

# **Kode Semu (Pseudocode) dan Diagram Alir (*Flowchart*)**

**Putra Pandu Adikara**

Program Studi S1 Teknik Informatika

Jurusan Teknik Informatika

Fakultas Ilmu Komputer UB

2019

# DAFTAR ISI

DAFTAR ISI.....	2
1    Pendahuluan .....	3
2    SELEKSI KONDISI.....	4
2.1    IF.....	4
2.2    IF..ELSE .....	5
2.3    IF..ELSE IF..ELSE .....	6
3    PENGULANGAN (ITERASI) .....	7
3.1    FOR.....	7
3.2    WHILE..DO.....	9
3.3    DO..WHILE atau REPEAT..UNTIL.....	10
4    PROSEDUR, FUNGSI, DAN PEMANGGILANNYA DI PROGRAM UTAMA.....	12
4.1    Prosedur ( <i>Procedure</i> ) .....	12
4.2    Fungsi ( <i>Function</i> ).....	12
4.3    Pemanggilan Prosedur/Fungsi di Program Utama.....	13
DAFTAR REFERENSI .....	14

# 1 PENDAHULUAN

Dalam cara berpikir komputasi (*computational thinking*), suatu masalah diselesaikan dengan langkah-langkah terstruktur (algoritmik) yang seefektif dan seefisien mungkin sehingga menghasilkan sebuah solusi masalah. Banyak masalah yang dapat diselesaikan dengan algoritme umum (*generic*), misalnya saja untuk menyelesaikan suatu masalah yang ternyata masalah dasarnya adalah suatu labirin (*maze*) maka bisa menggunakan proses *backtracking* dan rekursif, perhitungan deret Fibonacci, proses pemindahan di *Hanoi tower* menggunakan teknik rekursif, masalah yang lain dapat dipecahkan dengan cara teknik *brute force* meskipun tidak efektif, dll. Supaya suatu masalah dapat diselesaikan dengan baik, maka langkah penting yang tidak boleh dilupakan adalah perancangan. Setelah melalui proses perancangan baru dilakukan proses implementasi rancangan menjadi suatu program yang dapat menyelesaikan masalah.

Dalam pembelajaran pembuatan program di tahap awal, diajarkan mengenai pemrograman dengan pendekatan terstruktur, di tahap lanjut akan diajarkan dengan pendekatan berorientasi objek. Dalam pemrograman terstruktur ini, rancangan algoritme dapat dituliskan dalam bentuk kode semu atau yang dikenal dengan *pseudocode* dan dalam bentuk visual yang dikenal dengan diagram alir (*flowchart*). Dalam pemrograman berorientasi objek, rancangan program digambarkan menggunakan Unified Modelling Language (UML) yang sudah menjadi standar.

Dokumen ini menjelaskan mengenai dasar-dasar pembuatan diagram alir untuk mengetahui beberapa standar yang dapat digunakan dari beberapa sumber. Untuk dapat lebih memahami pembuatan diagram alir tersebut, maka disertakan juga contoh *pseudocode*-nya. Cakupan dalam dokumentasi pembuatan diagram alir ini hanya mengenai *program flowchart*. Topik-topik yang dibahas antara lain (1) struktur kendali berupa seleksi kondisi, (2) pengulangan, dan (3) fungsi/prosedur, dan cara pemanggilan fungsi/prosedur.

## 2 SELEKSI KONDISI

Seleksi kondisi digambarkan dengan notasi belah ketupat dengan kondisi benar/*true* atau salah. Kondisi Benar/*True* atau Salah/*False* harus dituliskan labelnya di alur yang sesuai. Perintah yang dijalankan di kondisi yang sesuai digambarkan secara: 1) simetris ke kanan dan kiri, atau 2) asimetris ke bawah dan ke kanan. Penggambaran ini disarankan secara simetris untuk lebih memudahkan pembacaan alur. Penggambaran ini dilakukan secara **konsisten**, artinya:

- Jika simetris maka harus simetris semua, begitu juga jika asimetris maka asimetris semua.
- Jika True ke kanan dan False ke kiri, maka semua diagram digambar dengan cara yang sama, begitu juga jika True ke bawah dan False ke kanan maka digambar semua seperti itu.

### 2.1 IF

#### Algoritme JikaHujan1()

{ Program yang akan menampilkan bawa payung atau tidak tergantung apabila hari sedang hujan atau tidak }

#### DEKLARASI

boolean hujan

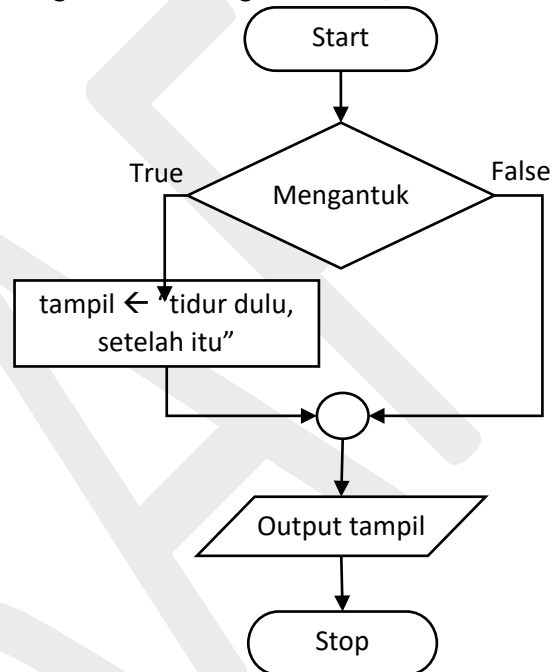
string tampil

#### DESKRIPSI

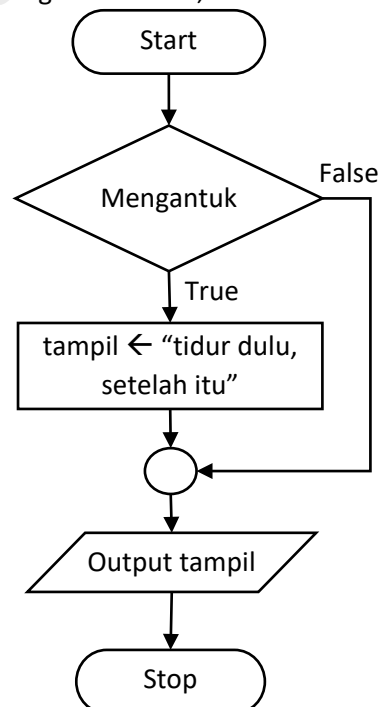
**if** mengantuk = True **then**  
    tampil ← "tidur dulu, setelah itu"

tampil ← tampil + "ayo jalan-jalan"  
Output tampil

Digambarkan dengan simetris, alur ke kanan dan ke kiri



Digambarkan dengan asimetris, alur ke bawah dan ke kanan



## 2.2 IF..ELSE

### Algoritme JikaHujan1()

{ Program yang akan menampilkan bawa payung atau tidak tergantung apabila hari sedang hujan atau tidak }

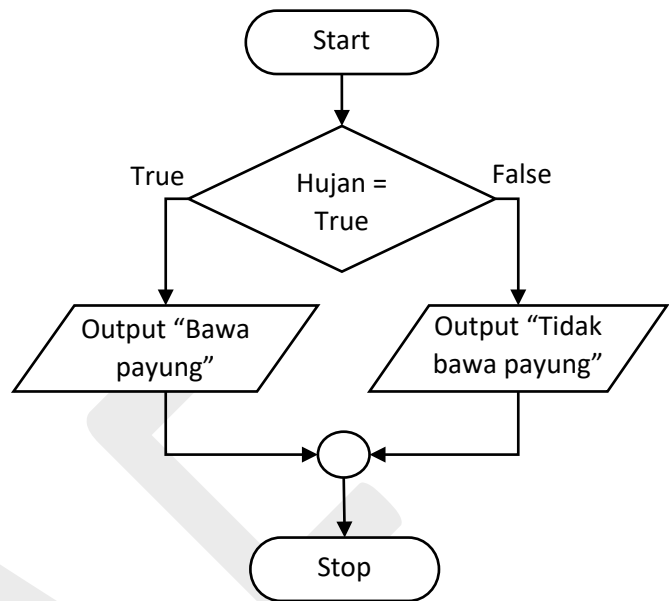
#### DEKLARASI

boolean hujan

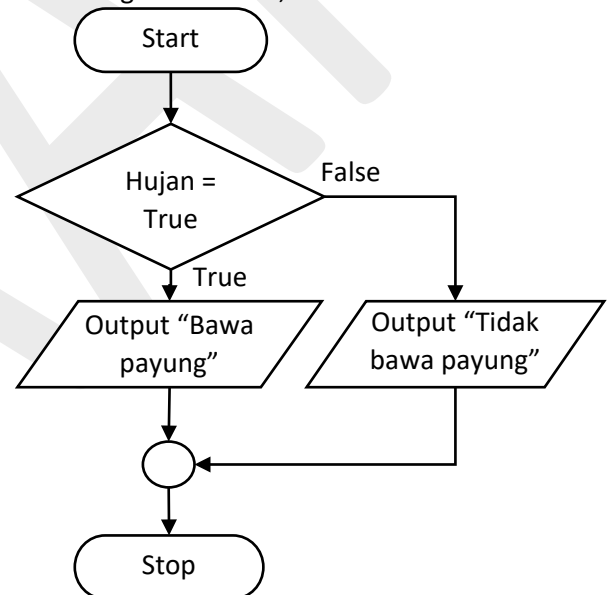
#### DESKRIPSI

```
if hujan = True then
    Output "Bawa payung"
else
    Output "Tidak Bawa payung"
```

Digambarkan dengan simetris, alur ke kanan dan ke kiri



Digambarkan dengan asimetris, alur ke bawah dan ke kanan



## 2.3 IF..ELSE IF..ELSE

### Algoritme JikaHujan1()

{ Program yang akan menampilkan kegiatan tergantung tiap harinya }

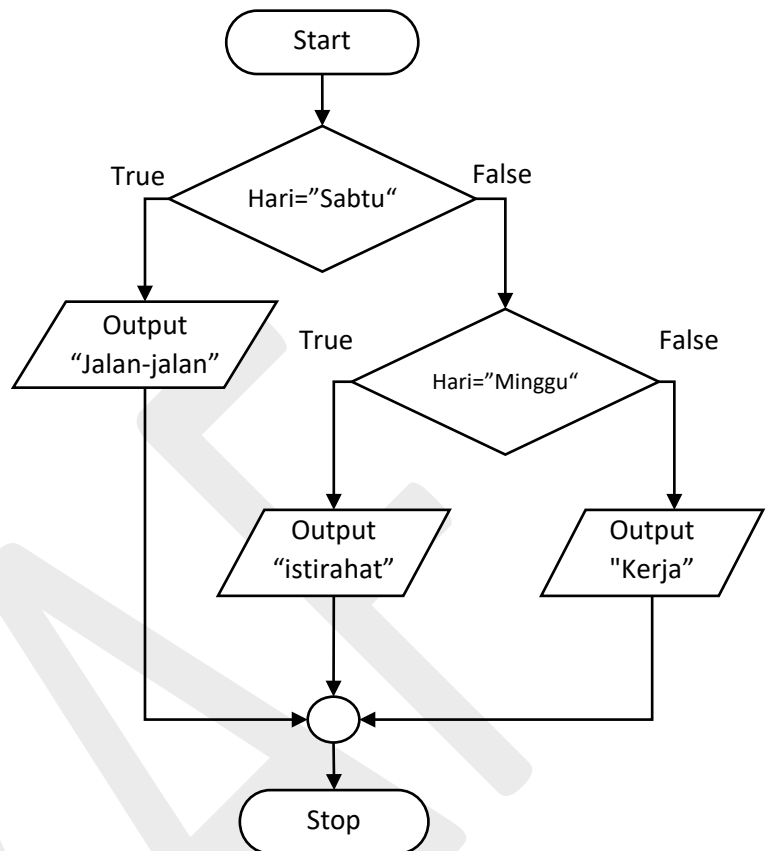
#### DEKLARASI

boolean Hari

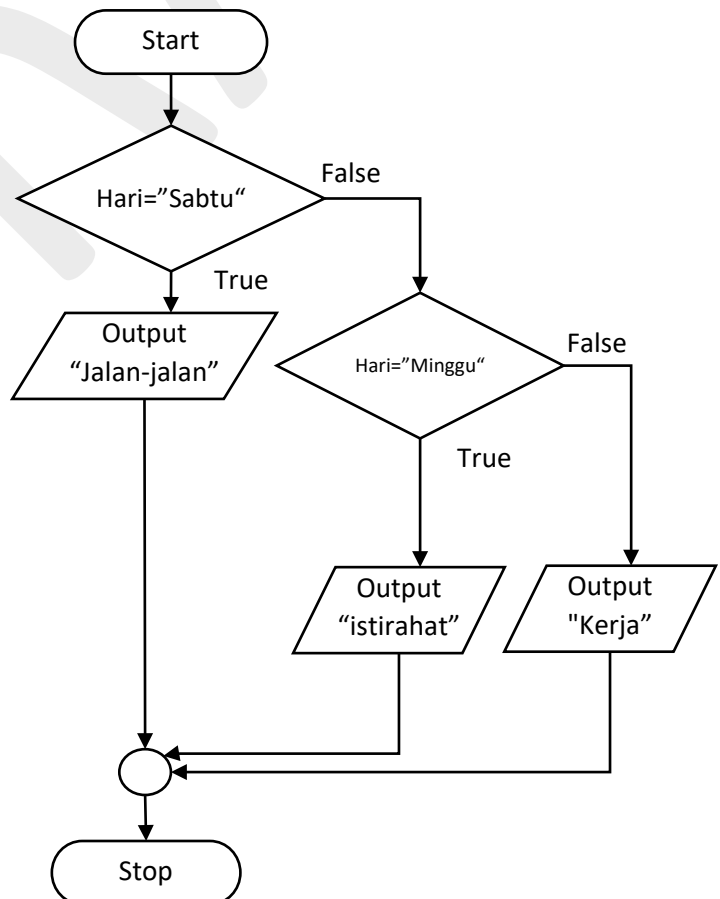
#### DESKRIPSI

```
if Hari = "Sabtu" then
    Output "Jalan-jalan"
else if Hari = "Minggu" then
    Output "Istirahat"
else
    Output "Kerja"
```

Digambarkan dengan simetris, alur ke kanan dan ke kiri



Digambarkan dengan asimetris, alur ke bawah dan ke kanan

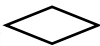




### 3 PENGULANGAN (ITERASI)

Struktur pengulangan dimaksudkan untuk melakukan proses pengulangan dari beberapa instruksi dalam sejumlah pengulangan tertentu

Jumlah pengulangan dapat ditentukan sebelumnya atau ditentukan dalam proses pelaksanaan pengulangan.

Untuk kasus pengulangan ini, ada beberapa varian penggambaran simbol untuk pengulangan, antara lain:

- Farrell (Farrell, 2014) dengan notasi 
- Flowgorithm (Flowgorithm, 2019) dengan notasi , dan
- ISO 5807-1985 (International Organization for Standardization, 1985) dengan notasi 

#### 3.1 FOR

FOR digunakan untuk pengulangan sebanyak sejumlah  $x$  kali (isi dari suatu pencacah/counter) dengan *pre-check loop*, artinya pengecekan dilakukan di awal pengulangan.

##### Algoritme PenjumlahanFor()

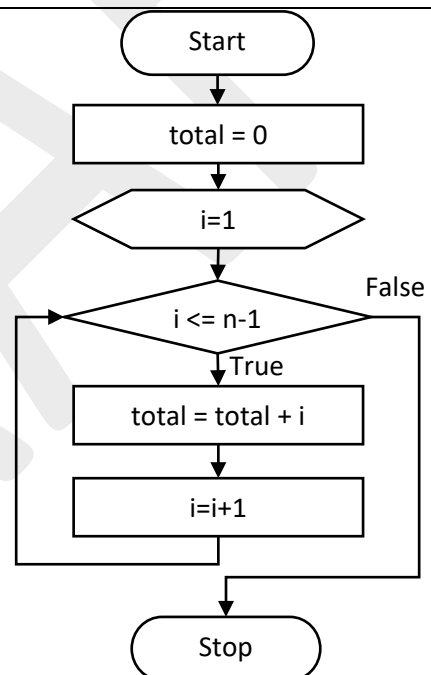
{Penjumlahan dilakukan dari angka 1 hingga  $n-1$ }

##### DEKLARASI

$i, n$ : integer  
total: integer

##### DESKRIPSI

```
total = 0
for  $i \leftarrow 1$  to  $n-1$  do
    total = total + i
end for
```



(Diadaptasi dari Farrell, J., 2014)

##### Algoritme PenjumlahanFor()

{Penjumlahan dilakukan dari angka 1 hingga  $n-1$ }

#### DEKLARASI

i, n:integer

total: integer

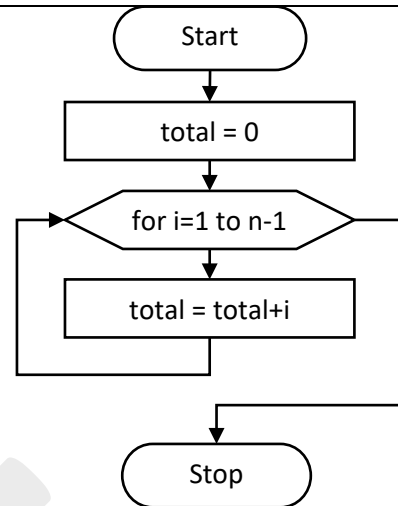
#### DESKRIPSI

total = 0

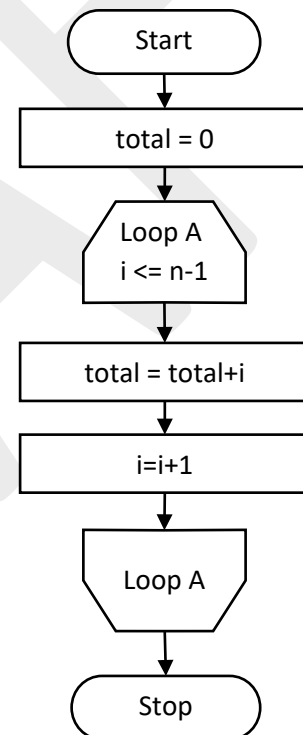
**for** i  $\leftarrow$  1 **to** n-1 **do**

total = total + i

**end for**



(Diadaptasi dari Flowgorithm)



(Diadaptasi dari ISO 5807-1985)



### 3.2 WHILE..DO

WHILE..DO digunakan untuk pengulangan selama kondisi tertentu terpenuhi. Pengulangan WHILE..DO juga *pre-check loop*.

#### Algoritme PenjumlahanWhile()

{Penjumlahan dilakukan dari angka 1 hingga  $n-1$ }

#### DEKLARASI

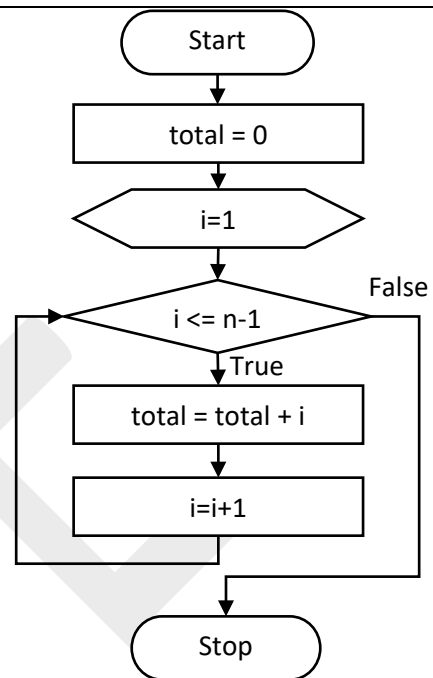
$i, n$ : integer  
total: integer

#### DESKRIPSI

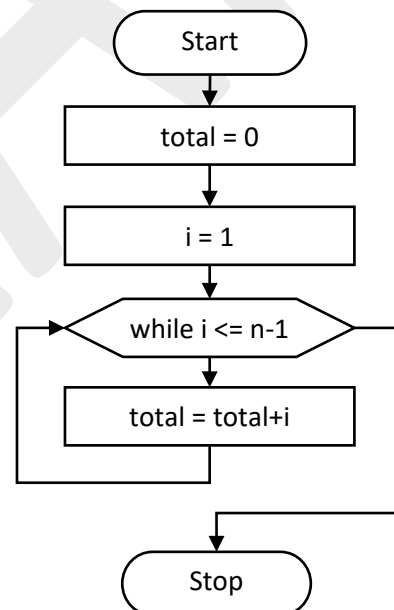
```
total = 0
i = 1
while i <= n-1 do
    total = total + i
    i = i+1
end while
```

#### NB:

Dalam kasus ini, flowchart antara *for* dan *while* mirip meski *pseudocode* yang hampir sama. Perhatikan baik-baik bedanya, yaitu inisialisasi variabel  $i$  dan isi simbol pengulangannya.



(Diadaptasi dari Farrell, J., 2014)



(Diadaptasi dari Flowgorithm)

#### Algoritme PenjumlahanWhile()

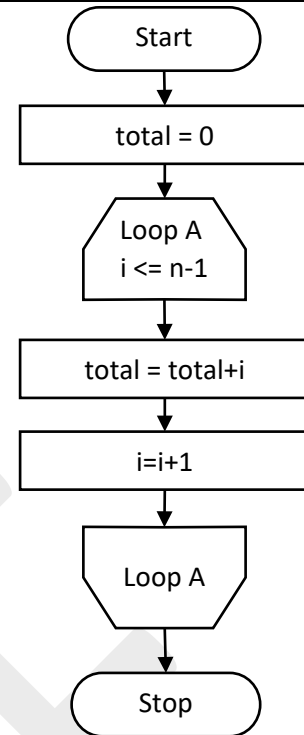
{Penjumlahan dilakukan dari angka 1 hingga n-1 }

#### DEKLARASI

i, n:integer  
total: integer

#### DESKRIPSI

```
total = 0
i = 1
while i <= n-1 do
    total = total + i
    i = i+1
end while
```



(Diadaptasi dari ISO 5807-1985)

### 3.3 DO..WHILE atau REPEAT..UNTIL

DO..WHILE digunakan untuk pengulangan selama kondisi tertentu terpenuhi namun bedanya DO..WHILE adalah *post-check loop*, artinya pengecekan kondisi dilakukan di akhir pengulangan. Dalam kasus ini pengulangan dilakukan minimal sekali meski kondisi tidak terpenuhi dari awal pengulangan. Dalam bahasa pemrograman tertentu DO..WHILE ini sinonim dengan REPEAT..UNTIL.

#### Algoritme PenjumlahanDoWhile()

{ Penjumlahan dilakukan dari angka 1 hingga n-1 }

#### DEKLARASI

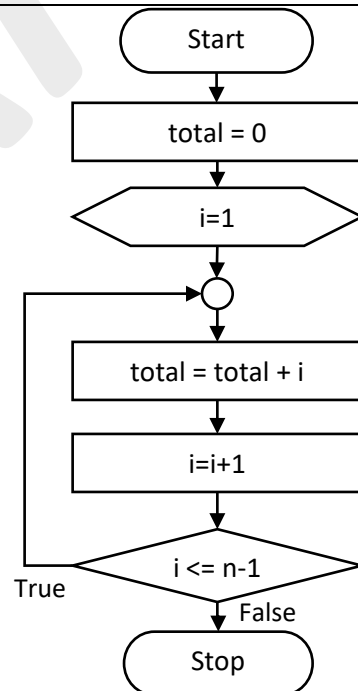
i, n:integer  
total: integer

#### DESKRIPSI

```
total = 0
i = 1
do
    total = total + i
    i = i+1
while i <= n-1
```

#### NB:

Dalam kasus ini, flowchart antara *for* dan *while* mirip meski *pseudocode* yang hampir sama. Perhatikan baik-baik bedanya, yaitu inisialisasi variabel *i* dan isi simbol pengulangannya.



(Diadaptasi dari Farrell, J., 2014)

#### Algoritme PenjumlahanDoWhile()

{ Penjumlahan dilakukan dari angka 1 hingga  $n-1$  }

#### DEKLARASI

$i, n$ : integer  
total: integer

#### DESKRIPSI

total = 0

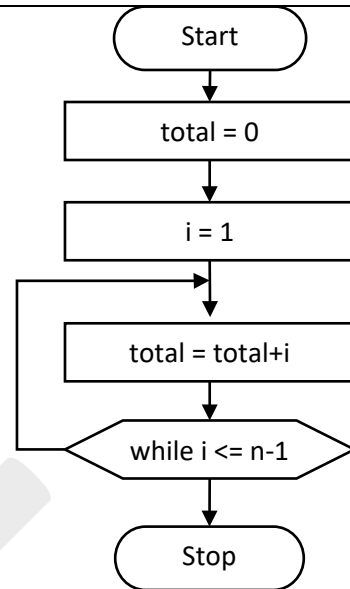
$i = 1$

**do**

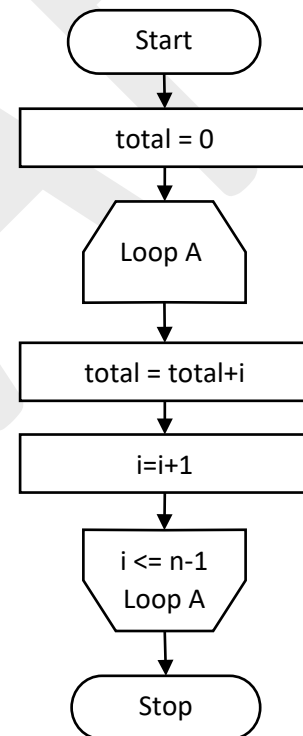
total = total +  $i$

$i = i + 1$

**while**  $i \leq n-1$



(Diadaptasi dari Flowgorithm)



(Diadaptasi dari ISO 5807-1985)

## 4 PROSEDUR, FUNGSI, DAN PEMANGGILANNYA DI PROGRAM UTAMA

### 4.1 Prosedur (Procedure)

**Prosedur:** diawali dengan nama prosedur beserta argumen (parameter formal) yang menyertainya, diakhiri dengan return tanpa mengembalikan variabel/nilai apapun. Dimungkinkan program keluar dengan lebih dari dua terminator return tapi disarankan hanya satu return untuk memudahkan pembacaan alur program.

<p><b>procedure CetakMenu()</b> { Prosedur untuk menampilkan menu, tidak mengembalikan nilai balik, berbeda dengan fungsi }</p> <p><b>DEKLARASI</b> String tampil</p> <p><b>DESKRIPSI</b> Tampil <math>\leftarrow</math> "Masukkan nilai x dan y (dipisahkan spasi) " Output tampil</p>	<pre>graph TD; A([CetakMenu()]) --&gt; B[tampil &lt;- "Masukkan nilai x dan y (dipisahkan spasi) "]; B --&gt; C[/Output tampil/]; C --&gt; D([return]);</pre>
---	---

### 4.2 Fungsi (Function)

**Fungsi:** diawali dengan nama prosedur beserta argumen (parameter formal) yang menyertainya dan diakhiri dengan return suatu variabel/nilai. Dimungkinkan program keluar dengan lebih dari dua terminator return tapi disarankan hanya satu return untuk memudahkan pembacaan alur program.

<p><b>function Max(x, y)</b> { Fungsi Max mengembalikan nilai maksimum dari x, y }</p> <p><b>DEKLARASI</b> num z { penampung sementara dari nilai maksimum }</p> <p><b>DESKRIPSI</b> if x &gt; y then     z <math>\leftarrow</math> x else     z <math>\leftarrow</math> y return z</p>	<pre>graph TD; A([Max(x, y)]) --&gt; B{x &gt; y}; B -- True --&gt; C[z &lt;- x]; B -- False --&gt; D[z &lt;- y]; C --&gt; E(( )); D --&gt; E; E --&gt; F([return z]);</pre>
---	---

<p>atau tanpa z, dengan dua return</p> <p><b>DESKRIPSI</b>  <b>if</b> <math>x &gt; y</math> <b>then</b>              <b>return</b> x  <b>else</b>              <b>return</b> y</p>	<p>atau tanpa z, dengan dua return (kurang disarankan)</p> <pre> graph TD     Start([Max(x, y)]) --&gt; Decision{x &gt; y}     Decision -- True --&gt; ReturnX([return x])     Decision -- False --&gt; ReturnY([return y]) </pre>
--	--

### 4.3 Pemanggilan Prosedur/Fungsi di Program Utama

**Program utama:** diawali dengan mulai/start dan diakhiri dengan selesai/stop.

Pemanggilan fungsi/prosedur menggunakan simbol

Di dalam simbol, apabila tidak ada nilai yang dikembalikan maka dikatakan sebagai prosedur. Apabila merupakan fungsi maka nilai yang dikembalikan dari fungsi akan disimpan ke suatu variabel.

<p><b>Algoritme HitungNilaiMax()</b>  <i>{ Program utama untuk menghitung dua nilai antara x dan y }</i></p> <p><b>DEKLARASI</b>  <i>num x { variabel penyimpan nilai pertama }</i>  <i>num y { variabel penyimpan nilai kedua }</i>  <i>num nilai { variabel penampung nilai maksimum }</i></p> <p><b>DESKRIPSI</b>            CetakMenu()   <i>{ memanggil prosedur CetakMenu() }</i>            input x            input y            nilai <math>\leftarrow</math> max(x, y)       <i>{ memanggil fungsi max() }</i>            output nilai</p>	<pre> graph TD     Start([start]) --&gt; CetakMenu[CetakMenu()]     CetakMenu --&gt; Nilai[nilai &lt;- max(x,y)]     Nilai --&gt; Output[/Output nilai/]     Output --&gt; Stop([stop]) </pre>
--	--

## DAFTAR REFERENSI

Farrell, J., 2014. *Programming Logic and Design, Comprehensive*. 8th ed. Delmar Learning.

Flowgorithm, 2019. *Documentation*. [online] Available at: <<http://www.flowgorithm.org/documentation/index.htm>> [Accessed 14 Oct. 2019].

International Organization for Standardization, 1985. *Information Processing - Documentation Symbols and Conventions for Data, Program and System Flowcharts, Program Network Charts and System Resources Charts (ISO 05807-1985)*.

DRAFT