



УНИВЕРЗИТЕТ У НОВОМ САДУ ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ИСПИТНИ РАД

Кандидат: Марко Драгићевић
Број индекса: SW74/2019

Предмет: Паралелно програмирање
Тема рада: Множење матрица

Ментор рада: проф. др Мирослав Поповић

Нови Сад, јун, 2021.

SADRŽAJ

1. Uvod.....	1
1.1 Matrice	1
1.2 Međusobno množenje matrica	2
1.3 Zadatak	2
2. Analiza problema.....	3
3. Koncept rešenja.....	4
4. Opis rešenja.....	5
4.1 Sekvencijalan program – moduli i osnovne metode (MatrixMulSerial).....	5
4.1.1 Modul glavnog programa (Main)	5
4.1.2 Modul za obradu programa (SerialTask)	5
4.1.2.1 Metoda za pokretanje množenja (doTask)	5
4.1.3 Modul za rukovanje matricom (MatrixInfo).....	5
4.1.3.1 Funkcija za množenje (serialMultiplication).....	5
4.2 Paralelni program (parallel_for) – moduli i osnovne metode	6
4.2.1 Modul glavnog programa (main).....	6
4.2.2 Modul za obradu programa (ParallelForTask)	6
4.2.2.1 Metoda za pokretanje množenja (doTask)	6
4.2.3 Modul za rukovanje matricom (PMatrixInfo)	6
4.2.3.1 Funkcija za množenje (prallelForMultiplication).....	6
4.3 Paralelni program (task_groups) – moduli i osnovne metode	7
4.3.1 Modul glavnog programa (main).....	7
4.3.2 Modul za obradu programa (TaskGroups)	7

4.3.2.1	Metoda za pokretanje množenja (doTask)	7
4.3.3	Modul za rukovanje matricom (TGMatrixInfo)	7
4.3.3.1	Funkcija za računanje jednog reda rezultujuće matrice po zadatku (oneRowOfResMatrix)	7
4.3.3.2	Funkcija za računanje jednog elementa rezultujuće matrice po zadatku (oneElementOfResMatrix)	7
4.3.3.3	Funkcija za računanje dela rezultujuće matrice po zadatku (partOfResMatrix)	7
5.	Testiranje	8
5.1	Testni slučajevi.....	8
5.1.1	Dimenzije matrica nisu dobro definisane za množenje	8
5.1.2	Dimenzije matrica dobro definisane za množenje	9
5.2	CheckOutputFiles	9
5.2.1	Modul CheckFiles	9
5.3	Rezultati i analiza vremena izvršenja	11
6.	Zaključak	13

SPISAK SLIKA

Slika 1 - Osnovni elementi matrice	1
Slika 2 - Izlaz programa za loše definisane dimenzije matrica	8
Slika 3 - Izlaz programa za loše definisane dimenzije matrica	8
Slika 4 - Izlaz programa za dobro definisane dimenzije matrica	9
Slika 5 - Izlaz programa za uspešnu proveru rešenja množenja	9
Slika 6 - Izlaz programa kada rešenja nisu jednaka	10
Slika 7 - Grafik zavisnosti vremena izvršavanja programa od implementacije i veličine matrice	12

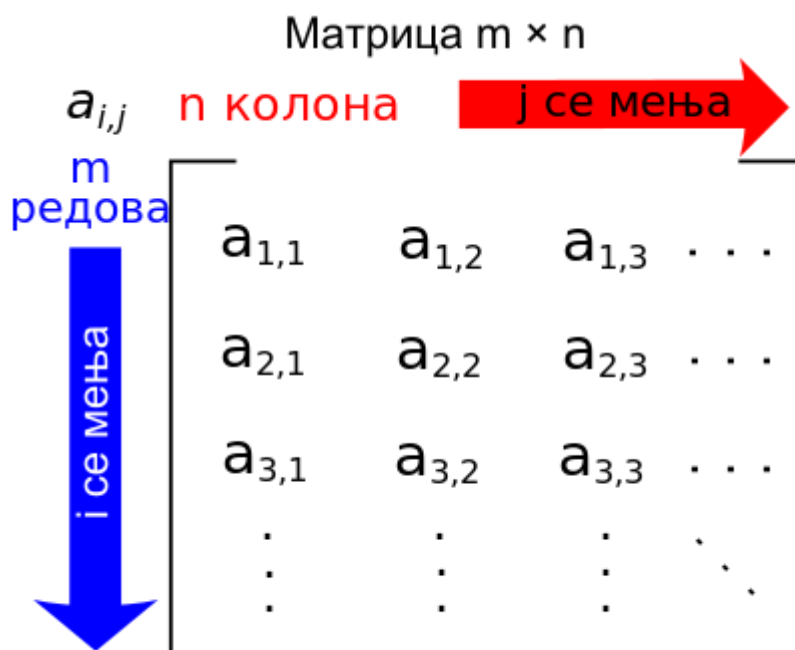
SPISAK TABELA

Tabela 1 - Vremena izvršavanja svih implementacija množenja u ms	11
--	----

1. Uvod

1.1 Matrice

Matrice se koriste da opišu linearne podatke, da se prate koeficijenti linearnih transformacija, kao i za čuvanje podataka koji zavise od dva parametra - što je u programiranju i najčešći slučaj.



Slika 1 - Osnovni elementi matrice

1.2 Međusobno množenje matrica

Množenje dve matrice je dobro definisano samo ako je broj kolona leve matrice jedan broju vrsta desne matrice. Ako je A matrica dimenzija $M \times N$, a B je matrica definisana sa $N \times P$, tada je njihov proizvod AB matrica dimenzija $M \times P$ dobijena na sledeći način:

$$(AB)[i,j]=A[i,1]B[1,j]+A[i,2]B[2,j]+\dots+A[i,n]B[n,j], \text{ za svaki par } i \text{ i } j.$$

1.3 Zadatak

Upotrebom TBB biblioteke realizovati algoritam množenja matrica celobrojnih vrednosti.

Potrebno je realizovati:

- Serijsku verziju programa za množenje matrica
- Paralelnu verziju algoritma upotrebom šablona visokog nivoa (`parallel_for`)
- Paralelnu verziju algoritma upotrebom TBB grupe zadataka (`task_group`)
- Skup ispitnih slučajeva koji potvrđuju ispravnost rada paralelne verzije

Matrice treba da se nalaze u zasebnom direktorijumu i da se učitaju na početku rada programa. Matrice mogu ali ne moraju biti kvadratne – voditi računa da dimenzije moraju biti odgovarajuće kako bi množenje bilo definisano. Veličinu matrica treba varirati od malih do veoma velikih (u zavisnosti od toga koliko radna memorija dozvoljava). Za svaki testni slučaj potrebno je izmeriti vreme i uporediti serijsku implementaciju, implementaciju uz pomoć *parallel_for*-a (koristiti sve podrazumevane vrednosti) i implementaciju uz pomoć TBB zadataka.

Što se tiče samih TBB zadataka, ispitajte rešenje sa različitom kompleksnošću zadataka. Jedan slučaj može da bude da svaki zadatak računa celu jednu vrstu ili kolonu u rezultatnoj matrici. Treći slučaj može da bude da svaki zadatak računa $1/N$ elemenata matrice, gde je N broj jezgara/hiperniti dostupnih na vašem računaru. Po želji možete praviti i dodatne slučajeve.

2. Analiza problema

Ako se zanemari algoritamski deo zadatka, glavni problem ostaje paralelizacija sekvencijalnog koda i verifikacija oba programa.

S obzirom da će se nakon sekvencijalnog programa praviti paralelni, sekvencijalan kod treba da bude modularan i sa jasno odvojenim blokovima obrade, tako da se kasnije može što lakše paralelizovati.

Prilikom paralelizacije, potrebno je uočiti regione koji se mogu paralelizovati, kao i odgovarajući tip paralelizacije (podaci, zadaci, i sl.). Nakon toga, treba odabrati način podele problema na zadatke, odrediti način kako će se zadaci formirati i povezati, odnosno kako će izgledati graf tih zadataka. Prilikom izdvajanja zadataka, nastaje problem sinhronizacije i komunikacije između zadataka, koje treba rešiti nekom od dostupnih metoda.

Neophodno je obezbediti isključiv pristup svim deljenim promenljivama, kako između zadataka, tako i na mestima gde se koristi paralelizam podataka, a zavisnost između nekih podataka postoji.

Takođe, potrebno je proveriti ispravnost rada paralelne implementacije i utvrditi da je program determinističan.

3. Koncept rešenja

Rešenje se zasniva na algoritmu za množenje matrica uz korišćenje `parallel_for` petlje i `task_groups`.

U prvom delu rešenja urađen je serijski algoritam i proveren određenim testnim podacima. Nakon toga, pomoću `parallel_for` je napravljena paralelizovana verzija. Kako bi algoritam bio još brži korišćeni su dinamički alocirani nizovi tipa `long long*`, dosta brži od `vector<vector<long long>>`.

Za jako velike dimenzije matrica (preko 2000x2000) efikasan je algoritam sa `task_groups` (grupe zadataka).

Algoritam sa grupom zadataka smo podelili na tri vrste:

- Algoritam gde svaki zadatak računa jedan red rezultujuće matrice
- Algoritam gde svaki zadatak računa jedan element rezultujuće matrice
- Algoritam gde delimo matricu na 4 dela i svaki zadatak računa jedan deo rezultujuće matrice

Posle pokretanja svih algoritama, vrši se provera ispravnosti istih tako što se dobijeni rezultati porede sa već proverenim rezultatom serijskog programa.

4. Opis rešenja

4.1 Sekvencijalan program – moduli i osnovne metode (MatrixMulSerial)

4.1.1 Modul glavnog programa (Main)

```
int main(int argc, char* argv[]);
```

Glavna funkcija serijskog programa. Ucitava nazive ulazne i izlazne datoteke i pokreće serijsku implementaciju.

4.1.2 Modul za obradu programa (SerialTask)

Modul za čitanje ulazne datoteke, ispisivanje rešenja u izlaznu datoteku i pokretanje množenja matrica.

4.1.2.1 Metoda za pokretanje množenja (doTask)

```
void SerialTask::doTask();
```

Može, ali ne mora da ispisuje matrice A, B i $A \times B$, pokreće funkciju `serialMultiplication()` i meri trajanje izvršenja te funkcije.

4.1.3 Modul za rukovanje matricom (MatrixInfo)

Modul koji je zadužen za ispravan rad matrice, kao i za algoritam za množenje.

4.1.3.1 Funkcija za množenje (serialMultiplication)

```
void serialMultiplication(MatrixInfo& B, MatrixInfo& resMatrix);
```

Obavlja množenje matrice A (objekat klase `MatrixInfo` nad kojim se poziva funkcija `serialMultiplication()`) i matrice B. Rezultat se smešta u matricu `resMatrix`.

4.2 Paralelni program (parallel_for) – moduli i osnovne metode

4.2.1 Modul glavnog programa (main)

```
int main(int argc, char* argv[]);
```

Glavna funkcija paralelnog programa. Učitava nazive ulazne i izlazne datoteke i pokreće implementaciju sa parallel_for.

4.2.2 Modul za obradu programa (ParallelForTask)

Modul za čitanje ulazne datoteke, ispisivanje rešenja u izlaznu datoteku i pokretanje množenja matrica.

4.2.2.1 Metoda za pokretanje množenja (doTask)

```
void ParallelForTask::doTask();
```

Može, ali ne mora da ispisuje matrice A, B i AxB, pokreće funkciju parallelForMultiplication() iz modula PMatrixInfo i meri trajanje izvršenja te funkcije.

4.2.3 Modul za rukovanje matricom (PMatrixInfo)

Modul koji je zadužen za ispravan rad matrice, kao i za algoritam za množenje koristeći parallel_for.

4.2.3.1 Funkcija za množenje (parallelForMultiplication)

```
void parallelForMultiplication(PMatrixInfo& B, PMatrixInfo& resMatrix);
```

Obavlja množenje matrice A (objekat klase PMatrixInfo nad kojim se poziva funkcija parallelForMultiplication) i matrice B. Rezultat se smešta u matricu resMatrix.

U pozivu funkcije parallel_for prosledjujemo dvodimenzionalni iteracioni opseg blocked_range2d<size_t> i funktor MatrixMul(PMatrixInfo*, PMatrixInfo*, PMatrixInfo*).

Klasa **MatrixMul** redefiniše metodu:

```
void operator()(const blocked_range2d<size_t>&) const
```

koja izvršava množenje matrica.

4.3 Paralelni program (task_groups) – moduli i osnovne metode

4.3.1 Modul glavnog programa (main)

```
int main(int argc, char* argv[]);
```

Glavna funkcija paralelnog programa. Ucitava nazive ulazne i izlazne datoteke i pokreće implementaciju sa task_groups.

4.3.2 Modul za obradu programa (TaskGroups)

Modul za čitanje ulazne datoteke, ispisivanje rešenja u izlaznu datoteku i pokretanje množenja matrica.

4.3.2.1 Metoda za pokretanje množenja (doTask)

```
void TaskGroups::doTask();
```

Može, ali ne mora da ispisuje matrice A, B i $A \times B$, pokreće funkciju tasksMultiplication() iz modula TGMATRIXINFO i meri trajanje izvršenja te funkcije.

4.3.3 Modul za rukovanje matricom (TGMATRIXINFO)

Modul koji je zadužen za ispravan rad matrice, kao i za tri algoritma za množenje koristeći task_groups.

4.3.3.1 Funkcija za računanje jednog reda rezultujuće matrice po zadatku (oneRowOfResMatrix)

```
void oneRowOfResMatrix(TGMATRIXINFO& B, TGMATRIXINFO& resMatrix);
```

Pravi se novi zadatak za svaki red rezultujuće matrice. Ovaj algoritam se ispostavio kao najefikasniji kod matrica većih dimenzija.

4.3.3.2 Funkcija za računanje jednog elementa rezultujuće matrice po zadatku (oneElementOfResMatrix)

```
void oneElementOfResMatrix(TGMATRIXINFO& B, TGMATRIXINFO& resMatrix);
```

Pravi se novi zadatak za svaki element rezultujuće matrice. Ovaj algoritam nije efikasan jer gubimo mnogo vremena u pravljenju zadataka.

4.3.3.3 Funkcija za računanje dela rezultujuće matrice po zadatku (partOfResMatrix)

```
void partOfResMatrix(TGMATRIXINFO& B, TGMATRIXINFO& resMatrix);
```

Rezultujuća matrica se deli na četiri dela i svaki zadatak računa jedan od delova.

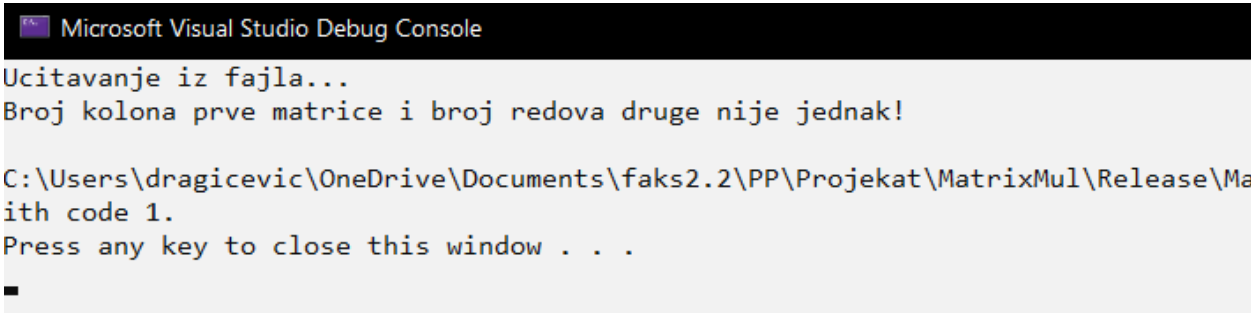
5. Testiranje

5.1 Testni slučajevi

5.1.1 Dimenzije matrica nisu dobro definisane za množenje

- U slučaju da broj kolona prve matrice i broj redova druge matrice nije jedan program izbacuje grešku.

Za matricu A: 3x5 i B: 6x3, izlaz je sledeći:



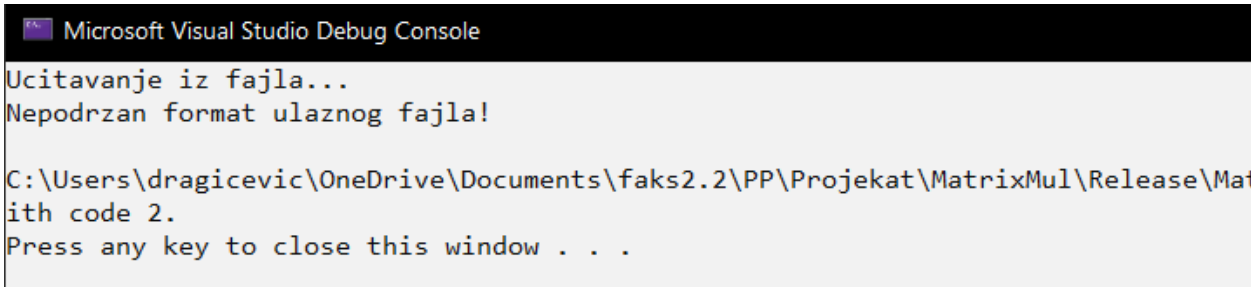
```
Microsoft Visual Studio Debug Console
Ucitavanje iz fajla...
Broj kolona prve matrice i broj redova druge nije jednak!

C:\Users\dragicevic\OneDrive\Documents\faks2.2\PP\Projekat\MatrixMul\Release\Mat
ith code 1.
Press any key to close this window . . .
```

Slika 2 - Izlaz programa za loše definisane dimenzije matrica

- U slučaju da je pročitani broj redova ili kolona matrice manji od 2, program izbacuje grešku.

Za matricu A: -3x4 i B: 4x3, izlaz je sledeći:



```
Microsoft Visual Studio Debug Console
Ucitavanje iz fajla...
Nepodrzan format ulaznog fajla!

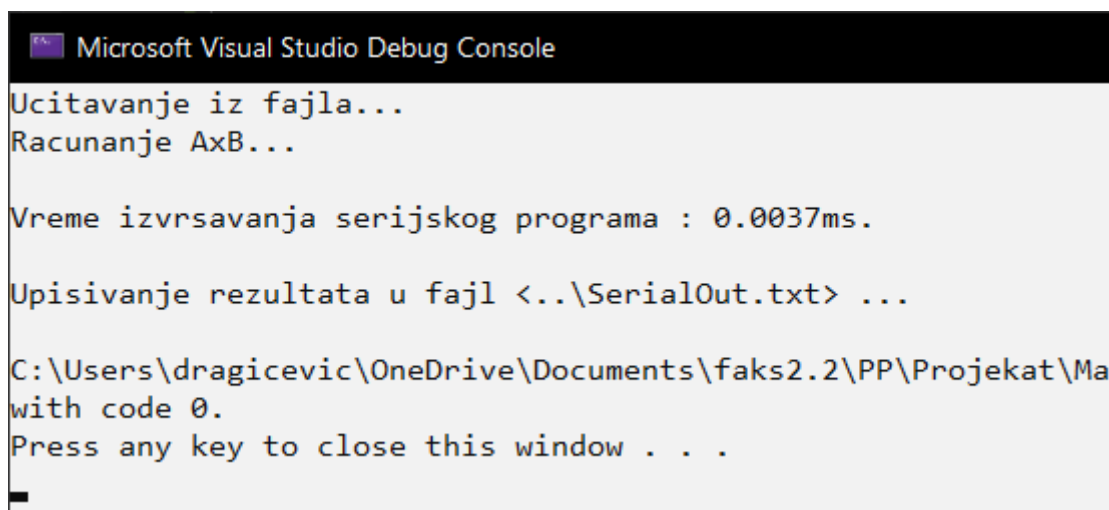
C:\Users\dragicevic\OneDrive\Documents\faks2.2\PP\Projekat\MatrixMul\Release\Mat
ith code 2.
Press any key to close this window . . .
```

Slika 3 - Izlaz programa za loše definisane dimenzije matrica

5.1.2 Dimenzije matrica dobro definisane za množenje

Za ispravne testne slučajeve program ispisuje vreme izvršavanja kao i poruku za upisivanje rezultata u prosleđenu izlaznu datoteku.

Za matricu A: 3x5 i B: 5x2, izlaz je sledeći:



```

Microsoft Visual Studio Debug Console
Ucitavanje iz fajla...
Racunanje AxB...

Vreme izvršavanja serijskog programa : 0.0037ms.

Upisivanje rezultata u fajl <..\SerialOut.txt> ...

C:\Users\dragicevic\OneDrive\Documents\faks2.2\PP\Projekat\Ma
with code 0.
Press any key to close this window . . .

```

Slika 4 - Izlaz programa za dobro definisane dimenzije matrica

5.2 CheckOutputFiles

CheckOutputFiles je projekat koji prima iz komandnih argumenata nazive ulaznih fajlova, poredi njihov sadržaj i ispisuje ukoliko se poklapaju ili ne.

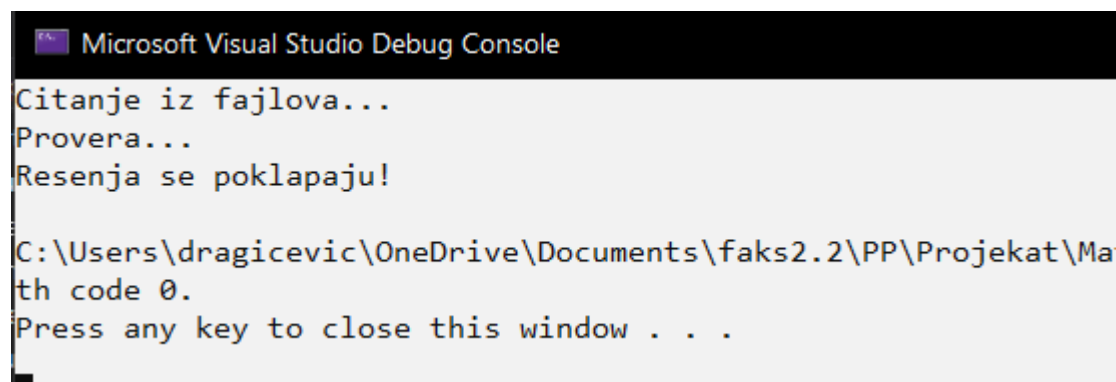
Smisao ovog projekta je da verifikuje ispravnost rada paralelnog programa.

Može porediti dve ili tri ulazne datoteke.

5.2.1 Modul CheckFiles

U ovom modulu se vrši provera tako što se otvaraju datoteke, čita se iz njih sadržaj i poredi medjusobno. Takođe, program broji koliko puta je došlo do neslaganja i to na kraju ispisuje.

Na slici je prikazan izlaz programa kada su rešenja svih implementacija jednaka:



```

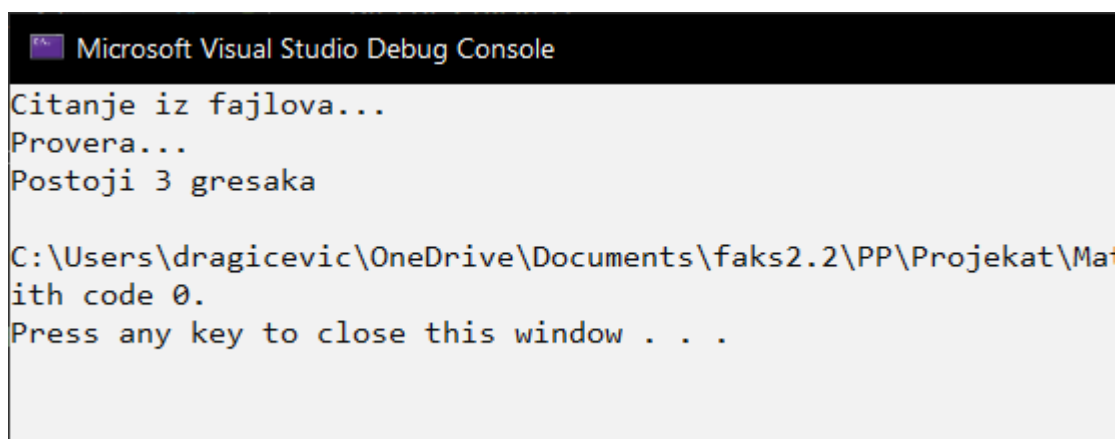
Microsoft Visual Studio Debug Console
Citanje iz fajlova...
Provera...
Resenja se poklapaju!

C:\Users\dragicevic\OneDrive\Documents\faks2.2\PP\Projekat\Ma
th code 0.
Press any key to close this window . . .

```

Slika 5 - Izlaz programa za uspešnu proveru rešenja množenja

A na sledećoj slici je prikazan izlaz programa kada rešenja nisu jednaka:

The image is a screenshot of the Microsoft Visual Studio Debug Console. The title bar at the top is black with the text 'Microsoft Visual Studio Debug Console' in white. The console area has a light gray background and displays the following text in a monospaced font: 'Citanje iz fajlova...', 'Provera...', 'Postoji 3 gresaka', 'C:\Users\dragicevic\OneDrive\Documents\faks2.2\PP\Projekat\Mat', 'ith code 0.', and 'Press any key to close this window . . .'.

```
Microsoft Visual Studio Debug Console
Citanje iz fajlova...
Provera...
Postoji 3 gresaka
C:\Users\dragicevic\OneDrive\Documents\faks2.2\PP\Projekat\Mat
ith code 0.
Press any key to close this window . . .
```

Slika 6 - Izlaz programa kada rešenja nisu jednaka

5.3 Rezultati i analiza vremena izvršenja

Svi testni slučajevi su uspešno izvršeni, što ukazuje na ispravnost verifikovanog programa nad definisanim skupom testova. Pod pretpostavkom da definisani skup testova obuhvata sve kritične slučajeve, sekvencijalan kod možemo smatrati ispravnim i uzeti za referentni.

Poređenjem izlaza sekvencijalnog i paralelnog programa, utvrđeno je da daju iste rezultate, odnosno da paralelni program isto kao i referentni, sekvencijalni.

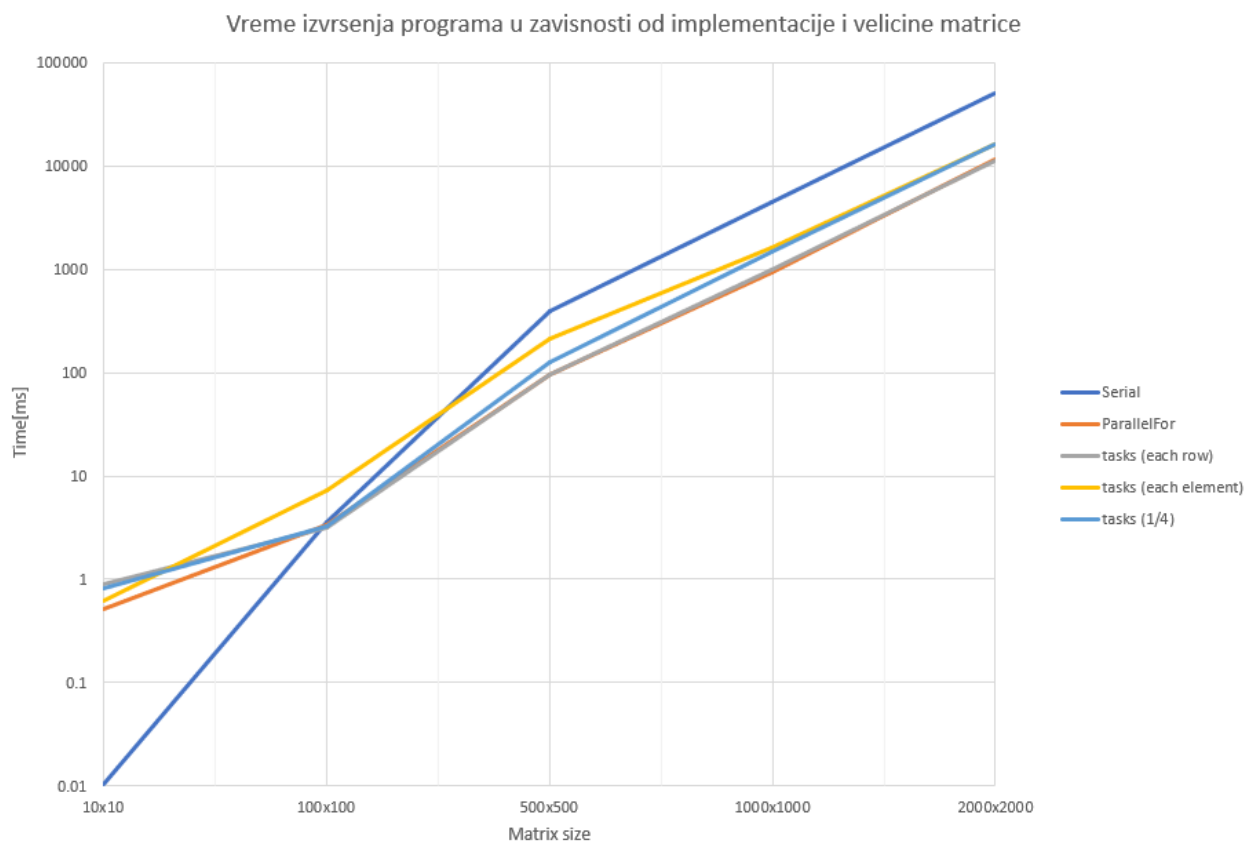
U cilju merenja vremena izvršenja i poređenja sekvencijalnog i paralelnog programa, oba su izvršavana na četvorojezgarnom procesoru *AMD Ryzen 5 3500*.

Izmerena vremena izvršavanja sekvencijalnog i paralelnog programa za određene dimenzije matrica prikazuje sledeća tabela:

	serial	parallelFor	tasks (each row)	tasks (each element)	tasks (1/4)
1	0.0097	0.5764	0.7124	0.5739	0.8153
2	0.0102	0.3596	0.9851	0.6238	0.6866
3	0.0102	0.7683	0.9077	0.6455	0.9044
4	0.0112	0.3791	0.9468	0.5982	0.9089
10x10 (avg)	0.010325	0.52085	0.888	0.61035	0.8288
1	3.4969	2.9367	3.223	7.2166	3.4587
2	4.0061	3.1282	3.1895	7.0766	3.4792
3	2.9709	3.9992	2.9317	7.8847	2.7782
4	3.8935	3.1975	3.2353	6.8137	3.4601
100x100 (avg)	3.59185	3.3154	3.144875	7.2479	3.29405
1	412.203	102.435	107.688	246.271	128.211
2	396.044	91.5582	86.2897	209.264	151.977
3	382.391	94.0756	84.1405	196.552	108.339
4	369.801	98.1857	100.502	192.637	116.508
500x500 (avg)	390.1098	96.563625	94.65505	211.181	126.2588
1	4724.88	915.421	1064.61	1706.34	1601.4
2	4119.03	912.178	952.957	1566.47	1561.11
3	4554.34	989.878	993.216	1577.83	1313.39
1000x1000 (avg)	4466.083	939.159	1003.594333	1616.88	1491.967
1	49548	11986.5	11158.6	17227.6	16674.3
3	53686.2	11357.5	11539.8	16338.3	15926.2
3	48909.9	11219	11023.2	14827.9	16733.3
2000x2000 (avg)	50714.7	11521	11240.53333	16131.26667	16444.6

Tabela 1 - Vremena izvršavanja svih implementacija množenja u ms

Grafik zavisnosti vremena izvršavanja programa od implementacije i veličine matrice dat je na sledećoj slici:



Slika 7 - Grafik zavisnosti vremena izvršavanja programa od implementacije i veličine matrice

6. Zaključak

Napravljen je jedan koncept za realizaciju datog problema. Napravljen je i verifikovan sekvencijalni i paralelni model koji obuhvata problem opisan u zadatku.

Poređenjem izlaza paralelnog i sekvencijalnog programa potvrđeno je njihovo podudaranje, odnosno ispravnost paralelnog programa.

Serijski i paralelni program su pokretani za iste testne matrice na istom računaru u cilju merenja vremena izvršenja.

Rezultati merenja (Tabela 1) pokazuju da se za male dimenzije matrica serijski program izvršava najbrže, dok se za velike dimenzije `parallel_for` i `task_groups` implementacija izvršava i do pet puta brže. Razlog tome je taj da je podela na zadatke skupa za male dimenzije.