



# DASAR PEMROGRAMAN

TIM AJAR

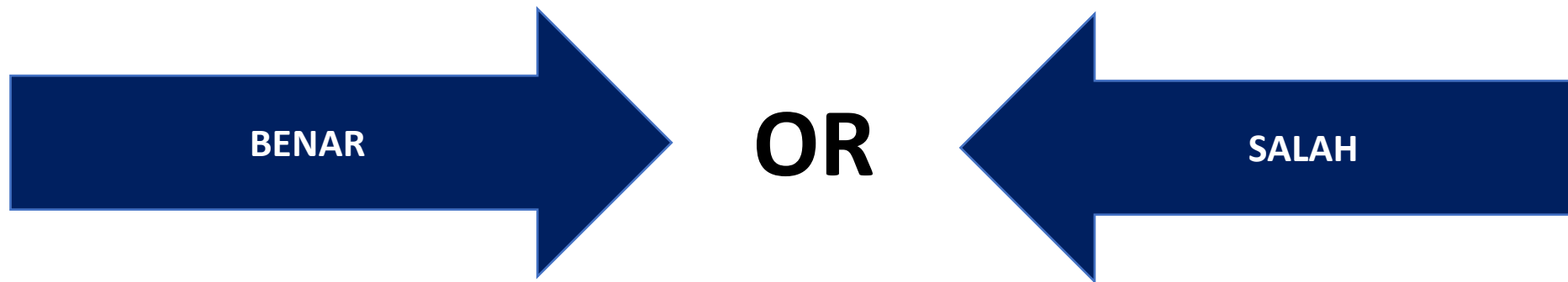
ALGORITMA DAN STRUKTUR DATA

2023/2024

# Pemilihan

- Pemilihan (*selection*) adalah instruksi untuk yang dipakai untuk memilih satu kemungkinan dari beberapa kondisi

Kondisi : suatu pernyataan atau ekspresi (pernyataan logika)



# PEMILIHAN

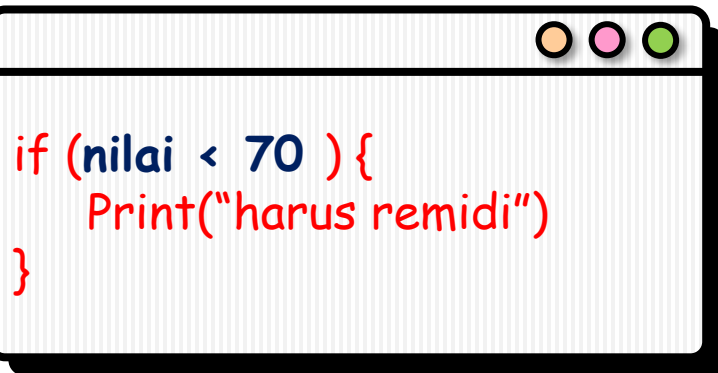
- Contoh:
  - **IF** your name starts with a 'J'
  - **THEN** raise your right hand
  - **ELSE** sit down

# Bentuk Sintaks Pemilihan

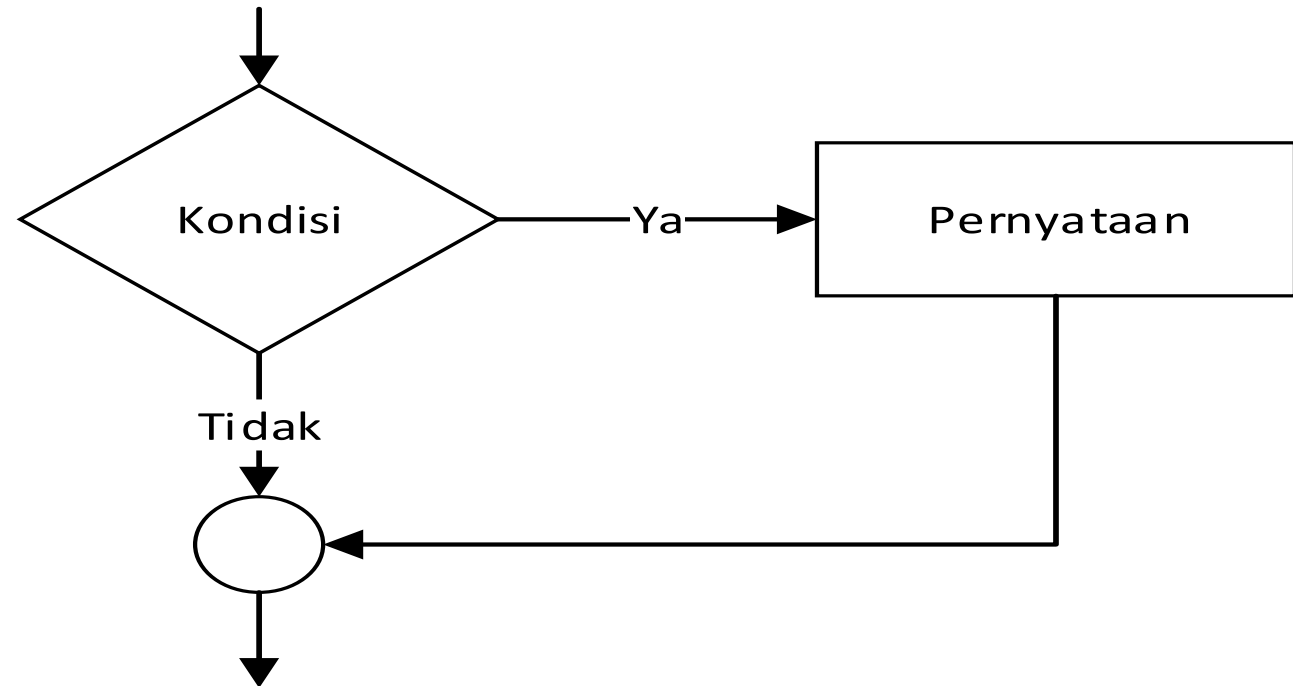
1. IF
2. IF...ELSE
3. IF...ELSE IF...ELSE...
4. SWITCH...CASE

# Sintaks Pemilihan if

```
if (Kondisi)
{
    Pernyataan;
}
```



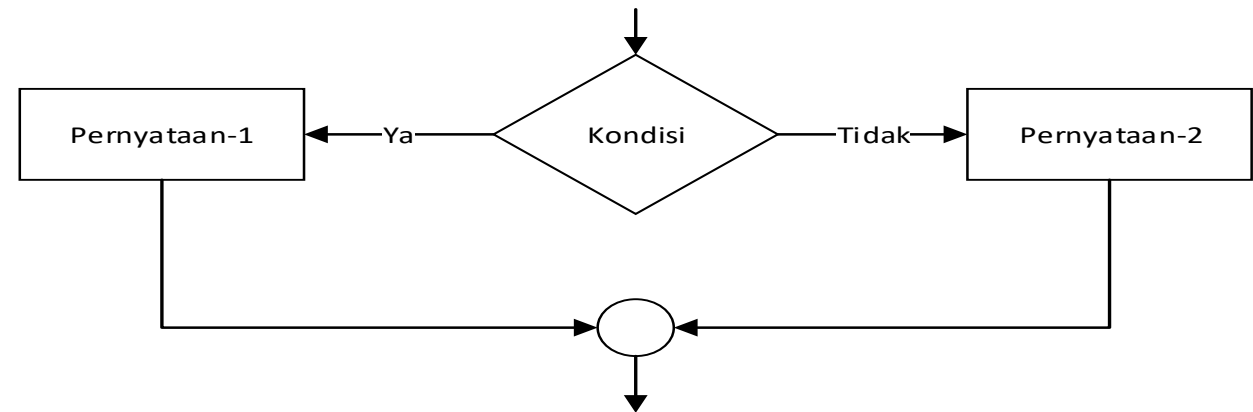
```
if (nilai < 70 ) {
    Print("harus remidi")
}
```



# Sintaks Pemilihan if...else

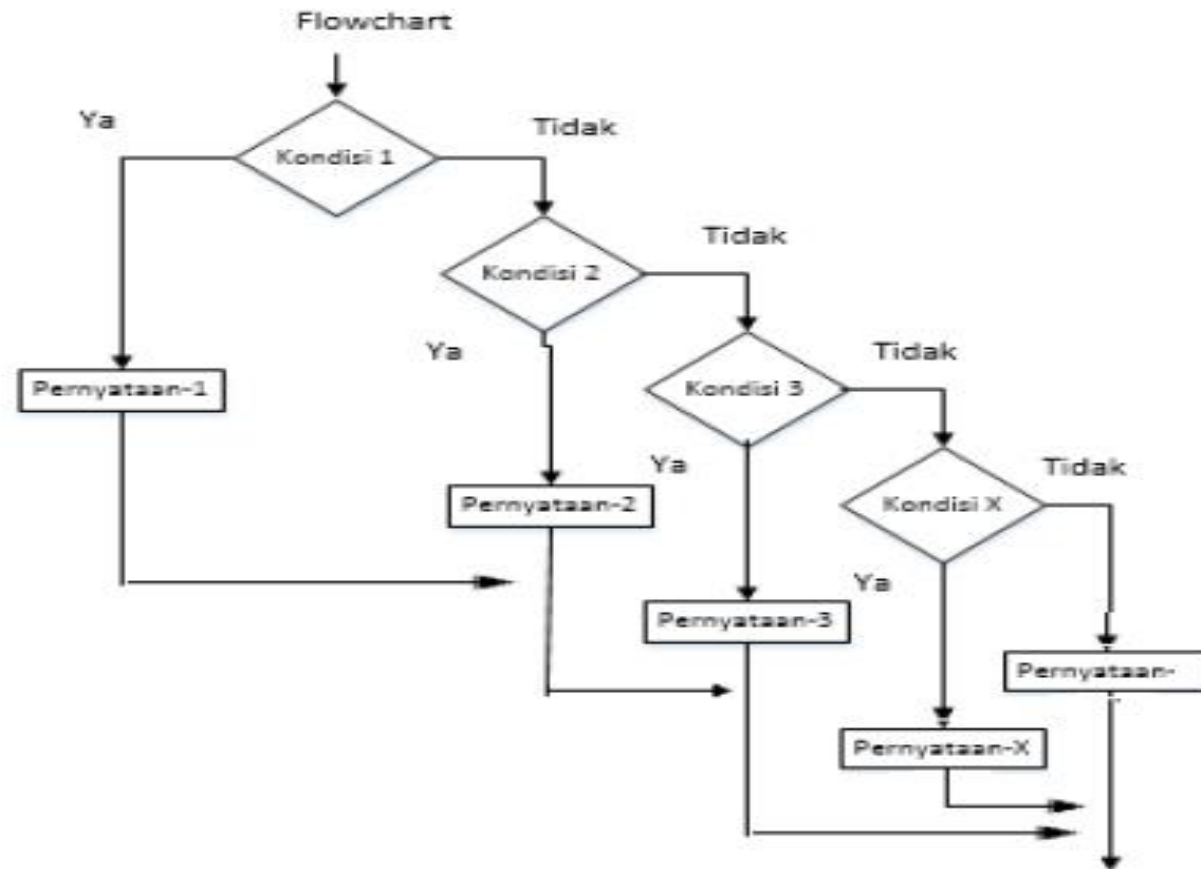
```
if (Kondisi)
{
    Pernyataan-1;
}
else
{
    Pernyataan-2;
}
```

```
if (nilai < 70 ) {
    Print("harus remidi")
}
else{
    Print("tidak remidi")
}
```



# Pemilihan if...else if...else

```
If ( kondisi 1 )  
{  
    pernyataan-1;  
}  
else if ( kondisi 2 )  
{  
    pernyataan-2;  
}  
else if ( kondisi 3 )  
{  
    pernyataan-3;  
}  
.....  
.....  
else if ( kondisi X )  
{  
    pernyataan-X;  
}  
Else  
{  
    pernyataan;  
}
```



# Pemilihan if...else if...else

```
if (nilai > 80 ) {  
    print("excellent")  
}  
else if (nilai > 70) {  
    print("good")  
}  
else {  
    print("poor")  
}
```

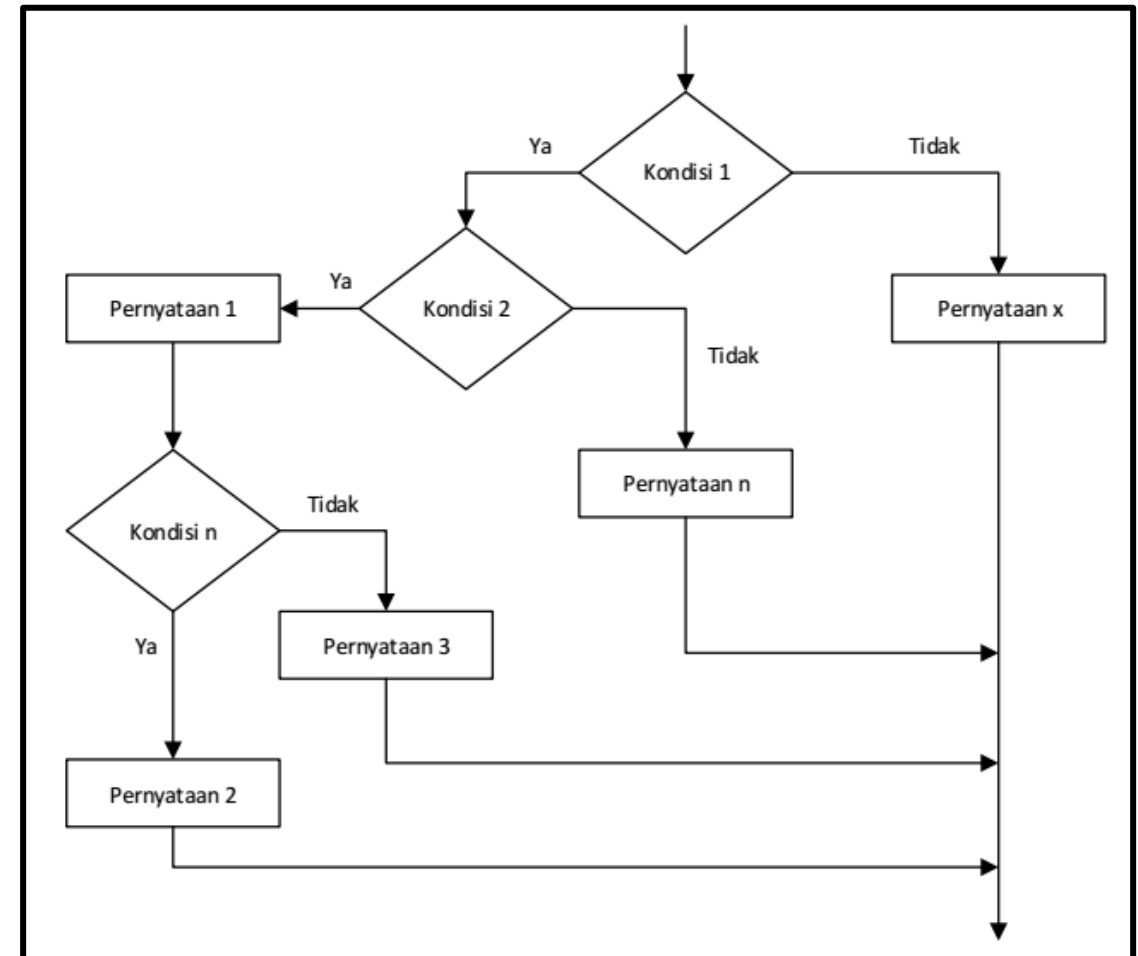


# Switch case

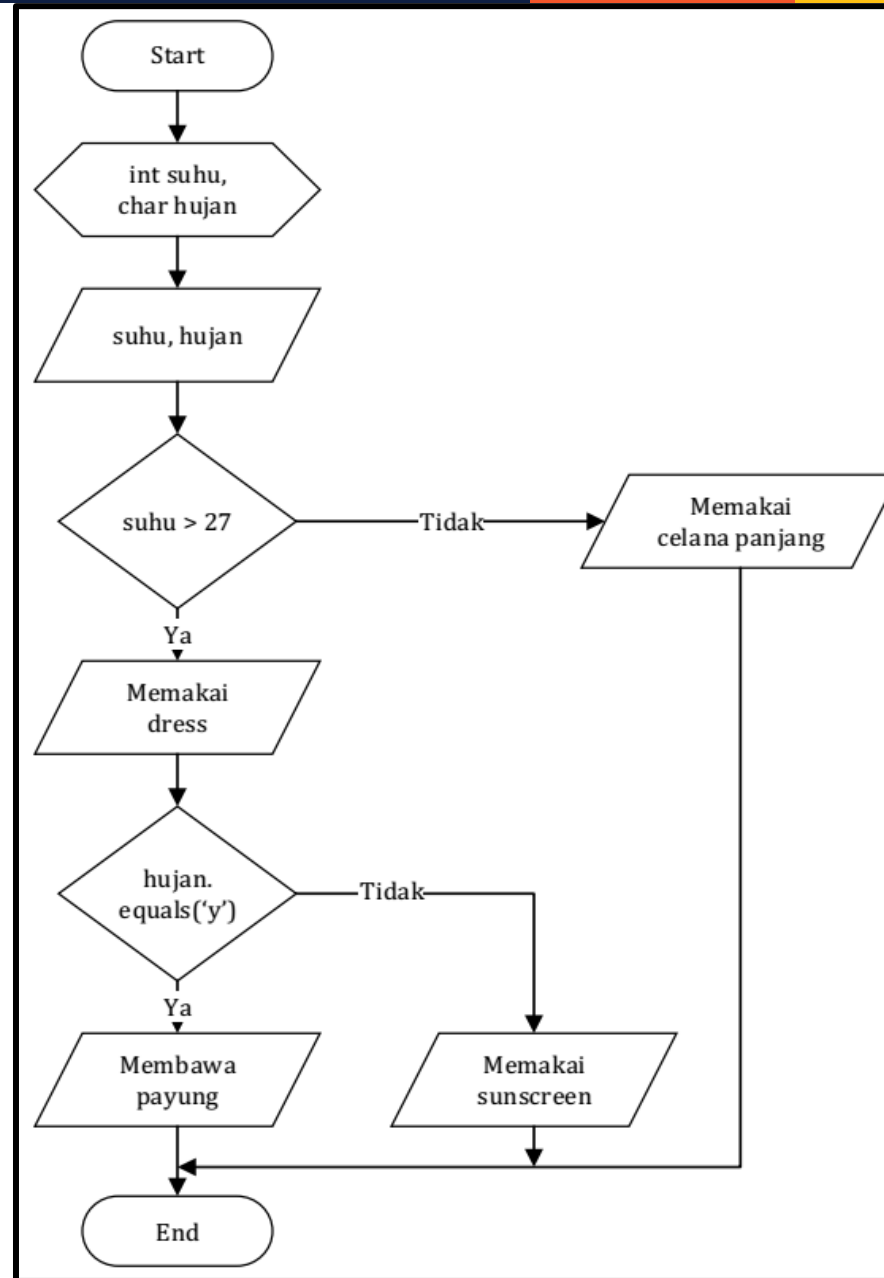
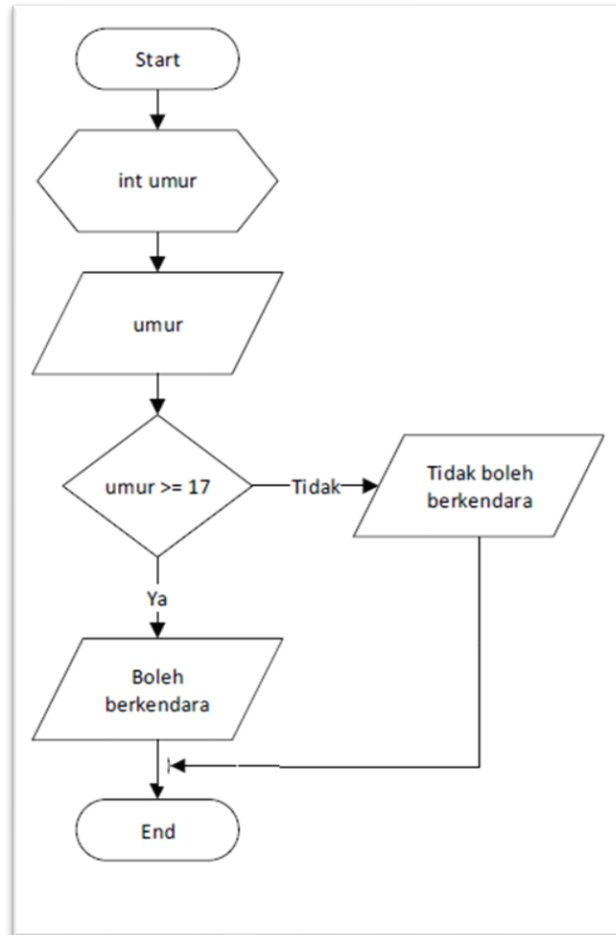
```
switch(platNomor)
{
    case 'L': print("Surabaya");
    break;
    case 'B': print("Jakarta");
    break;
    case 'D': printf("Bandung");
    break;
    default: printf("Karakter tidak diketahui");
}
```

# Pemilihan Bersarang

```
if (kondisi 1){  
    if (kondisi 2){  
        pernyataan 1;  
        ...  
        ...  
        if (kondisi n){  
            pernyataan 2;  
        } else {  
            pernyataan 3;  
        }  
    } else {  
        pernyataan n;  
    }  
} else {  
    pernyataan x;  
}
```

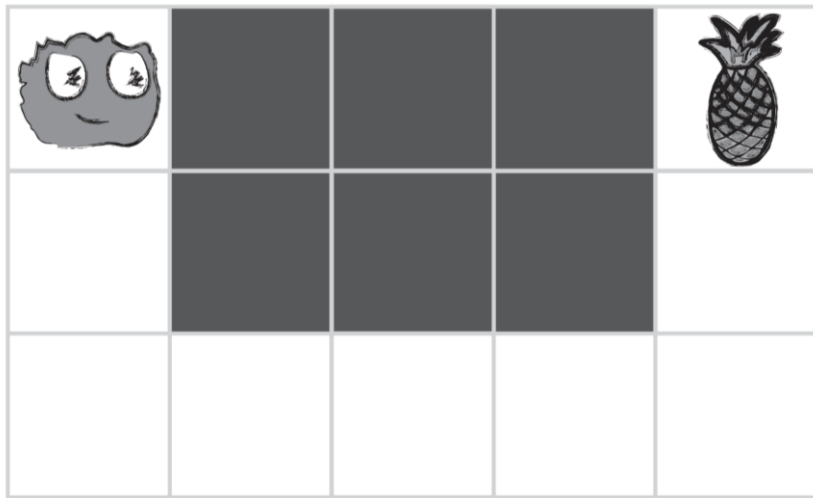


# Contoh flowchart Pemilihan

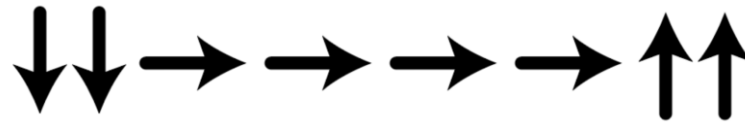


# Perulangan

- Perintah perulangan atau iterasi (loop) adalah perintah untuk mengulang satu atau lebih statement sebanyak beberapa kali
- MONSTER pada gambar berikut harus mencapai tempat NANAS



Instruksi yang dapat dilakukan:

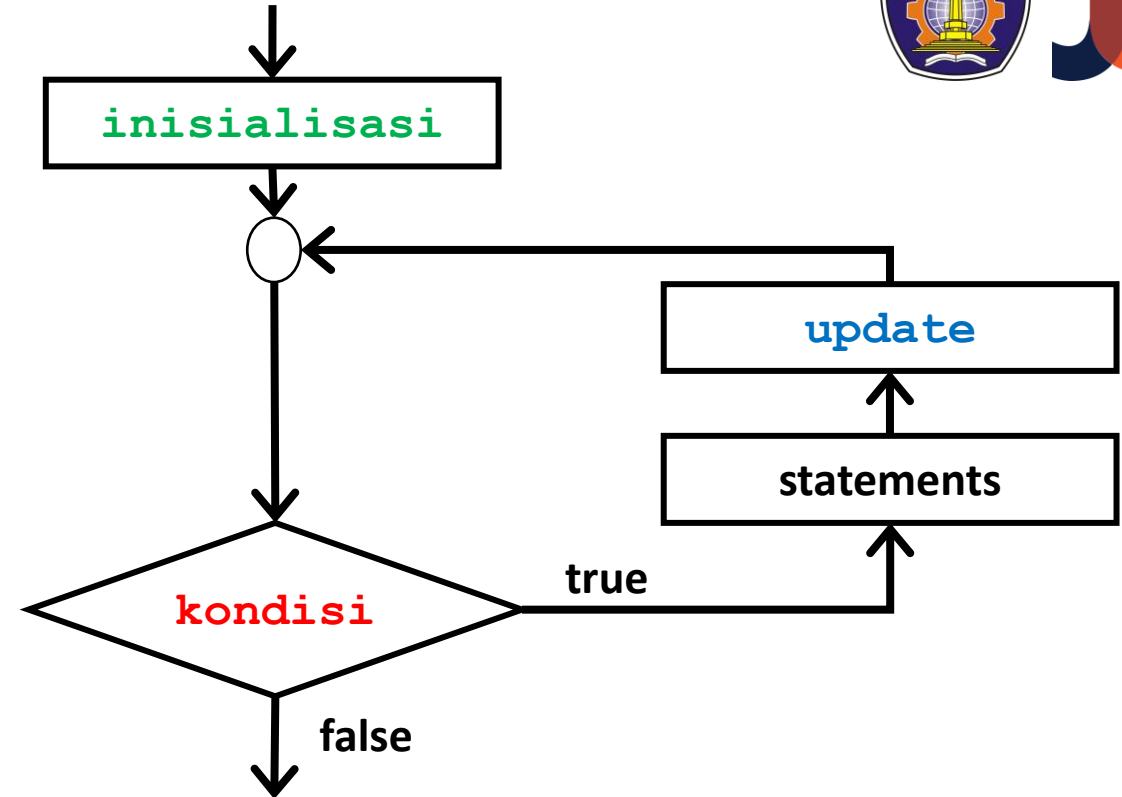


Banyak symbol yang diulang? Bagaimana jika symbol yang diulang dituliskan dengan satu icon symbol?



# Perulangan For

- Loop yang memiliki awal, akhir, dan perubahan nilai
- **inisialisasi**: deklarasi dan inisialisasi variabel counter (variabel pengontrol perulangan)
- **kondisi**: batas atau syarat agar perulangan tetap dieksekusi
- **update**: perubahan nilai variabel counter pada setiap putaran perulangan (increment atau decrement)

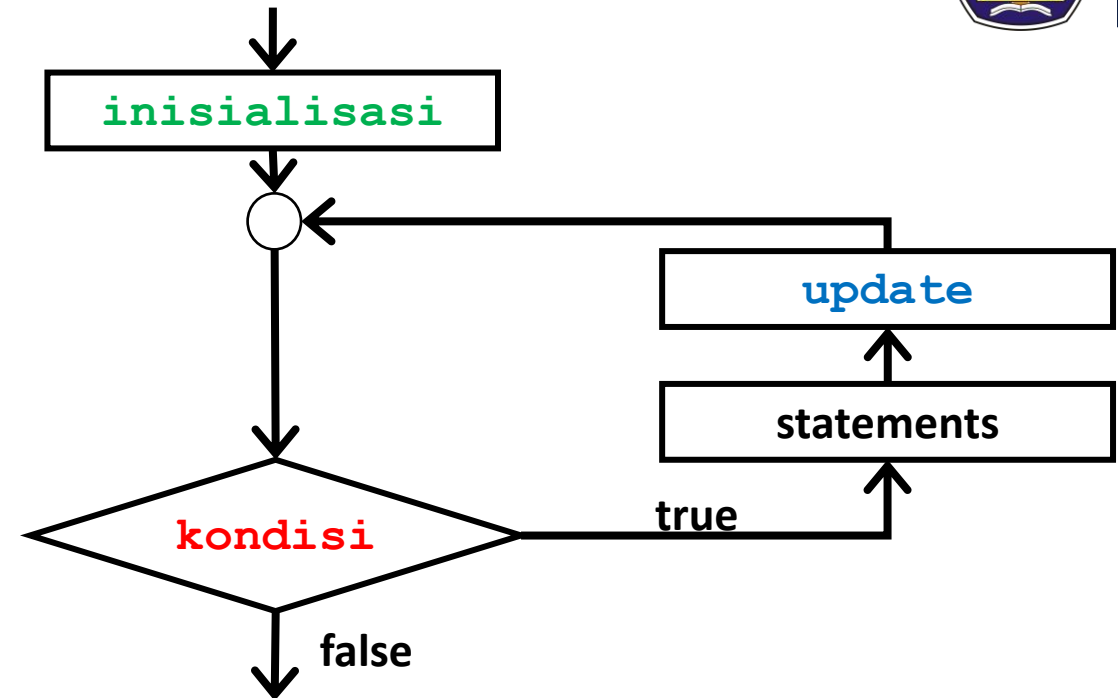


```
for (inisialisasi; kondisi; iterasi)
{
    //statement yang akan diulang
}
```

# Perulangan while

- Syntax dari perintah while():

```
inisialisasi  
while (kondisi) {  
    //statement yang diulang  
    update  
}
```

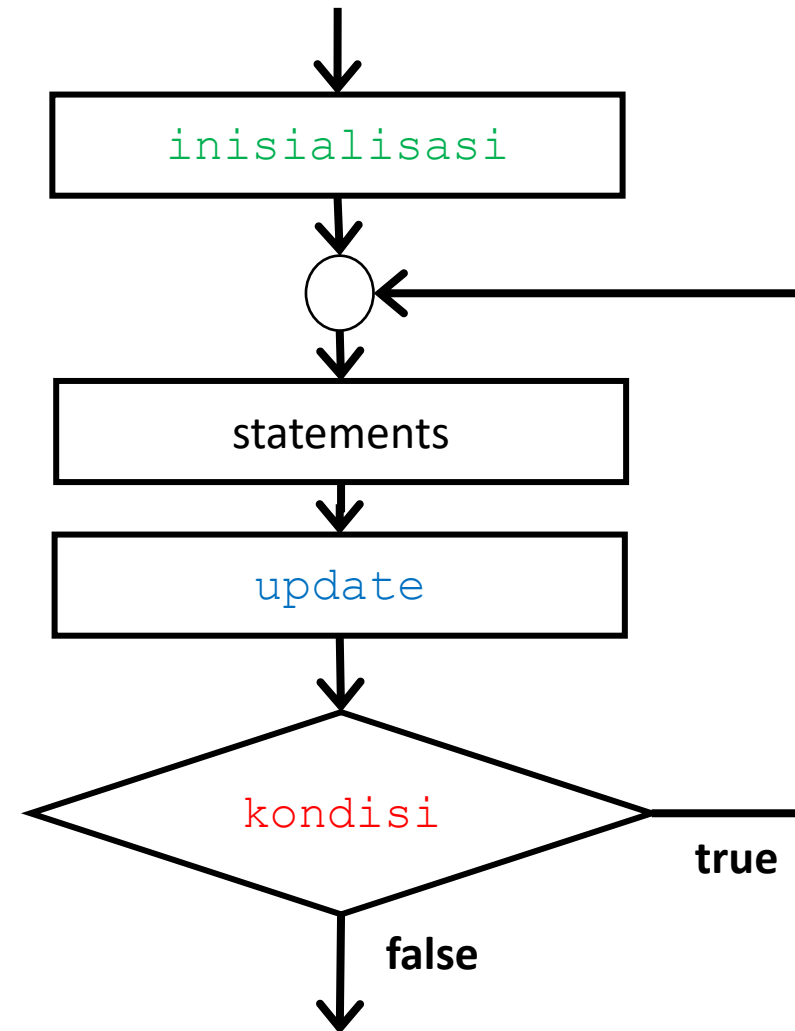


- Syarat perulangan adalah syarat yang harus dipenuhi agar perulangan tetap dilakukan
- Perulangan while akan terus dijalankan selama syarat perulangan bernilai TRUE

# Perulangan Do-while

- Perintah `do-while()` akan menjalankan statementnya sebanyak satu kali, meskipun syarat pengulangan tidak terpenuhi.

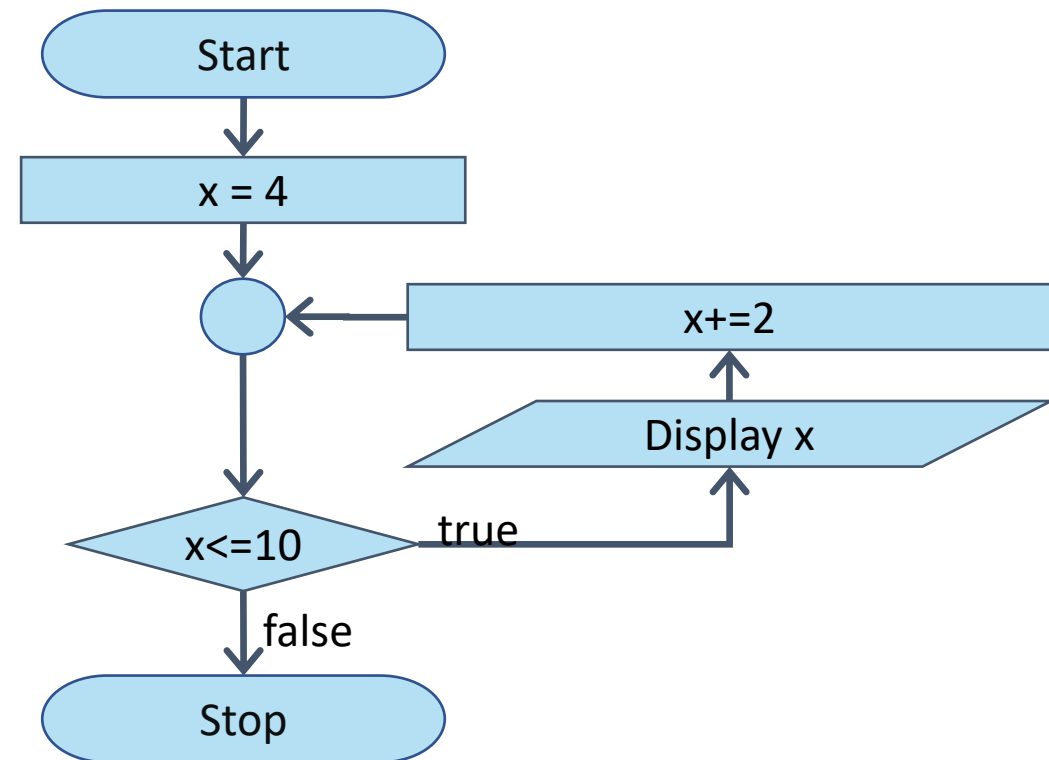
```
inisialisasi  
do {  
    //statement yang akan diulang  
    ...  
    update  
} while (kondisi);
```



# Contoh

```
int x;  
for (x=4; x<=10; x+=2){  
    System.out.println(x);  
}
```

```
int x=4;  
while (x<=10){  
    System.out.println(x);  
    x+=2  
}
```





# Perulangan Bersarang (1)

- “for bersarang”

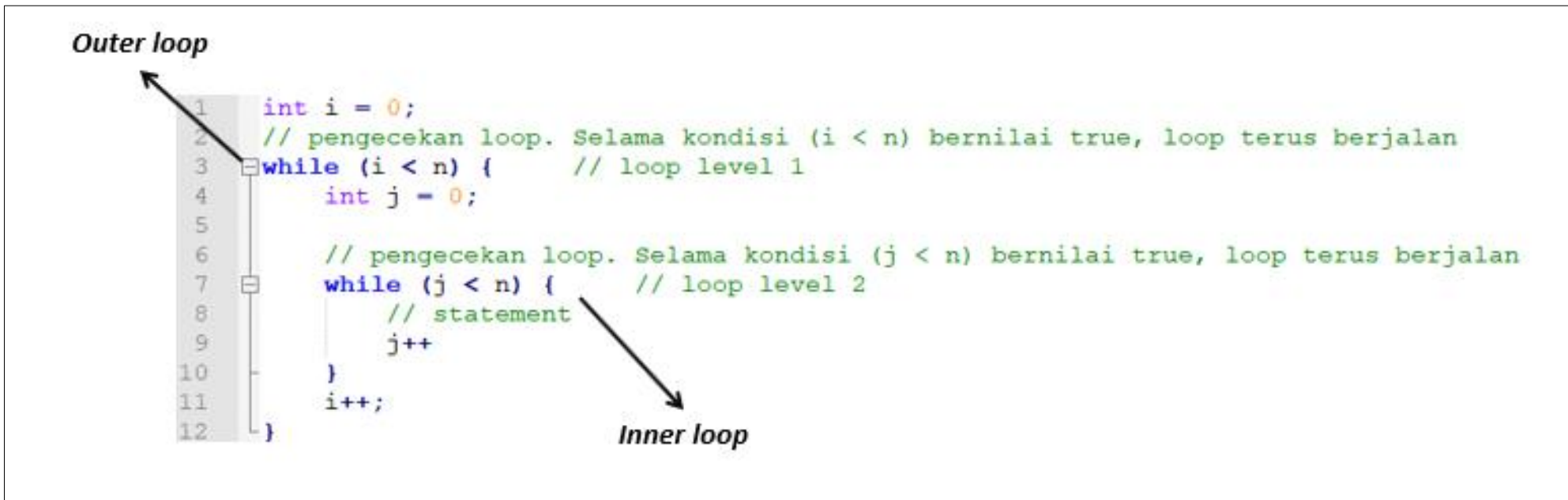
```
for (int i = 0; i < n; i++) {    //loop level 1
    for (int j = 0; j < n; j++) {    //loop level 2
        for (int k = 0; k < n; k++) {    ///loop level 3
            //statement
        }
    }
    for (int j = 0; j < n; j++) {    //loop level 2
    }
}
```

Outer Loop

Inner Loop

# Perulangan Bersarang (2)

- “while bersarang”



# Perulangan Bersarang (3)

- “do-while bersarang”

```
1  int i = 0;
2  do {    // loop level 1
3      int j = 0;
4
5      do {    // loop level 2
6          // statement
7          j++;
8
9          // pengecekan loop. Selama kondisi (j < n) bernilai true, loop terus berjalan
10         } while (j < n);
11         i++;
12
13         // pengecekan loop. Selama kondisi (i < n) bernilai true, loop terus berjalan
14     } while (i < n);
```

Outer loop

Inner loop

# Array Satu Dimensi

## Deklarasi

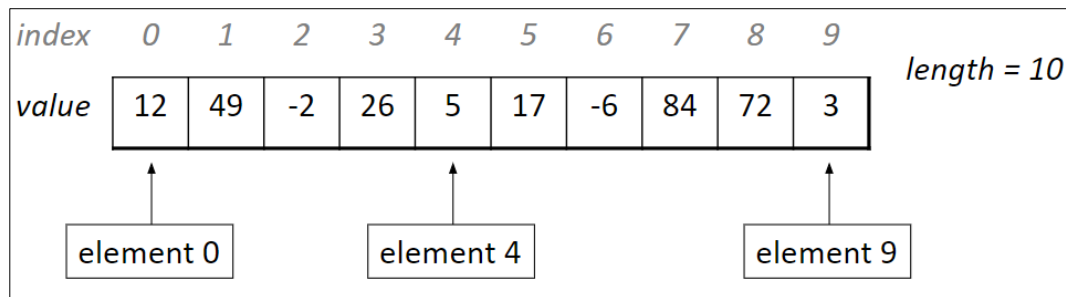
```
type namaArray[];
```

atau

```
type[] namaArray;
```

contoh: `int a[]; int[] a;`

- **Type** adalah tipe data dari array yang akan dibuat.
- **namaArray** adalah nama dari array yang akan dibuat.



# Array Satu Dimensi

- Instansiasi objek array:
  - Ketika sebuah array dideklarasikan, hanya referensi dari array yang dibuat. untuk alokasi memori dilakukan dengan menggunakan kunci kata *new*
  - Cara Instansiasi variabel array:

```
namaArray = new tipe[jumlah elemen];
```

```
contoh:   a = new int[10];
```

# Array Satu Dimensi

- Deklarasi dan instansiasi objek array dapat digabungkan dalam sebuah instruksi sbb.:

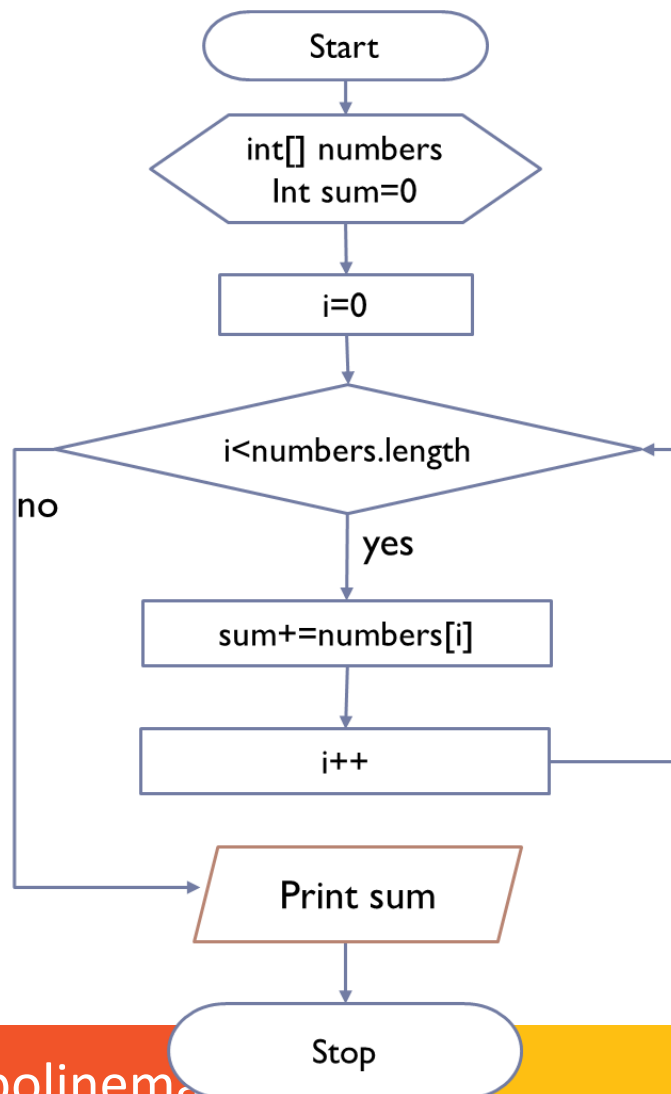
```
type[] namaArray = new type[jumlah_elemen];
```

**atau**

```
type namaArray[] = new type[jumlah_elemen];
```

- Contoh :
  - `int[] a = new int[10];`      atau
  - `int a[] = new int[10];`

# Contoh Array



```
public static void main(String[] args) {  
    int[] numbers = {1,2,3,4,5};  
    int sum = 0;  
    for (int i = 0; i < numbers.length; i++) {  
        sum += numbers[i];  
    }  
    System.out.println(sum);  
}
```

# Array 2 Dimensi

- Array 2 dimensi adalah sebuah array yang penomoran indeksinya menggunakan 2 angka, satu untuk baris dan satu lagi untuk kolom
- Contoh

Baris      Kolom

rating[0][2]=4  
rating[1][3]=3

Rating



Penonton  
(*baris*)

Film(*kolom*)

	0	1	2	3
0	4	3	4	4
1	1	1	2	3
2	1	2	3	4



# Mendeklarasikan Array 2D

- Untuk mendeklarasikan variable array 2D, sama dengan array 1D. Hanya berbeda dengan jumlah kurung sikunya “[]”
- Bentuk umumnya

```
data_type[][] array_name = new data_type[x][y];
```

x = jumlah baris

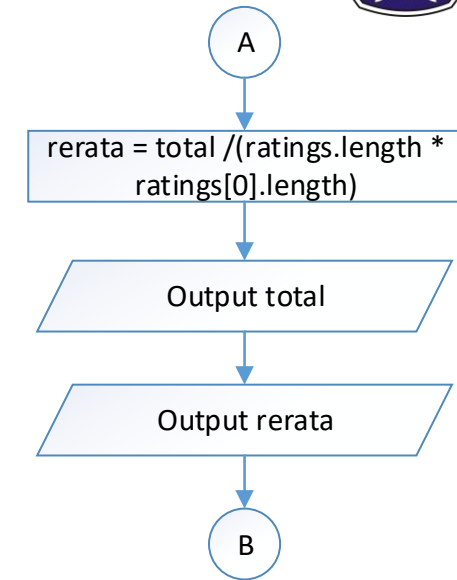
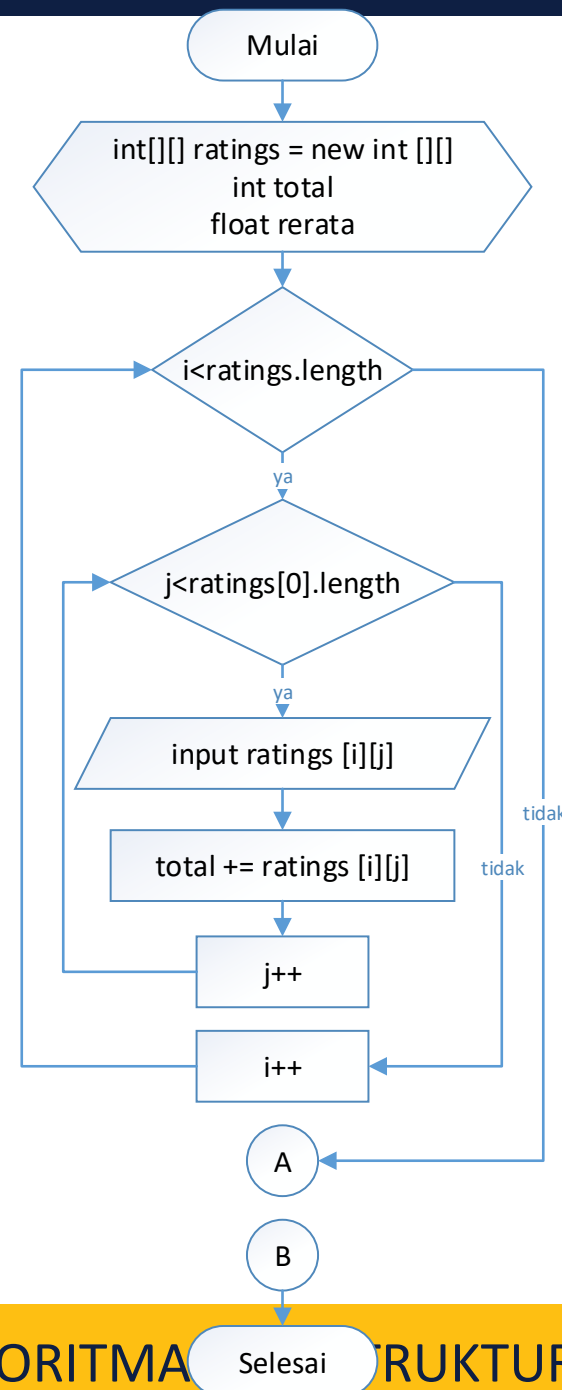
y = jumlah kolom

Contoh:

```
int[][] arr = new int[10][20];
```

# Contoh Flowchart

- Buatlah flowchart untuk menghitung rata-rata rating yang diberikan setiap penonton pada array 2 dimensi pada tabel rating film yang terdiri dari 3 baris (penonton pemberi rating) dan 4 kolom (judul film)!



# Fungsi

```
static TypeDataKembalian namaFungsi() {  
    //statement  
    //statement  
}
```

Keterangan:

- **Static** : Jenis fungsi yang dibuat bersifat *static*, agar dapat secara langsung di panggil di fungsi main yang juga bersifat *static*
- **TypeDataKembalian**: tipe data dari nilai yang dikembalikan (*output*) setelah fungsi dieksekusi
- **namaFungsi()**: nama fungsi yang dibuat

# Contoh Fungsi:

## Pembuatan Fungsi:

```
static void berisalam() {  
    System.out.println("Halo! Selamat Pagi");  
}
```

## Pemanggilan Fungsi:

```
public static void main(String[] args) {  
    berisalam();  
}
```

# Fungsi

- Fungsi **void** tidak memerlukan return.
- Fungsi yang memiliki tipe data fungsi **selain void** memerlukan return.
- **Nilai yang di-return-kan** dari suatu fungsi harus **sesuai dengan tipe data fungsi**.
- **Variabel lokal**: variabel yang dideklarasikan dalam suatu fungsi, dan hanya bisa **diakses** atau dikenali dari **dalam fungsi itu sendiri**.
- **Variabel global**: variabel yang dideklarasikan di **luar blok fungsi**, dan bisa diakses atau **dikenali dari fungsi manapun**.

# Fungsi yang mengembalikan Nilai

**Pembuatan Fungsi dengan parameter dan return value:**

```
static int luasPersegi(int sisi){  
    int luas = sisi * sisi;  
    return luas;  
}
```

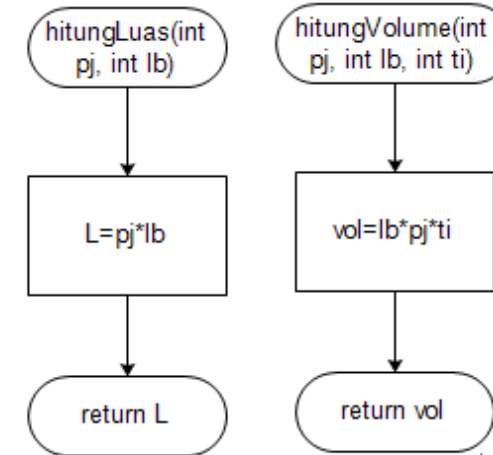
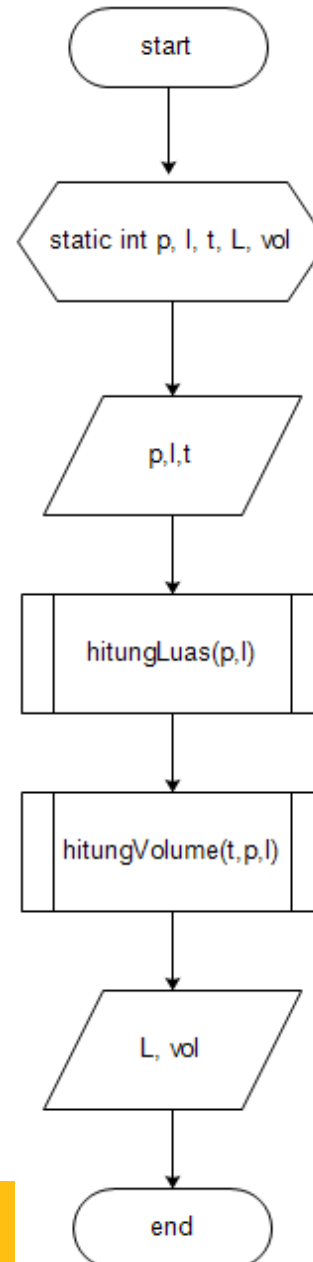
**Pemanggilan Fungsi dan memberi nilai parameter:**

```
System.out.println("Luas Persegi dengan sisi 5 = " + luasPersegi(5));  
  
int luasan = luasPersegi(6);
```

# Contoh Fungsi

- Buatlah flowchart untuk menghitung luas persegi dan volume balok menggunakan fungsi.

Flowchart : main()



Flowchart : hitungLuas (int pj, int lb)

Flowchart : hitungVolume (int pj, int lb, int ti)

# Fungsi Rekursif

- Biasanya sebuah fungsi akan dipanggil (di-CALL) oleh fungsi lain
- Pada fungsi rekursif, di dalam sebuah fungsi terdapat perintah untuk memanggil fungsi itu sendiri (dirinya sendiri). Dengan demikian, proses pemanggilan fungsi akan terjadi secara berulang-ulang
- Bentuk umum:

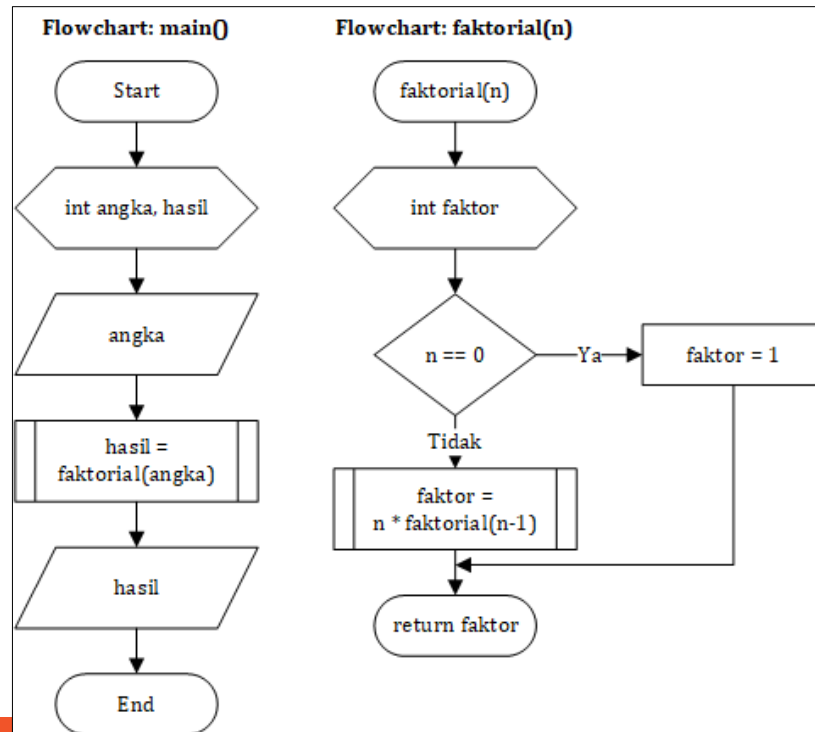
```
static tipe_data_kembalian nama_fungsi (parameter) {  
    ...  
    nama_fungsi (...)  
    ...  
}
```



# Contoh Fungsi Rekursif

## Fungsi faktorial

- Base case:  $n = 0$
- Recursion call:  $f(n) = n * f(n-1)$



```
public class faktorial {  
  
    public static void main(String[] args) {  
        System.out.println(faktorialRekursif(5));  
    }  
  
    static int faktorialRekursif(int n) {  
        if (n == 0) {  
            return 1;  
        } else {  
            return (n * faktorialRekursif(n - 1));  
        }  
    }  
}
```

# LATIHAN

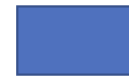
Buatlah *flowchart/pseudocode* untuk menyelesaikan permasalahan berikut ini:

- Menampilkan deretan bilangan dari angka 1 sampai 15 kecuali angka 6 dan 10, angka ganjil dicetak dengan asterik "\*", angka genap dicetak sesuai bilangan aslinya.

Contoh : \* 2 \* 4 \* \* 8 \* \* 12 \* 14 \*

- Permasalahan di bawah ini menggunakan konsep fungsi:
  - Menghitung rata-rata rating untuk setiap movie
  - Mencari movie yang memiliki rata-rata rating paling tinggi dan paling rendah

Rating



Film(*kolom*)

	0	1	2	3
0	4	3	4	4
1	1	1	2	3
2	1	2	3	4

Penonton  
(*baris*)