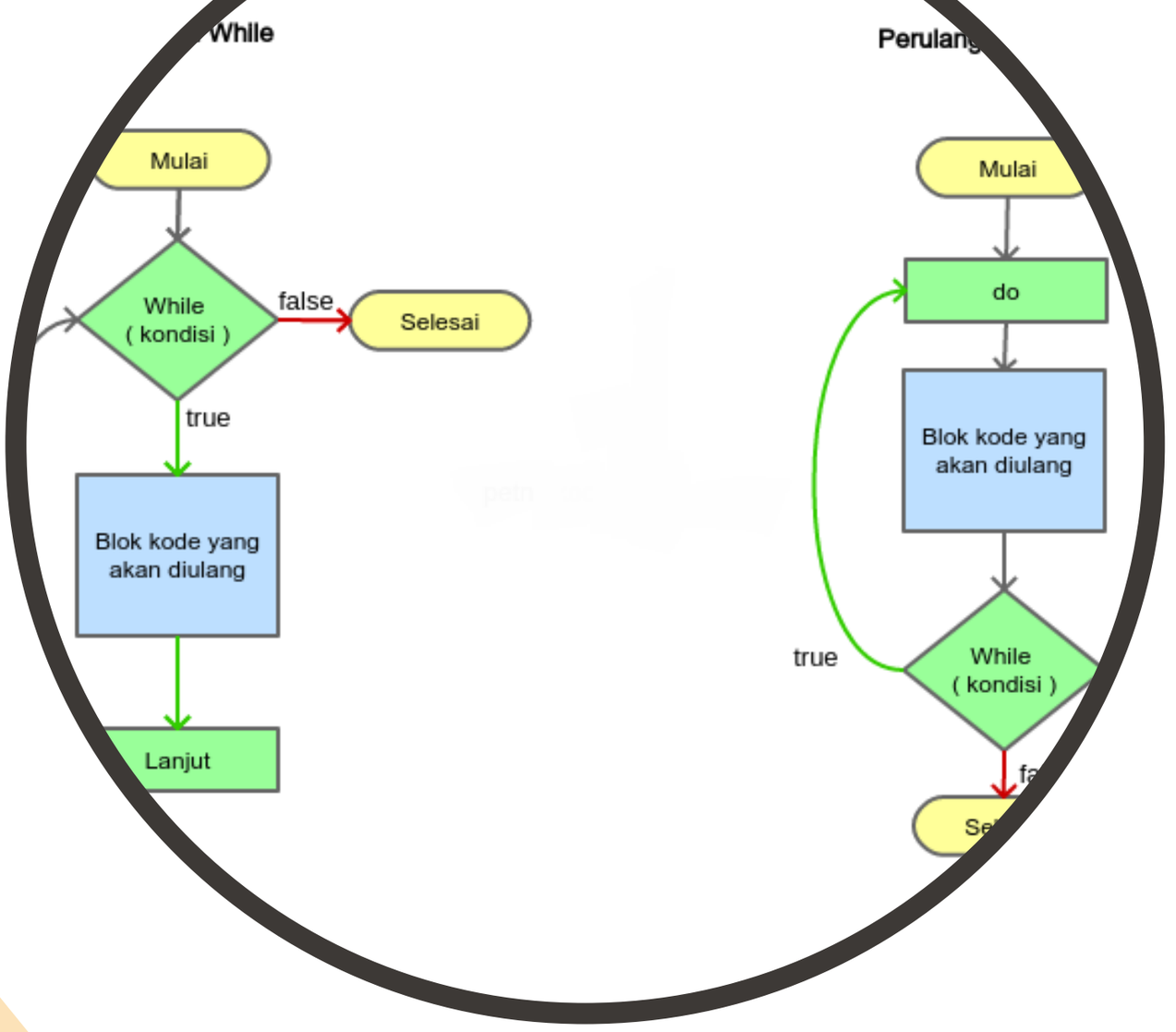
Modul Perulangan

Bagian I



Disusun Oleh :

Johannes Alexander Putra

Perulangan

Kalian akan belajar memahami konsep dan cara kerja perulangan dalam pemrograman. Setelah menyelesaikan aktivitas ini, kalian diharapkan mampu menerapkan perulangan dan menghubungkannya dengan kejadian sehari- hari.
a. Pernahkah kalian melakukan aktivitas yang sama berulang kali? Eits... tidak perlu jauh-jauh, untuk melangkah saja, kita mengulang pergerakan kaki kanan dan kiri. Apakah kalian selalu berpikir sebelum melakukan setiap langkah? Pertanyaan terakhir tersebut dapat ditanyakan pada semua kegiatan perulangan yang kalian lakukan.
b. Membaca buku - kalian membalikkan halaman buku setiap selesai membaca halaman tersebut

Salah satu keunggulan program komputer daripada manusia ialah kemampuannya untuk mengolah data yang berukuran besar atau melaksanakan suatu aksi berulang kali dalam periode waktu yang lama tanpa merasa bosan atau lelah. Hal ini dimungkinkan dengan adanya suatu kontrol perulangan. Pernyataan perulangan atau loop merupakan struktur program untuk keperluan iterasi, yaitu memproses satu atau beberapa pernyataan secara berulang (looping) berdasarkan kondisi tertentu. Program

Perulangan (Looping) merupakan suatu bagian yang bertugas untuk melakukan kegiatan mengulang suatu proses sesuai dengan yang diinginkan. Banyak dari aplikasi perangkat lunak yang melakukan pekerjaan berulang-ulang sampai sebuah kondisi yang diinginkan, oleh karena itu pengulangan merupakan bagian penting dalam pemrograman (Sukamto, 2018)

Jenis Perulangan

Terdapat beberapa jenis perulangan:

- Perulangan For

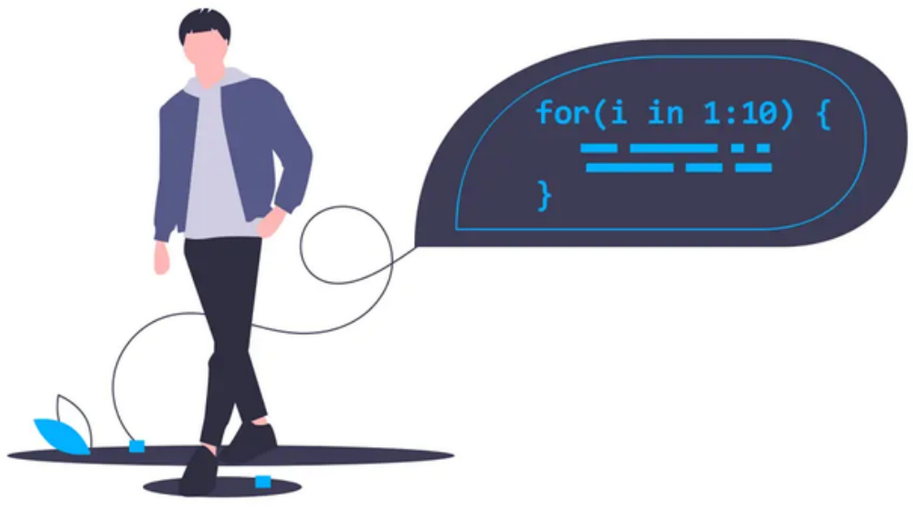
Perulangan for ini biasanya digunakan untuk perulangan yang sudah jelas perlu dilakukan berapa kali.

- Perulangan While

Perulangan While biasanya digunakan jika jumlah perulangan tidak diketahui atau memiliki kemungkinan untuk dapat dilakukan kurang dari batas perulangan

- Perulangan Do While

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

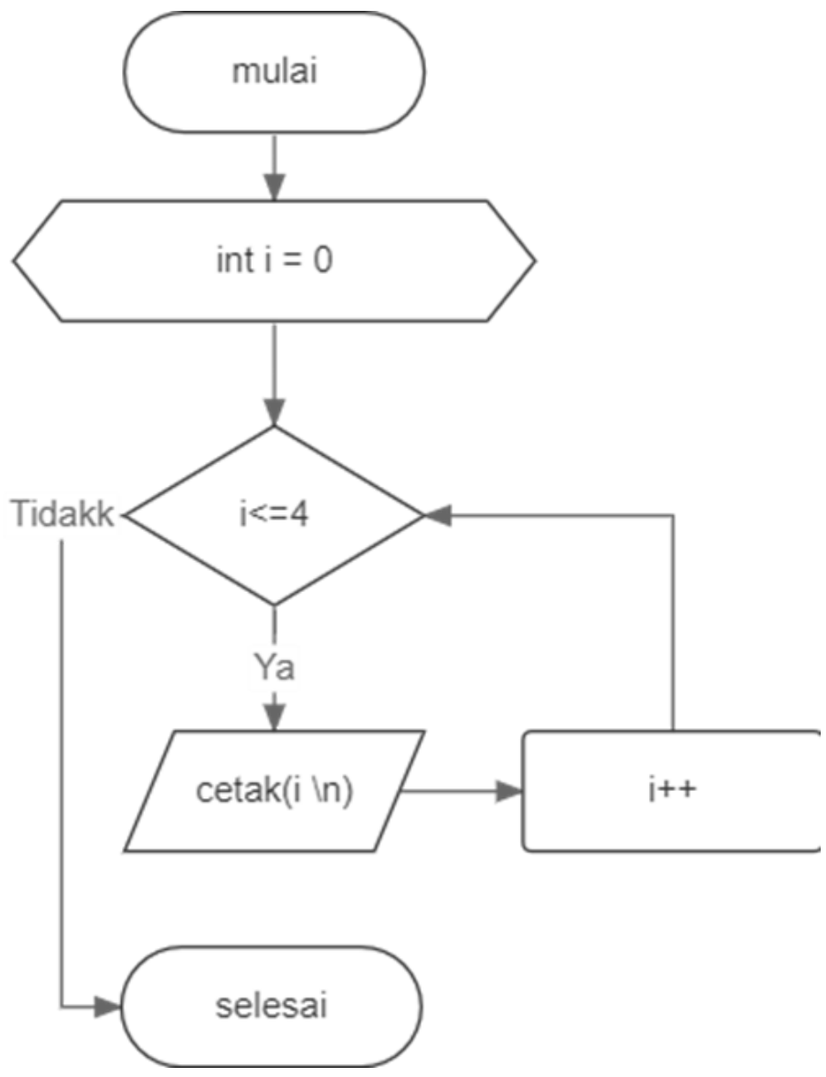


Perulangan For

Perulangan menggunakan for biasanya digunakan untuk perulangan yang sudah jelas dilakukan berapa kali. Dengan kata lain jumlah perulangan sudah jelas dan diketahui oleh pembuat program dari awal.

Perulangan for dibagi menjadi dua
1. For untuk hitung naik
2. For untuk hitung turun

Perulangan For untuk Perhitungan Naik



Gambar 1 Flowchart For mencetak 0 sampai 4

```
for (expr1; expr2; expr3)
{
    <statement>;
    ...
}
```

Gambar 2 struktur for

```cpp
#include <iostream>

using namespace std;

int main() {
    for(int i = 0;i<=4;i++){
        cout<<i<<endl;
    }

}
```

Gambar 3 Code For Hitung Naik dari 0 Sampai 4 pada C++

jdoodle.com/ia/NeC

Keterangan:

- int i = 0 (merupakan bagian inisialisasi)
- cout<<i<<endl (merupakan bagian proses)
- i++ (merupakan increment)
- i<=4 (merupakan terminasi/kondisi perulangan)

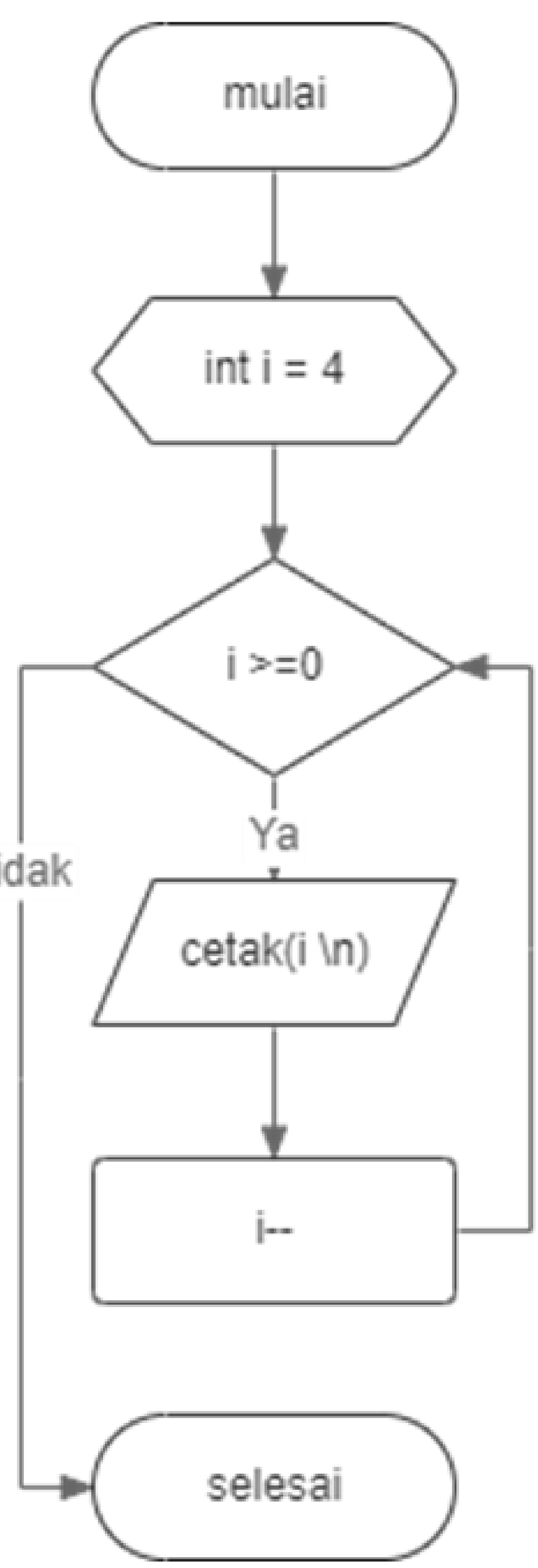
Inisialisasi : Tahapan persiapan membuat kondisi awal sebelum melakukan perulangan. Misalnya mengisi variable dengan nilai awal

Proses: Tahapan proses terjadi di dalam bagian perulangan di mana berisi semua proses yang perlu dilakukan secara berulang

increment: Kondisi penambahan agar perulangan dapat terus berjalan

Terminasi: Kondisi berhenti dari perulangan. Kondisi berhenti ini penting agar tidak terjadi perulangan yang tanpa henti. Kondisi perulangan merupakan kondisi yang harus dipenuhi oleh jalannya algoritma untuk masuk ke dalam blok perulangan (Sukamto, 2018)

Maka output yang akan dicetak adalah
0
1
2
3
4
Pengulangan For untuk perhitungan turun



Gambar 4 Flowchart Perulangan For untuk Perhitungan turun

```cpp
#include <iostream>

using namespace std;

int main() {
    for(int i = 4;i>=0;i--){
        cout<<i<<endl;
    }
}
```

Gambar 5 Code Perulangan For untuk Perhitungan turun

i = 4 (merupakan inisialisasi/nilai awal)
i>=0 (merupakan terminasi/batas atas)
i-- (merupakan iterasi)
cout<<i<<endl (merupakan proses)

jdoodle.com/ia/NeD

Output:
4
3
2
1
0

Contoh Soal dan Penyelesaian

Miftah ingin membuat sebuah aplikasi perulangan dalam bahasa pemrograman C++ untuk menuliskan angka-angka secara berurutan. Diketahui angka yang akan dicetak adalah bilangan bulat dari 0 sampai 4 dengan new line. Diketahui bahwa batas awal dan akhir sudah diinisialisasi pada variable i. Buatkan codenya dalam bahasa C++?

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.Apakah saja data yang diketahui dan penting?

-Miftah ingin membuat sebuah aplikasi perulangan yang menuliskan angka-angka secara berurutan

-Miftah mengetahui bahwa angka yang dituliskan adalah dari angka 0 sampai angka 4 tanpa new line.

-Batas awal dan akhir sudah diinisialisasi dengan variable i dengan tipe data int (bilangan bulat)

-Bahasa Pemrograman yang digunakan adalah C++

2. Apakah data yang diberikan sudah cukup untuk menjawab permasalahan?

Data yang diberikan sudah dapat menjawab permasalahan. Data-data yang diberikan pada soal sudah cukup untuk menentukan algoritma yang paling sesuai, pembuatan flowchart, dan pembuatan 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 for .. ?”, Jika pada soal ditambahkan keterangan, perulangan akan melakukan pengecekan kondisi di awal dan perulangan ini batas atas dan bawahnya tidak hanya berupa angka bisa saja suatu kondisi. 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 for? Bila pada soal diberikan informasi: Batas atas perulangan yang digunakan ini hanyalah berupa angka dan bisa juga char. Namun diberikan informasi juga bahwa batas atas ini bisa berupa boolean yang membandingkan a % 2 == 0. Maka data yang diberikan akan kontradiktif karena pada kalimat pertama dikatakan **Hanya berupa angka dan bisa juga char.**

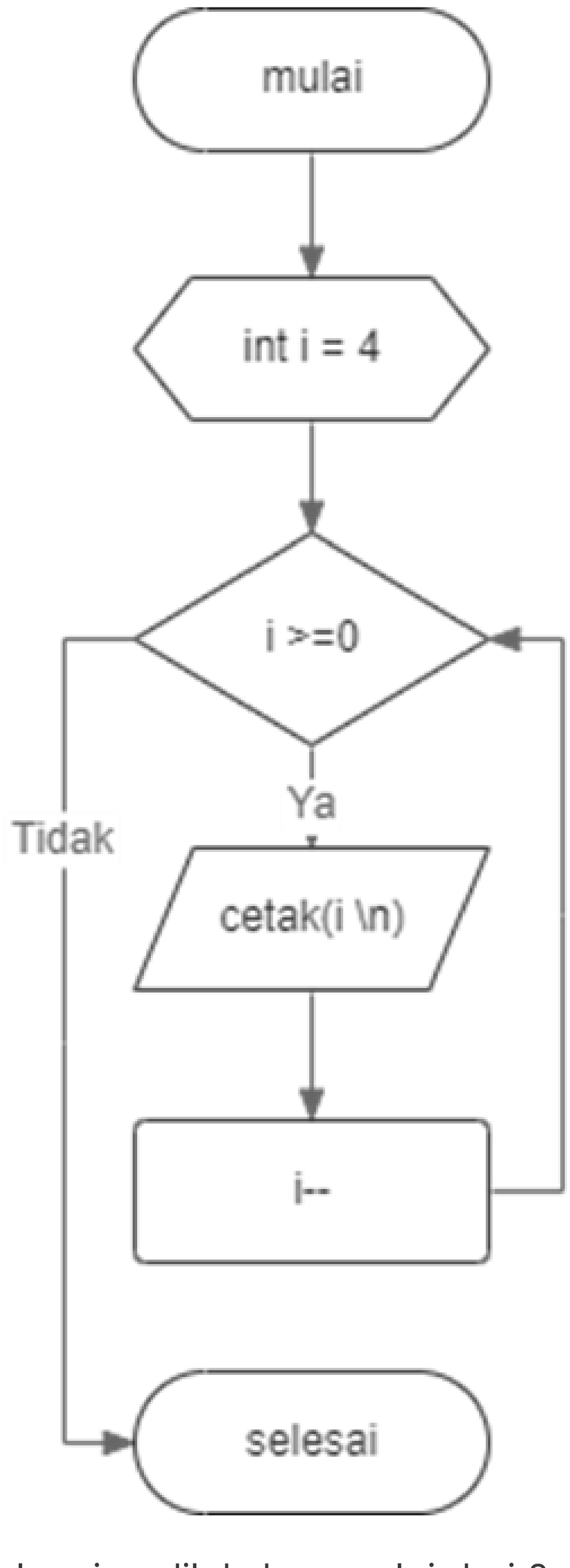
Kondisi dinyatakan cukup bila data yang diberikan sudah cukup untuk menjawab pertanyaan. Contohnya terdapat soal: “Apakah kondisi tersebut cukup untuk menentukan bahwa perulangan yang digunakan adalah perulangan for ?”, Bila pada soal tersebut disebutkan bahasa pemrograman yang digunakan dan disebutkan juga batas-batasnya berupa integer atau char

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 for?”, Bila pada soal tidak diberikan bahasa pemrogramannya yang jelas atau ketika batas-batasnnya tidak jelas.

Kondisi dinyatakan tidak ada konteks yang tepat bila tidak menggambarkan sama sekali perulangan

3. Bila pada soal tidak diberi tahu batas atas perulangannya apakah data yang diberikan cukup untuk menentukan perulangan yang digunakan adalah for?
Bila tidak diberikan batas atas perulangan maka kita tidak dapat menentukan perulangan yang digunakan, maka data yang diberikan tidak cukup

Tahap 2 (Membuat Rencana Penyelesaian Masalah):
1. Kira-kira kondisi apakah yang mirip dengan kasus tersebut?
Jawab: For, Dikarenakan tadi telah disebutkan bahwa terdapat batas awal dan batas akhir yang jelas yaitu menggulang dari angka 0 sampai 4
2. Kira-kira bagaimana bentuk flowchartnya
Jawab:



Artinya proses looping dilakukan mulai dari 0 sampai dengan 4 karena batas terminasinya i<=4 (I lebih kecil atau sama dengan 4) yang artinya 4 ikut termasuk yang dicetak.

Tahap 3 (Melaksanakan Rencana Penyelesaian Masalah):
1. Bagaimana implementasi masalah tersebut dalam pemrograman C++?

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4      for(int i = 0;i<=4;i++){
5          cout<<i<<endl;
6      }
7      return 0;
8  }
```

Tahap 4 (Memeriksa Kembali):
1. Apakah kode tersebut sudah benar dan menghasilkan output dengan benar. Misal kodenya begini ?

```
1  #include <iostream>
2  using namespace std;
3  int main(){
4      for(int i = 0;i<=4;i++){
5          cout<<i<<endl;
6      }
7      return 0;
8  }
```

Jawab: Kode tersebut tidak menghasilkan error dan hasilnya pun tepat
0
1
2
3
4

2. Bagaimana agar kode tersebut menghasilkan output yang berbeda misalnya

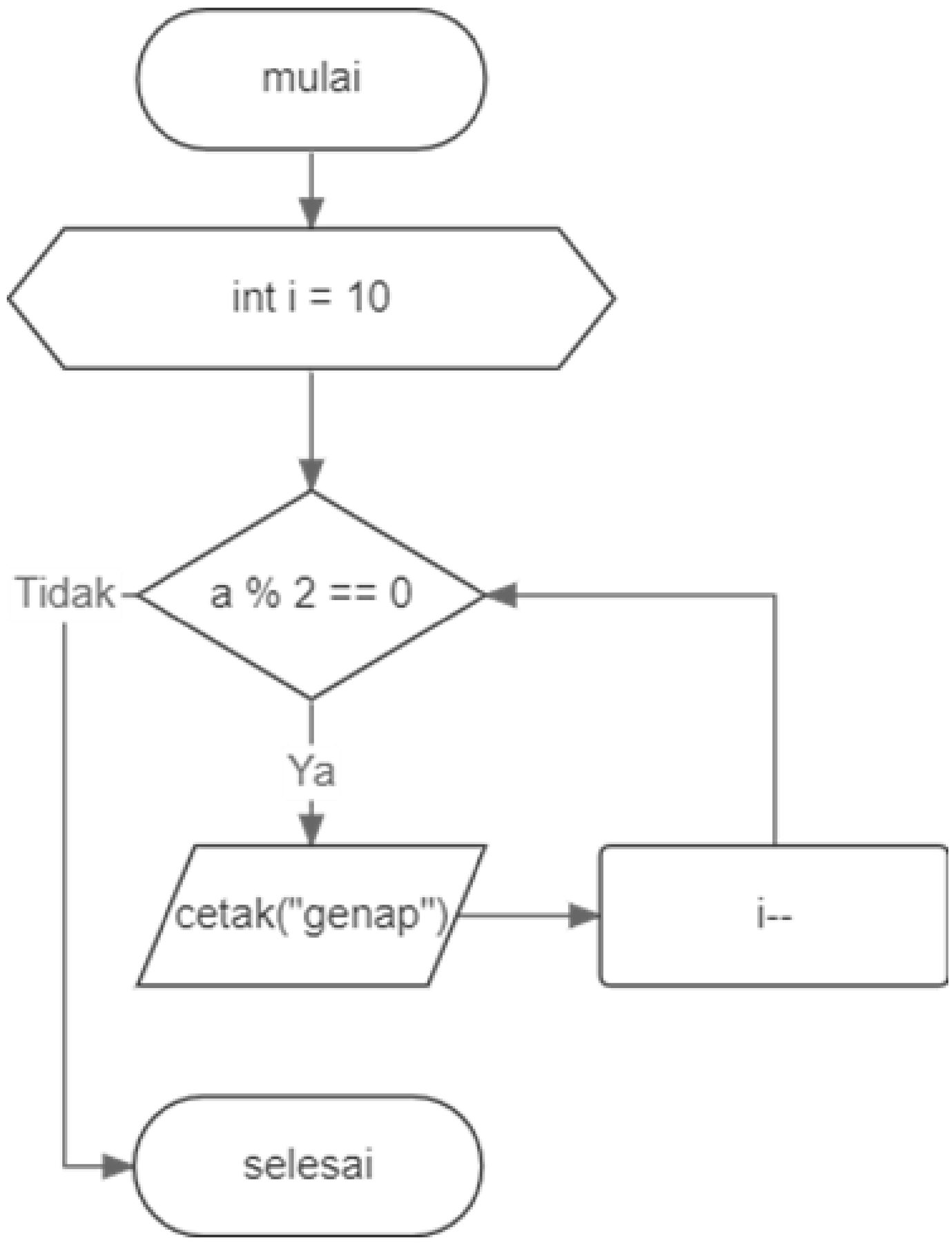
4
3
2
1
0

Jawab: Bisa tinggal ubah for menaik menjadi for menurun

```
#include <iostream>
using namespace std;
int main() {
    for(int i = 4;i>=0;i--){
        cout<<i<<endl;

    }
    return 0;
}
```

Perulangan While

Perulangan while biasa digunakan jika jumlah perulangan tidak diketahui atau memilikin kemungkinan dapat dilakukan kurang dari batas perulangan. Perulangan while ini akan melakukan perulangan selama kondisi perulangan terpenuhi. (Sukamto, 2018)



Gambar 6 Flowchart Perulangan While untuk Mencetak Genap Selama Bilangan Genap

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

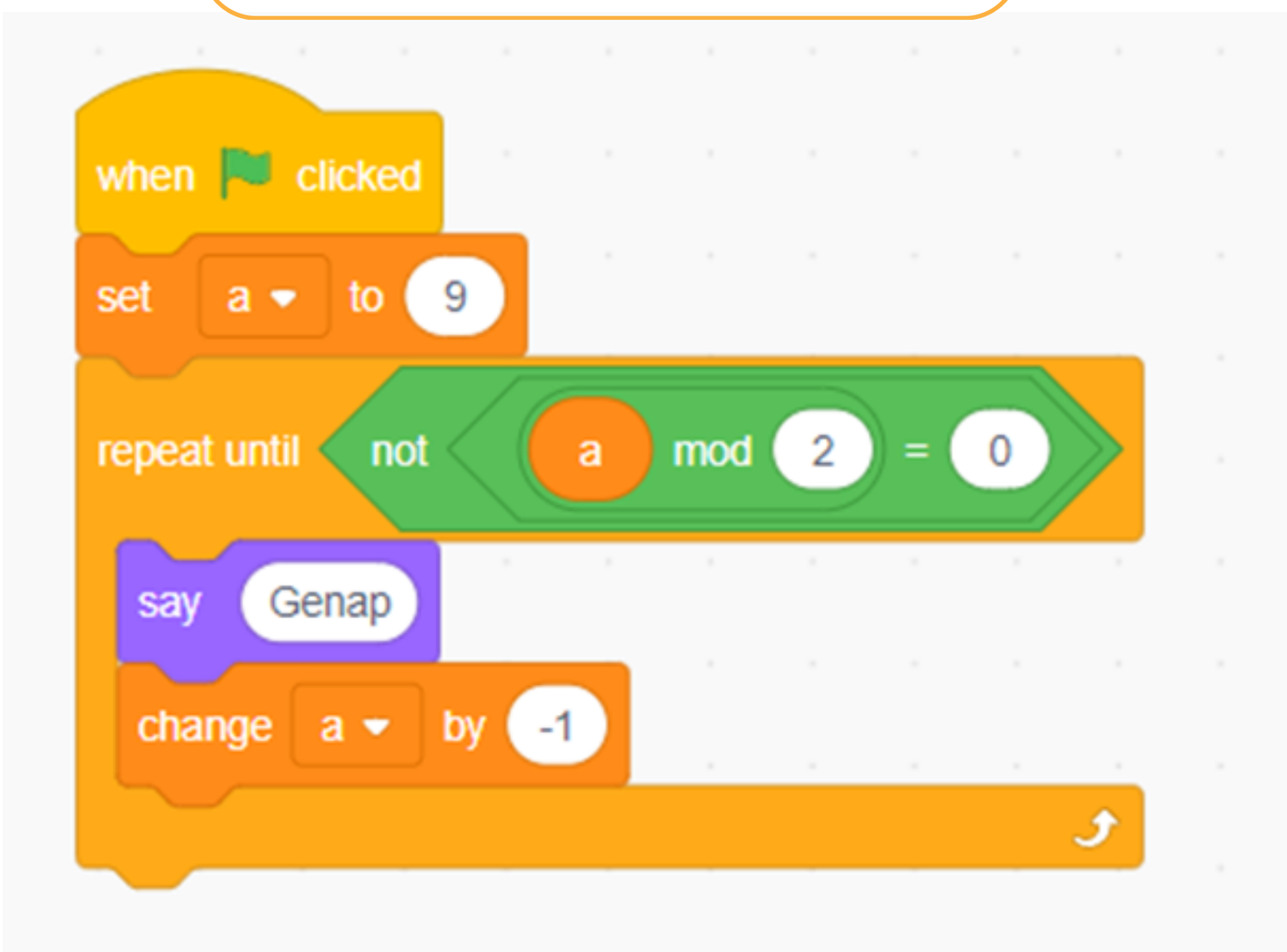
Gambar 7 struktur while

```cpp
#include <iostream>

using namespace std;

int main() {
    int a = 10;
    while(a % 2 == 0){
        cout<<"genap"<<endl;
        a--;
    }
    return 0;

}
```

Gambar 8 Code Perulangan While untuk Mencetak Genap Selama Bilangan Genap Pada C++

jdoodle.com/ia/NeI

Penjelasan: Sebelum memasuki badan perulangan kondisi terlebih dahulu diperiksa apakah memenuhi atau tidak, Aksi dikerjakan berulang kali selama kondisinya bernilai benar. Jika kondisi bernilai salah maka kondisi tidak dikerjakan atau berhenti dikerjakan (Munir & Lidya, 2016) Dalam kasus tersebut outputnya adalah Genap kenapa hanya satu kali diulang, karena pada perulangan kedua kondisi sudah tidak dipenuhi karena a = 9 dan 9 % 2 tidak sama dengan 0



Gambar 9 Code Perulangan While (Repeat Until) Untuk Mencetak Genap Selama Bilangan Genap Pada Scratch

https://s.id/1Z3f5

Penjelasan kode tersebut akan melakukan pengecekan di awal repeat until (dikerjakan sampai) dengan a mod 2 ! = 0. Perhatikan ini berbeda dengan while loop kalau while loop perulangan dilakukan jika kondisi terpenuhi. Sedangkan repeat until, perulangan dilakukan sampai. Dalam kasus ini Perulangan dilakukan sampai a mod 2 tidak sama dengan 0. Dalam hal ini perulangan tidak dilakukan karena pengulangan dilakukan sampai a mod 2 != 0 dan memang benar 9 mod 2 != 0.

Tahap 2 (Merencanakan Pemecahan Masalah)

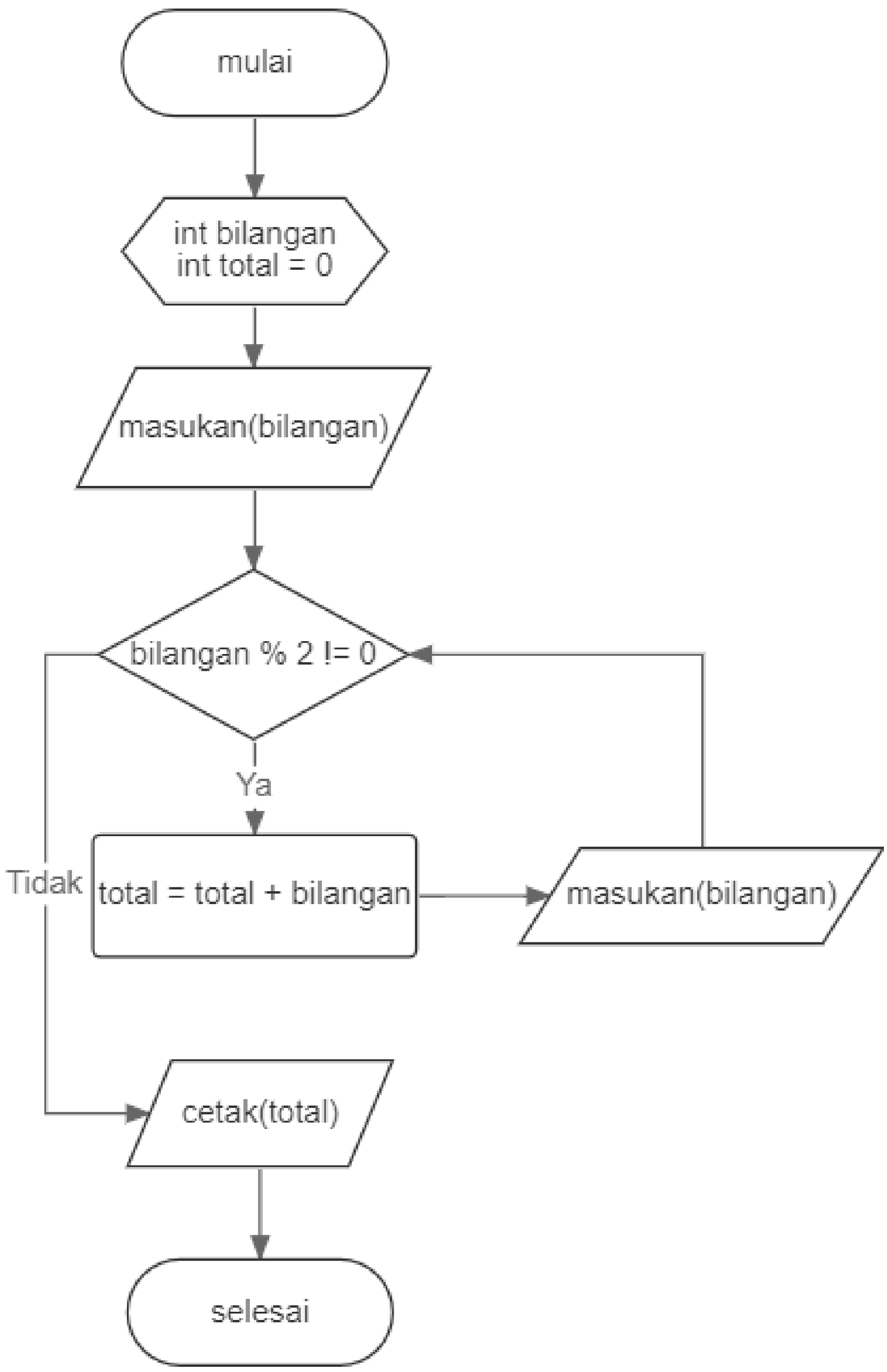
1. Kondisi apakah yang mirip dengan kasus tersebut

Jawab:

Perulangan While dikarenakan kita tidak mengetahui berapa kali perulangan yang dilakukan dan pengecekan dilakukan di awal sebelum masuk ke dalam blok perulangan.

2. Bagaimana flowchart diagramnya? (untuk bahasa C++)

Jawab:



Tahap 3 (Melaksanakan Pemecahan Masalah)
Jawab:

```cpp
#include <iostream>
using namespace std;
int main(){
    int bilangan;
    int total = 0;
    cin>>bilangan;
    while(bilangan % 2 != 0){
        total = total + bilangan;
        cin>>bilangan;
    }
    cout<<total;
}
```

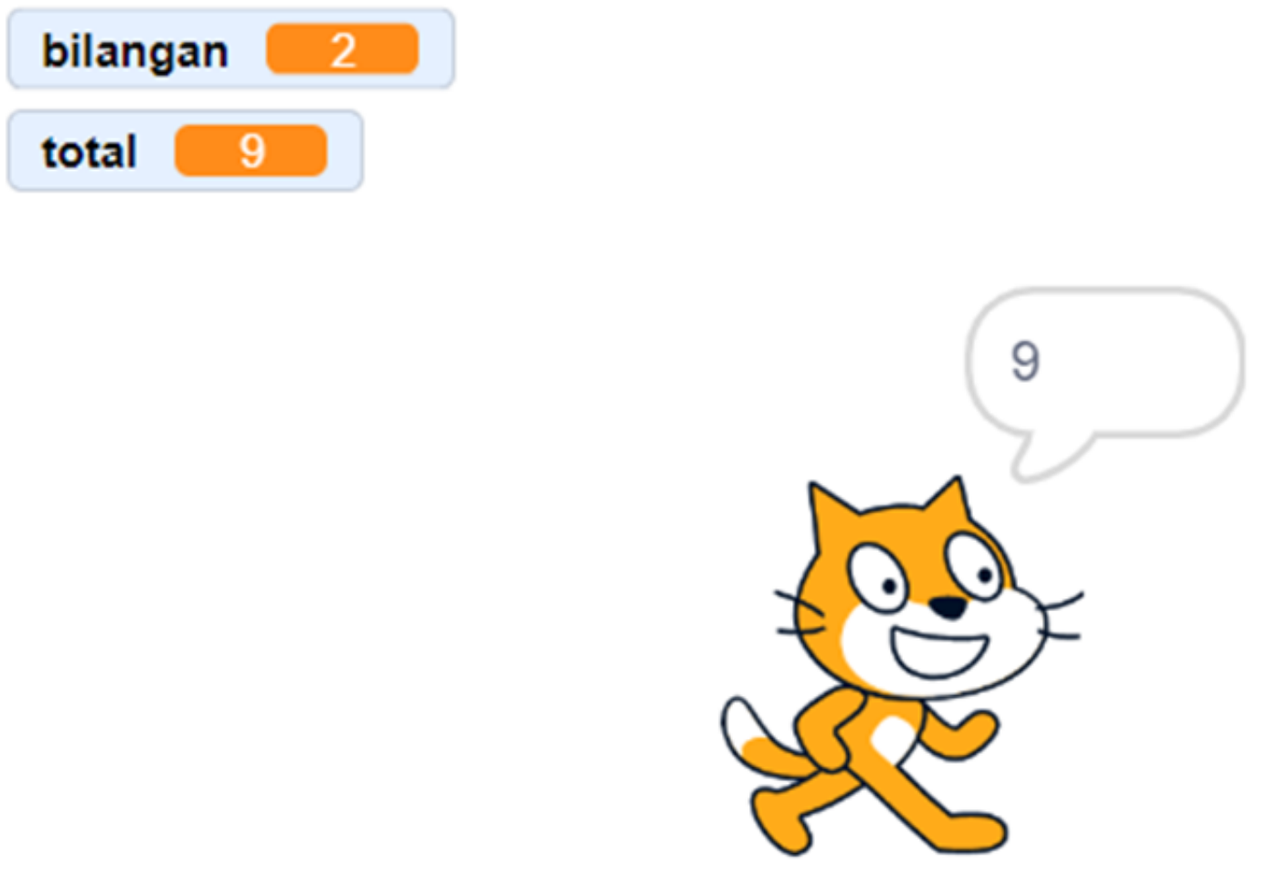
Tahap 4 (Memeriksa Kembali)

1. Apakah hasil pemecahan masalah sudah benar?

Jawab:

Pemecahan masalah tersebut sudah sesuai dengan yang diminta yaitu menghitung total dari bilangan ganjil dan berhenti ketika yang dimasukan adalah bilangan genap

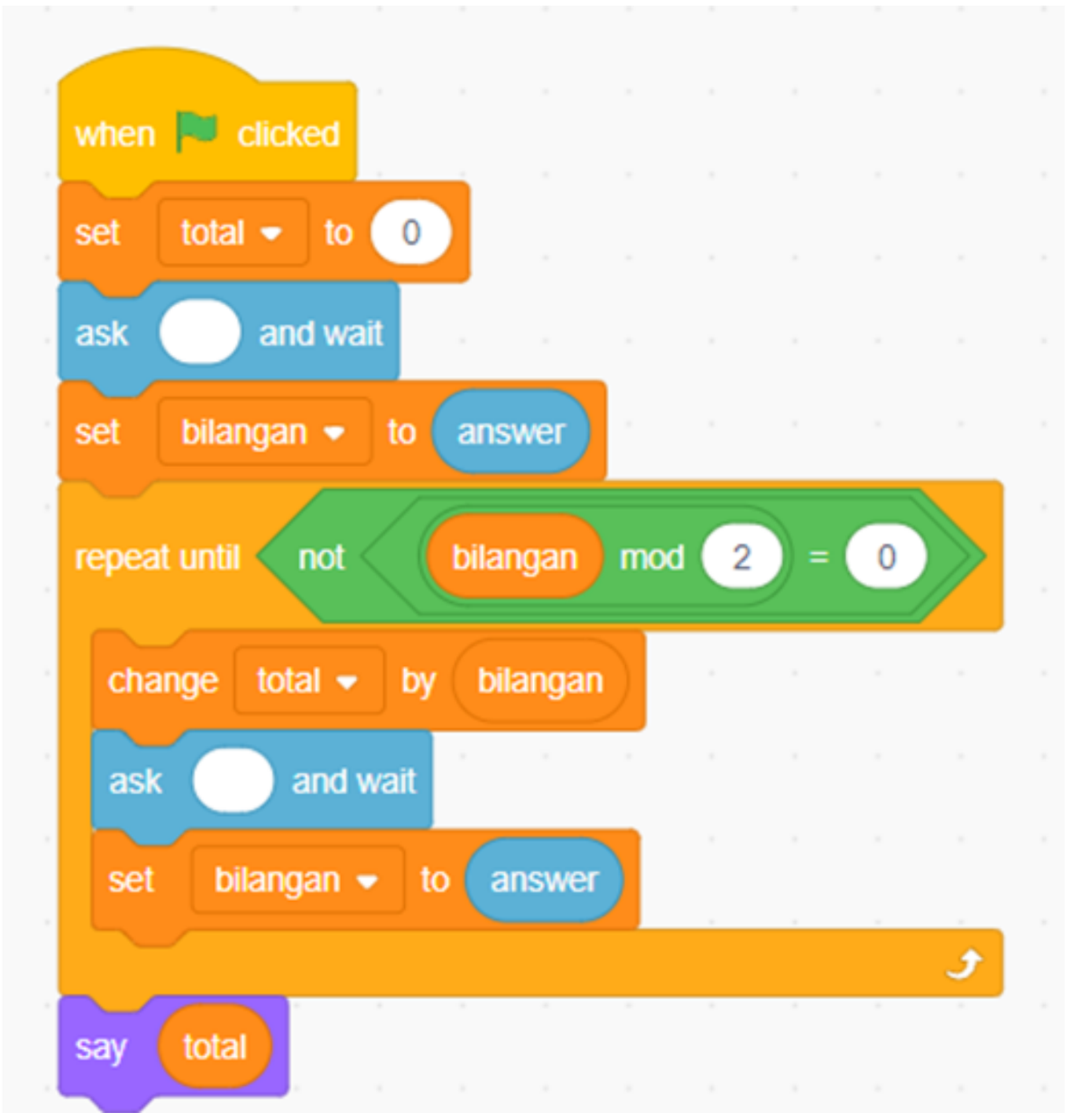
```
1
3
5
2
9
--------------------------------
Process exited after 2.715 seconds with return value 0
Press any key to continue . . .
```



2. Bagaimana agar kode tersebut menghasilkan total dari bilangan genap dan berhenti ketika yang dimasukan adalah bilangan ganjil?

```cpp
#include <iostream>
using namespace std;
int main() {

    int bilangan;
    int total = 0;
    cin>>bilangan;
    while(bilangan % 2 ==0){
        total = total + bilangan;
        cin>>bilangan;
    }
    cout<<total;
    return 0;
}
```

Dengan menggubah kondsi pada while yang sebelumnya bilangan % 2 != 0 menjadi bilangan % 2==0



Dengan menggubah kondisi pada repeat until menjadi not bilangan mod 2 = 0 dari sebelumnya bilangan mod 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.

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