

VERSION 1.0  
SEPTEMBER 1, 2021



# ALGORITMA PEMROGRAMAN

Pengenalan Algoritma dan Kompleksitas Algoritma | Modul 1

DISUSUN OLEH:

- FARKHAN HAMZAH FIRDAUS
- GILLY HUGA ANARGYA

DIAUDIT OLEH:

- DIDIH RIZKI CHANDRANEGARA, S.KOM., M.KOM
- CHRISTIAN SRI KUSUMA, S.KOM, M.KOM

PRESENTED BY: TIM LAB-IT

UNIVERSITAS MUHAMMADIYAH MALANG

## ALGORITMA PEMROGRAMAN

---

### PERSIAPAN MATERI

#### 1. Konsep Dasar Algoritma

Algoritma adalah serangkaian proses yang dilakukan secara berurutan untuk menyelesaikan sebuah permasalahan. Algoritma bisa bermacam-macam tergantung kepada siapa yang membuat algoritma tersebut. Contoh :

- Algoritma Mendapatkan Diskon 20% Di Supermarket Dengan Minimal Belanja 300 Ribu
  1. Anda harus masuk ke supermarketnya terlebih dahulu.
  2. Ambil keranjang atau troli belanja.
  3. Pilih barang-barang yang ingin Anda beli
  4. Lihat harga yang tercantum di setiap rak tempat Anda mengambil barang.
  5. Setelah semua barang sudah masuk ke dalam troli dan totalnya masih belum mencapai 300 ribu, cari barang lainnya.
  6. Ketika total harga barang yang Anda beli sudah mencapai 300 ribu atau lebih dan dirasa cukup, berjalanlah menuju meja kasir.
  7. Tunggu kasir menghitung kembali semua barang belanjaan Anda.
  8. Keluarkan dompet Anda dan bayarkan sesuai total harga belanjaan yang sudah dihitung oleh kasir.
  9. Periksa kembali struk yang diberikan oleh kasir dan pastikan sudah mendapat diskon 20%.
  10. Ketika pembayaran selesai, jangan lupa ambil belanjaan Anda.
  11. Keluar supermarket dan selesai.
- Algoritma Menghitung Luas Persegi
  1. Masukkan nilai Panjang dan nilai Lebar.
  2. Kalikan nilai Panjang dengan nilai Lebar.
  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 Mendapatkan Diskon 20% Dengan Minimal Belanja 300 Ribu dan Algoritma Menghitung Luas Persegi) merupakan contoh penulisan algoritma deskriptif.

## b) Pseudocode

Notasi ini merupakan notasi yang sangat mirip dengan penulisan bahasa pemrograman, di dalam 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 //deklarasi variabel hasil dengan tipe data String

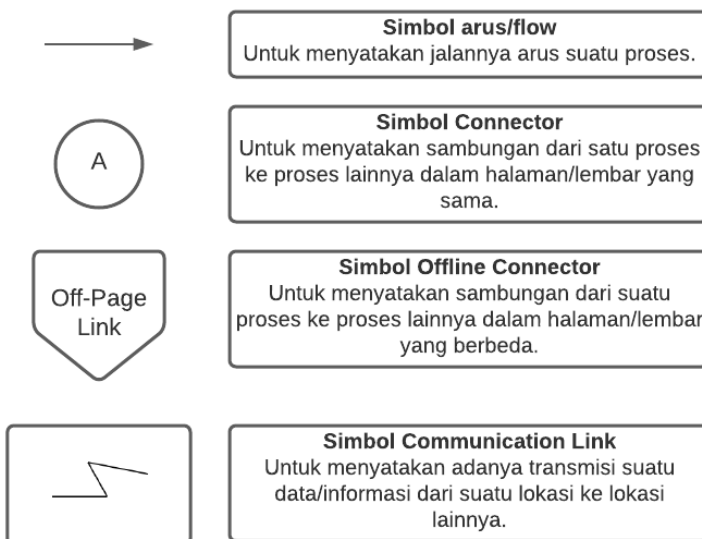
Deskripsi
read(x) //input variabel x
if (x%2==0) //mengecek apakah nilai x habis dibagi 2
    hasil <- 'bilangan genap' //jika iya, maka variabel hasil akan diisi dengan keterangan "bilangan genap"
else
    hasil <- 'bilangan ganjil' //jika tidak, maka variabel hasil akan diisi dengan keterangan "bilangan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

```

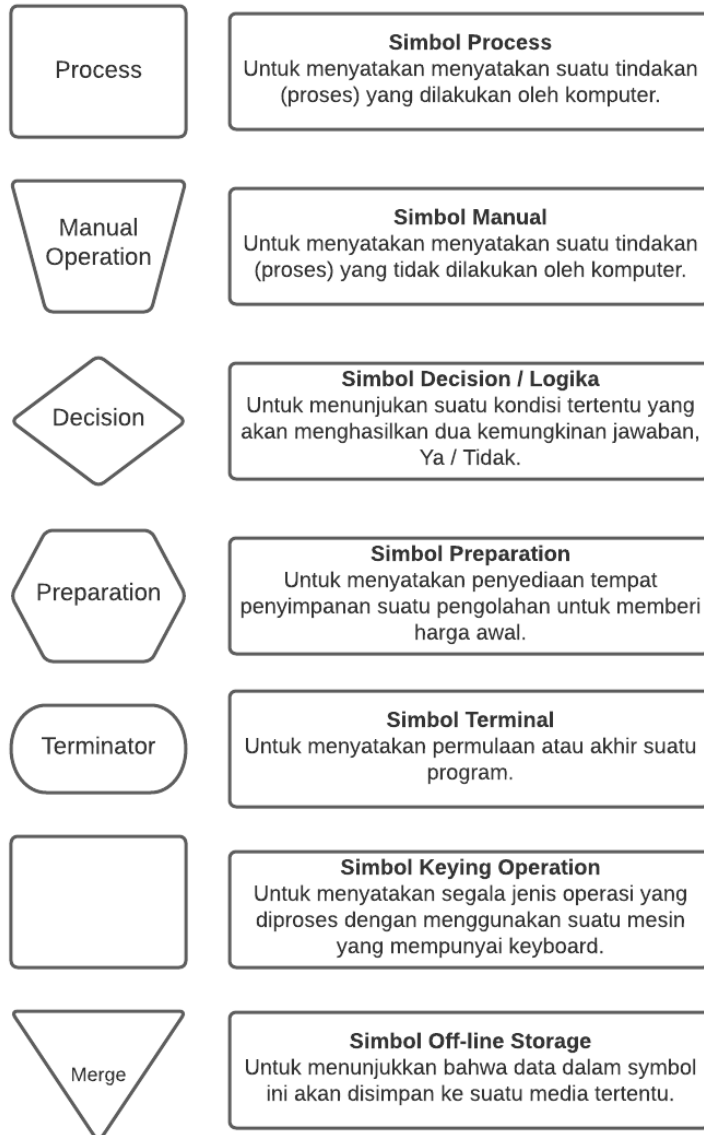
## c) Flowchart

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 grafis. 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 :

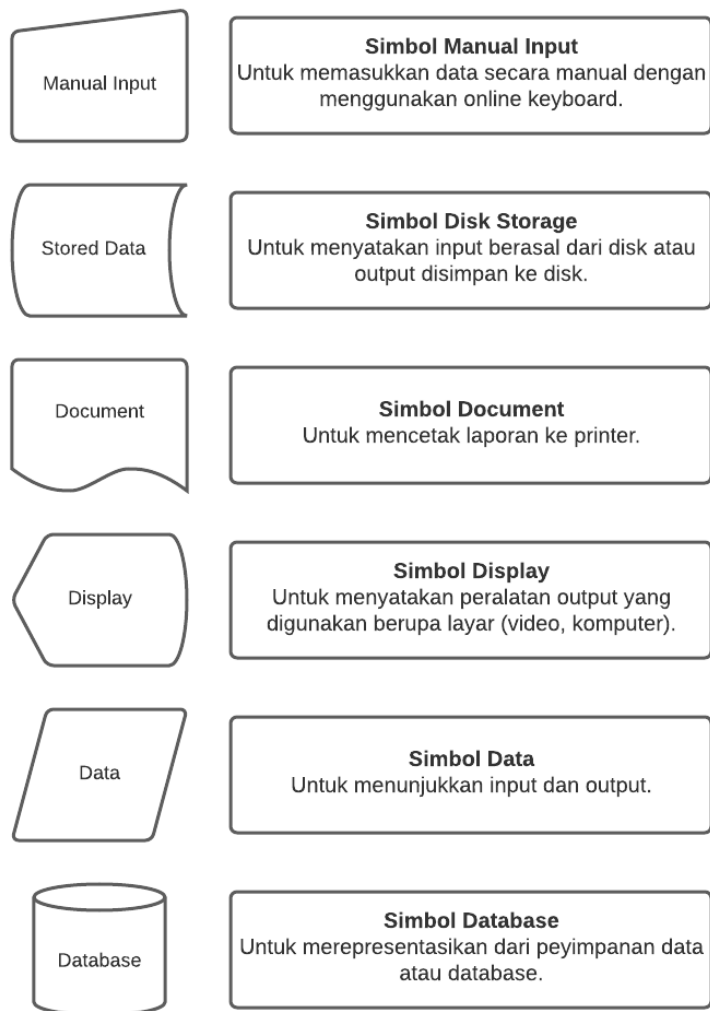
- Flow Direction Symbol



- Processing symbol



- Processing symbol



### 3. Kompleksitas Algoritma

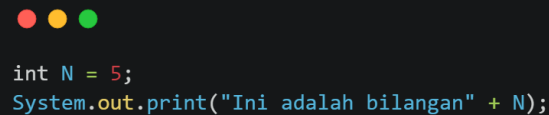
Sebuah masalah dapat mempunyai banyak algoritma penyelesaian. Contoh :

- Masalah membuat Indomie Goreng (ada banyak cara untuk memasak seperti, 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. Berikut macam – Macam kompleksitas waktu :

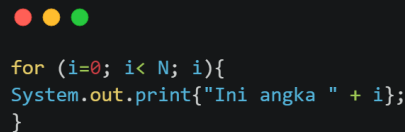
#### a) Konstan



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

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

#### b) Linear



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

Kode di atas 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 nilai  $N$  bertambah, maka waktu eksekusi akan bertambah juga.

#### c) 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  $N^2$  .

d) 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 :

a) Looping

```
1 for (i=0;i<n;i++) //program akan dijalankan sebanyak n
2 {
3     a = b+c; // baris ini akan dijalankan secara konstan, sebut saja baris ini x
4 }
```

- Dari for di atas maka  $x$  akan dijalankan sebanyak  $n$  kali (sebanyak jumlah  $n$ )
- Maka kompleksitas waktu bisa ditulis  $xn$  ( $x \times n$ )
- Dalam notasi big O,  $x$  dalam  $xn$  dapat diabaikan, sehingga tinggal  $n$  saja
- Sehingga notasi big O nya adalah  $O(n)$

b) Nested Loop

```
1 for (i=0;i<n;i++) //program akan dijalankan sebanyak n
2 {
3     for (j=0;j<n;j++) //program akan dijalankan sebanyak n
4     {
5         a = b+c; // baris ini akan dijalankan secara konstan, sebut saja baris ini x
6     }
7 }
8
```

- Dari for di atas maka  $x$  akan dijalankan sebanyak ( $n \times n$  ( $n^2$ )) kali ( $n \times n$  karena ada dua faktor loop)
- Maka kompleksitas waktu bisa ditulis  $xn^2$  ( $x \times n^2$ )
- Dalam notasi big O,  $x$  dalam  $xn$  dapat diabaikan, sehingga tinggal  $n^2$  saja
- Sehingga notasi big O nya adalah  $O(n^2)$

## c) 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 }

```

- Program diatas memiliki 3 bagian
  - statement baris 1, kompleksitas waktunya ditulis dengan  $n$
  - for loop baris 3-6, kompleksitas waktunya ditulis dengan  $yn$
  - for loop baris 8-11, kompleksitas waktunya ditulis dengan  $zn$
- Kita menjumlahkan notasi kompleksitas waktunya =  $x + yn + zn$
- $x, y, z$  dapat diabaikan, sehingga tinggal  $n + n$  saja atau hanya  $n$  saja
- Sehingga notasi big O nya adalah  $O(n)$

## d) 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 }

```

- Pada if else diatas kita anggap sudah menghitung masing masing nilai big O pada if dan else
- Untuk menentukan nilai big O dari if-else, kita memilih nilai big O nya terbesar, yaitu  $O(n^2)$
- Sehingga nilai dari big O dari if-else diatas adalah  $O(n^2)$

---

**TUJUAN**

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



---

## TARGET MODUL

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

---

## PERSIAPAN SOFTWARE/APLIKASI

Eclipse/ Netbeans/IntelliJ IDEA/ dsb (IDE Bahasa Java)

---

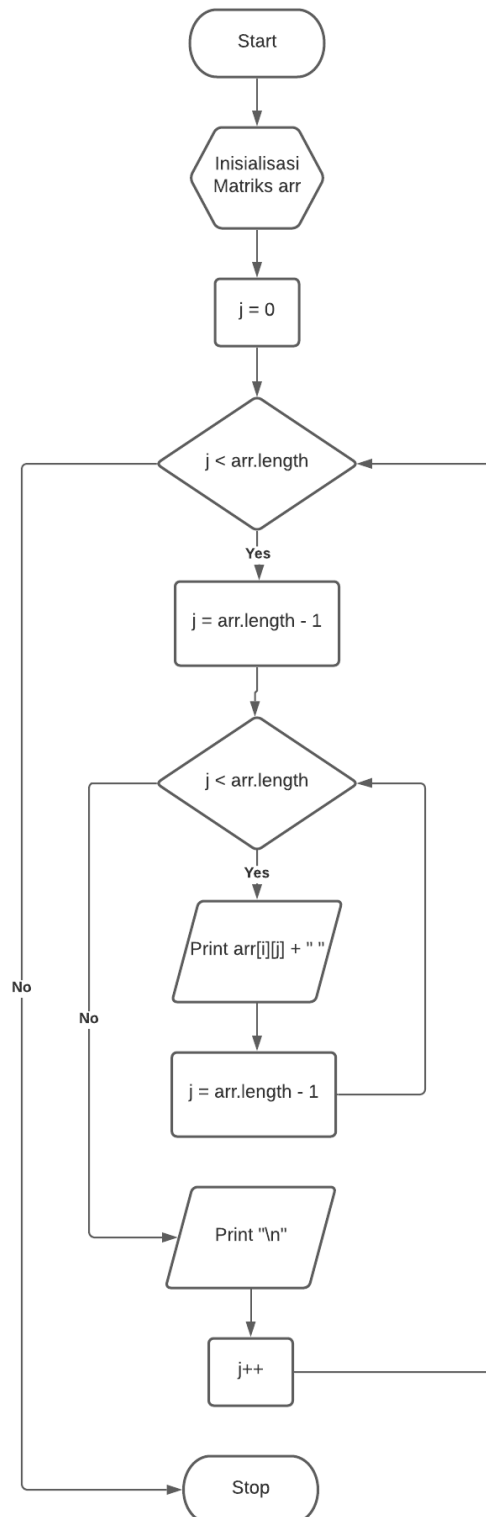
## KEGIATAN PERCOBAAN

### PERCOBAAN 1

Pahami algoritma dan flowchart di bawah ini untuk memutar matriks 3x3 sebanyak 90 derajat. Serta ketikkan ulang program di bawah menggunakan laptop/pc masing- masing.

1. Algoritma
  - Step 1 : Start
  - Step 2 : Inisialisasi Matriks arr
  - Step 3 : For j dari 0 sampai kurang dari Panjang matriks
  - Step 4 : For i dari Panjang matriks dikurangi 1 sampai i kurang dari atau sama dengan 0
  - Step 5 : Print arr[i][j] + " "
  - Step 6 : Print "\n"
  - Step 7 : Stop

## 2. Flowchart



## 3. Program

```

1 import java.io.*;
2
3 public class matriks {
4
5     public static void main(String[] args)
6     {
7         //Inisialisasi isi matriks
8         int arr[][] = { { 1, 2, 3 },
9                         { 5, 6, 7 },
10                        { 9, 10, 11 } };
11
12        //Proses menampilkan hasil dan memutar matriks sebanyak 90 derajat
13        for (int j = 0; j < arr.length; j++) {
14            for (int i = arr.length - 1; i >= 0; i--)
15                System.out.print(arr[i][j] + " ");
16            System.out.println();
17        }
18    }
19 }

```

## TUGAS

1. Membuat algoritma dengan notasi pseudocode dan flowchart serta program Java untuk program memutar matriks  $n \times n$  sebanyak **180 derajat**, yang bersifat dinamis dengan **inputan ordo & elemen matriks dari user**.

Ketentuan :

1. Menggunakan Java OOP, yaitu dengan membuat satu kelas main dan beberapa pendukung.
2. Hasil yang diharapkan dari program:

```

Masukkan jumlah ordo matriks:
5
Masukkan elemen matriks:
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25

Hasil:
[25, 24, 23, 22, 21]
[20, 19, 18, 17, 16]
[15, 14, 13, 12, 11]
[10, 9, 8, 7, 6]
[5, 4, 3, 2, 1]

```

2. Analisis kompleksitas waktu pada program dari tugas nomor 1 dan jelaskan kepada asisten saat demo.

---

#### DETAIL PENILAIAN TUGAS

1. Menjelaskan notasi pseudocode dan flowchart baik secara tertulis atau lisan. (15 poin)
2. Menyelesaikan tugas praktikum sesuai dengan ketentuan. (35 poin)
3. Menjelaskan program dari tugas praktikum yang dibuat. (20 poin)
4. Menjelaskan kompleksitas waktu dari program yang dibuat. (30 poin)