

# Perulangan 1

Tim Teaching Dasar Pemrograman 2024

# Tujuan Pembelajaran

Di akhir pertemuan, mahasiswa diharapkan mampu :

- Memahami algoritma perulangan (for, while, do-while)
- Memahami statement break dan continue
- Menyelesaikan studi kasus perulangan dengan menggunakan pseudocode/flowchart

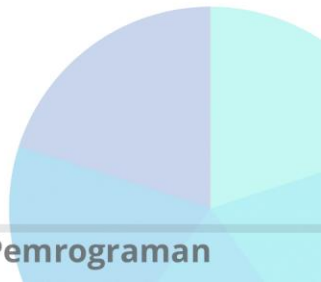
# Definisi Perulangan

- Perintah perulangan atau iterasi (loop) adalah perintah untuk mengulang satu atau lebih statement sebanyak beberapa kali
- Loop statement digunakan agar kita tidak perlu menuliskan satu/sekumpulan statement berulang-ulang. Dengan begitu maka kesalahan pengetikan bisa dikurangi
- Tipe perulangan:
  - Definite loop
  - Indefinite loop



# Tipe Perulangan – Definite Loop

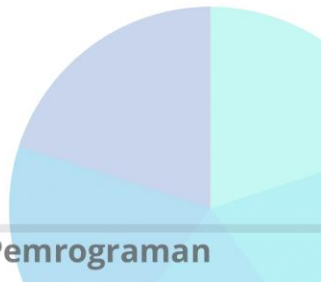
- Perulangan yang jumlah eksekusinya **telah diketahui sebelumnya**
- Biasanya ditandai dengan “ulangi sebanyak \_\_ kali”
- Contoh:
  - Ulangi pernyataan ini untuk setiap bilangan genap antara 8 dan 26
  - Ulangi pernyataan ini untuk setiap mahasiswa di dalam kelas
  - Ulangi pernyataan ini untuk setiap item menu yang tersedia di kafe





# Tipe Perulangan – Indefinite Loop

- Perulangan yang jumlah eksekusinya **tidak dapat ditentukan sebelum dilakukan**
- Perulangan dieksekusi selama kondisi bernilai benar (TRUE), atau sampai kondisi menjadi salah (FALSE)
- Contoh:
  - Ulangi pernyataan ini selama bilangan n bukan bilangan prima
  - Ulangi pernyataan ini selama pengguna belum memilih opsi keluar dari menu
  - Ulangi pernyataan ini selama pelanggan ingin melakukan pemesanan



# Jenis Perintah Perulangan

Dalam bahasa Java, ada 3 macam perintah perulangan yang umum digunakan yaitu:

- Perintah FOR
- Perintah WHILE
- Perintah DO-WHILE



# Konsep Dasar Perulangan

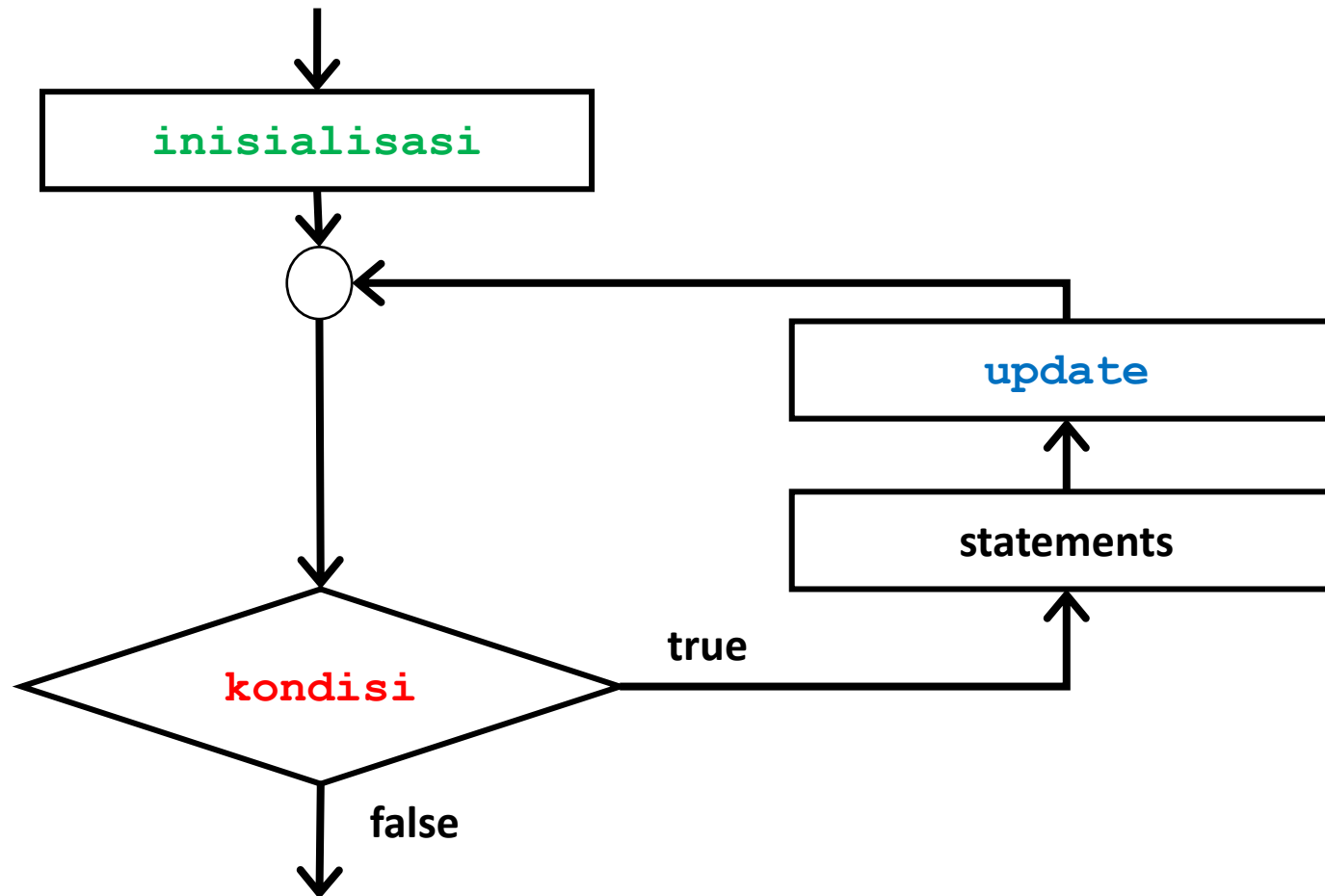
Cetak bilangan 1 sampai 5

<b>Cara 1 :</b> 1. Cetak "1" 2. Cetak "2" 3. Cetak "3" 4. Cetak "4" 5. Cetak "5"	<b>Cara 2 :</b> 1. Int i = 1 2. Cetak i 3. i++ 4. Cetak i 5. i++ 6. Cetak i 7. i++ 8. Cetak i 9. i++ 10. Cetak i 11. i++	<b>Cara 3 :</b> 1. Int i = 1 2. Jika i <= 5 lanjut langkah 3, jika tidak langkah selesai 3. Cetak i 4. i ++ 5. Kembali ke Langkah 2
---	---	--

# Struktur Perulangan FOR



# Flowchart FOR



# Perulangan FOR

- FOR umumnya digunakan pada pengulangan yang jumlah perulangannya sudah pasti atau sudah diketahui sebelumnya
- Sintaks FOR

```
for (inisialisasi; kondisi; update) statement;
```

atau:

```
for (inisialisasi; kondisi; update) {  
    statement1;  
    statement2;  
    .....  
}
```

# Perulangan FOR

- `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)

`inisialisasi` dan `update` bersifat optional (boleh ada atau tidak)

# Alur Perulangan FOR

1. Perulangan diawali dengan melakukan `inisialisasi`
2. Evaluasi `kondisi`
  - Jika kondisi bernilai TRUE, eksekusi semua statement di dalam perulangan. Lakukan `update`. Ulangi kembali langkah nomor 2
  - Jika kondisi bernilai FALSE, hentikan perulangan

# Studi Kasus Perulangan FOR

Buatlah pseudocode dan flowchart untuk memasukkan nilai akhir seorang mahasiswa pada 5 mata kuliah. Kemudian, hitung dan tampilkan nilai rata-rata dari kelima mata kuliah tersebut!

# Pseudocode dan Flowchart FOR

Set matkul = 5

Set total = 0

**FOR** i **FROM** 1 to matkul **DO**

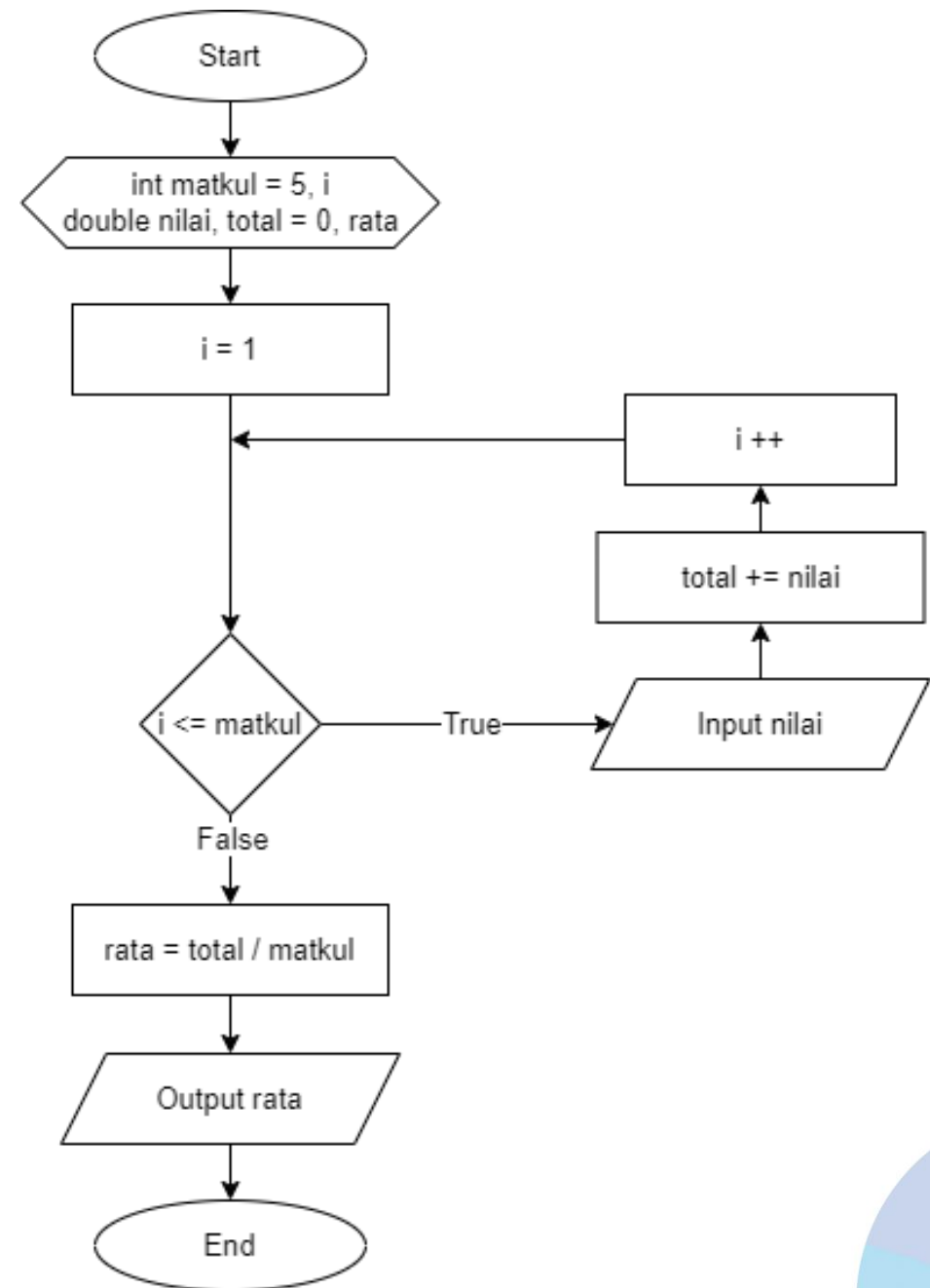
    Input nilai

    Hitung total += nilai

**END FOR**

Hitung rata = total / matkul

Output rata





# Kode Program Perulangan FOR

```
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
    int matkul = 5;  
    double nilai, rata, total = 0;  
    for (int i = 1; i <= matkul; i++) {  
        System.out.print("Masukkan nilai mata kuliah ke-" + i + ": ");  
        nilai = sc.nextDouble();  
        total += nilai;  
    }  
    rata = total / matkul;  
    System.out.println("Rata-rata nilai: " + rata);  
    sc.close();  
}
```

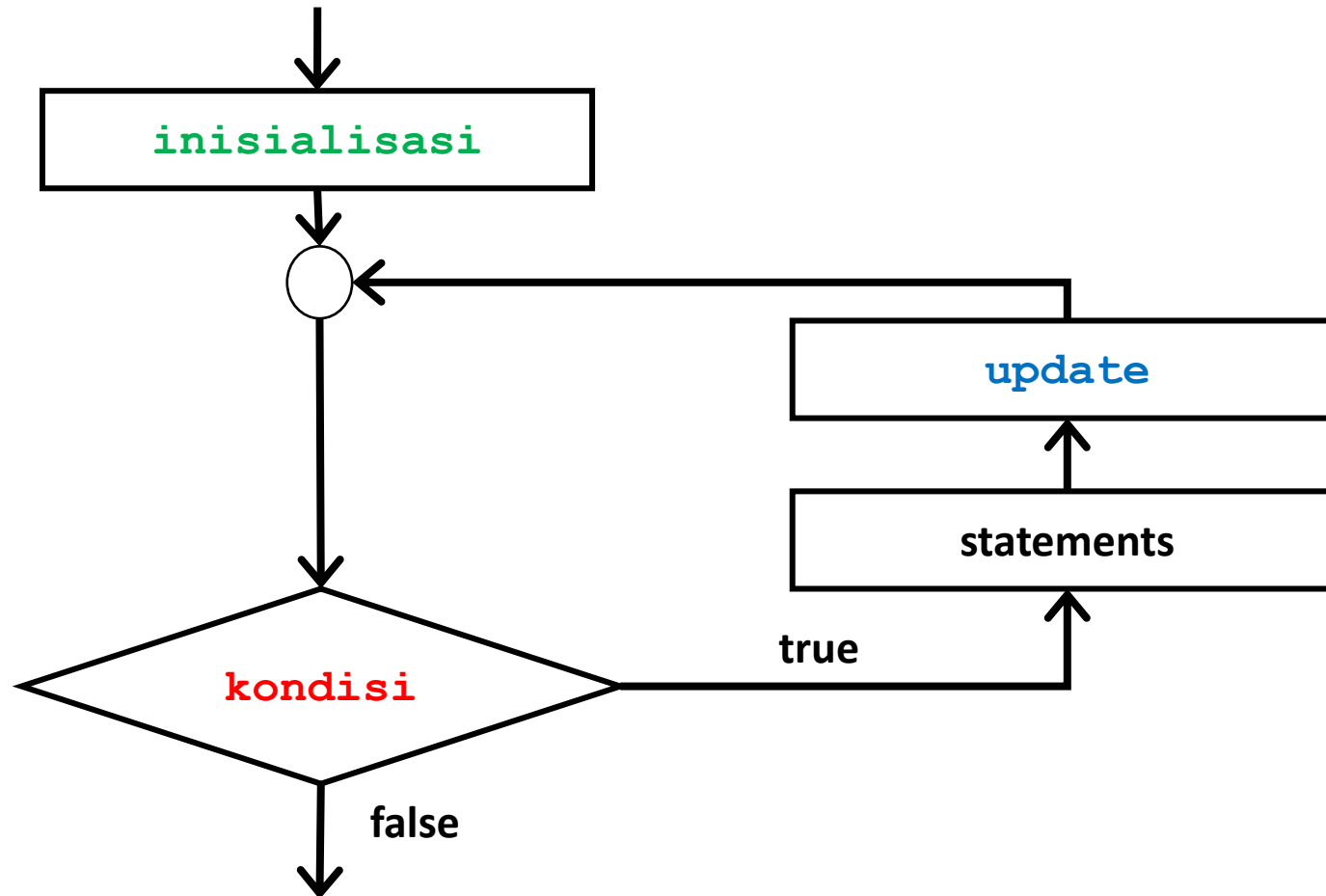
## Output

```
Masukkan nilai mata kuliah ke-1: 87.2  
Masukkan nilai mata kuliah ke-2: 84.2  
Masukkan nilai mata kuliah ke-3: 78.7  
Masukkan nilai mata kuliah ke-4: 73.4  
Masukkan nilai mata kuliah ke-5: 90.8  
Rata-rata nilai: 82.86
```

# Struktur Perulangan **WHILE**



# Flowchart WHILE





# Perulangan WHILE

- WHILE cocok digunakan untuk perulangan yang jumlahnya tidak diketahui sebelumnya (indefinite loop)
- Sintaks WHILE

```
inisialisasi;  
while (kondisi) {  
    statement1;  
    statement2;  
    ...  
    update;  
}
```

Perulangan dengan `while` akan terus dijalankan selama `kondisi` bernilai TRUE

# Perbandingan FOR dan WHILE

## WHILE

```
inisialisasi;  
  
while (kondisi)  
{  
    statement1;  
    statement2;  
    ...  
    update  
}
```

setara

## FOR

```
for (inisialisasi; kondisi; update)  
{  
    statement1;  
    statement2;  
    ...  
}
```

Contoh:

```
int x = 1;  
while (x <= 10) {  
    _____  
    _____  
    x++;  
}
```

sama

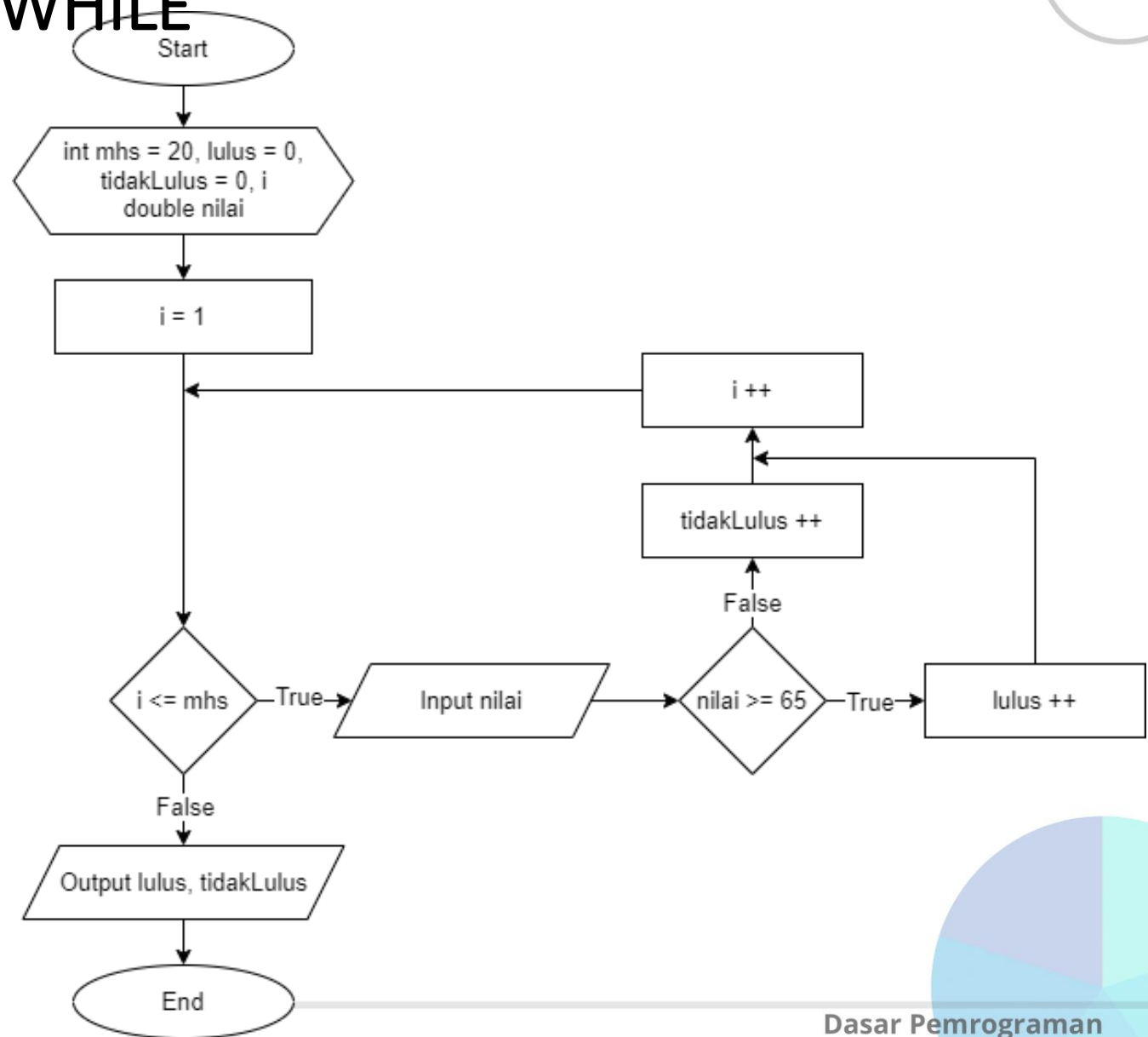
```
for (int x = 1 ; x <= 10 ; x++){  
    _____  
    _____  
}
```

# Studi Kasus Perulangan WHILE

Buatlah pseudocode dan flowchart untuk memasukkan nilai 20 orang mahasiswa dalam satu kelas dan menghitung banyaknya mahasiswa yang lulus (nilai akhir  $\geq 65$ ) dan tidak lulus. Kemudian, tampilkan banyaknya mahasiswa yang lulus dan tidak lulus tersebut!

# Pseudocode dan Flowchart WHILE

```
Set mhs = 20
Set lulus = 0
Set tidakLulus = 0
Set i = 1
WHILE i <= mhs DO
    Input nilai
    IF nilai >= 65 THEN
        lulus ++
    ELSE
        tidakLulus ++
    END IF
    i ++
END WHILE
Output lulus, tidakLulus
```





# Kode Program Perulangan WHILE

```
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
    int mhs = 20, lulus = 0, tidakLulus = 0, i;  
    double nilai;  
    i = 1;  
    while (i <= mhs) {  
        System.out.print("Masukkan nilai mahasiswa " + i + ": ");  
        nilai = sc.nextDouble();  
        if (nilai >= 65) {  
            lulus++;  
        } else {  
            tidakLulus++;  
        }  
        i++;  
    }  
    System.out.println("Banyaknya mahasiswa lulus: " + lulus);  
    System.out.println("Banyaknya mahasiswa tidak lulus: " + tidakLulus);  
    sc.close();  
}
```

## Output

```
Masukkan nilai mahasiswa 1: 89.2  
Masukkan nilai mahasiswa 2: 70.5  
Masukkan nilai mahasiswa 3: 64.4  
Masukkan nilai mahasiswa 4: 60.2  
Masukkan nilai mahasiswa 5: 90.3  
Masukkan nilai mahasiswa 6: 90.5  
Masukkan nilai mahasiswa 7: 81.7  
Masukkan nilai mahasiswa 8: 69.2  
Masukkan nilai mahasiswa 9: 50.5  
Masukkan nilai mahasiswa 10: 55.4  
Masukkan nilai mahasiswa 11: 63.8  
Masukkan nilai mahasiswa 12: 79.9  
Masukkan nilai mahasiswa 13: 81.2  
Masukkan nilai mahasiswa 14: 86.4  
Masukkan nilai mahasiswa 15: 95.3  
Masukkan nilai mahasiswa 16: 98.1  
Masukkan nilai mahasiswa 17: 87.4  
Masukkan nilai mahasiswa 18: 64.3  
Masukkan nilai mahasiswa 19: 60.8  
Masukkan nilai mahasiswa 20: 71.7  
Banyaknya mahasiswa lulus: 13  
Banyaknya mahasiswa tidak lulus: 7
```

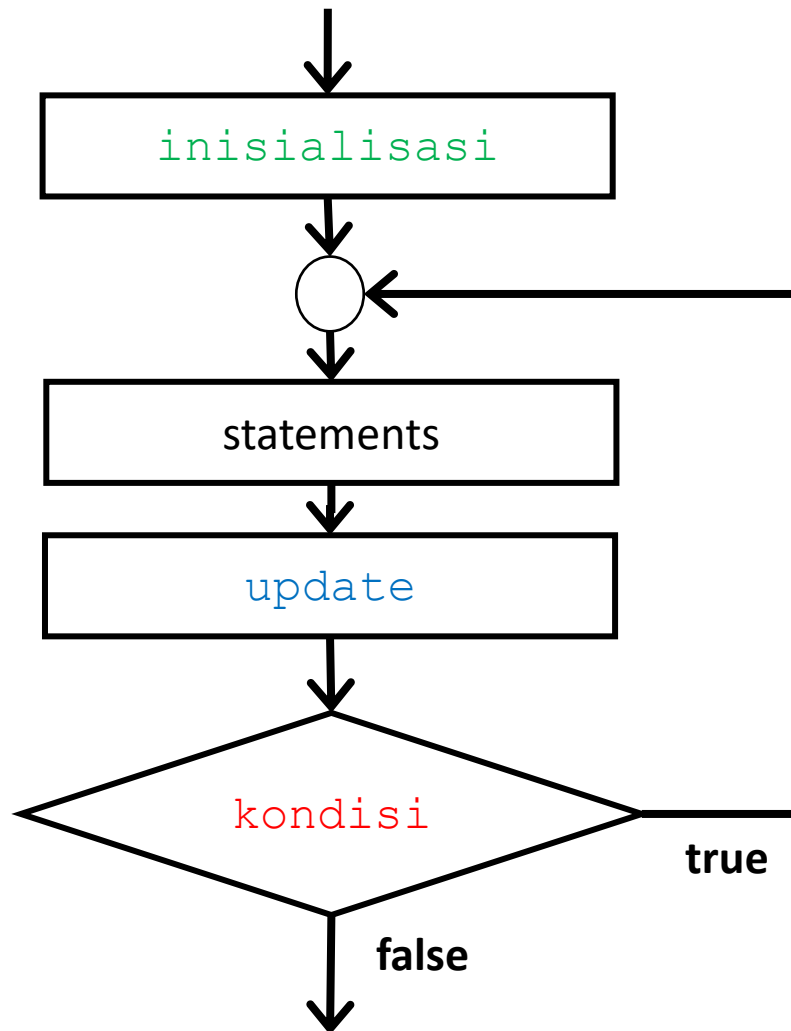
# Struktur Perulangan DO-WHILE

# Perulangan DO-WHILE

- Perintah DO-WHILE hampir sama dengan perintah WHILE
- Perbedaannya:
  - DO-WHILE **mengeksekusi statementnya terlebih dahulu**, lalu mengevaluasi kondisi
  - WHILE **mengevaluasi kondisi** sebelum mengeksekusi statement
- Oleh karena itu, perintah DO-WHILE akan mengeksekusi block statement **minimal 1 kali**, meskipun kondisi **tidak terpenuhi**



# Flowchart DO-WHILE





# Perulangan DO-WHILE

- Sintaks DO-WHILE

```
Inisialisasi;  
do {  
    statement1;  
    statement2;  
    ...  
    update;  
} while (kondisi);
```

Eksekusi statement minimal 1 kali.  
Selama **kondisi** bernilai TRUE,  
maka perulangan akan terus dijalankan

# Studi Kasus Perulangan DO-WHILE

Buatlah pseudocode dan flowchart untuk memasukkan harga minuman yang dibeli di kafe. Kasir terus menerus memasukkan harga minuman hingga selesai (input harga = 0 untuk berhenti). Hitung total pembayaran dan tampilkan hasilnya!

# Pseudocode dan Flowchart DO-WHILE

Set total = 0

Set i = 1

**DO**

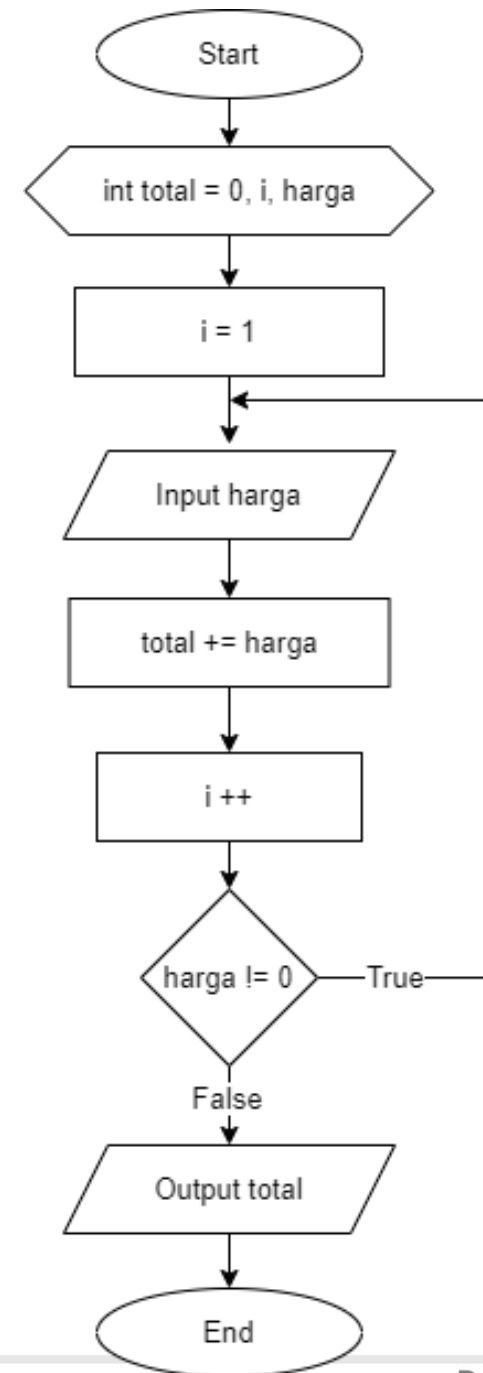
Input harga

total += harga

i ++

**WHILE** harga != 0

Output total





# Kode Program Perulangan DO-WHILE

```
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
    int total = 0, harga, i;  
    i = 1;  
    do {  
        System.out.print("Masukkan harga ke-" + i + ": ");  
        harga = sc.nextInt();  
        System.out.println("Harga ke-" + i + " sebesar " + harga);  
        total += harga;  
        i++;  
    } while (harga != 0);  
    System.out.println("Total harga: " + total);  
    sc.close();  
}
```

## Output

Masukkan harga ke-1: 15000  
Harga ke-1 sebesar 15000  
Masukkan harga ke-2: 23000  
Harga ke-2 sebesar 23000  
Masukkan harga ke-3: 7500  
Harga ke-3 sebesar 7500  
Masukkan harga ke-4: 0  
Harga ke-4 sebesar 0  
Total harga: 45500

## Output

Masukkan harga ke-1: 0  
Harga ke-1 sebesar 0  
Total harga: 0

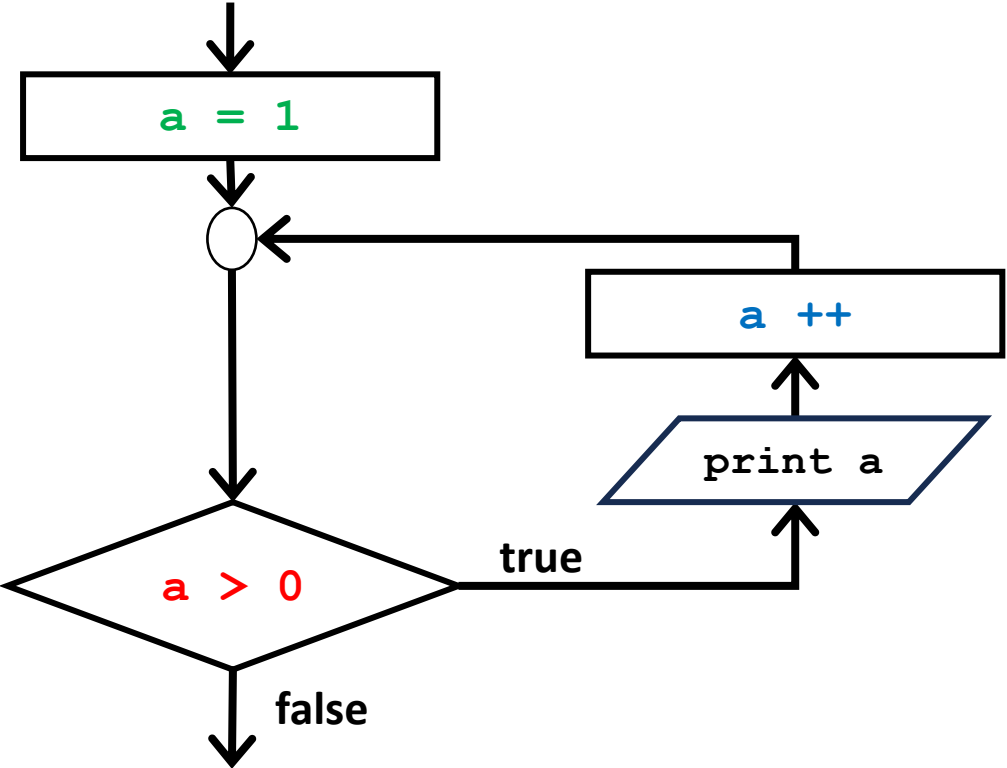
# Infinite Loop

# Infinite Loop

- Saat melakukan eksekusi statement di dalam perulangan, harus terdapat kondisi yang menjadikan **kondisi** bernilai FALSE
- Jika tidak ada (**kondisi** terus menerus bernilai TRUE), maka hal ini disebut **infinite loop**, yaitu perulangan yang akan dijalankan terus menerus tanpa batas sampai pengguna menghentikan program
- Logika program harus selalu diperiksa ulang untuk memastikan bahwa loop akan berakhir



# Flowchart Infinite Loop







# Contoh Kode Program Infinite Loop

```
public static void main(String[] args) {  
    int n = 1;  
    while (n >= 0) {  
        System.out.println("Pesanan #" + n + " diproses");  
        n++;  
    }  
    System.out.println(x: "Transaksi selesai");  
}
```

Loop akan berjalan terus menerus sampai pengguna memaksa menghentikannya

## Output

```
Pesanan #113025 diproses  
Pesanan #113026 diproses  
Pesanan #113027 diproses  
Pesanan #113028 diproses  
Pesanan #113029 diproses  
Pesanan #113030 diproses  
Pesanan #113031 diproses  
Pesanan #113032 diproses  
Pesanan #113033 diproses  
Pesanan #113034 diproses  
Pesanan #113035 diproses  
Pesanan #113036 diproses  
Pesanan #113037 diproses  
Pesanan #113038 diproses  
Pesanan #113039 diproses  
...
```

# Penghentian Perulangan dalam Program Interaktif

# Penghentian Perulangan

Untuk menghentikan pengulangan dalam program interaktif, alternatif yang dapat dilakukan antara lain:

- Menambahkan **Sentinel** atau pembatas dengan kode khusus
- Menambahkan **Pertanyaan** sebagai penentu dilanjutkan atau tidaknya perulangan

# Menambahkan Sentinel

- **Sentinel** adalah nilai yang menandakan **akhir dari input pengguna**
- **Sentinel loop** menyatakan perulangan yang akan terus berjalan sampai nilai sentinel ditemukan



# Contoh Kode Program Penerapan Sentinel

```
Scanner sc = new Scanner(System.in);
int totalNilai = 0, mhs = 0, nilai;
do {
    System.out.print(s:"Masukkan nilai di luar rentang (0-100) untuk berhenti: ");
    nilai = sc.nextInt();
    if (nilai >= 0 && nilai <= 100) {
        totalNilai += nilai;
        mhs++;
    }
} while (nilai >= 0 && nilai <= 100);
if (mhs > 0) {
    double rataRata = (double) totalNilai / mhs;
    System.out.println("Rata-rata nilai: " + rataRata);
} else {
    System.out.println(x:"Tidak ada nilai yang valid dimasukkan.");
}
sc.close();
```

**Sentinel**

## Output

Masukkan nilai di luar rentang (0-100) untuk berhenti: 60  
Masukkan nilai di luar rentang (0-100) untuk berhenti: 73  
Masukkan nilai di luar rentang (0-100) untuk berhenti: 105  
Rata-rata nilai: 66.5

# Menambahkan Pertanyaan

- **Pertanyaan** digunakan untuk memberikan pilihan kepada pengguna apakah pengguna masih akan melanjutkan perulangan
- Apabila **kondisi** pada perulangan bernilai TRUE berdasarkan jawaban pertanyaan dari pengguna, maka perulangan dilanjutkan
- Contoh:
  - Apakah Anda akan melanjutkan perulangan?
  - Apakah Anda akan menambahkan barang baru?



# Contoh Kode Program Penerapan Pertanyaan

```
public static void main(String[] args) {  
    Scanner input = new Scanner(System.in);  
    String minuman, pilihan;  
    int count = 0;  
    do {  
        System.out.print(s:"Masukkan nama minuman: ");  
        minuman = input.nextLine();  
        System.out.println(minuman + " dipesan");  
        count++;  
        System.out.print(s:"Apakah ada minuman lain yang dipesan? (ya/tidak): ");  
        pilihan = input.nextLine();  
    } while (pilihan.equalsIgnoreCase(anotherString:"ya"));  
    System.out.println("Total variasi minuman yang dibeli: " + count);  
    input.close();  
}
```

## Output

```
Masukkan nama minuman: matcha frappe  
matcha frappe dipesan  
Apakah ada minuman lain yang dipesan? (ya/tidak): YA  
Masukkan nama minuman: milk tea  
milk tea dipesan  
Apakah ada minuman lain yang dipesan? (ya/tidak): Ya  
Masukkan nama minuman: smoothies  
smoothies dipesan  
Apakah ada minuman lain yang dipesan? (ya/tidak): Tidak  
Total variasi minuman yang dibeli: 3
```

## Pertanyaan

# Statement **BREAK dan CONTINUE**



# Statement BREAK

- BREAK digunakan untuk keluar dari perulangan sepenuhnya, **menghentikan** eksekusi loop meskipun kondisinya belum terpenuhi.
- Setelah BREAK, program akan melanjutkan ke pernyataan setelah perulangan.
- Selain digunakan untuk keluar dari SWITCH (pemilihan SWITCH-CASE), BREAK juga digunakan untuk keluar dari perulangan (FOR, WHILE, dan DO-WHILE)

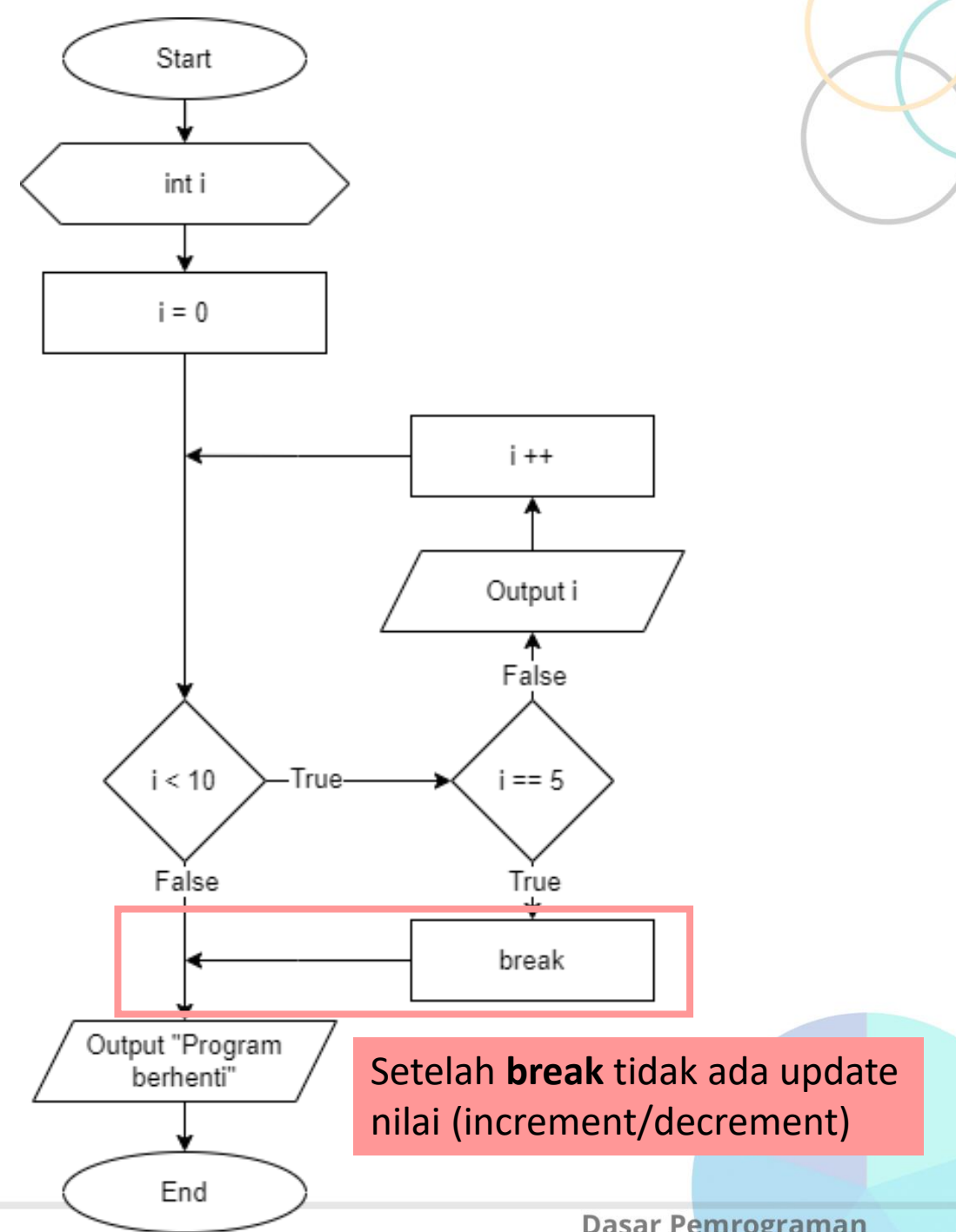
# Contoh Penggunaan BREAK

```
for (int i = 0; i < 10; i++) {  
    if (i == 5) {  
        break;  
    }  
    System.out.print(i + " ");  
}  
System.out.println("\nProgram berhenti");
```

Keluar dari loop

## Output

0 1 2 3 4  
Program berhenti



# Statement CONTINUE

- CONTINUE digunakan untuk **melewati (*skip*)** iterasi saat ini (1 iterasi saja) dan melanjutkan ke iterasi berikutnya.
- Setelah CONTINUE, program tidak akan menjalankan perintah di bawahnya dalam loop, tetapi akan langsung melanjutkan ke iterasi berikutnya.

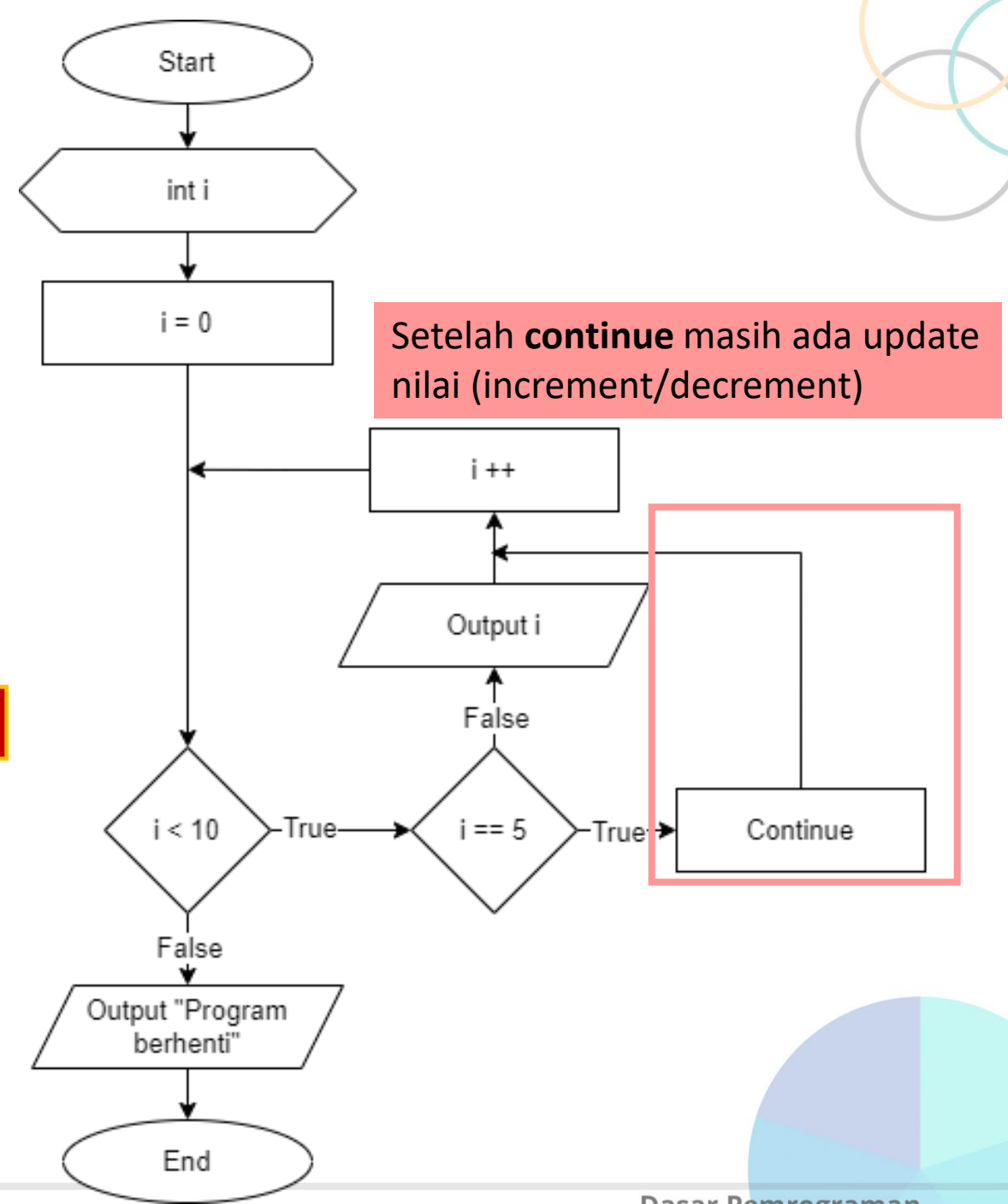
# Contoh Penggunaan CONTINUE

```
for (int i = 0; i < 10; i++) {  
    if (i == 5) {  
        continue;  
    }  
    System.out.print(i + " ");  
}  
System.out.println("\nProgram berhenti");
```

Skip perulangan sesuai kondisi

## Output

0 1 2 3 4 6 7 8 9  
Program berhenti



# LATIHAN

# Latihan 1

Buatlah flowchart untuk melacak waktu belajar siswa di rumah. Setiap hari siswa memasukkan jumlah jam belajar harian mereka. Program harus menghitung dan menampilkan total jam belajar selama seminggu. Jika siswa belajar lebih dari 10 jam dalam sehari, program harus mengabaikan input tersebut dan meminta input ulang, karena dianggap tidak mungkin.

*Catatan: Perulangan dapat menggunakan for, while, atau do-while. Penambahan break atau continue jika diperlukan*

## Latihan 2

Buatlah flowchart untuk menentukan bonus yang diterima pegawai perusahaan. Setiap pegawai yang bekerja lebih dari 8 jam per hari akan menerima bonus sebesar 20% dari gaji harian. Program akan meminta input jumlah jam kerja dan gaji harian pegawai, lalu menghitung bonusnya. Program akan berhenti jika pengguna memasukkan jam kerja negatif, dan pegawai tersebut tidak diproses.

*Catatan: Perulangan dapat menggunakan for, while, atau do-while.  
Penambahan break atau continue jika diperlukan*