Perulangan 1

Tim Ajar Dasar Pemrograman 2023

Tujuan

Di akhir pertemuan, mahasiswa diharapkan mampu :

- Memahami algoritma perulangan (for, while, do-while)
- Memberikan contoh sederhana perulangan
- Menggambarkan permasalahan studi kasus perulangan dengan menggunakan 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 sebanyak n kali
- Ulangi pernyataan ini untuk setiap bilangan genap antara 8 dan 26

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 sampai pengguna memasukkan bilangan bulat yang valid

Jenis Perintah Perulangan

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

- Perintah WHILE
- Perintah DO-WHILE

Struktur Perulangan 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

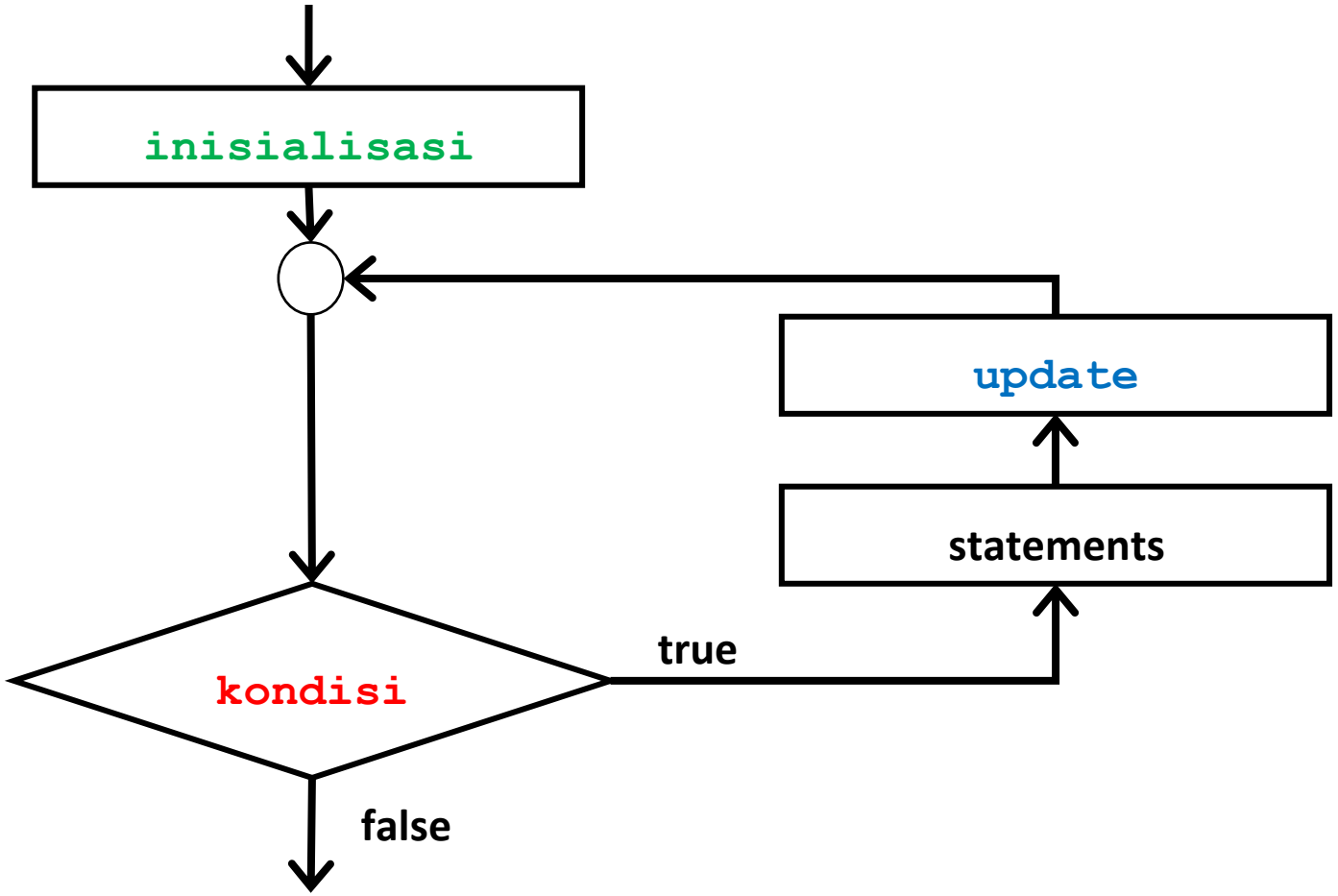
- `inisialisasi`: deklarasi dan inisialisasi variabel counter (variabel pengontrol perulangan)
- `kondisi`: batas atau syarat agar perulangan tetap dieksekusi
- `update`: perubahan nilai variabel counter pada setiap putaran perulangan (increment atau decrement)

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

Alur Perulangan FOR

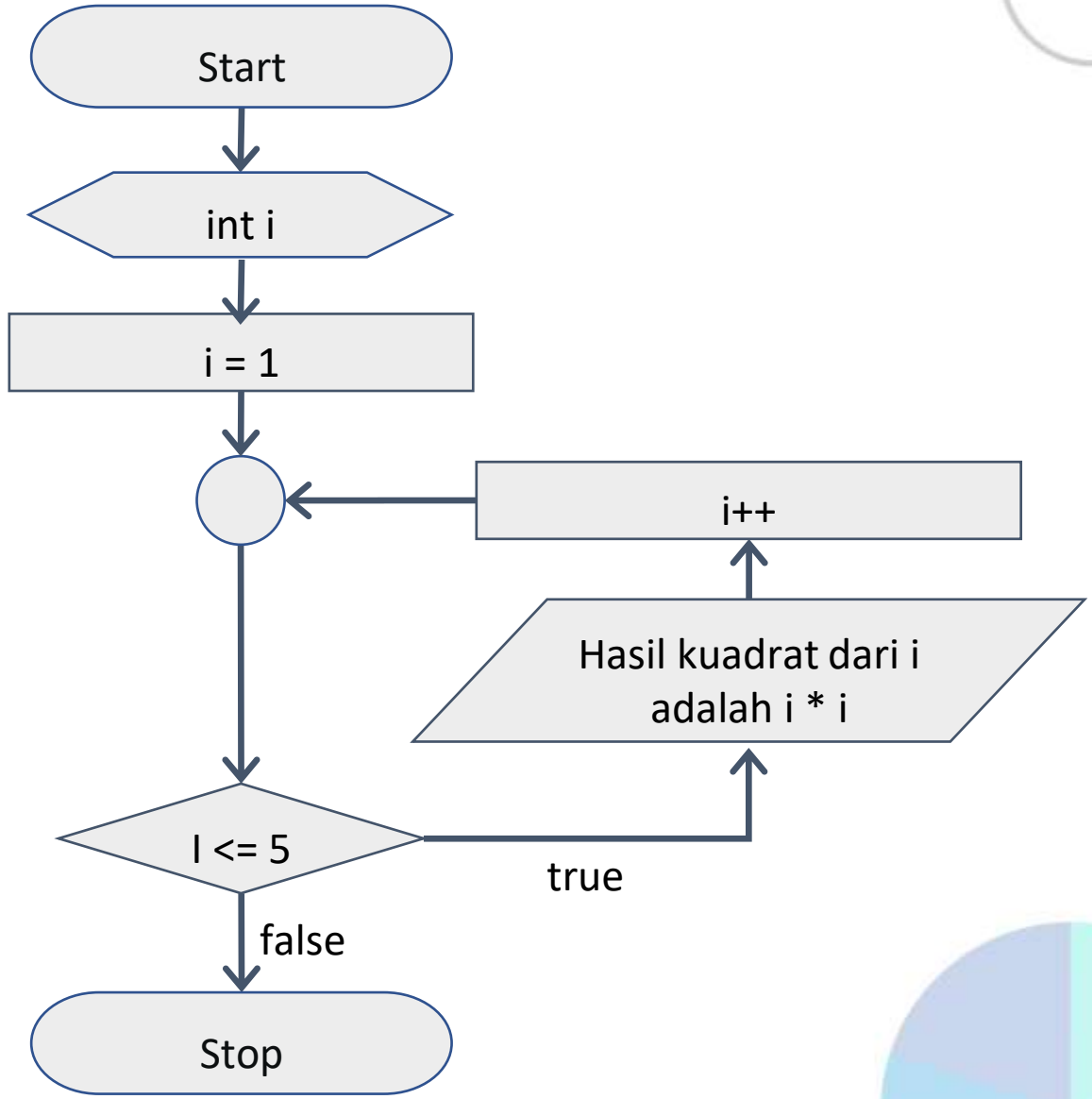
1. Perulangan diawali dengan melalukan `inisialisasi`
2. Evaluasi `kondisi`
   - Jika kondisi bernilai TRUE, eksekusi semua statement di dalam perulangan. Lakukan `update`. Ulangi kembali langkah nomor 2
   - Jika kondisi bernilai FALSE, hentikan perulangan

Flowchart FOR



Contoh Perulangan FOR

Buatlah flowchart dan kode program untuk menampilkan bilangan dan hasil kuadratnya dengan rentang nilai bilangan 1 sampai 5!



Contoh Perulangan FOR

Kode Program

```
for (int i = 1; i <= 5; i++) {
    System.out.println("Hasil kuadrat dari " + i + " adalah " + (i * i));
}
```

Output

```
Hasil kuadrat dari 1 adalah 1
Hasil kuadrat dari 2 adalah 4
Hasil kuadrat dari 3 adalah 9
Hasil kuadrat dari 4 adalah 16
Hasil kuadrat dari 5 adalah 25
```

Variasi Perulangan FOR – Variasi 1

`inisialisasi` dan `update` boleh terdiri dari beberapa ekspresi yang dipisahkan dengan tanda koma

```
for (int i = 1, j = 10; i < j; i++, j--) {
    System.out.printf("%03d -- %03d\n", i, j);
}
```

Output

```
001 -- 010
002 -- 009
003 -- 008
004 -- 007
005 -- 006
```

Variasi Perulangan FOR – Variasi 2

- `inisialisasi` dan `update` dapat dikosongi, sesuai dengan kebutuhan
- Contoh:

```
Scanner sc = new Scanner(System.in);
int bil;
boolean berhenti = false;
for (; !berhenti;) {
    System.out.print("Masukkan bilangan: ");
    bil = sc.nextInt();
    System.out.println("Bilangan yang Anda masukkan: " + bil);
    if (bil == 0) {
        berhenti = true;
    }
}
System.out.println("Program berakhir");
```

```
Masukkan bilangan: 4
Bilangan yang Anda masukkan: 4
Masukkan bilangan: 1
Bilangan yang Anda masukkan: 1
Masukkan bilangan: 0
Bilangan yang Anda masukkan: 0
Program berakhir
```

Output

Variasi Perulangan FOR – Variasi 3

Seperti halnya kondisi dalam `if`, `kondisi` pada `for` juga dapat menggunakan variable bertipe boolean Contoh:

```
Scanner sc = new Scanner(System.in);
int bil, n;
boolean berhenti = false;
for (n = 0; !berhenti; n++) {
    System.out.print("Masukkan bilangan: ");
    bil = sc.nextInt();
    System.out.println("Bilangan yang Anda masukkan: " + bil);
    if (bil < n) {
        berhenti = true;
    }
}
System.out.println("Program berakhir");
```

```
Masukkan bilangan: 2
Bilangan yang Anda masukkan: 2
Masukkan bilangan: 5
Bilangan yang Anda masukkan: 5
Masukkan bilangan: 3
Bilangan yang Anda masukkan: 3
Masukkan bilangan: 2
Bilangan yang Anda masukkan: 2
Program berakhir
```

Output

Struktur Perulangan WHILE

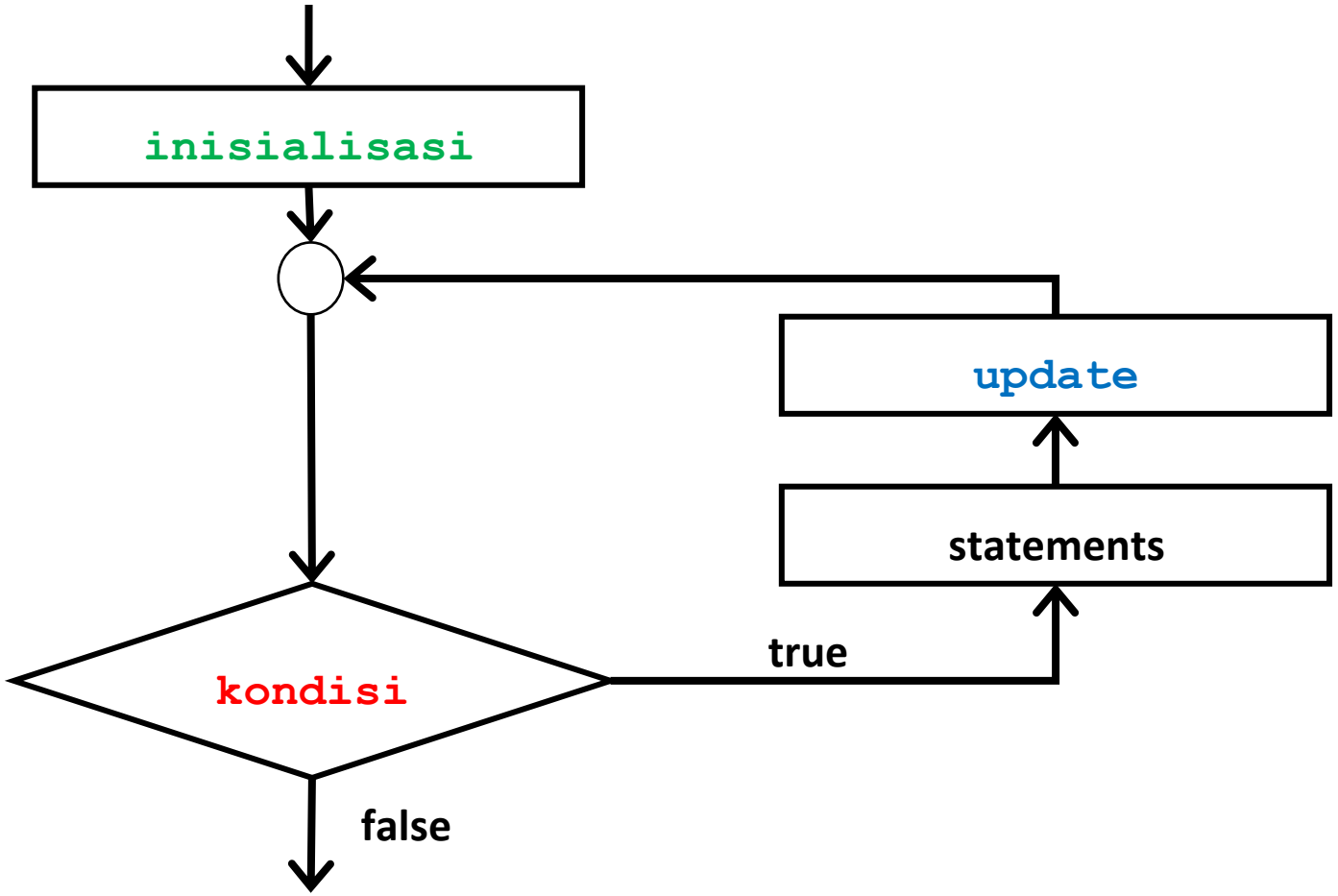
Perulangan WHILE

- 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

Flowchart WHILE



Perbandingan FOR dan WHILE

WHILE

```
inisialisasi;

while
(kondisi)
{
    statement1;
    statement2;
    …
    update
}
```

setara

FOR

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

```
        a
        t
        e
}
```

Contoh:

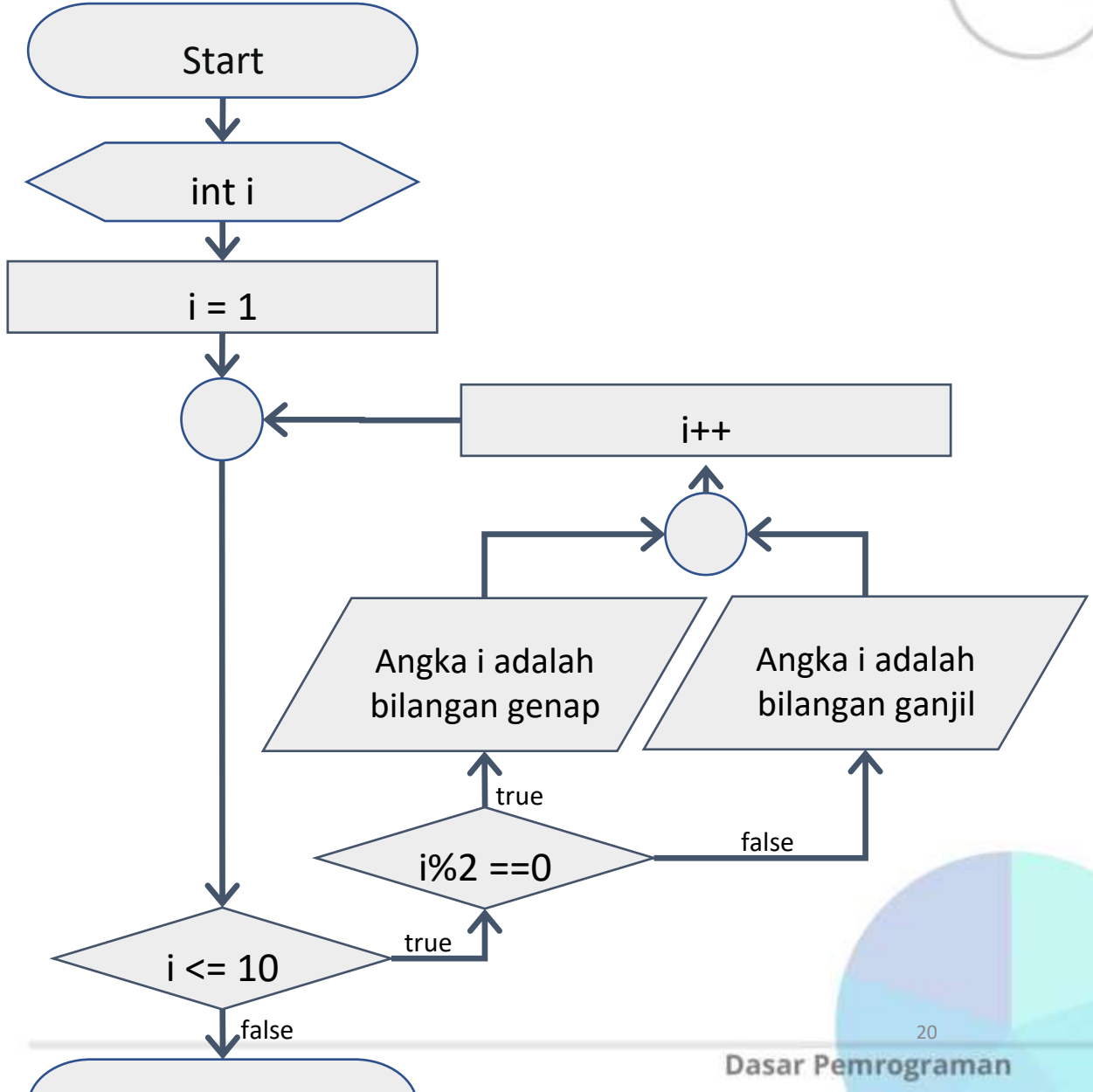
```
int x = 1;

while (x <= 10) {
      ______
      ______
      x++;
 }
```

sama

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

Contoh Perulangan WHILE



Buatlah flowchart dan kode program untuk menampilkan keterangan bilangan ganjil dan genap dengan rentang nilai bilangan 1 sampai 10!

Contoh Perulangan WHILE

Kode Program

```
int i = 1;
while (i <= 10) {
    if (i % 2 == 0) {
        System.out.println("Angka " + i + " adalah bilangan genap");
    } else {
        System.out.println("Angka " + i + " adalah bilangan ganjil");
    }
    i++;
}
```

Output

```
Angka 1 adalah bilangan ganjil
Angka 2 adalah bilangan genap
Angka 3 adalah bilangan ganjil
Angka 4 adalah bilangan genap
Angka 5 adalah bilangan ganjil
Angka 6 adalah bilangan genap
Angka 7 adalah bilangan ganjil
Angka 8 adalah bilangan genap
Angka 9 adalah bilangan ganjil
Angka 10 adalah bilangan genap
```

Struktur Perulangan DO-WHILE

Perulangan DO-WHILE

- Pada prinsipnya, perintah DO-WHILE sama dengan perintah WHILE
- Perbedaanya:
  - 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**

Perulangan DO-WHILE

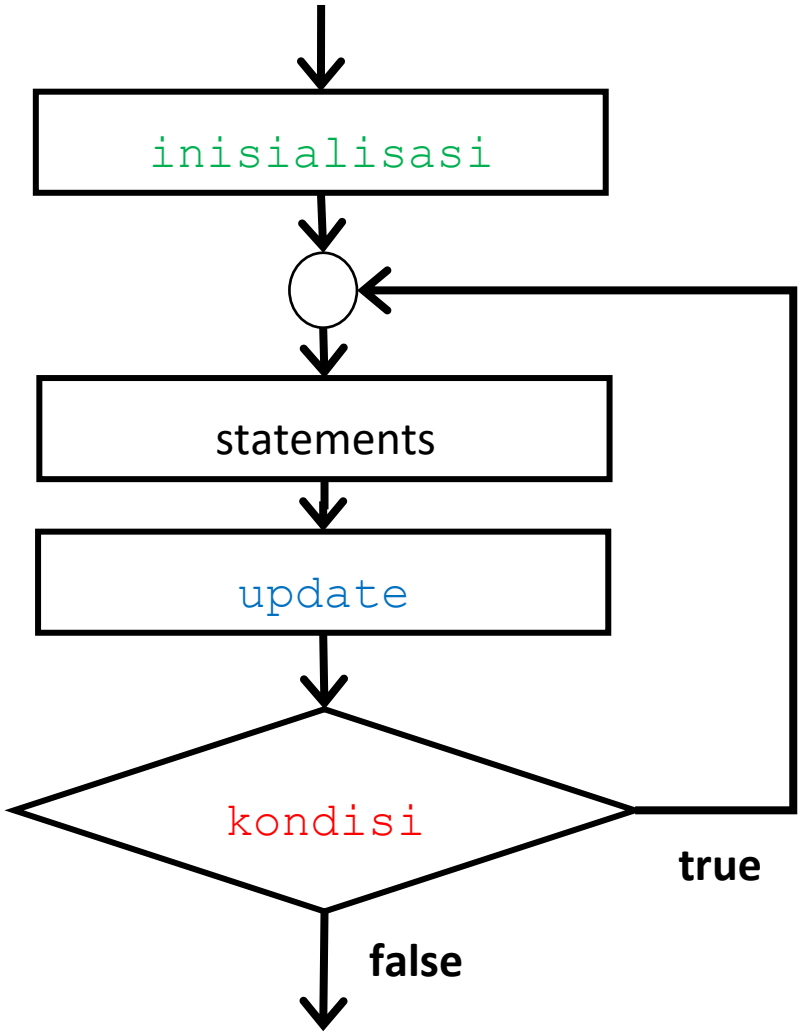
- Sintaks DO-WHILE

```
Inisialisasi;
do{
```

```
        statement1; statement2; … update;
} while (kondisi);
```

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

Flowchart DO-WHILE



Contoh Perulangan DO-WHILE

Kode Program

```
int x = 0;
do {
    System.out.println(x);
} while (++x <= 8);
System.out.println("Program berhenti");
```

Kode Program Output

```
0
1
2
3
4
5
6
7
8
Program berhenti
```

```
int x = 10;
do {
    System.out.println(x);
} while (++x <= 8);
System.out.println("Program berhenti");
```

```
10
Program berhenti
```

Output

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

Contoh Infinite Loop

Kode Program

```
int hitung = 1;
while (hitung <= 25) {
    System.out.println(hitung);
    hitung = hitung - 1;
}
```

Output

```
1 0 -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 -26 -27 -28 -29 -30 -31 -32 -33 -34 -35 -36 -37 -38 -39 -40 -41 -42 -43
```

Loop akan berjalan terus menerus sampai pengguna menghentikannya

Cara Menghentikan Perulangan

Cara Menghentikan Perulangan

Beberapa cara untuk menghentikan pengulangan untuk program interaktif, di antaranya dapat dilakukan dengan:

- 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

Menambahkan Sentinel – Contoh Kode Program

Tuliskan kode program untuk menerima input (integer positif) dari pengguna sampai pengguna memasukkan -1 untuk berhenti. Cetak jumlah dan rata-rata dari angka-angka yang telah dimasukkan.

```
Scanner sc = new Scanner(System.in);
int jumlah = 0, counter = 0, angka;
float rata = 0;
do {
    System.out.print("Masukkan integer positif (-1 untuk berhenti): ");
    angka = sc.nextInt();
    if (angka >= 0) {
        jumlah += angka;
        ++counter;
    }
} while (angka != -1);
rata = jumlah / counter;
System.out.printf("Jumlah dari %d angka adalah %d\n", counter, jumlah);
System.out.printf("Rata-rata dari %d angka adalah %.3f\n", counter, rata);
```

Output

```
Masukkan integer positif (-1 untuk berhenti): 10
Masukkan integer positif (-1 untuk berhenti): 20
Masukkan integer positif (-1 untuk berhenti): 30
Masukkan integer positif (-1 untuk berhenti): 40
Masukkan integer positif (-1 untuk berhenti): 50
Masukkan integer positif (-1 untuk berhenti): -1
Jumlah dari 5 angka adalah 150
Rata-rata dari 5 angka adalah 30.000
```

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?

Menambahkan Pertanyaan – Contoh Kode Program

Tuliskan kode program untuk menerima input sejumlah nama pelanggan. Cetak jumlah pelanggan yang telah dimasukkan.

```
Scanner sc = new Scanner(System.in);
String nama;
char jawab;
int jml = 0;
do {
    System.out.print("Masukkan nama pelanggan: ");
    nama = sc.nextLine();
    jml++;
    System.out.print("Apakah Anda ingin memasukkan nama pelanggan baru (Y/T)? ");
    jawab = sc.nextLine().charAt(0);
} while (jawab == 'y' || jawab == 'Y');
System.out.println("Jumlah pelanggan yang Anda masukkan = " + jml);
```

Output

```
Masukkan nama pelanggan: Afi
Apakah Anda ingin memasukkan nama pelanggan baru (Y/T)? y
Masukkan nama pelanggan: Brian
Apakah Anda ingin memasukkan nama pelanggan baru (Y/T)? Y
Masukkan nama pelanggan: Dewi
Apakah Anda ingin memasukkan nama pelanggan baru (Y/T)? t
Jumlah pelanggan yang Anda masukkan = 3
```

Statement BREAK dan CONTINUE

Statement BREAK

- Terkadang suatu program perlu untuk keluar dari perulangan
- Pernyataan BREAK akan **menghentikan paksa** perulangan, kemudian kode di luar perulangan akan dieksekusi

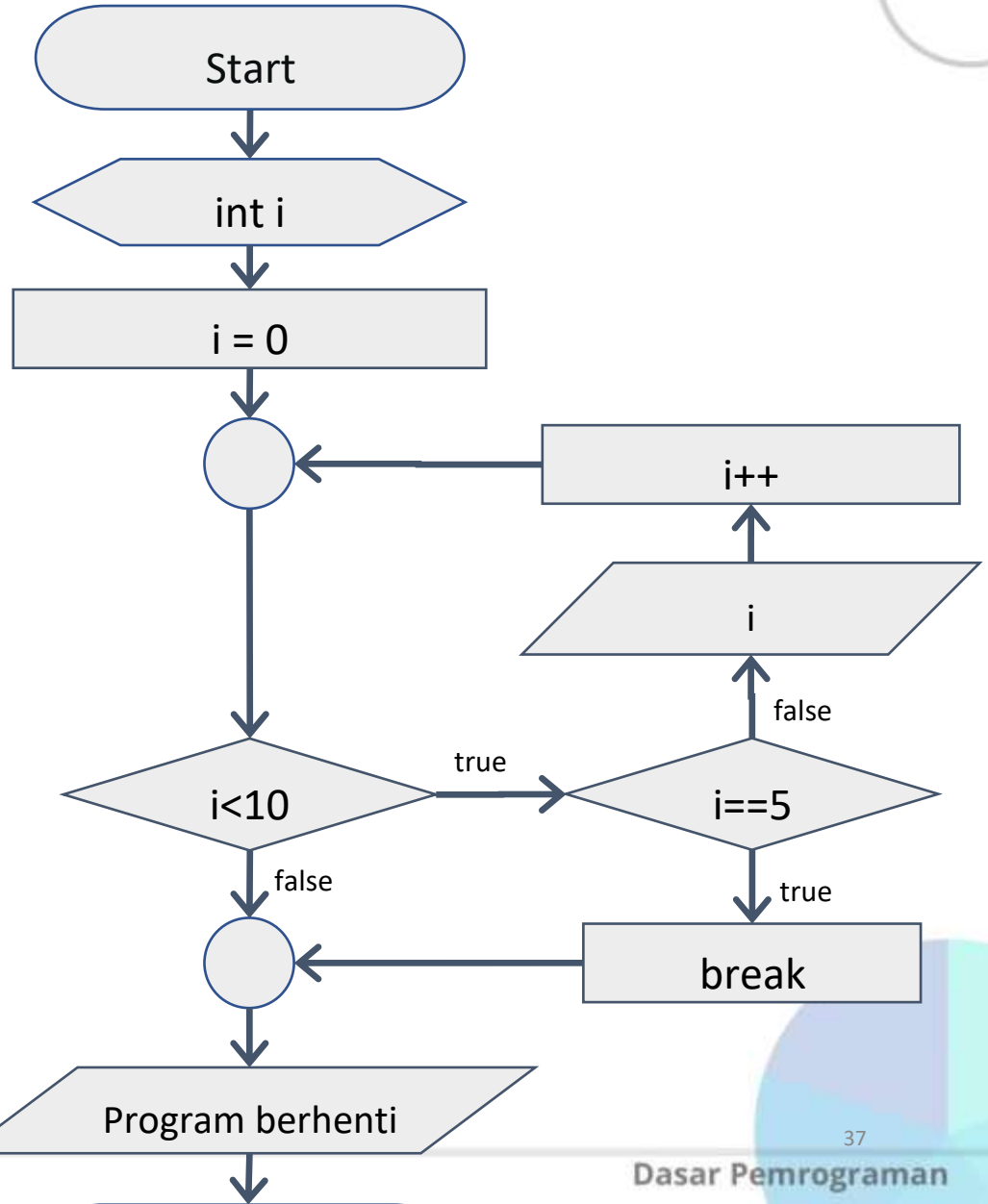
- Selain digunakan untuk keluar dari SWITCH (pemilihan SWITCHCASE), BREAK juga digunakan untuk keluar dari perulangan (FOR, WHILE dan DO-WHILE)

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

```
0 1 2 3 4
Program berhenti
```



Kode Program

Output

Statement CONTINUE

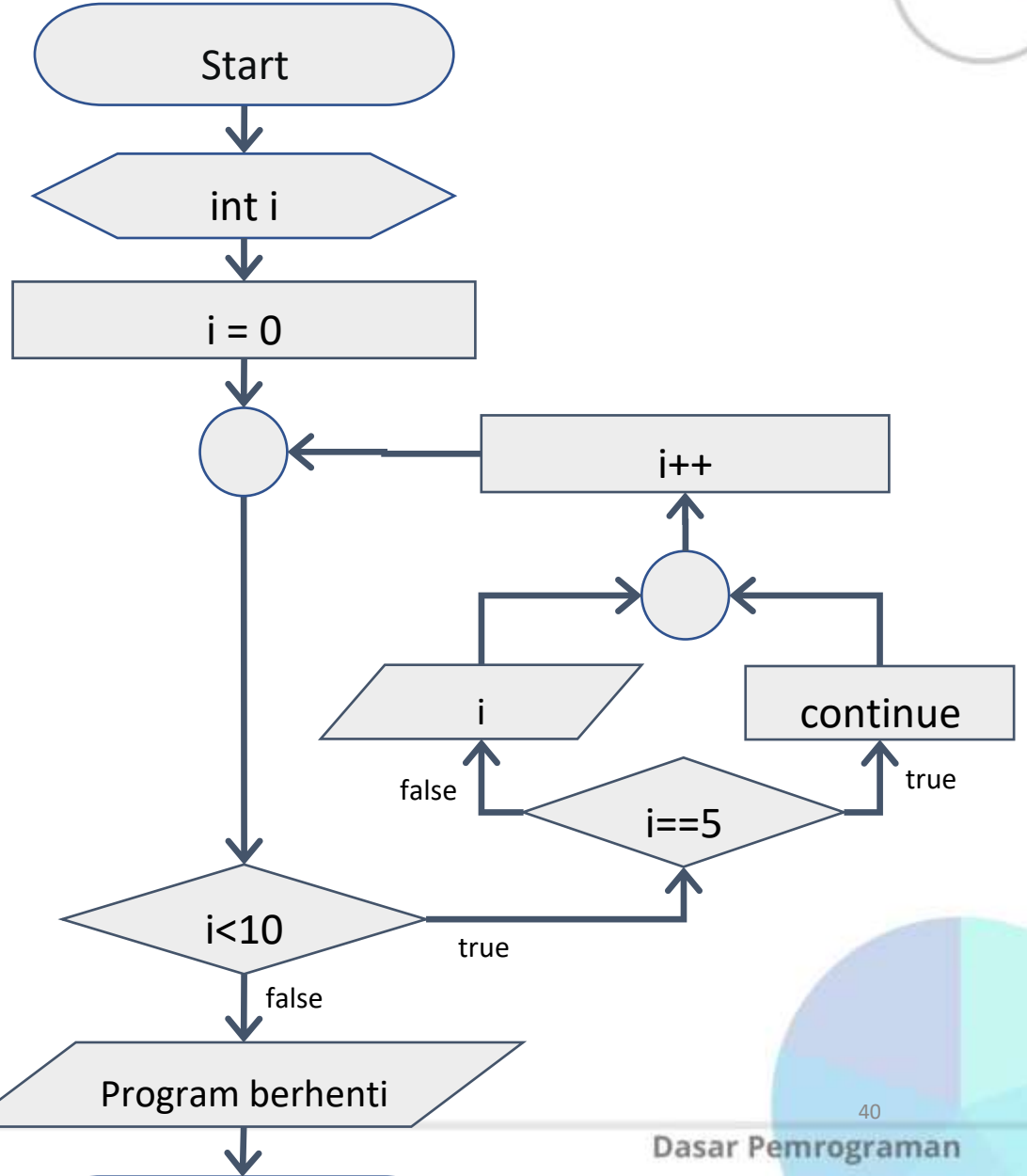
- Digunakan untuk menghentikan perulangan yang saat ini terjadi (1 iterasi saja), kemudian melanjutkan perulangan iterasi berikutnya
- Melewati (skip) sisa statement dalam loop, dan eksekusi loop berjalan ke tahap selanjutnya

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

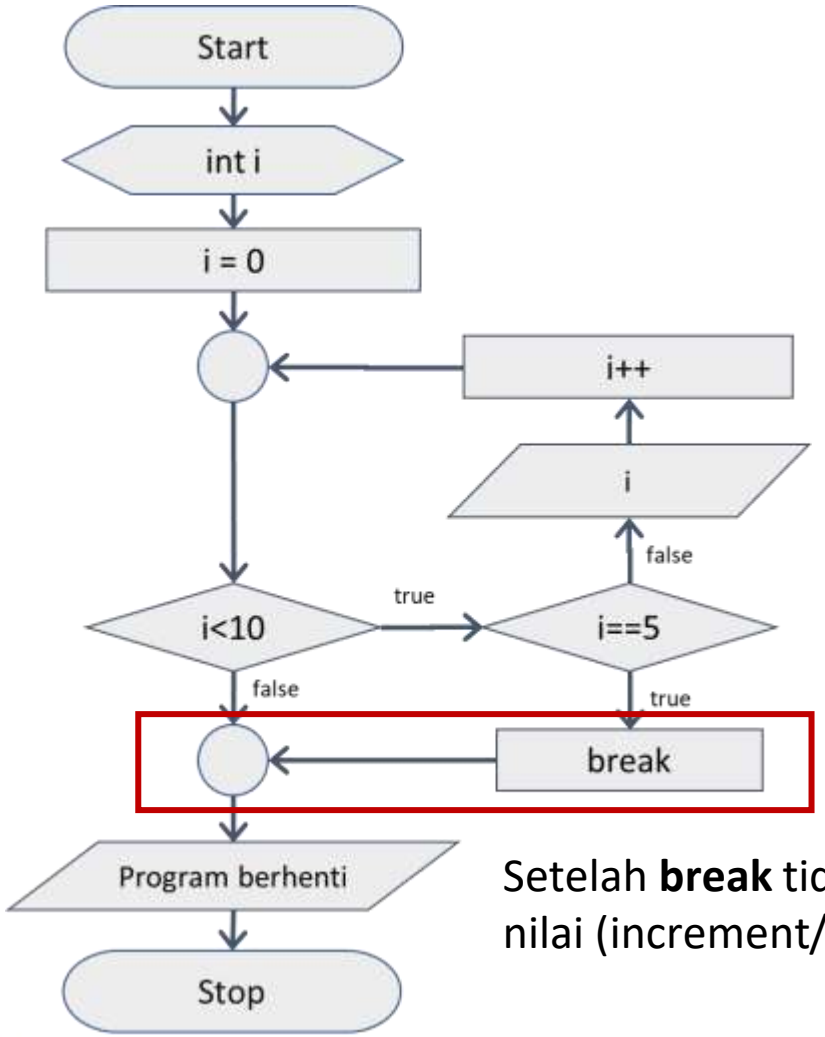
```
0 1 2 3 4 6 7 8 9
Program berhenti
```



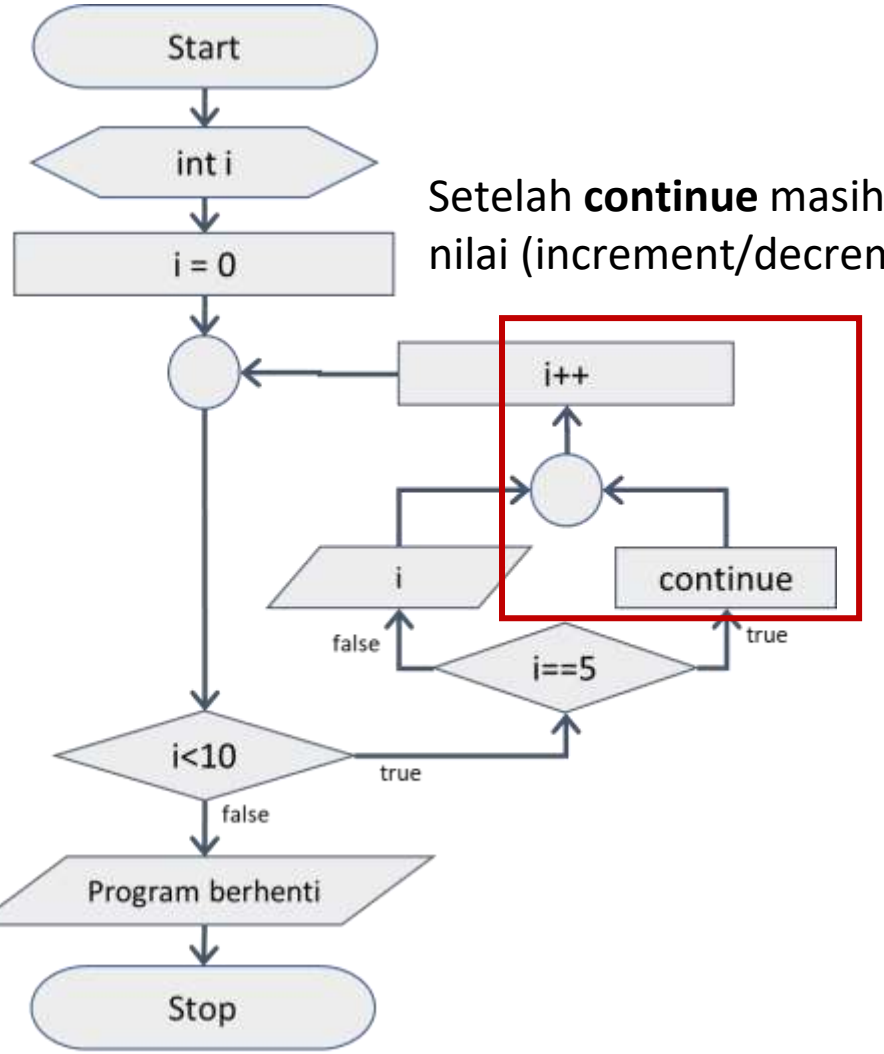
Kode Program

Output

Perbedaan BREAK dan CONTINUE pada Flowchart



Setelah **break** tidak ada update nilai (increment/decrement)

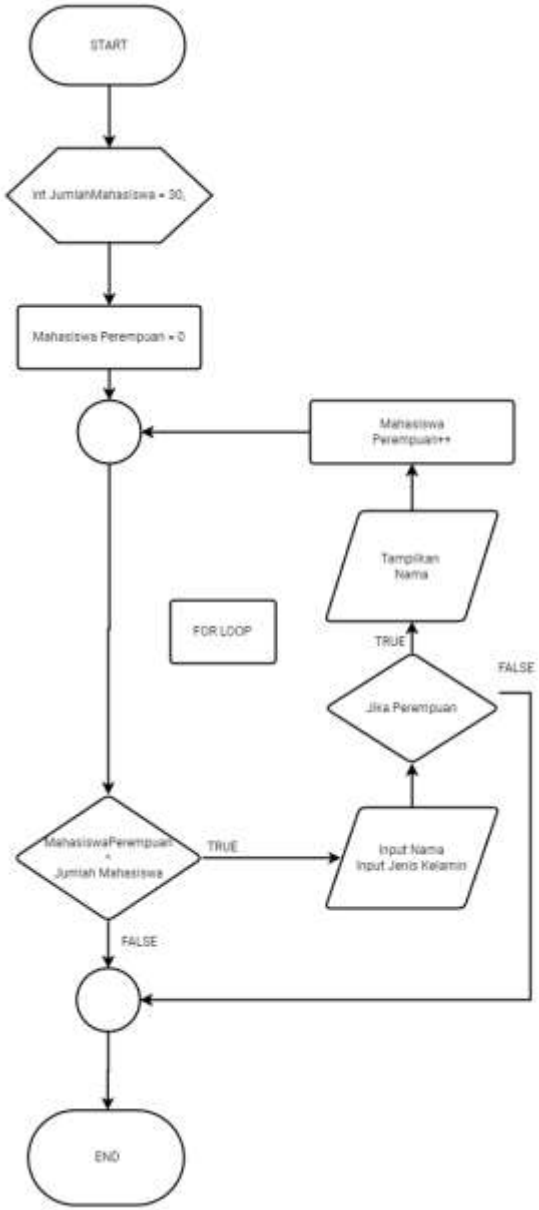


Setelah **continue** masih ada update nilai (increment/decrement)

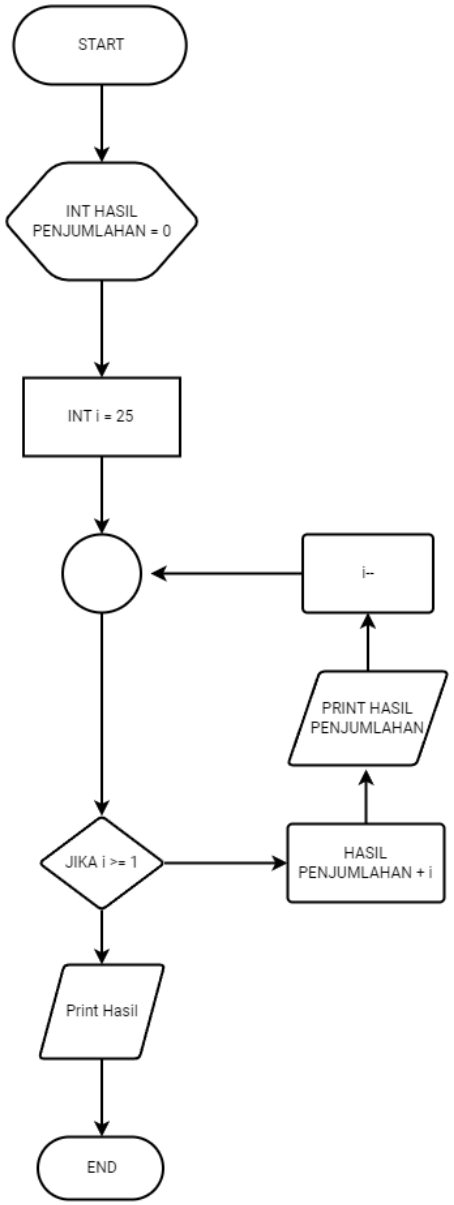
Latihan Mandiri

Buatlah flowchart dari pernyataan berikut dengan menggunakan FOR, WHILE, atau DO-WHILE:

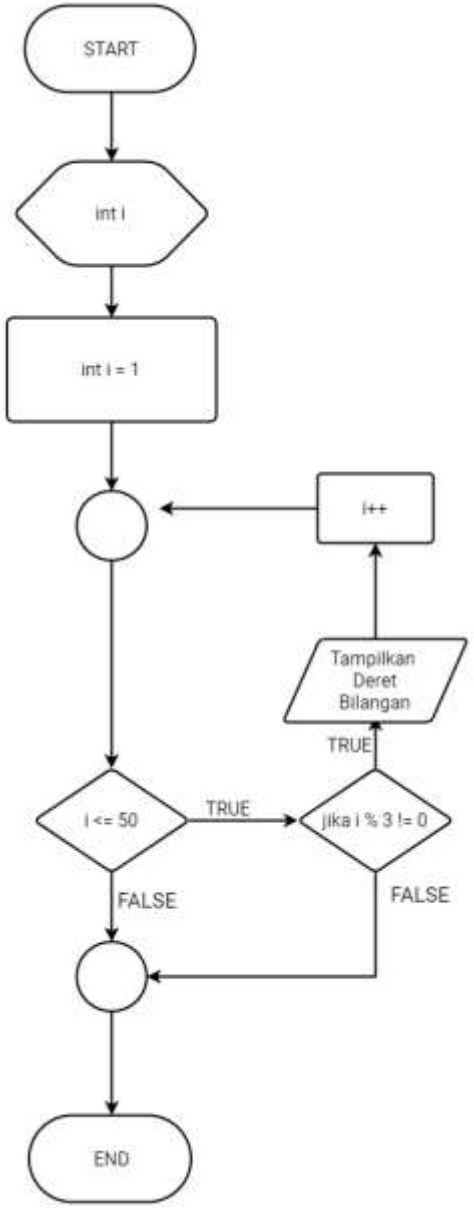
a. Pengguna memasukkan nama dan jenis kelamin dari 30 mahasiswa di suatu kelas. Nama-nama mahasiswa yang ditampilkan hanya yang berjenis kelamin perempuan


START
int JumlahMahasiswa = 30;
Mahasiswa Perempuan = 0
Mahasiswa Perempuan++
Tampilkan Nama
FOR LOOP
TRUE
FALSE
Jika Perempuan
MahasiswaPerempuan < Jumlah Mahasiswa
TRUE
Input Nama
Input Jenis Kelamin
FALSE
END

b. Menampilkan hasil penjumlahan deret bilangan 25 sampai dengan 1


START
INT HASIL PENJUMLAHAN = 0
INT i = 25
i--
PRINT HASIL PENJUMLAHAN
JIKA i >= 1
HASIL PENJUMLAHAN + i
Print Hasil
END

c. Menampilkan deret bilangan 1 sampai 50, kecuali bilangan kelipatan 3 (1 2 4 5 7 8 10 … 47 49 50)


START
int i
int i = 1
i++
Tampilkan Deret Bilangan
TRUE
i <= 50
TRUE
jika i % 3 != 0
FALSE
FALSE
END

Tugas Kelompok

1. Identifikasi sesuai project masing-masing fitur apa saja yang membutuhkan konsep perulangan
2. Tentukan bentuk perulangan yang sesuai (FOR, DO-WHILE, WHILE-DO)
3. Gunakan sentinel/BREAK/CONTINUE jika diperlukan
4. Buatlah algoritma dalam bentuk flowchart