
- `printf ("%s\n", str );`

- `%s` → αντιστοιχεί σε αλφαριθμητικό
- `str[0]` είναι ο πρώτος χαρακτήρας
- `str` είναι η **διεύθυνση του πρώτου στοιχείου**
  - str είναι το **ίδιο** με &str[0]
  - ισχύει για κάθε τύπο πίνακα

Διαφορετική σημασία απλών / διπλών εισαγωγικών στη C

- Διαφορετικά από python
- Απλά εισαγωγικά → ένας χαρακτήρας

```
char ch;
ch = 'a';
```

- Διπλά εισαγωγικά → ακολουθία χαρακτήρων που τερματίζεται με την αξία 0

```
char name[10] = "myname";
printf("stay safe\n");
printf("hello");
```

- Χαρακτήρας και κώδικας ascii

Αρχικοποίηση string

Ορισμός String

Ένας πίνακας από χαρακτήρες που τελειώνει με τον χαρακτήρα **NULL '\0' ,**

π.χ. |Hello\0|, |Hi there\0|, |\0|

(Προσοχή το \0 δεν φαίνεται στην οθόνη)

Υπάρχουν διάφοροι τρόποι να ορίσουμε ένα String στην C, ανάλογα με το αν γνωρίζουμε το string προτού την μεταγλώττιση ή όχι.

Α) Αρχικοποίηση (γνωρίζοντας εκ'των πρότερων το String)

char msg[ ]="Hello";

Δημιουργεί αυτόματα το:

0	1	2	3	4	5
H	E	L	L	O	\0

Αρχικοποίηση string

Συνίσταται αυτός ο ορισμός
char msg[ ]="Hello"

αλλά υπάρχουν και άλλοι τρόποι…

Σωστό
```
char msg[6];
msg[0] = 'H';
msg[1] = 'e';
msg[2] = 'l';
msg[3] = 'l';
msg[4] = 'o';
msg[5] = '\0';
```

Σωστό
char msg[ ]={'H','e','l','l','o','\0'};

Σωστό
char msg[6]={'H','e','l','l','o','\0'};

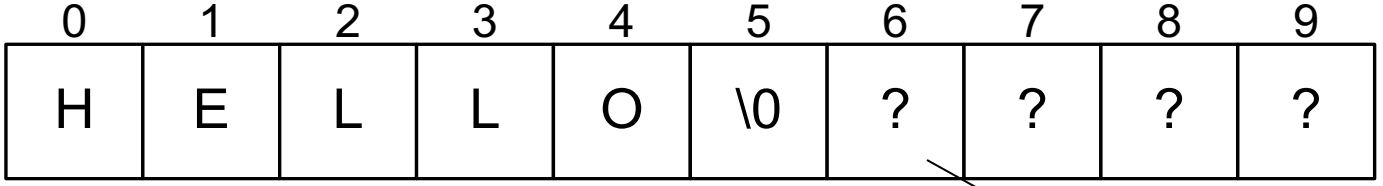
Σωστό αλλά σπάταλο
char msg[40]="Hello"

Αρχικοποίηση string

Ας δούμε πως μοιάζουν εικονικά οι ακόλουθοι ορισμοί

char msg[10]="Hello";

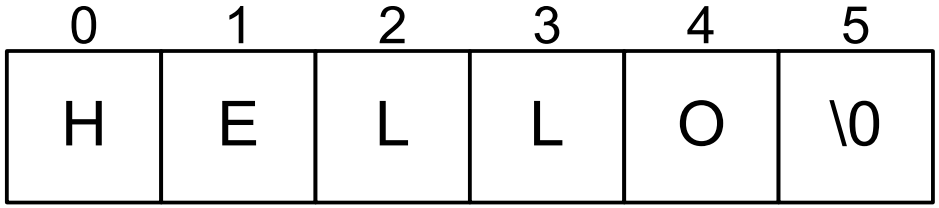
SIZE=10



Το '?' είναι απροσδιόριστος χαρακτήρας

char msg[ ]="Hello";

SIZE=6



Παράδειγμα


```c
#include <stdio.h>
#include <string.h>
int main ( ) {
   char msg[6]={'H','e','l','l','o','\0'};
   int i=0;
   printf("%p\n", msg);
   while ( msg[i] != '\0'){
            printf("%p - %c\n", &msg[i],msg[i]);
            i++;
    }
    printf("%p - %c\n", &msg[i],msg[i]);
}
```

```
0x7ffee7222ad6
0x7ffee7222ad6 - H
0x7ffee7222ad7 - e
0x7ffee7222ad8 - l
0x7ffee7222ad9 - l
0x7ffee7222ada - o
0x7ffee7222adb -
```

Αρχικοποίηση string

Ερώτηση

- Τι γίνεται αν δεν ξέρουμε το String εκ' των πρότερων;
- Πόσο χώρο πρέπει να δεσμεύσουμε;

Απάντηση

- Αρκετό για να χωρέσει διάφορα δεδομένα εισόδου που πρόκειται να εισάγουμε.

  π.χ. αν πρόκειται να αποθηκεύσουμε κάποιο **όνομα** τότε 20 χαρακτήρες φαίνεται να αρκούν.

- Στην πραγματικότητα χρησιμοποιούμε **δυναμική δέσμευση μνήμης**, η οποία μας επιτρέπει να δεσμεύσουμε τον απαιτούμενο χώρο κατά την διάρκεια εκτέλεσης του προγράμματος (θα μιλήσουμε για αυτό το θέμα στην συνέχεια του μαθήματος)!

Προτού δούμε ένα παράδειγμα, ας δούμε πως διαβάζουμε strings από τον χρήστη και πως τα εκτυπώνουμε....

Ανάγνωση/Εκτύπωση String

scanf (“%s”, name)

ΠΡΟΣΟΧΗ: Δεν χρησιμοποιείτε το &, γιατί το name είναι πίνακας.

Θυμηθείτε ότι σε άλλους τύπους δεδομένων χρησιμοποιείται το & π.χ. **scanf(“%d”, &a);**

- Για εισαγωγή συμβολοσειράς με κενά χρησιμοποιείται η
  - #define MAX 50

    scanf("%" MAX "[^\n]", name);

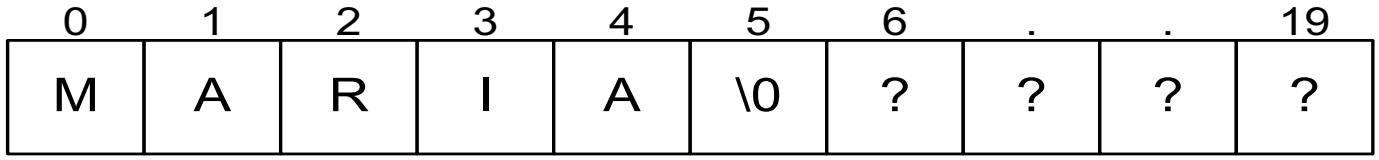
printf (“%s”,name)

- Για εκτύπωση συμβολοσειρών

Ανάγνωση/Εκτύπωση string

```
#include <stdio.h>
int main()
{
    char name[20];
    scanf("%s", name);
    printf("%s", name);
    return 0;
}
```

Το printf γνωρίζει ότι η εκτύπωση πρέπει να γίνει μέχρι το \0

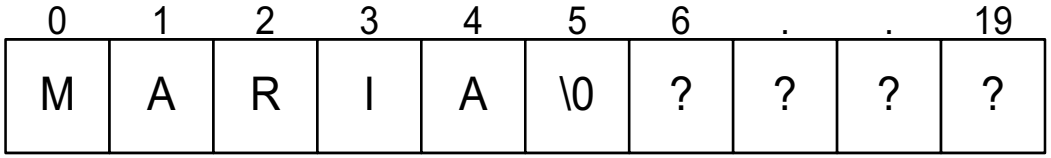


Ανάγνωση/Εκτύπωση string

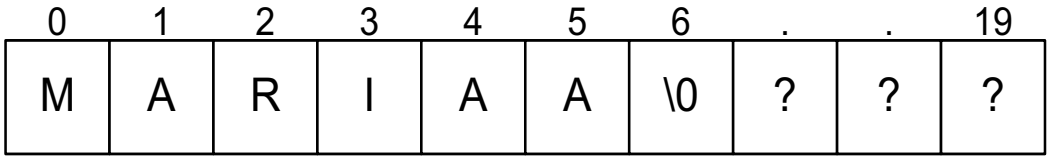
Πρόγραμμα που προσθέτει ένα «Α» στην θέση 5 και τερματίζει το string

```
#include <stdio.h>
int main()
{
        char name[20];
        scanf("%s", name);
        name[5]='A';
        name[6]='\0';
        printf("%s", name);
        return 0;
}
```

Πρίν



Μετά



Κοινό Λάθος (Ανάθεση σε String)

Υποθέστε ότι έχουμε την πιο κάτω δήλωση

char name [50];

Δεν επιτρέπετε να αναθέσουμε ένα string από ένα άλλο (όπως στους υπόλοιπους τύπους)

Π.χ.

- name=“hydrogen” **ΛΑΘΟΣ (compile error)**
- name={‘h’,’y’,’d’,’r’,’o’,’\0’} **ΛΑΘΟΣ (compile error)**
- **name[0]=“h”; name[1]=“y”; …. ΟΡΘΟ**

Όμως, υπάρχει ευκολότερος τρόπος τον οποίο θα δούμε σε επόμενες διαλέξεις (με χρήση strcpy)

Άσκηση Εξάσκησης

- Δηλώστε έναν μονοδιάστατο πίνακα **x** ακεραίων αριθμών (**N** θέσεων) και αρχικοποιήστε τον με 0. Στη συνέχεια να γράψετε μια συνάρτηση **initialize** η οποία να αρχικοποιεί κάθε στοιχείο **i** του μονοδιάστατου πίνακα ακεραίων αριθμών με την τιμή $\mathbf{i^2}$ **.** Στη συνέχεια να γράψετε μια συνάρτηση **print** η οποία να εκτυπώνει τα στοιχεία του πίνακα.

Άσκηση Εξάσκησης


```c
#include <stdio.h>
#include <stdlib.h>
#define N 7
void initialize(int x[N]);
void print(const int x[N]);

int main()
{
    int x[N]={};
    initialize(x);
    print(x);
    return 0;
}

void initialize(int x[N]){
   int i;
   for(i=0;i<N;i++){
       x[i]=i*i;
   }
   return;
}

void print(const int x[N]){
   int i;
   for(i=0;i<N;i++){
       printf("\n %d cell, memory address %p value %d ",i,&x[i],x[i]) ;
   }
   return;
}
```

```
0 cell, memory address 0x7ffee0dc69b0 value 0
1 cell, memory address 0x7ffee0dc69b4 value 1
2 cell, memory address 0x7ffee0dc69b8 value 4
3 cell, memory address 0x7ffee0dc69bc value 9
4 cell, memory address 0x7ffee0dc69c0 value 16
5 cell, memory address 0x7ffee0dc69c4 value 25
6 cell, memory address 0x7ffee0dc69c8 value 36
7 cell, memory address 0x7ffee0dc69cc value 49
8 cell, memory address 0x7ffee0dc69d0 value 64
9 cell, memory address 0x7ffee0dc69d4 value 81
10 cell, memory address 0x7ffee0dc69d8 value 100
11 cell, memory address 0x7ffee0dc69dc value 121
12 cell, memory address 0x7ffee0dc69e0 value 144
13 cell, memory address 0x7ffee0dc69e4 value 169
14 cell, memory address 0x7ffee0dc69e8 value 196
15 cell, memory address 0x7ffee0dc69ec value 225
16 cell, memory address 0x7ffee0dc69f0 value 256
17 cell, memory address 0x7ffee0dc69f4 value 289
18 cell, memory address 0x7ffee0dc69f8 value 324
19 cell, memory address 0x7ffee0dc69fc value 361
20 cell, memory address 0x7ffee0dc6a00 value 400
21 cell, memory address 0x7ffee0dc6a04 value 441
22 cell, memory address 0x7ffee0dc6a08 value 484
23 cell, memory address 0x7ffee0dc6a0c value 529
24 cell, memory address 0x7ffee0dc6a10 value 576
25 cell, memory address 0x7ffee0dc6a14 value 625
26 cell, memory address 0x7ffee0dc6a18 value 676
27 cell, memory address 0x7ffee0dc6a1c value 729
28 cell, memory address 0x7ffee0dc6a20 value 784
29 cell, memory address 0x7ffee0dc6a24 value 841
30 cell, memory address 0x7ffee0dc6a28 value 900
```

Άσκηση Εξάσκησης

- Υλοποιήστε συνάρτηση **sum_array** που θα υπολογίζει το άθροισμα των στοιχείων του μονοδιάστατου πίνακα και θα επιστρέφει το αποτέλεσμα ως ακέραιο αριθμό. Επίσης, υλοποιήστε συνάρτηση **average_array** που θα υπολογίζει τον μέσο όρο των στοιχείων του μονοδιάστατου πίνακα και θα επιστρέφει το αποτέλεσμα ως πραγματικό αριθμό.

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5
void initialize(int x[N]);
int sum_array (const int x[N]);
float average_array(const int x[N]);

int main()
{
    int x[N]={};


    initialize(x);
    printf("\n avg value %.3f ", average_array(x)) ;
    return 0;
}

void initialize(int x[N]){
   int i;
   for(i=0;i<N;i++){
       x[i]=i*i;
   }
   return;
}

float average_array(const int x[N]){
   return (float)sum_array (x)/N;
}

int sum_array (const int x[]){
   int i, sum;
   sum=0;
   for(i=0;i<N;i++){
      sum=sum+x[i];
   }
   return sum;
}
```

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/