



**2017 - 2018**

**PROGRAMLAMA Z |||**

**PROJE 1 RAPORU**

**Moaaz Ahmed**

**1421221039**

istenilen :-

5000 X 5000 ‘lik A ve B matrisleri çarpılarak yeni 5000 x 5000 lik bir C matrisi oluşturulacaktır. Ve çarpılırken indisleri her türlü değiştirerek süresi oluşturup bir biriyle kıyaslanması yapılacaktır. En kısa süre belirletip açıklanması yapılacaktır.

Çalışma prensibi :-

İlk adımda malloc() kullanarak 2 boyutlu A,B ve C diye 3 dizi oluşturdum. Oluşturduğum 3 diziye random() ile rastgele double sayılar üretip A ve B matrisleri çarparak C dizisi kurdum. Çarpım metot süresi clock metodu çağırarak çarpma işlemi hesapladım. Hesapladığım süreleri çarpım metodu indisleri değiştirerek bir biriyle kıyasladım. Kıyasları da ölçüm kıyaslamaları kısmında açıklanmıştır.

Kod :-

/\*

============================================================================

Name : BP3\_Proje1.c

Author : Moaaz Ahmed

Version :

Copyright : BP3\_Proje1

Description : Hello World in C, Ansi-style

============================================================================

\*/

**#include** <Windows.h>

**#include** <stdio.h>

**#include** <stdlib.h>

**#include** <time.h>

**#include** "myHeader.h"

// this is to generate a random double number

**double** **random**() {

**struct** timespec ts;

//TIMEVAL t1;

//gettimeofday(&t1,NULL);

**srand**((time\_t)ts.tv\_nsec);

**srand**((**time**(NULL )\*((**double**) **rand**() / RAND\_MAX \* 5.0)));

ns();

**return** (**double**) **rand**() / RAND\_MAX \* 5.0;

}

// this is to print any matrix

**void** **printMatris**(**double** satir, **double** sutun, **double** \*\*matris) {

**int** i, j;

**for** (i = 0; i < satir; ++i) {

**for** (j = 0; j < sutun; ++j) {

**double** a=\*(\*(matris+j)+i);

**printf**(" %lf\t\t",a );

}

**puts**("");

**puts**("");

}

}

// this is to set the heap space free

**void** **free\_matrix**(**int** rows, **double** \*\*mat){

**int** i=0;

**for**(i=0;i<rows;i++)

**free**(mat[i]);

**free**(mat);

}

**int** **main**(**void**) {

**int** satir=5000;

**int** sutun=5000;

**double** \*\*A = (**double** \*\*)**malloc**(satir \* **sizeof**(**double**\*));

**double** \*\*B = (**double** \*\*)**malloc**(satir \* **sizeof**(**double**\*));

**double** \*\*C = (**double** \*\*)**malloc**(satir \* **sizeof**(**double**\*));

**for**(**int** i = 0; i < satir; i++){

//\*(A+i)= (double \*)malloc(satir \* sizeof(double));

A[i] = (**double** \*)**malloc**(sutun \* **sizeof**(**double**));

//\*(B+i)= (double \*)malloc(satir \* sizeof(double));

B[i] = (**double** \*)**malloc**(sutun \* **sizeof**(**double**));

//\*(C+i)= (double \*)malloc(satir \* sizeof(double));

C[i] = (**double** \*)**malloc**(sutun \* **sizeof**(**double**));

}

**puts**("Memory Allocation OK");

// this is to setting the initial values of matricies

**int** y,u;

**for** (y = 0; y < satir; ++y) {

**for** (u = 0; u < sutun; ++u) {

A[y][u] =random();

B[y][u] =random();

C[y][u] =0.0;

}

}

clock\_t start = **clock**();

// this is to multiply two matrices

**int** i, j, k;

**for** (k = 0; k < satir; ++k) {

**for** (j = 0; j < sutun; ++j) {

**for** (i = 0; i < sutun; ++i) {

C[k][i]+=A[j][i]\*B[k][j];

//\*(\*(C+k)+i) +=\*(\*(A+j)+i) \* \*(\*(B+k)+j);

}

}

}

clock\_t end = **clock**();

**double** seconds = (**double**)(end - start) / CLOCKS\_PER\_SEC;

**printf**("\n For k,j,i multiplication you spent %lf seconds !\n",seconds);

// this is to print any matrix

//puts("-------this is A matrix-------");

//printMatris(satir, sutun, A);

//puts("------this is B matrix--------");

//printMatris(satir, sutun, B);

//puts("------this is C matrix--------");

//printMatris(satir, sutun, C);

// this is setting the heap meomory free

free\_matrix(satir,A);

free\_matrix(satir,B);

free\_matrix(satir,C);

**return** 0;

}

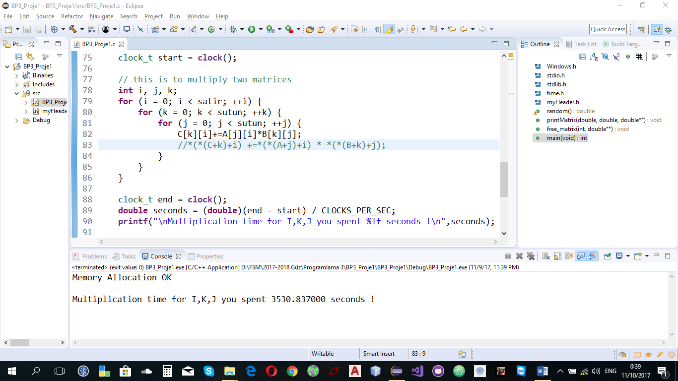
Ekran çıktıları :-

İ j k sırası için :-

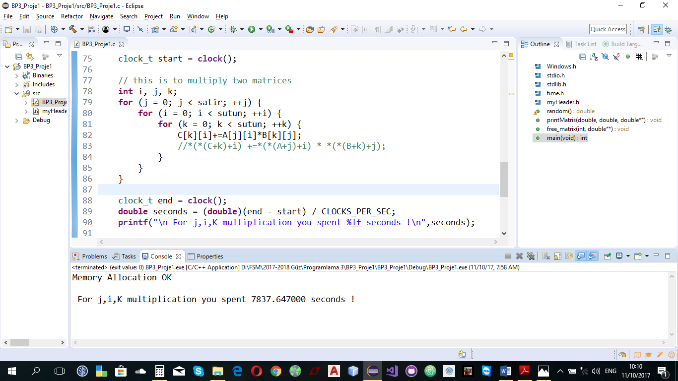
ekran görüntüsü içeren bir resim

Çok yüksek güvenilirlikle oluşturulmuş açıklama

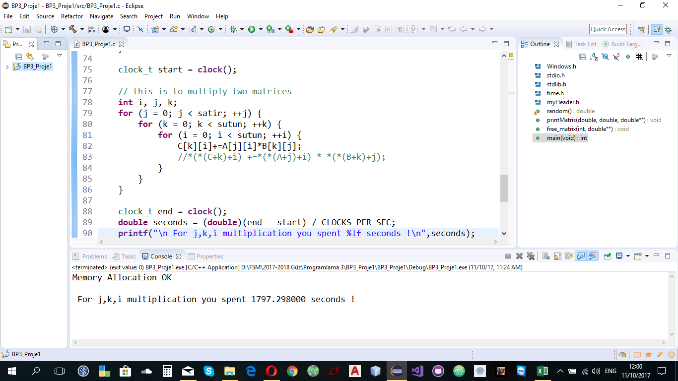
İ k j sırası için :-



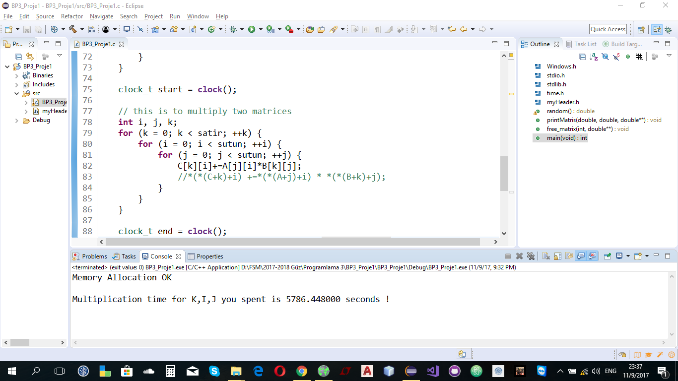
j i k sırası için :-



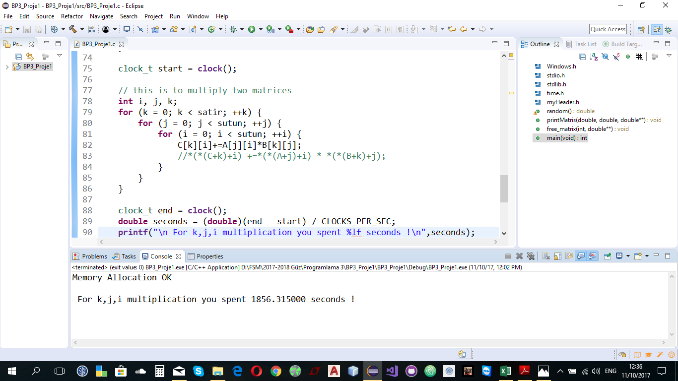
j k i sırası için :-



k i j sırası için :-



k j i sırası için :-



Ölçüm kıyaslamaları :-

C[K][i]+=A[j][i]\*B[k][j]

|  |  |
| --- | --- |
| İndis | Süre |
| İ j k | 7686 saniye |
| İ k j | 3530 saniye |
| J i k | 7837 saniye |
| J k i | 1797 saniye |
| K i j | 5786 saniye |
| K j i | 1856 saniye |

İlk olarak en dıştaki döngü eğer C matrisin indislerinden bir tanesi değil ise (j k i sırasında ) çok fazla sürmediğini görüyoruz. Çünkü çarpım adımlarında önce tüm elemanların çarpma işlemi yapıyor sonra toplam işlemi yapıyor. Fakat En fazla sürdüğü sıra ( j i k ) C matriste her elemanı için çarpma hemen ardında toplama işlemi yapıp ancak böyle bir sonraki elemana geçebiliyor.

Sonuç ve çıkarımlar :-

\*\* Aynı cinsinden olan işlemleri art arda yapmak işlemciye göre daha kolay ve daha hızlıdır.

\*\* Değişik işlemleri çeşitli sıralarda yapmayı işlemciye göre daha yavaş ve maliyetlidir.