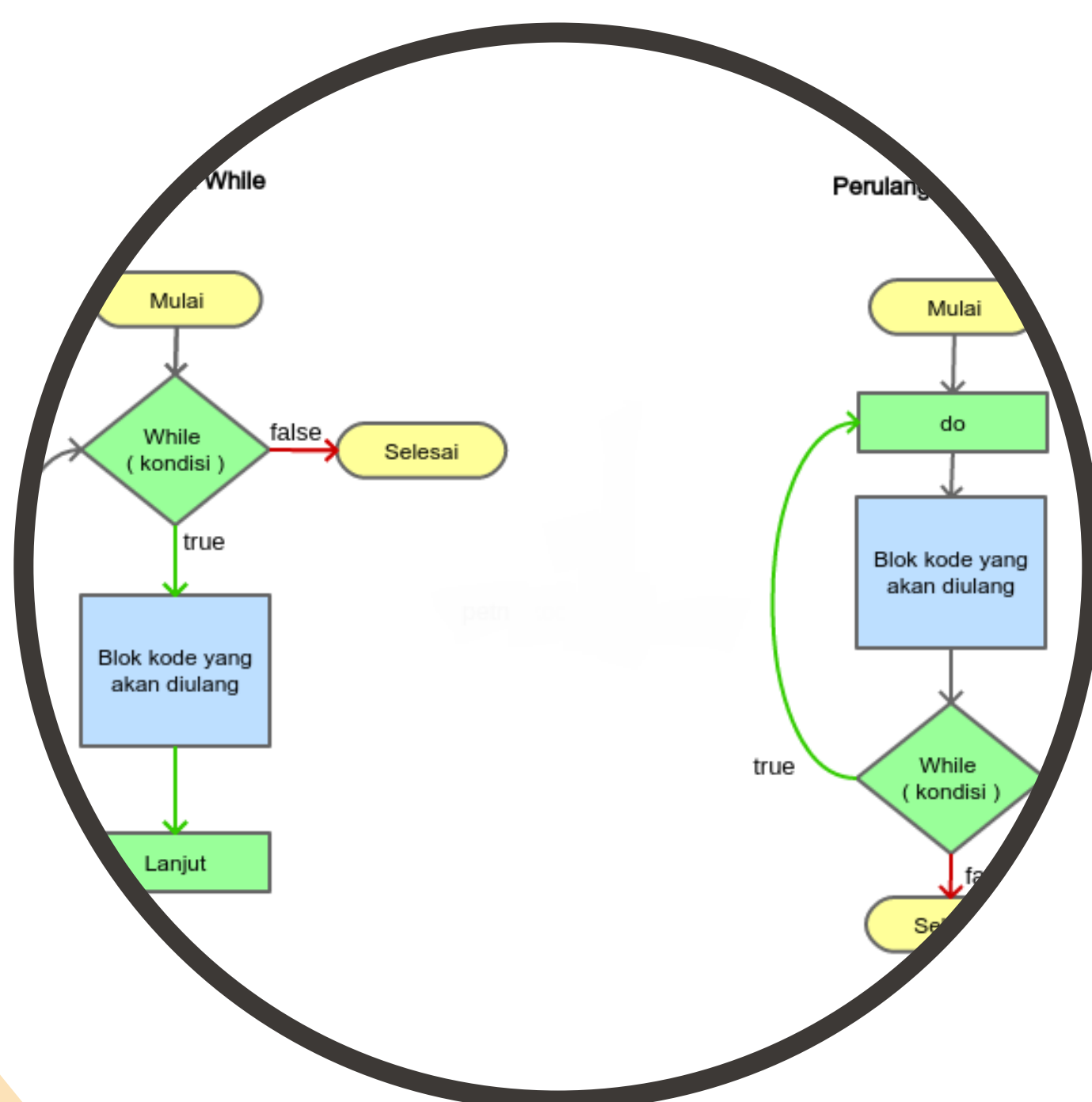


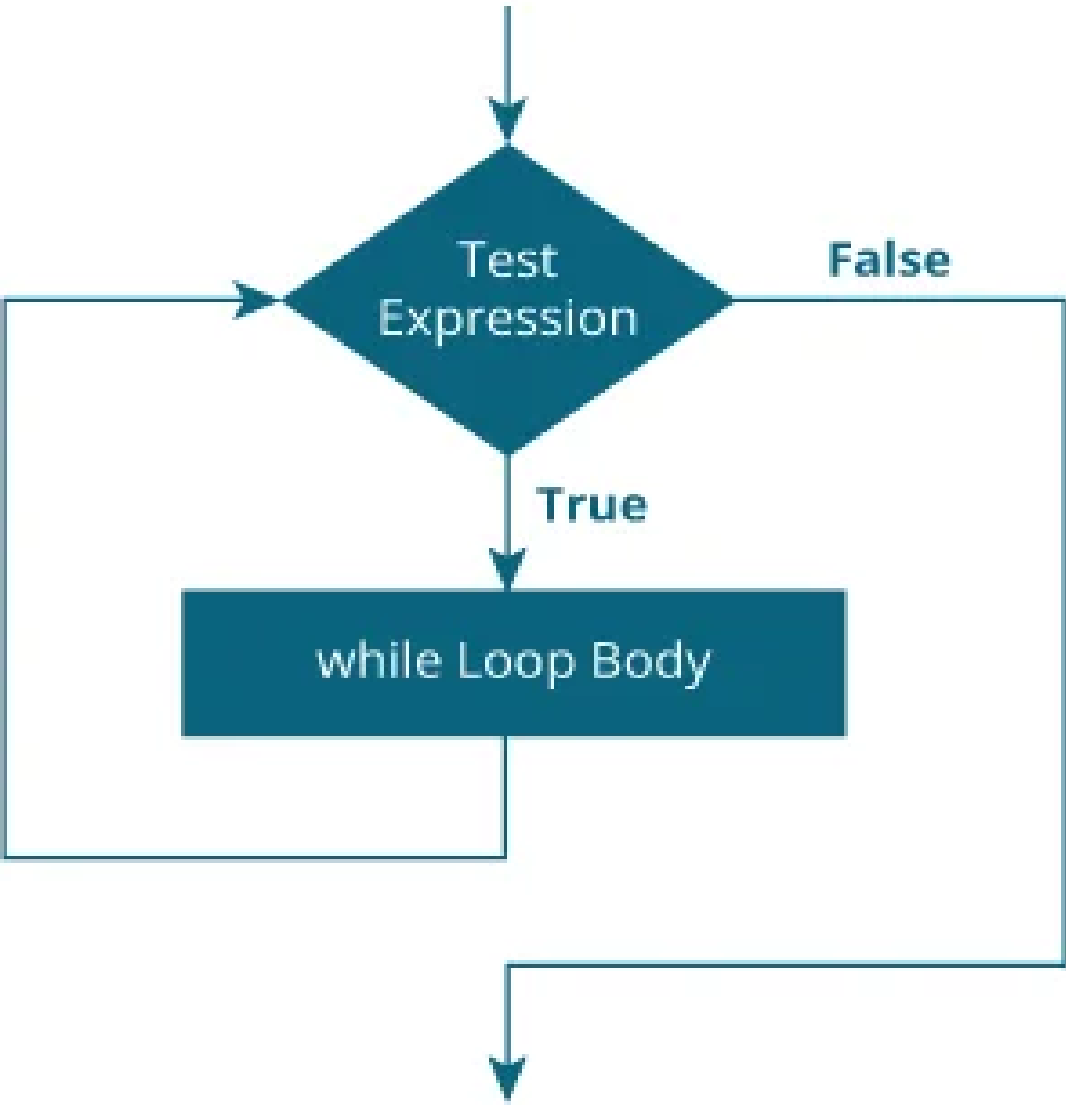
Modul Perulangan Bagian II



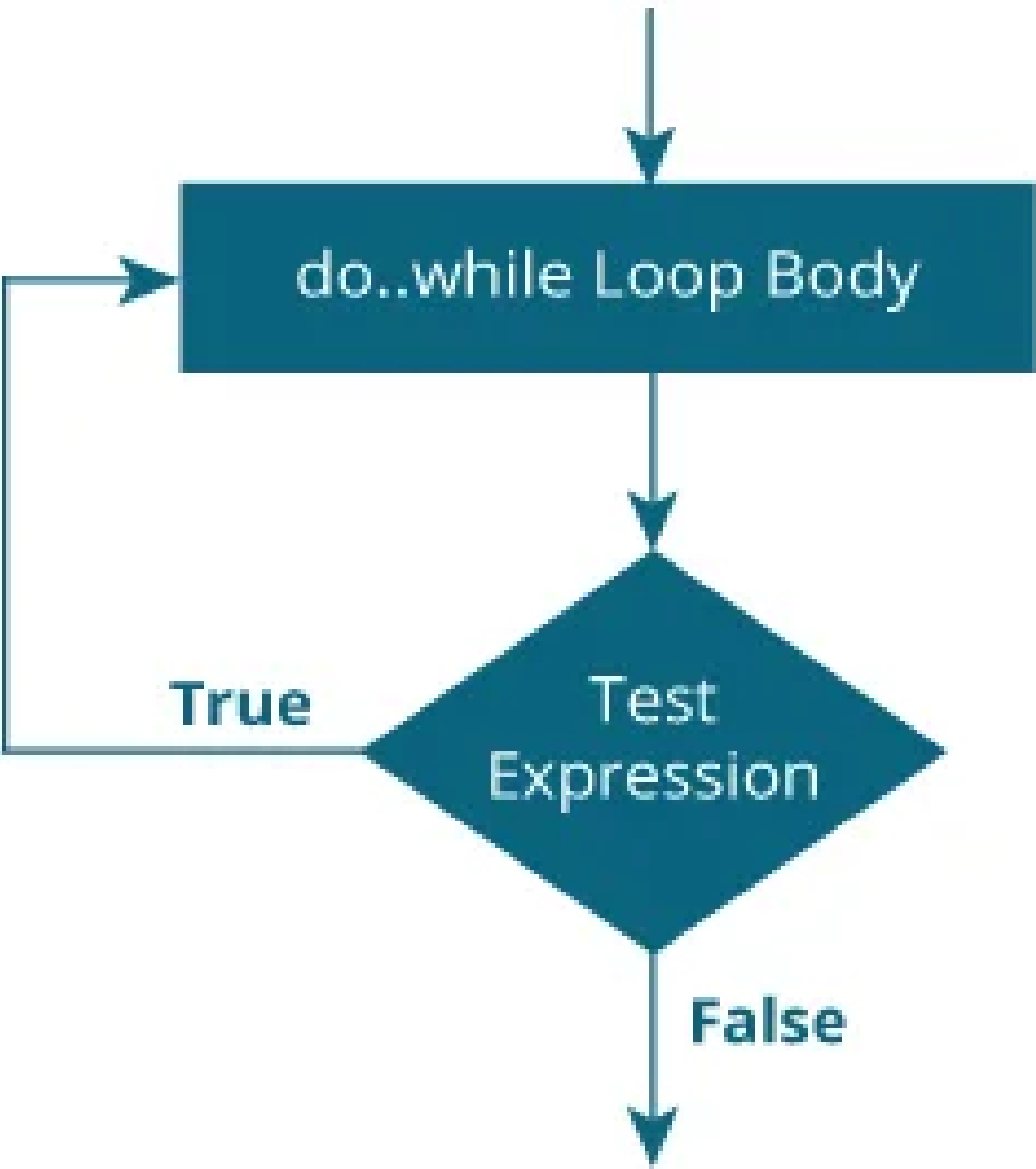
Disusun Oleh :
Johannes Alexander Putra

Pengantar

Pertemuan sebelumnya kalian sudah mempelajari definisi perulangan dan beberapa jenis perulangan seperti for dan while. Pada pertemuan terakhir ini kalian akan mempelajari mengenai perulangan do while. Mari kita simak perbedaan antara perulangan while dan do while berdasarkan flowchart berikut ini:



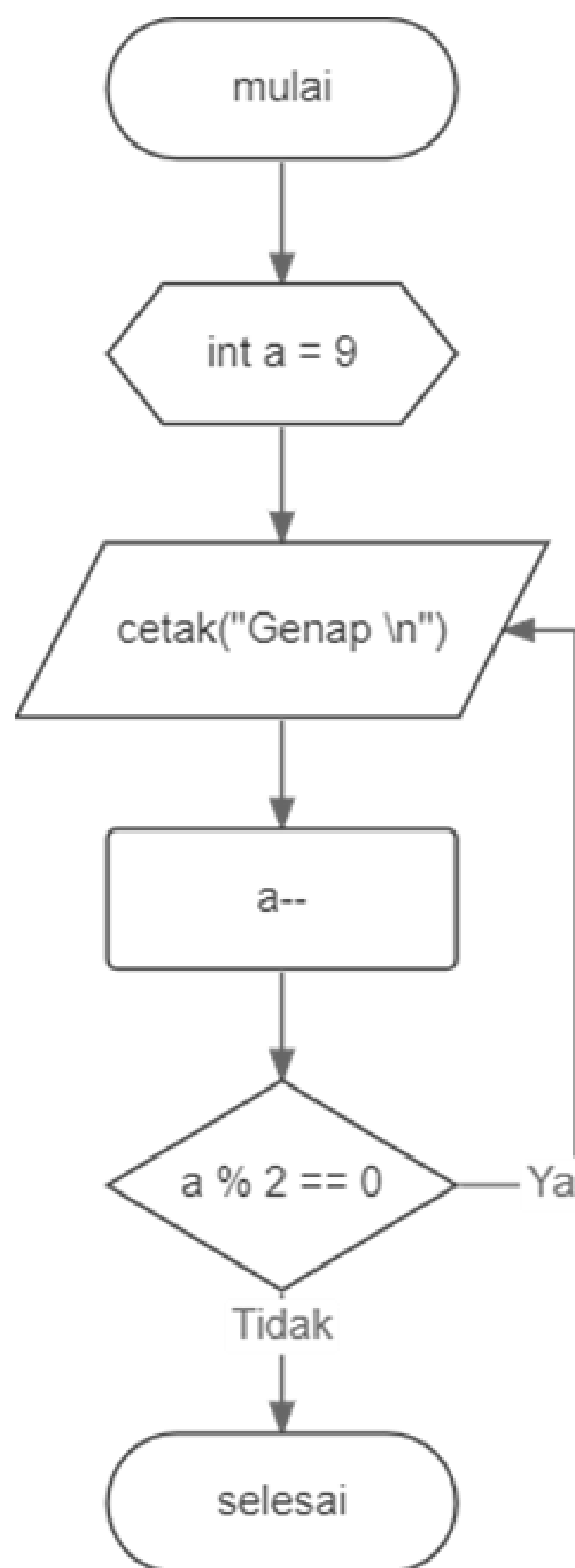
Gambar 1 Flowchart While



Gambar 2 Flowchart Do While

Perulangan Do While

Do While Perulangan Do While biasanya digunakan jika jumlah perulangannya tidak diketahui namun berbeda dengan while. karena kondisi perulangan ada di bagian bawah. Perulangan ini minimal dilakukan satu kali (Sukamto, 2018)



Gambar 3 Flowchart Do While Menentukan Bilangan Genap

```
do {  
    <pernyataan>;  
} while (ekspresi kondisi);
```

Gambar 4 Struktur Do While

```

1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int a = 9;
7      do{
8          cout<<"genap"<<endl;
9          a--;
10
11      }while(a % 2 == 0);
12
13      return 0;
14
15  }

```

Gambar 5 Code Do While untuk Memprint Genap ketika Angkanya Genap pada C++



jdoodle.com/ia/Nho

Penjelasan

Konstruksi do while mendasarkan pengulangan pada kondisi bernilai boolean. Pemeriksaan kondisi dilakukan di akhir setiap pengulangan. Maka output yang akan dicetak adalah

Genap

Genap

Karena proses dikerjakan terlebih dahulu walaupun 9 bukan bilangan genap.

Contoh Soal

Miftah ingin membuat suatu aplikasi dalam bahasa pemrograman C++ yang menerapkan prinsip perulangan yang iterasinya berdasarkan kondisi dan tidak hanya sekadar angka awal dan angka akhir saja dan ingin mengulang minimal satu kali, Jika angka awal adalah 9. Perulangan akan mencetak "Genap" minimal satu kali dan nilai bilangan selalu berkurang 1 dan batas looping adalah ketika bilangan bernilai ganjil, penamaan variable dibebaskan, apakah informasi sudah cukup untuk menyelesaikan permasalahan, perulangan apa yang dipergunakan dan bagaimana code programnya?

Ada 4 Tahapan Penyelesaian masalah menurut (Polya, 2014)

Memahami Masalah, Merencanakan Penyelesaian Masalah,

Melaksanakan Rencana Penyelesaian Masalah, dan Memeriksa Kembali

Tahap 1 (Memahami Masalah):

1. Apa saja data yang diketahui dan penting?

Aplikasi perulangan dalam bahasa C++. Iterasi perulangannya berdasarkan suatu kondisi bukan hanya sekadar angka awal dan angka akhir. Perulangan yang dilakukan minimal satu kali. Mencetak genap minimal satu kali dan berhenti ketika angka bernilai ganjil. Penamaan variable dibebaskan

2. Apakah data tersebut sudah dapat menyelesaikan pemecahan permasalahan?

Data tersebut sudah cukup untuk menyelesaikan pemecahan masalah, dengan data tersebut kita dapat menentukan perulangan yang dipergunakan, pembuatan flowchart diagram, dan pembuatan code program.

Keterangan:

Kondisi dinyatakan berlebihan bila data yang diberikan berlebihan

contohnya, Jika terdapat soal: "Apakah kondisi tersebut cukup untuk menentukan bahwa perulangan yang digunakan adalah perulangan do while .. ?", Jika pada soal ditambahkan keterangan, perulangan akan melakukan pengecekan kondisi di akhir dan perulangan ini batas atas dan bawahnya hanya berupa angka saja. Maka data yang diberikan akan berlebihan.

Kondisi dinyatakan kontradiktif. Misalnya jika terdapat soal: "Apakah kondisi yang diberikan cukup untuk menjawab bahwa perulangan tersebut adalah perulangan do while? Bila pada soal diberikan informasi: perulangan minimal dilakukan satu kali dan pengecekan kondisi dilakukan di awal.

Kondisi dinyatakan cukup bila data yang diberikan sudah cukup untuk menjawab pertanyaan. Contohnya terdapat soal: “Apakah kondisi tersebut cukup untuk menentukan bahwa perulangan yang digunakan adalah perulangan do while ?”, Bila pada soal tersebut disebutkan bahasa pemrograman yang digunakan dan disebutkan ciri-ciri do while seperti perulangan minimal dilakukan satu kali

Kondisi dinyatakan tidak cukup bila data yang diberikan tidak cukup untuk menjawab pertanyaan. Contoh terdapat soal: “Apakah kondisi tersebut cukup untuk menentukan bahwa perulangannya adalah perulangan do while?”, Bila batas perulangan tidak jelas

Kondisi dinyatakan tidak ada konteks yang jelas bila tidak menggambarkan sama sekali perulangan do while

3. Jika pada soal diberikan tambahan, data berupa, “pengecekan dilakukan satu kali” apakah data tersebut sudah dapat menentukan perulangan yang digunakan adalah perulangan do while & menulisnya? Jika pada soal diberikan tambahan data, perulangan minimal dilakukan satu kali maka, data yang diberikan adalah kontradiktif. Hal tersebut dikarenakan pada soal sudah diberikan data perulangan minimal dilakukan satu kali dan jika ditambah pengecekan dilakukan di awal, maka keduanya adalah kontradiktif.

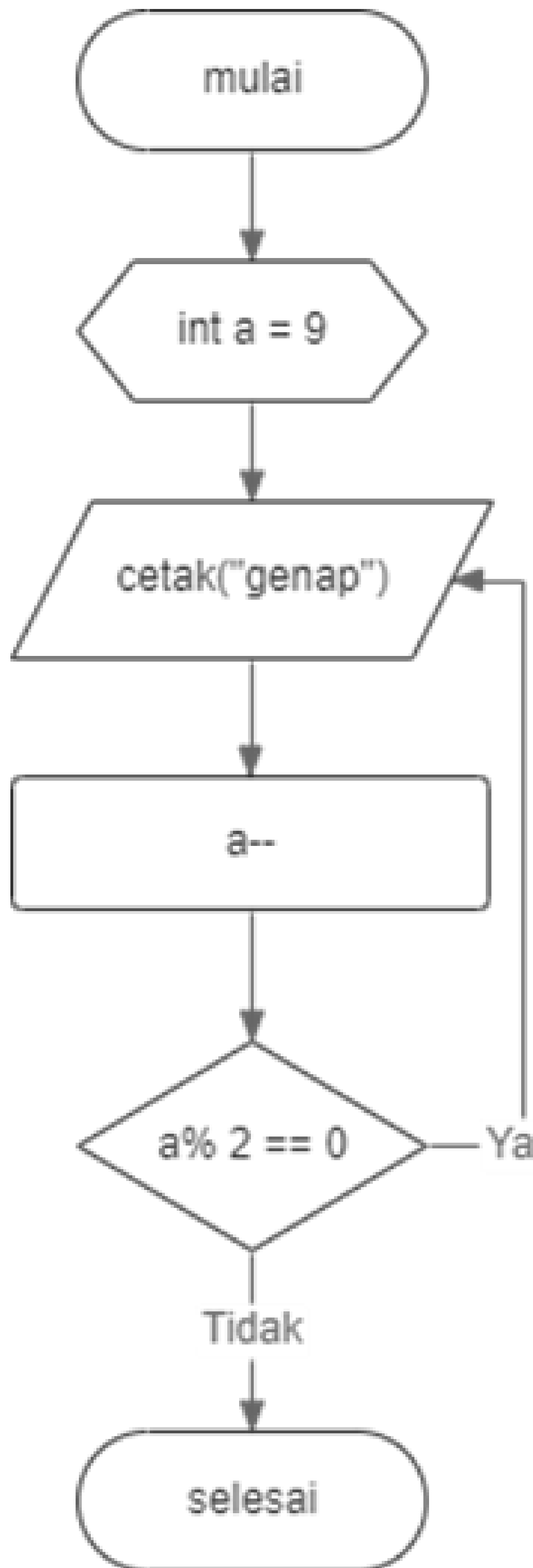
Tahap 2 (Membuat Rencana Penyelesaian Masalah):

1. Kira-kira kondisi apakah yang mirip dengan kasus tersebut?

Jawab: kondisi yang mirip adalah perulangan do while karena kondisi pengecekan dilakukan di akhir.

2. Bagaimana flowchartnya?

Jawab:



Tahap 3 (Melaksanakan Rencana Penyelesaian Masalah):

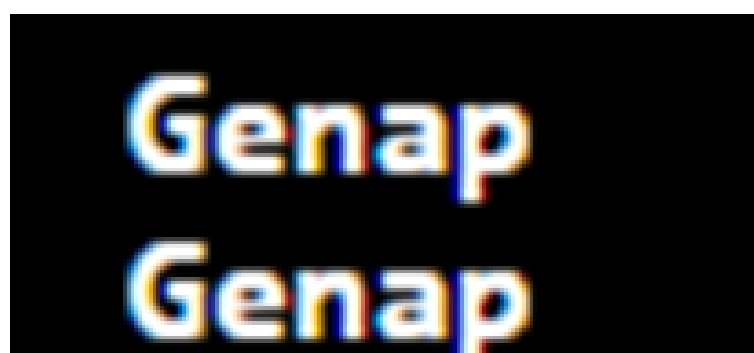
1. Bagaimana kodenya?

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4      int a = 9;
5      do{
6          cout<<"Genap \n";
7          a--;
8      }while(a % 2 == 0);
9      return 0;
10 }
11 }
```

Tahap 4 (Memeriksa Kembali):

1. Apakah kode tersebut sudah benar dan menghasilkan output dengan benar?

Berdasarkan kode tersebut dapat disimpulkan bahwa perulangan yang dihasilkan sudah benar. Bahwa minimal diadakan perulangan minimal 1 kali.



2. Bagaimana agar perulangan seperti code tersebut mencetak "Ganjil" dan berhenti ketika angka bernilai genap, nilai bilangan selalu berkurang 1?

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4      int a = 9;
5      do{
6          cout<<"Ganjil \n";
7          a--;
8      }while(a % 2 != 0);
9      return 0;
10 }
11 }
```

Ubah kata yang dicetak dari sebelumnya genap menjadi ganjil dan ubah kondisi whilenya dari sebelumnya $a \% 2 == 0$ menjadi $a \% 2 != 0$

Daftar Pustaka

Munir, R., & Lidya, L. (2016). Algoritma Dan Pemrograman Dalam Bahasa Pascal, C, C++ Edisi Keenam. Bandung: Informatika.

Polya, G. (2014). How to Solve It: A New Aspect of Mathematical Method. New Jersey: Princeton University Press.

Sukanto, R. A. (2018). Logika Algoritma dan Pemograman Dasar. Bandung: Modula.