



ALGORITMA PEMROGRAMAN

Pengenalan Algoritma dan Kompleksitas Algoritma

TIM PENYUSUN: - DOSEN
- MOCH. DAFFA SHAFWAN CHAIRULLAH

PRESENTED BY: LAB. TEKNIK INFORMATIKA
UNIVERSITAS MUHAMMADIYAH MALANG

ALGORITMA PEMROGRAMAN

CAPAIAN PEMBELAJARAN MATA KULIAH

1. Mahasiswa mampu menjelaskan dan menerapkan algoritma untuk menyelesaikan masalah yang dihadapi.
2. Mahasiswa mampu menganalisa kompleksitas algoritma dalam menyelesaikan masalah.

SUB CAPAIAN PEMBELAJARAN MATA KULIAH

1. Mahasiswa mampu memahami peranan algoritma dalam komputasi dan menjelaskan konsep-konsep dasar analisa algoritma.
2. Mahasiswa mampu menjabarkan di dalam menganalisa kompleksitas algoritma.
3. Mahasiswa mampu membuat flowchart dari algoritma yang telah dibuat.
4. Mahasiswa mampu membuat program melalui algoritma dan flowchart yang dibuat menggunakan bahasa pemrograman Java untuk menyelesaikan masalah.
5. Mahasiswa mampu membuat algoritma, flowchart, dan program operasi matematika matriks.

KEBUTUHAN HARDWARE & SOFTWARE

- Laptop/ PC
- Eclipse/ Netbeans/ dsb (IDE Bahasa Java).

MATERI POKOK

1. Konsep Dasar Algoritma

Algoritma adalah **langkah-langkah logis** penyelesaian masalah yang disusun secara sistematis. Contoh :

- Algoritma Membuat Indomie Goreng
 - 1) Rebus mie dalam air mendidih
 - 2) Tuangkan bumbu dan lainnya pada piring
 - 3) Ketika mie sudah matang, tiriskan
 - 4) Tuangkan mie pada piring
 - 5) Aduk
- Algoritma Mencari Kuadrat Suatu Bilangan
 - 1) Masukkan bilangan X yang ingin dikuadratkan
 - 2) kalikan A dengan A
 - 3) tampilkan hasil
 - 4) selesai

2. Notasi Algoritma

Algoritma dapat dipresentasikan dalam 3 notasi (format penulisan), antara lain :

a. Notasi Algoritma Deskriptif

Merupakan notasi yang paling simpel dan mudah ditulis, penulisan dua algoritma diatas (Algoritma Membuat Indomie Goreng dan Algoritma Mencari Kuadrat Suatu Bilangan) merupakan contoh penulisan algoritma deskriptif.

b. Pseudocode

Notasi ini merupakan notasi yang sangat mirip dengan penulisan bahasa pemrograman, didalam notasi pseudocode ada for, if, else if, while, dll. Dalam penulisannya, dibagi menjadi tiga bagian, ada *Title of Algorithm* (Nama Algoritma), *Declaration* (Deklarasi), *Description* (Deskripsi)

Contoh :

```

Algoritma penentuan_bilangan_ganjil_genap

Deklarasi
x : Integer //deklarasi variabel x dengan tipe data integer
hasil : String

Deskripsi
read(x) //input variabel x
if (x%2==0)//mengecek apakah x habis dibagi 2
hasil <- 'bilangan genap' //jika iya, maka variabel hasil akan diisi dengan keterangan bil. genap
else
hasil <- 'bilangan ganjil' //jika tidak, variabel hasil akan diisi dengan keterangan bil. ganjil
end if
write(hasil) //tampilkan hasil

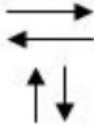


```

c. Flowchart







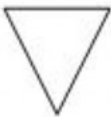

Notasi algoritma yang paling banyak digunakan adalah flowchart karena bentuknya yang sederhana dan mudah dipahami. Flowchart (diagram alir) adalah sebuah jenis diagram yang mewakili algoritma, alir kerja atau proses, yang menampilkan langkah-langkah dalam bentuk graf. Diagram ini mewakili penggambaran penyelesaian masalah. Diagram alir digunakan untuk menganalisa, mendesain, mendokumentasi atau manajemen sebuah proses atau program di berbagai bidang.

Flowchart diawali dengan penerimaan masukan (input), pemroses masukan dan diakhiri dengan menampilkan hasilnya (output). Adapun simbol-simbol yang sering digunakan untuk menyusun flowchart adalah sebagai berikut :

Flow Direction Symbol

	<p>Simbol arus /flow Untuk menyatakan jalannya arus suatu proses.</p>
	<p>Simbol Communication link Untuk menyatakan bahwa adanya transmisi suatu data/informasi dari satu lokasi ke lokasi lainnya</p>
	<p>Simbol Connector Untuk menyatakan sambungan dari satu proses ke proses lainnya dalam halaman/lembar yang sama.</p>
	<p>Simbol Offline Connector Untuk menyatakan sambungan dari satu proses ke proses lainnya dalam halaman/lembar yang berbeda.</p>

Processing symbol

	Simbol Process Untuk menyatakan suatu tindakan (proses) yang dilakukan oleh komputer.
	Simbol Manual Untuk menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer (manual).
	Simbol Decision / logika Untuk menunjukkan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban, ya / tidak.
	Simbol Predefined Proses Untuk menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberi harga awal.
	Simbol Terminal Untuk menyatakan permulaan atau akhir suatu program.
	Simbol Keying Operation Untuk menyatakan segala jenis operasi yang diproses dengan menggunakan suatu mesin yang mempunyai keyboard
	Simbol off-line storage Untuk menunjukkan bahwa data dalam symbol ini akan disimpan ke suatu media tertentu.
	Simbol Manual input Untuk memasukkan data secara manual dengan menggunakan online keyboard.

Input output

**Simbol Input-output**

Untuk menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya.

**Simbol Punched Card**

Untuk menyatakan input berasal dari kartu atau output ditulis ke kartu.

**Simbol Magnetic-tape unit**

Untuk menyatakan input berasal dari pita magnetic atau output disimpan ke pita magnetic.

**Simbol Disk storage**

Untuk menyatakan input berasal dari disk atau output disimpan ke disk.

**Simbol Document**

Untuk mencetak laporan ke printer.

**Simbol Display**

Untuk menyatakan peralatan output yang digunakan berupa layar (video, komputer).

3. Kompleksitas Algoritma

- Sebuah masalah dapat mempunyai banyak algoritma penyelesaian. Contoh :
 - Masalah membuat Indomie Goreng, ada banyak cara untuk memasak (direbus atau di microwave)
 - Masalah perkalian matriks (ada banyak cara untuk menyelesaikan)
- Sebuah algoritma tidak tidak saja harus benar, akan tetapi harus mangkus (efisien).
- Algoritma yang bagus adalah algoritma yang mangkus (efisien).
- Kemangkusan algoritma dapat diukur dari waktu dari waktu (waktu eksekusi program) dan ruang (penggunaan memory/RAM).
- algoritma yang mangkus adalah algoritma yang meminimumkan kebutuhan waktu dan ruang.
- Kebutuhan waktu dan ruang pada suatu algoritma bergantung pada ukuran masukan / input (n). n adalah jumlah data yang akan diproses.
- Pada modul ini, kita akan belajar tentang kebutuhan waktu (kompleksitas waktu / time complexity) pada suatu algoritma.
- Kompleksitas Waktu
 - Merupakan jumlah waktu yang dibutuhkan oleh suatu program untuk berjalan hingga program tersebut selesai.
 - Kompleksitas waktu biasa ditulis dengan notasi Big O.
 - Macam – Macam kompleksitas waktu
 - Konstan

```
int N = 5;
System.out.print("Ini adalah bilangan" + N);
```

Dalam Kode diatas, kompleksitas waktunya adalah konstan. Karena ketika nilai N diganti, waktu eksekusi program tidak akan terpengaruh.

- Linear

```
for (i=0; i < N; i++){
    System.out.print("Ini angka " + i);
}
```

Kode diatas adalah contoh kompleksitas waktu linear, karena semakin banyak nilai N maka looping akan semakin banyak dilakukan, dengan semakin banyak looping yang dilakukan, maka waktu eksekusi akan bertambah. Jika nialai N bertambah, maka waktu eksekusi akan bertambah juga.

- Kuadratik

```

for(i=0; i < N; i++)
{
    for(j=0; j < N; j++)
    {
        statement;
    }
}

```

Kode diatas merupakan kode dari *nested loop* (for dalam for). Kompleksitas waktunya adalah kuadratik. Karena waktu eksekusi dari dua for loop adalah N^2 . Ketika nilai N bertambah, maka waktu eksekusi akan bertambah senilai N^2 .

- Logaritmik

```

while(low <= high)
{
    mid = (low + high) / 2;
    if (target < list[mid])
        high = mid - 1;
    else if (target > list[mid])
        low = mid + 1;
    else break;
}

```

Kode diatas merupakan program dengan kompleksitas waktu logaritmik. Disebut logaritmik karena setiap looping dieksekusi nilai mid dibagi menjadi dua.

- Penulisan Big O

- Looping

```

for (i=0; i<n; i++) //program akan dijalankan sebanyak n
{
    a = b+c; //baris ini akan dijalankan secara konstan, sebut saja baris ini x
}

//dari for diatas maka x akan dijalan sebanyak n kali (sebanyak jumlah n)
//maka kompleksitas waktu bisa ditulis xn (x * n)
//dalam notasi big O, x dalam xn dapat diabaikan, sehingga tinggal n saja
//sehingga notasi big O nya adalah O(n)

```


▪ Nested Loop

```

for (i=0; i<n; i++) //program akan dijalankan sebanyak n
{
    for (j=0; j<n; j++) //program akan dijalankan sebanyak n
    {
        a = b+c; //baris ini akan dijalankan secara konstan, sebut saja baris ini x
    }
}

//dari for diatas maka x akan di jalan sebanyak n * n (n^2) kali (n * n karena ada dua for loop)
//maka kompleksitas waktu bisa ditulis xn^2 (x * n^2)
//dalam notasi big O, x dalam xn dapat diabaikan, sehingga tinggal n^2 saja
//sehingga notasi big O nya adalah O(n^2)

```

▪ Gabungan

```

1 c = d + e; //sebut saja baris ini x
2
3 for (i=0; i<n; i++) //program akan dijalankan sebanyak n
4 {
5     f = g+h; //baris ini akan dijalankan secara konstan, sebut saja baris ini y
6 }
7
8 for (j=0; j<n; j++) //program akan dijalankan sebanyak n
9 {
10     a = b+c; //baris ini akan dijalankan secara konstan, sebut saja baris ini z
11 }
12
13 //program diatas memiliki 3 bagian (statement baris 1, for loop baris 3-6, for loop baris 8-11)
14 //pada statement baris 1, kompleksitas waktunya ditulis dengan x
15 //pada for loop baris 3-6, kompleksitas waktunya ditulis dengan yn
16 //pada for loop baris 8-11, kompleksitas waktunya ditulis dengan zn
17 //kita menjumlahkan notasi kompleksitas waktunya
18 // = x + yn + zn
19 //x, y, dan z dapat diabaikan, sehingga hanya sisa n + n saja atau hanya ada n saja
20 //sehingga notasi big O nya adalah O(n)

```

▪ If-Else

```

1 if (condition)
2 {
3     //misal baris ini big O nya adalah O(n)
4 }else
5 {
6     //misal baris ini big O nya adalah O(n^2)
7 }
8
9 //pada if else diatas kita anggap sudah menghitung masing masing nilai big O pada if dan else
10 //untuk menentukan nilai big O dari if-else, kita memilih yang nilai big O nya terbesar, yaitu O(n^2)
11 //sehingga nilai big O dari if-else diatas adalah O(n^2)

```

MATERI PRAKTIKUM

1. Matriks

Matriks dalam matematika merupakan kumpulan bilangan, simbol atau ekspresi berbentuk persegi panjang yang disusun menurut baris dan kolom. Bilangan-bilangan yang terdapat pada suatu matriks disebut dengan elemen atau disebut juga anggota dari suatu matriks.

2. Operasi Matriks

a. Penjumlahan dan Pengurangan

Penjumlahan serta pengurangan dalam matriks hanya dapat dilakukan apabila kedua matriks mempunyai ukuran atau tipe yang sama. Elemen-elemen dalam suatu matriks yang dijumlahkan atau dikurangkan yaitu elemen yang memiliki posisi/letak yang sama.

$$a_{ij} \pm b_{ij} = c_{ij}$$

representasi dekoratifnya sebagai berikut :

$$\begin{bmatrix} (a_{11} \pm b_{11}) & (a_{12} \pm b_{12}) & (a_{13} \pm b_{13}) \\ (a_{21} \pm b_{21}) & (a_{22} \pm b_{22}) & (a_{23} \pm b_{23}) \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

b. Perkalian

Perkalian matriks dilakukan dengan cara tiap baris dikalikan dengan tiap kolom, selanjutnya dijumlahkan pada kolom yang sama

$$c_{ij} = \sum_{k=1}^m a_{ik} \cdot b_{kj}$$

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad B = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$$

Maka :

$$A \times B = \begin{pmatrix} ap + br & aq + bs \\ cp + dr & cq + ds \end{pmatrix}$$

c. Pembagian

Secara teknis, matriks tidak bisa dibagi. Pembagian satu matriks dengan matriks lainnya tidak dapat didefinisikan. Cara yang paling mendekati adalah mengalikan dengan invers matriks lainnya.

Untuk materi yang lebih lengkap tentang pembagian matriks, dapat dipelajari pada halaman berikut : <https://id.wikihow.com/Membagi-Matriks>

LEMBAR KERJA**KEGIATAN 1**

Pahami algoritma dan flowchart dibawah ini untuk operasi matematika matriks 2x2 serta ketikkan ulang program dibawah menggunakan laptop/pc masing- masing.

1. Algoritma**a. Penjumlahan**

```

Step 1      : Start
Step 2      : Inisialisasi Matriks A dan B
Step 3      : For i dari 0 sampai kurang dari panjang baris matriks A
Step 4      :       For j dari 0 sampai kurang dari panjang kolom matriks A
Step 5      :       Atur matriks C [i][j] = matriks A [i][j] + matriks B
[i][j]
Step 6      : Print matriks C
Step 7      : Stop

```

b. Pengurangan

```

Step 1      : Start
Step 2      : Inisialisasi Matriks A dan B
Step 3      : For i dari 0 sampai kurang dari panjang baris matriks A
Step 4      :       For j dari 0 sampai kurang dari panjang kolom matriks A
Step 5      :       Atur matriks C [i][j] = matriks A [i][j] - matriks B [i]
[j]
Step 6      : Print matriks C
Step 7      : Stop

```

c. Perkalian

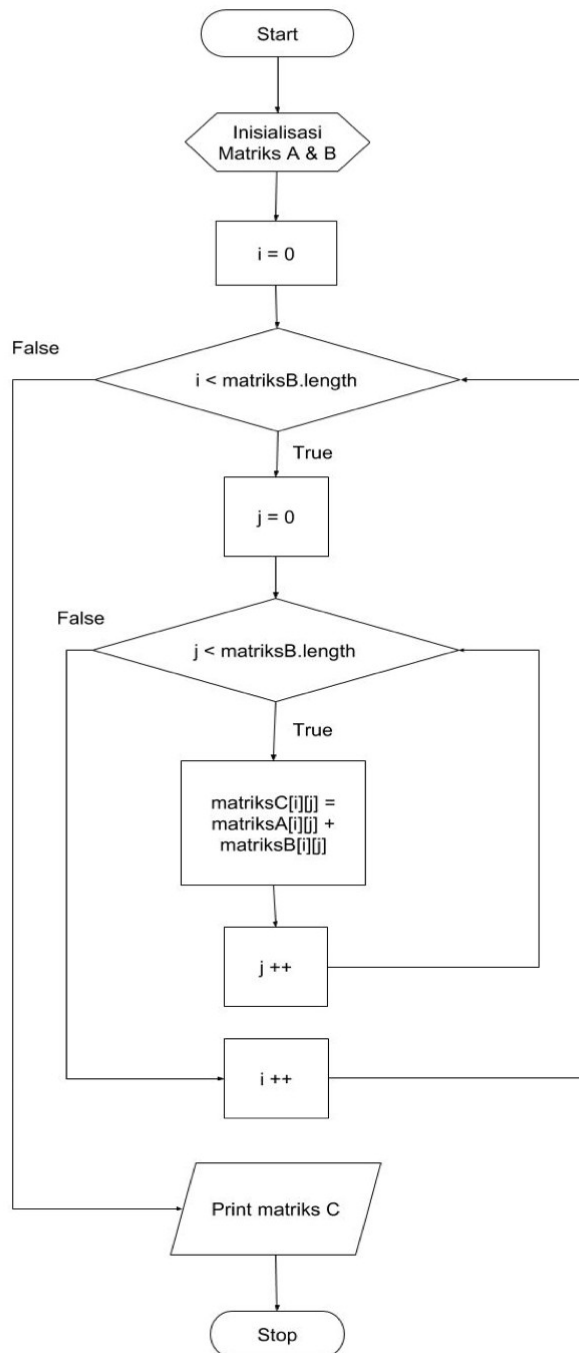
```

Step 1      : Start
Step 2      : Inisialisasi Matriks A, B dan variabel total = 0
Step 3      : For i dari 0 sampai kurang dari panjang baris matriks A
Step 4      :       For j dari 0 sampai kurang dari panjang kolom matriks A
Step 5      :       For k dari 0 sampai kurang dari panjang baris
matriks A
Step 6      :       Atur total = total + (matriks A[i][k] * matriks
B[k][j]
Step 7      :       Atur matriks C [i][j] = total
Step 8      : Print matriks C
Step 9      : Stop

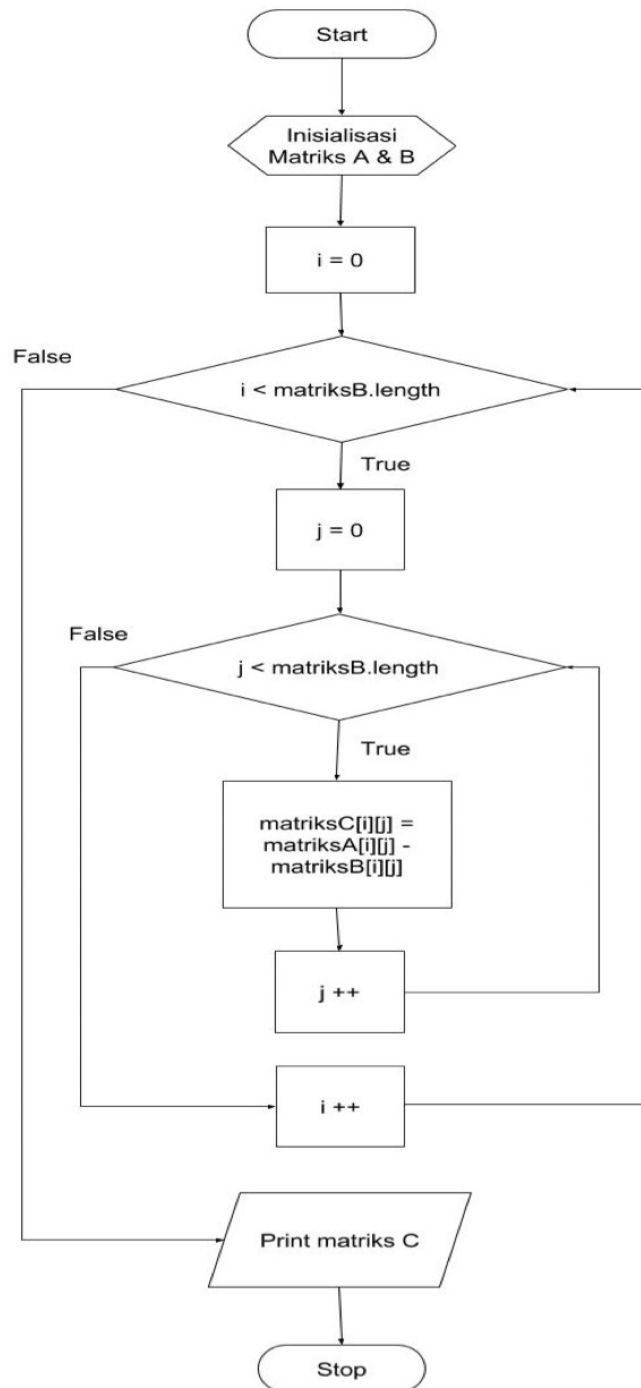
```

2. Flowchart

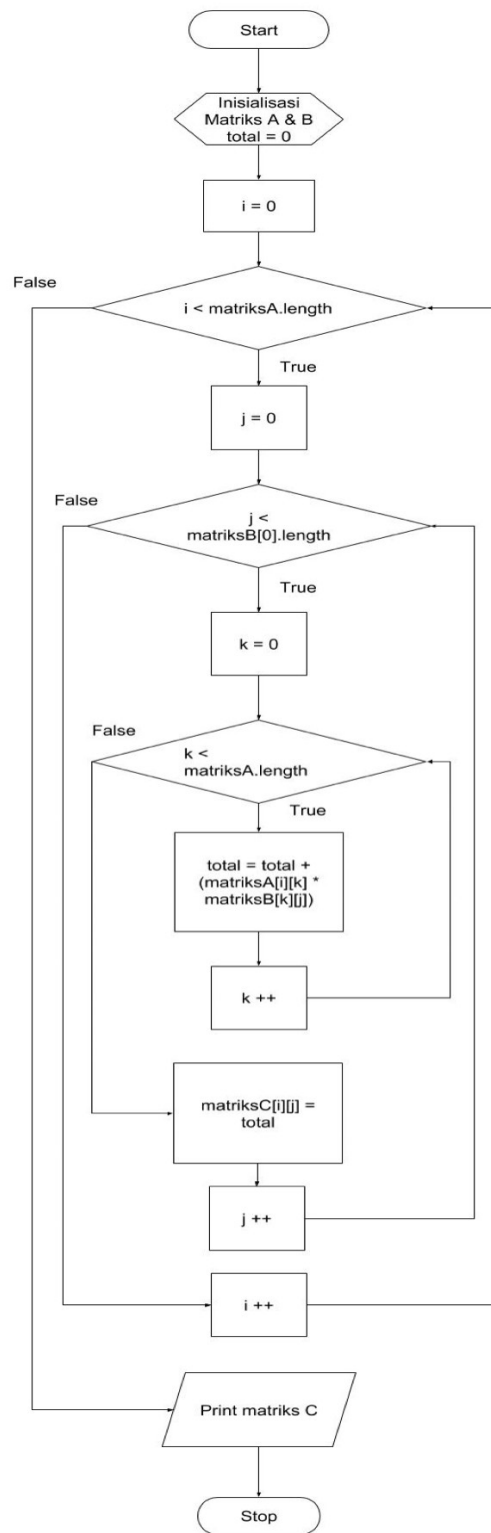
a. Penjumlahan



b. Pengurangan



c. Perkalian



3. Program
 - a. Penjumlahan Matriks 2x2

```
// inisiasi matriks 2x2
int matriksA[][] = { { 2, 5 }, { 4, 6 } };
int matriksB[][] = { { 5, 3 }, { 4, 4 } };
int matriksC[][] = new int[2][2];

// proses penjumlahan
for (int i = 0; i < matriksA.length; i++) {
    for (int j = 0; j < matriksA.length; j++) {
        matriksC[i][j] = matriksA[i][j] + matriksB[i][j];
    }
}

// menampilkan hasil
for (int i = 0; i < matriksC.length; i++) {
    for (int j = 0; j < matriksC.length; j++) {
        System.out.print(matriksC[i][j] + " ");
    }
    System.out.println("");
}
```

- b. Pengurangan Matriks 2x2

```
// inisiasi matriks 2x2
int matriksA[][] = { { 2, 5 }, { 4, 6 } };
int matriksB[][] = { { 5, 3 }, { 4, 4 } };
int matriksC[][] = new int[2][2];

// proses pengurangan
for (int i = 0; i < matriksB.length; i++) {
    for (int j = 0; j < matriksB[0].length; j++) {
        matriksC[i][j] = matriksA[i][j] - matriksB[i][j];
    }
}

// menampilkan hasil
for (int i = 0; i < matriksC.length; i++) {
    for (int j = 0; j < matriksC[0].length; j++) {
        System.out.print(matriksC[i][j] + " ");
    }
    System.out.println("");
}
```

c. Perkalian Matriks 2x2

```

//inisiasi matriks 2x2
int matriksA[][] = { { 2, 5 }, { 4, 6 } };
int matriksB[][] = { { 5, 3 }, { 4, 4 } };
int matriksC[][] = new int[2][2];
int total;

//proses perkalian
for (int i = 0; i < matriksA.length; i++) {
    for (int j = 0; j < matriksA.length; j++) {
        total = 0;
        for (int k = 0; k < matriksA.length; k++) {
            total = total + (matriksA[i][k] * matriksB[k][j]);
        }
        matriksC[i][j] = total;
    }
}

//menampilkan hasil
for (int i = 0; i < matriksC.length; i++) {
    for (int j = 0; j < matriksC.length; j++) {
        System.out.print(matriksC[i][j] + " ");
    }
    System.out.println();
}

```


TUGAS PRAKTIKUM

1. Membuat algoritma dengan notasi pseudocode dan flowchart serta program Java untuk program operasi matriks **m x n** (penjumlahan, pengurangan, perkalian, *pembagian*) yang **bersifat dinamis** dengan **inputan ordo & elemen matriks dari user**.

Ketentuan :

1. Menggunakan Java OOP, yaitu dengan membuat Satu kelas main dan beberapa kelas pendukung.
 2. **Khusus program pembagian** menjadi **tugas opsional**, tetapi **program penjumlahan, pengurangan dan perkalian** merupakan **tugas wajib**.
 3. Jika ingin mengerjakan Program pembagian, matriks nya ordo 2 x 2, hanya elemen matriks saja yang dinamis (inputan dari user), sementara ordo tidak.
2. Analisis kompleksitas waktu pada program operasi matriks dari tugas nomor 1 dan jelaskan kepada asisten saat demo.

RUBRIK PENILAIAN

1. Menjelaskan notasi pseudocode dan flowchart baik secara tertulis atau lisan. (15 poin).
2. Menyelesaikan tugas praktikum wajib sesuai dengan ketentuan. (25 poin)
3. Menyelesaikan tugas praktikum optional sesuai dengan ketentuan. (10 poin)
4. Menjelaskan program dari tugas praktikum yang dibuat. (20 poin)
5. Menjelaskan kompleksitas waktu dari program yang dibuat. (30)

Semangat guys ! ^-^