



Παράλληλος Προγραμματισμός 2018

Προγραμματιστική Εργασία #1

Σεβαστοπούλου Ελένη

A.M.: Π2013128

α)

Προσπέλαση όλων των στοιχείων ενός πίνακα δύο διαστάσεων :

ΚΑΤΑ ΓΡΑΜΜΗ

```
7
8 #define NCOLS 100 // παραμένει σταθερό
9 #define NROWS 100000
10 void get_walltime(double *wct) {
11     struct timeval tp;
12     gettimeofday(&tp, NULL);
13     *wct = (double)(tp.tv_sec+tp.tv_usec/1000000.0);
14 }
15
16
17 int main(int argc, char *argv[]) {
18     int i,j;
19     double sum = 0.0;
20     double *table;
21     double ts, te, Maccess;
22
23
24     table = (double *)malloc(NCOLS*NROWS*sizeof(double)); // δυναμική δεσμευση μνήμης
25     if (table==NULL) {
26         printf("alloc error!\n");
27         exit(1);
28     }
29
30
31
32     for(i=0; i<NCOLS*NROWS; i++){
33         table[i]=1.0;
34     }
35
36     get_walltime(&ts);
37
38     // προσπέλαση κατά γραμμή
39
40
41     for (i=0; i<NROWS; i++){
42         for(j=0; j<NCOLS; j++){
43             sum+=table[j*NCOLS+i];
44         }
45     }
46     get_walltime(&te);
47
48     printf("Accesses = %f\n", sum);
49
50     //To Maccess είναι τα Memory_accesses per second
51     Maccess=((double)NROWS*NCOLS)/((te-ts)*1e6);
52
53     printf("Maccesses/sec = %f\n", Maccess);
54
55     free(table);
56
57     return 0;
58 }
```

Εικόνα (1). Προσπέλαση κατά γραμμή

ΚΑΤΑ ΣΤΗΛΗ

```
[*] Untitled1.cpp  Untitled1.cpp
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/time.h>
4
5  #define NCOLS 100
6  #define NROWS 100000
7  void get_walltime(double *wct) {
8      struct timeval tp;
9      gettimeofday(&tp, NULL);
10     *wct = (double)(tp.tv_sec+tp.tv_usec/1000000.0);
11 }
12
13 int main(int argc, char *argv[]) {
14     int i,j;
15     double sum = 0.0;
16     double *table;
17     double ts, te, Maccess;
18
19     table = (double *)malloc(NCOLS*NROWS*sizeof(double));
20     if (table==NULL) {
21         printf("alloc error!\n");
22         exit(1);
23     }
24
25     for(i=0; i<NCOLS*NROWS; i++){
26         table[i]=1.0;
27     }
28
29     get_walltime(&ts);
30
31     // προσπέλαση κατά στήλη
32
33     for (i=0; i< NCOLS; i++){
34         for(j=0; j< NROWS; j++){
35             sum+=table[j*NCOLS+i];
36         }
37     }
38     get_walltime(&te);
39
40     printf("Accesses = %f\n", sum);
41
42     //To Maccess είναι τα Memory_accesses per second
43
44     Maccess=((double)NROWS*NCOLS)/((te-ts)*1e6);
45
46     printf("Maccesses/sec = %f\n", Maccess);
47
48     free(table);
49
50     return 0;
51 }
52 }
```

Εικόνα (2). Προσπέλαση κατά στήλη

Πίνακας 1

	ΓΡΑΜΜΕΣ	ΣΤΗΛΕΣ	ΓΡΑΜΜΗ ΑΝΑ ΓΡΑΜΜΗ	ΣΤΗΛΗ ΑΝΑ ΣΤΗΛΗ
MEMORY ACCESS	100	100	1.#INF00	1.#INF00
	200	100	1.290.396	19.958.620
	1000	100	100.007.248	1.#INF00
	10.000	100	63.815.960	63.837.329
	100.000	100	146.139.432	63.955.402

Πίνακας 2

	ΓΡΑΜΜΕΣ	ΣΤΗΛΕΣ	ΓΡΑΜΜΗ ΑΝΑ ΓΡΑΜΜΗ	ΣΤΗΛΗ ΑΝΑ ΣΤΗΛΗ
TIME	100	100	0.1047	0.04543
	200	100	0.04838	0.03267
	1000	100	0.06382	0.04843
	10.000	100	0.08372	0.08384
	100.000	100	0.1503	0.2998

Προσέλαση κατά γραμμή δυσδιάστατου πίνακα 100.000 γραμμών

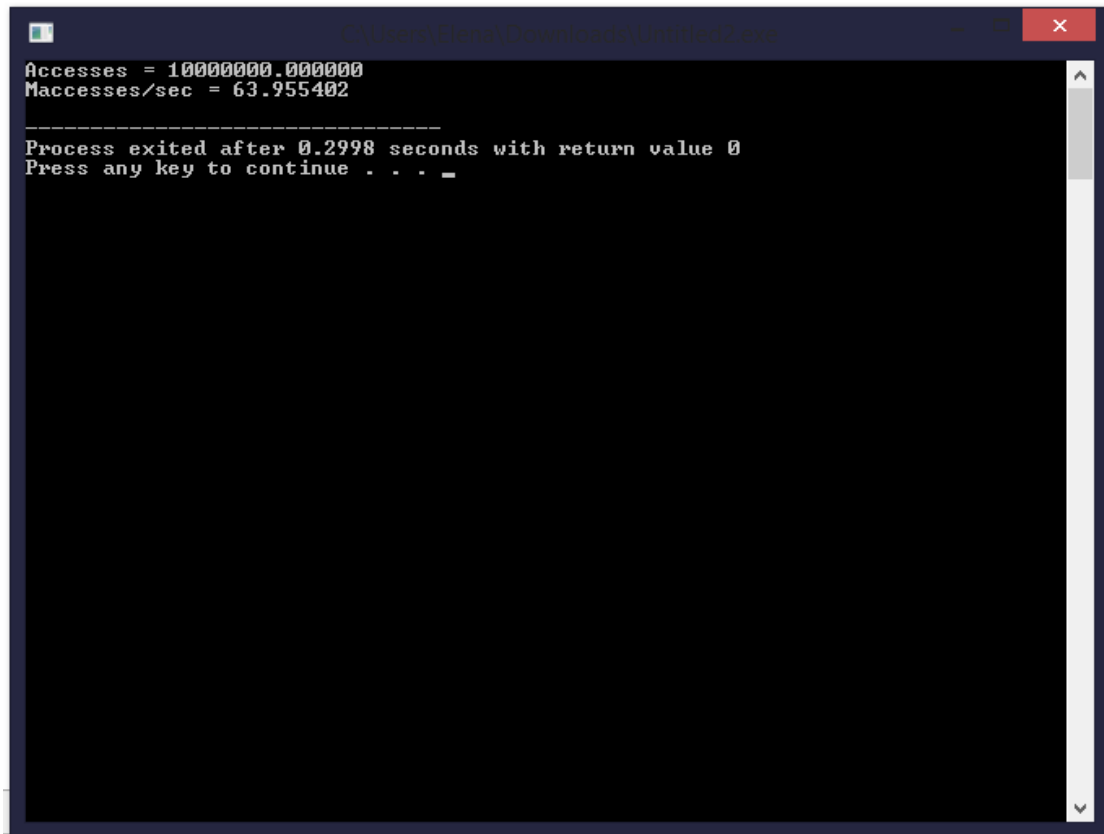
```

C:\Users\Elena\Downloads\Πρόγραμμα Πίνακα\ΠΡΟΓΡΑΜΜΑ ΠΙΝΑΚΑ\MATRIX 1\Untitled1...
Accesses = 100000000.000000
Maccesses/sec = 146.139432

-----
Process exited after 0.1503 seconds with return value 0
Press any key to continue . . . _
  
```

Εικόνα (3)

Προσπέλαση κατά στήλη δυσδιάστατου πίνακα 100.000 γραμμών



```
C:\Users\Elena\Downloads\Untitled2.exe
Accesses = 10000000.000000
Maccesses/sec = 63.955402

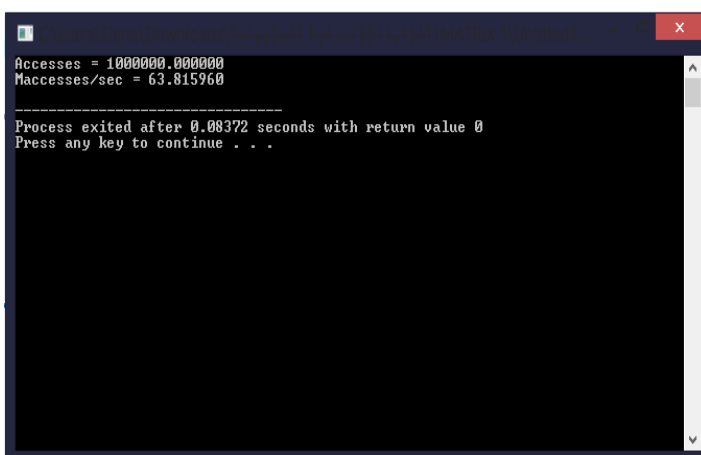
-----
Process exited after 0.2998 seconds with return value 0
Press any key to continue . . . _
```

Εικόνα (4)

ΠΡΟΣΠΕΛΑΣΗ ΚΑΤΑ ΓΡΑΜΜΗ :

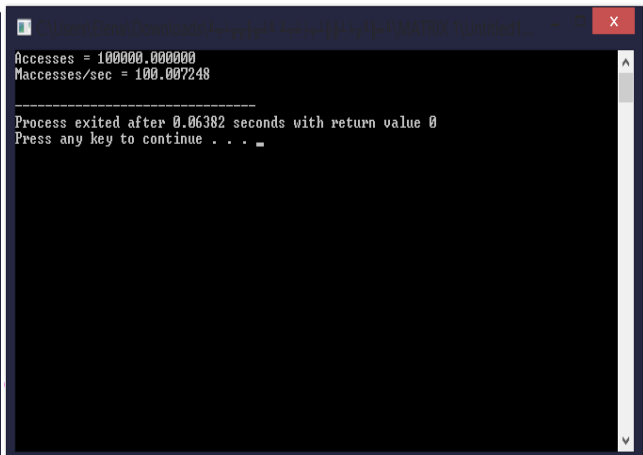
10.000 γραμμών

1000 γραμμών



```
C:\Users\Elena\Downloads\Untitled2.exe
Accesses = 1000000.000000
Maccesses/sec = 63.815960

-----
Process exited after 0.08372 seconds with return value 0
Press any key to continue . . . _
```



```
C:\Users\Elena\Downloads\Untitled2.exe
Accesses = 100000.000000
Maccesses/sec = 100.007248

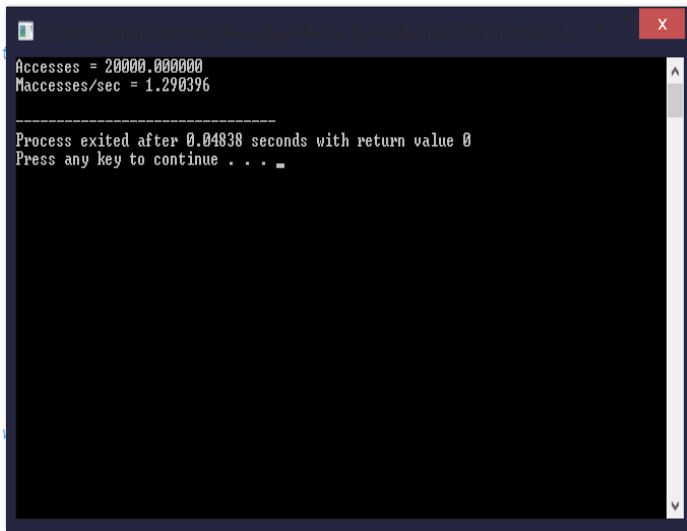
-----
Process exited after 0.06382 seconds with return value 0
Press any key to continue . . . _
```

Εικόνα (5)

Εικόνα (6)

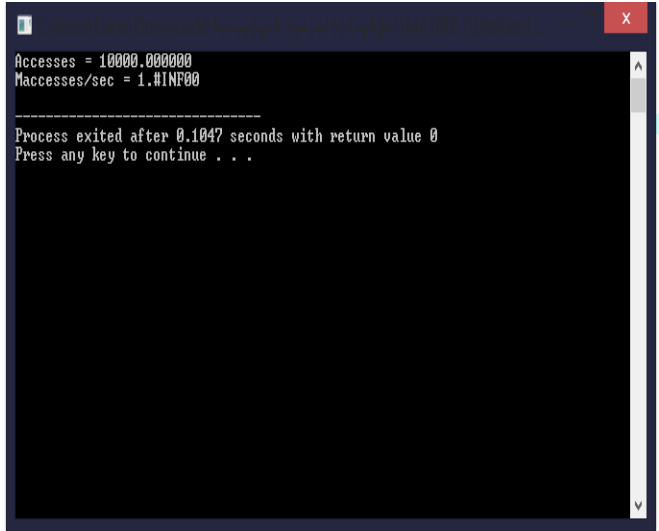
200 γραμμών

100 γραμμών



```
Accesses = 20000.000000
Maccesses/sec = 1.290396

-----
Process exited after 0.04638 seconds with return value 0
Press any key to continue . . . _
```



```
Accesses = 10000.000000
Maccesses/sec = 1.#INF00

-----
Process exited after 0.1047 seconds with return value 0
Press any key to continue . . .
```

Εικόνα (7)

Εικόνα(8)

ΠΡΟΣΠΕΛΑΣΗ ΚΑΤΑ ΣΤΗΛΗ

10.000 γραμμών

1000 γραμμών

```
Accesses = 1000000.000000
Maccesses/sec = 63.815960

-----
Process exited after 0.08372 seconds with return value 0
Press any key to continue . . .
```

Εικόνα(9)

```
Accesses = 100000.000000
Maccesses/sec = 100.007248

-----
Process exited after 0.06382 seconds with return value 0
Press any key to continue . . .
```

Εικόνα(10)

200 γραμμών

100 γραμμών

```
Accesses = 20000.000000
Maccesses/sec = 19.958620

-----
Process exited after 0.03267 seconds with return value 0
Press any key to continue . . .
```

Εικόνα(11)

```
Accesses = 10000.000000
Maccesses/sec = 1.41NF00

-----
Process exited after 0.04543 seconds with return value 0
Press any key to continue . . .
```

Εικόνα (12)

Με βάση τα αποτελέσματα των εικόνων παρατηρούμε ότι:

- α) για τον ίδιο αριθμό γραμμών γίνονται περισσότερες προσπελάσεις ανά δευτερόλεπτο με την προσπέλαση κατά γραμμή,
- β) στην προσπέλαση κατά γραμμή, η διαφορά των προσπελάσεων ανά δευτερόλεπτο για διαφορετικό αριθμό

γραμμών αρχικά δεν φαίνεται να είναι μεγάλη, μετά όμως τις 10.000 γραμμές απαιτούνται αρκετές περισσότερες προσπελάσεις,

γ) στην προσπέλαση κατά στήλη παρατηρούμε, πως όσο αυξάνεται ο αριθμός των γραμμών μειώνονται και οι προσπελάσεις ανά δευτερόλεπτο. Έτσι, συμπεραίνουμε πως οι δυο τρόποι προσπέλασης έχουν διαφορά στην απόδοσή τους.

β)

Το πρόγραμμα υλοποιήθηκε και εκτελέστηκε σε λειτουργικό Windows 8.1 Pro 64-bit με

επεξεργαστή AMD Athlon(tm)X4 860K Quad Core Processor 3.7GHz.

Στην C, οι πίνακες αποθηκεύονται κατά γραμμή. Συνεπώς αν προσπελάσουμε ένα στοιχείο $a[i][j]$, τότε η προσπέλαση του $a[i][j+1]$ πιθανόν να βρεθεί στην κρυφή μνήμη. Δεν θα γίνει πρόσβαση στην κύρια μνήμη. Η κρυφή μνήμη είναι γρηγορότερη από την κύρια συνεπώς ο τρόπος προσπέλασης έχει σημασία.