DAFTAR ISI

BAB IV : PEMILIHAN
A. Pengertian

B. Pemilihan 1 Kasus
1. Pendahuluan
2. Algoritma Pengecekan Negatif
3. Algoritma Pemberian Diskon

C. Pemilihan 2 Kasus
1. Pendahuluan
2. Algoritma Ganjil Genap
3. Algoritma Terbesar dari 2 Bilangan
4. Algoritma Menghitung Uang Lembur
5. Algoritma Tahun Kabisat
6. Algoritma Syarat Keanggotaan Suatu Jangkauan/Range
7. Algoritma Syarat Keanggotaan Diluar Range

D. Pemilihan Lebih dari 2 Kasus
1. Pendahuluan
2. Algoritma Menentukan Positif, Nol, atau Negatif
3. Algoritma Gaji Berdasarkan Golongan
4. Algoritma Penentuan Nomor Kuadran Titik Koordinat
5. Algoritma Tahun Kabisat
6. Algoritma Menentukan Grade Nilai Akhir
7. Algoritma Penambahan 1 Detik Pada Waktu
8. Algoritma Terbesar dari 3 Bilangan (Bandingkan Satu-Satu)
9. Algoritma Terbesar dari 3 Bilangan (Pohon Keputusan)
10. Algoritma Terbesar Dari 3 Bilangan (Sekuensial)

DOSEN : HERJUN JURISANTO PUR
Herjun_jp@yahoo.co.id
Jalan Letjend. R. Soeprapto, Muka Kuning
Telp : (0778) 364 035, 450 111 Fax : (0778) 364 187
Batam - Indonesia

BAB IV
PEMILIHAN

A. Pengertian

Dalam bab **runtunan**, setiap baris proses dalam algoritma akan selalu dikerjakan. Namun, ada saat dimana kita menginginkan satu atau beberapa proses dalam algoritma tersebut **hanya dikerjakan apabila memenuhi syarat atau kondisi tertentu**.

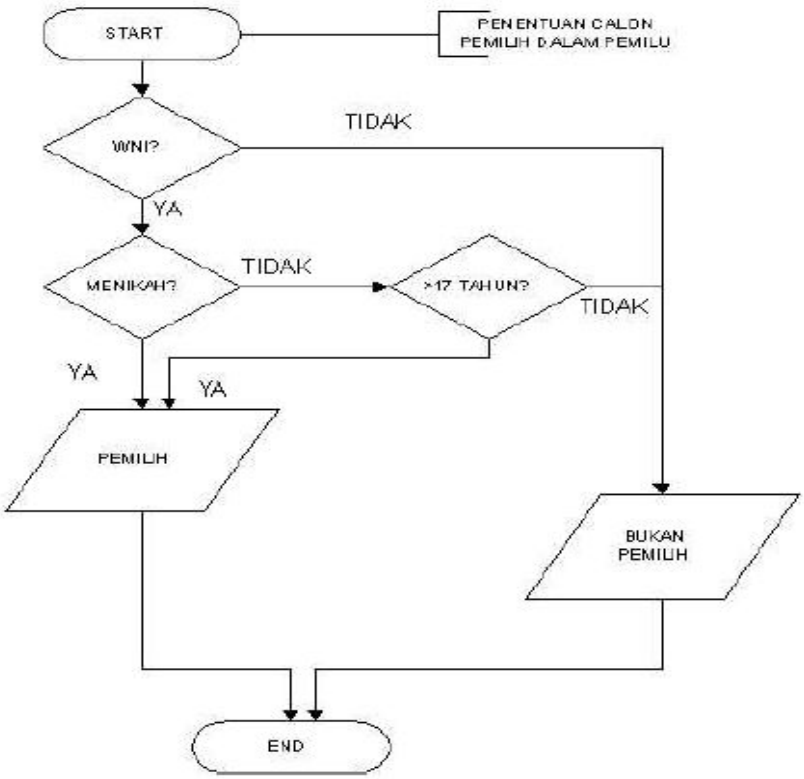
Dalam algoritma, kita dapat menggunakan struktur percabangan IF.

Bentuk penulisan struktur IF adalah

```
IF kondisi THEN
proses_1
...
proses_n
END IF
```

Atau

```
 IF kondisi THEN
proses_1
...
proses_n ELSE
proses_a
 ...
proses_x
END IF
```



Kondisi dalam struktur IF di atas adalah suatu nilai boolean yang dapat bernilai TRUE atau bernilai FALSE. Kondisi dalam hal ini dapat berupa **suatu ekspresi yang menghasilkan nilai boolean (ekspresi boolean)** atau juga dapat berupa **suatu variabel yang bertipe boolean**. Dalam struktur IF di atas, proses_1 hingga proses_n **hanya akan dikerjakan apabila kondisi bernilai TRUE**. Sedangkan proses_a hingga proses_x **hanya akan dikerjakan apabila kondisi bernilai FALSE**. Hal yang perlu diperhatikan dalam struktur IF adalah bahwa struktur IF tidak harus memiliki bagian ELSE. Bagian ELSE digunakan apabila ada proses yang hendak dikerjakan untuk kondisi bernilai FALSE. Perhatikan pula bahwa proses ditulis sedikit menjorok ke dalam. Hal ini disebut dengan *indent*. Indent sangat diperlukan untuk memudahkan dalam pembacaan suatu algoritma. Mulai saat ini, perhatikan setiap indent yang terdapat pada contoh algoritma dalam buku ini.

B. Pemilihan 1 Kasus

1. Pendahuluan

Dalam algoritma, terkadang suatu proses hanya dikerjakan bila memenuhi kondisi tertentu, misalnya mencetak pesan kesalahan bila terjadi kesalahan input dari user. Tentu saja bila user tidak melakukan kesalahan input, maka pesan kesalahan tersebut tidak boleh tercetak. Contoh lainnya adalah pemberian diskon belanja untuk pembeli yang membeli dengan jumlah harga minimal tertentu. Tentu saja bila pembeli tidak mencapai jumlah pembelian minimum yang telah ditentukan, maka pembeli tidak akan diberikan diskon. Contoh-contoh ini adalah contoh-

contoh algoritma untuk satu kasus, yaitu kasus suatu proses hendak dikerjakan atau tidak dikerjakan.

2. Algoritma Pengecekan Negatif

Buatlah algoritma untuk mencetak kata "negatif" apabila user menginputkan suatu bilangan negatif. Analisis: Suatu bilangan dikatakan negatif apabila bilangan tersebut kurang daripada nol.

Langkah pengerjaan:

1. Input suatu bilangan, misalkan n
2. Cek kondisi n<0. Apabila bernilai true, cetaklah kata "negatif".

```
10 Algoritma Pengecekan_Negatif
20 {Diinput suatu bilangan, apabila negatif, maka cetaklah kata "negatif"}
30 Deklarasi:
40 n : real
50 Deskripsi
60 Read(n)
70 If n<0 then
80 write('negatif')
90 end if
```

3. Algoritma Pemberian Diskon

Sebuah toko yang sedang melakukan promosi memberikan diskon sebesar 10% untuk pembeli yang melakukan transaksi minimal Rp100.000,00.

Buatlah algoritma untuk menampilkan diskon dan total yang harus dibayar pembeli.
Analisis: input: Total belanja pembeli (belanja) output: besarnya diskon (diskon) dan total pembayaran (total)

Langkah pengerjaan:

1. input total belanja pembeli
2. cek kondisi belanja ≥ Rp100.000,00. Apabila bernilai true maka menghitung diskon dengan rumus diskon = 10% x belanja.
3. Hitung total pembayaran dengan rumus total = belanja – diskon
4. Cetak nilai diskon dan total pembayaran.

```
10 Algoritma Pemberian_Diskon
20 {Diinput total belanja pembelian. Diskon sebesar 10% diberikan dengan syarat total belanja minimal Rp100.000,-. Algoritma memberikan output besarnya diskon yang diperoleh dan total yang harus dibayarkan pembeli.}
```

```
30 Deklarasi:
40 belanja : integer
50 diskon, total : real
60 Deskripsi
70 read(belanja)
80 if belanja >= 100000 then
90 diskon ←0.1 * belanja
100 end if
110 total ←belanja – diskon
120 write(diskon, total)
```

Dalam algoritma di atas, baris ke-09 **hanya akan dikerjakan** apabila memenuhi kondisi belanja ≥ 100000. Akibatnya, apabila kondisi belanja ≥ 100000 bernilai FALSE, maka perhitungan baris ke-11 adalah ilegal karena variabel diskon belum diberi nilai. Maka variabel diskon sebelumnya haruslah diberi nilai awal (inisialisasi).

Algoritma yang lengkap dapat dilihat di bawah ini.

```
10 Algoritma Pemberian_Diskon
20 {Diinput total belanja pembelian. Diskon sebesar 10% diberikan dengan syarat total belanja minimal Rp100.000,-. Algoritma memberikan output besarnya diskon yang diperoleh dan total yang harus dibayarkan pembeli.}
30 Deklarasi:
40 belanja : integer
50 diskon, total : real
60 Deskripsi
70 read(belanja)
80 diskon ←0 {inisialisasi}
90 if belanja >= 100000 then
100 diskon ←0.1 * belanja
110 end if
120 total ←belanja – diskon
130 write(diskon, total)
```

C. Pemilihan 2 Kasus

1. Pendahuluan

Kadang terdapat 2 proses yang saling berlawanan, dan tidak boleh dikerjakan dalam satu saat. Hal inilah yang disebut pemilihan untuk 2 kasus. Proses dipilih berdasarkan syarat atau kondisi, bila memenuhi maka dikerjakan proses pertama, jika tidak memenuhi maka dikerjakan proses kedua.

Sebagai contohnya, misalnya dalam menentukan apakah suatu bilangan bulat yang diinputkan user adalah bilangan genap atau bilangan ganjil.

Apabila yang diinputkan adalah bilangan genap, maka dicetak keterangan yang menyatakan bilangan genap.
Sebaliknya jika bukan bilangan genap, pastilah bilangan tersebut merupakan bilangan ganjil.

2. Algoritma Ganjil Genap

Buatlah algoritma untuk mencetak kata "GENAP" apabila bilangan bulat yang diinputkan user adalah bilangan genap, dan mencetak kata "GANJIL" apabila bilangan bulat yang diinputkan user adalah bilangan ganjil. Analisis: input: suatu bilangan bulat (n) Kita tahu bahwa suatu bilangan bulat apabila bukan genap, pastilah merupakan bilangan ganjil.
Jadi, kasus ganjil dan genap merupakan sesuatu yang berlawanan. Hal ini tergolong ke dalam struktur pemilihan 2 kasus. Syarat suatu bilangan dikatakan genap adalah bilangan tersebut **habis dibagi 2**. Dengan kata lain, bilangan n dibagi dengan 2 bersisa 0.
Operator untuk menghitung sisa hasil bagi adalah MOD. Jadi syarat bahwa n bilangan genap adalah n mod 2 = 0.

```
10 Algoritma Menentukan_Genap_Ganjil
20 {Mencetak "Genap" bila bilangan yang diinput user adalah genap, dan "Ganjil" bila ganjil}
30 Deklarasi:
40 n : integer
50 Deskripsi
60 read(n)
70 if n mod 2 = 0 then
80    write("GENAP")
90 else
100   write("GANJIL")
110 end if
```

Algoritma di atas dapat juga ditulis menjadi

```
10 Algoritma Menentukan_Genap_Ganjil
20 {Mencetak "Genap" bila bilangan yang diinput user adalah genap, dan "Ganjil" bila ganjil}
30 Deklarasi:
40 n : integer
50 genap : boolean
60 Deskripsi
70 read(n)
80 genap ←n mod 2 = 0
90 if genap = TRUE then
100   write("Genap")
110 else
120   write("Ganjil")
130 end if
```

Perhatikan baris ke-08. Kondisi atau syarat genap ganjilnya bilangan *n* dapat kita tampung di dalam suatu variabel bertipe boolean, dalam hal ini kita namakan *genap*.

Misalnya user menginput bilangan 4. Maka *n* berisikan bilangan bulat 4. Diperoleh 4 mod 2 = 0 adalah bernilai TRUE. Kemudian nilai TRUE tersebut akan diisikan ke variabel *genap*. Selanjutnya dalam baris ke-09, isi variabel *genap* dibandingkan dengan nilai boolean TRUE. Karena TRUE = TRUE adalah TRUE, maka algoritma menghasilkan output "GENAP".

Perhatikan tabel berikut:

Genap	Genap = TRUE	Not Genap	Genap = FALSE
TRUE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE

Ternyata nilai dari ekspresi *Genap = TRUE* adalah sama dengan nilai dari variabel *genap*. Berdasarkan pengamatan ini, maka baris ke-09 pada algoritma di atas dapat diganti menjadi *if genap then*.

```
...
if genap then
...
```

Algoritma di atas juga dapat ditulis dari sudut pandang GANJIL, yaitu syaratnya diganti menjadi *genap = FALSE*. Maka

```
...
if genap = FALSE then
        write('GANJIL')
else
        write("GENAP")
end if
```

dapat diganti menjadi

```
...
if not genap then
...
```

3. Algoritma Terbesar dari 2 Bilangan

Diinputkan dua buah bilangan bulat dari user. Buatlah algoritma untuk mencetak bilangan yang terbesar dari dua bilangan tersebut.

Analisis: Input: 2 bilangan bulat (*a* dan *b*) Apabila *a*>*b*, maka yang dicetak adalah *a*. Apabila *a*<*b*, maka yang dicetak adalah *b*. Tetapi apabila *a* = *b*, maka kita dapat mencetak *a* atau *b*. Oleh

karena itu, kita dapat menyimpulkan bahwa kita mencetak *a* apabila *a*≥*b*, dan mencetak *b* apabila terjadi sebaliknya.

```
10 Algoritma Menentukan_Terbesar_dari_2_Bilangan_Bulat
20 {Diinput 2 bilangan bulat. Kemudian mencetak yang terbesar}
30 Deklarasi:
40 a, b, terbesar : integer
50 Deskripsi
60 read(a,b)
70 if a >= b then
80 terbesar ←a
90 else
100 terbesar ←b
110 end if
120 write(terbesar)
```

4. Algoritma Menghitung Uang Lembur

Karyawan PT "ABC" digaji berdasarkan jumlah jam kerjanya selama satu minggu. Upah per jam adalah Rp2.000,00. Bila jumlah jam kerja lebih besar dari 48 jam, maka sisanya dianggap sebagai jam lembur. Upah lembur adalah Rp3.000,00.
Buatlah algoritma untuk menampilkan upah normal, uang lembur, dan total upah yang diterima karyawan.
Analisis:
input: jumlah jam kerja (n) output: upah normal (upah), uang lembur (lembur), dan total upah (total) Upah per jam, upah lembur, dan batas jam lembur dapat dijadikan sebagai konstanta. Kasus ini memberikan kita dua kemungkinan, yaitu apakah karyawan menerima lembur atau tidak menerima lembur (lembur = 0).

Syarat seorang karyawan menerima lembur adalah apabila n > 48. Uang lembur yang diterima adalah selisih jam kerja dengan batas jam lembur dikalikan dengan upah lembur. Oleh karena itu, dapat kita simpulkan bahwa rumus yang dipakai adalah

Apabila karyawan tidak mendapat uang lembur, maka
    lembur = 0, upah = n x 2000
Apabila karyawan mendapatkan uang lembur, maka
    lembur = (n – 48) x 3000, sedangkan upah = 48 x 2000, **bukan upah = n x 2000**.

```
10 Algoritma Upah_Karyawan
20 {menentukan upah mingguan karyawan. Upah normal Rp2000,-/jam dan upah lembur
Rp3000,-/jam. Apabila jam kerja karyawan lebih dari 48, maka sisanya dihitung lembur.
Algoritma menghasilkan output jumlah upah normal, jumlah uang lembur, dan total upah yang
diterima karyawan}
```

```
30 Deklarasi:
40 const upah_per_jam = 2000
50 const upah_lembur = 3000
60 const batas_lembur = 48
70 n, upah, lembur, total : integer
80 Deskripsi:
90 read(n)
100 if n > 48 then {menerima lembur}
110 upah ←batas_lembur * upah_per_jam
120 lembur ← (n – batas_lembur)*upah_lembur
130 else {tidak menerima lembur}
140 upah ←n * upah_per_jam
150 lembur ←0 {penting}
160 end if
170 total ←upah + lembur
180 write(upah, lembur, total)
```

5. Algoritma Tahun Kabisat

Tahun kabisat adalah tahun yang memenuhi syarat berikut: (a) kelipatan 4; (b) bukan kelipatan 100; (c) kelipatan 400. Jadi, perhatikan tabel berikut:

Tahun Kabisat?

Tahun	Kabisat?
1996	Ya
1900	Bukan
2000	Ya

Berdasarkan keterangan di atas, buatlah algoritma untuk menentukan apakah suatu tahun merupakan tahun kabisat atau bukan. Analisis: input: tahun (n) Perhatikan syarat tahun kabisat bagian (a) dan (b). Dari 2 syarat tersebut dapat kita simpulkan bahwa syaratnya adalah kelipatan 4 **dan bukan** kelipatan 100, **atau** kelipatan 400. Dengan demikian, maka syaratnya yang lengkap adalah (n mod 4 = 0) and (n mod 100 <> 0) or (n mod 400 = 0)

Percobaan:

Tahun (n)	n mod 4 = 0	n mod 100 <> 0	n mod 400 = 0	Hasil
1996	TRUE	TRUE	FALSE	TRUE
1900	TRUE	FALSE	FALSE	FALSE
2000	TRUE	FALSE	TRUE	TRUE

```
10 Algoritma Menentukan_Kabisat
20 {menentukan kabisat atau tidak dari suatu tahun}
```

```
30 Deklarasi:
40 n : integer
50 kabisat : boolean
60 Deskripsi
70 read(n)
80 kabisat ← (n mod 4 = 0) and (n mod 100 <> 0) or (n mod 400 = 0)
90 if kabisat then
100 write("KABISAT")
110 else
120 write("BUKAN KABISAT")
130 end if
```

6. Algoritma Syarat Keanggotaan Suatu Jangkauan/Range

Diketahui bahwa himpunan penyelesaian suatu pertidaksamaan adalah *HP* ={*x* | -3 ≤ *x* ≤3}. Diinputkan suatu bilangan x. Buatlah algoritma untuk mencetak "Penyelesaian" bila *x* € *HP* dan mencetak "Bukan Penyelesaian" bila *x* €/ *HP*.

Analisis:

Persyaratan -3 ≤ x ≤ 3 tidak memenuhi bentuk penulisan eksresi perbandingan yang benar. Maka persyaratan tersebut harus dipisah menjadi 2 bagian, yaitu x≥-3 dan x ≤ 3.

Kedua persyaratan tersebut harus digabung dengan menggunakan operator Boolean **AND** Oleh karena itu, persyaratan keanggotaan x secara lengkap adalah (*x* ≥ -3) *AND* (*x* ≤ 3).

```
10 Algoritma Menentukan_Keanggotaan_range
20 {Diinput suatu bilangan x. Algoritma menghasilkan TRUE apabila x
adalah anggota HP = {x | -3 <= x <= 3} dan mencetak FALSE bila
bukan}
30 Deklarasi:
40 x : real
50 syarat : boolean
60 Deskripsi
70 read(x)
80 syarat←x >= -3) and (x <= 3)
90 if syarat then
100 write(„Penyelesaian")
110 else
120 write(„Bukan Penyelesaian")
130 end if
```

7. Algoritma Syarat Keanggotaan Diluar Range

Diketahui bahwa himpunan penyelesaian suatu pertidaksamaan adalah *HP* ={*x* | *x*>3}U{*x* | *x* <- 3}. Diinputkan suatu bilangan x.

Buatlah algoritma untuk mencetak "Penyelesaian" bila *x* € *HP* dan mencetak "Bukan Penyelesaian" bila *x* €/ *HP*.
Analisis:
Perhatikan bahwa himpunan HP merupakan **gabungan** (**union**) dari dua himpunan, yaitu {x | x > 3} dan {x | x < -3}. Himpunan yang pertama mempunyai syarat x > 3 dan himpunan ke dua mempunyai syarat x < -3. Maka kedua persyaratan tersebut harus digabung dengan menggunakan operator boolean **OR** (Mengapa?). Oleh karena itu, persyaratan keanggotaan x secara lengkap adalah
(*x* > 3) *OR* (*x* < -3)

```
10 Algoritma Menentukan_Keanggotaan_range
20 {Diinput suatu bilangan x. Algoritma menghasilkan TRUE apabila x
adalah anggota HP = {x | x > 3} U {x | x < -3} dan mencetak FALSE
bila bukan}
30 Deklarasi:
40 x : real
50 syarat : boolean
60 Deskripsi
70 read(x)
80 syarat ← (x > 3) or (x < -3)
90 if syarat then
100 write("Penyelesaian")
110 else
120 write("Bukan Penyelesaian")
130 end if
```

D. Pemi l ihan Lebih dar i 2 Kasus

1. Pendahuluan

Terkadang kita harus memilih lebih dari 2 pilihan yang ada. Misalnya menentukan apakah suatu bilangan bulat yang diinputkan user adalah bilangan positif, negatif, atau nol. Contoh lainnya adalah dalam menentukan gaji pokok karyawan berdasarkan golongan (kepangkatan).

Apabila terdapat lebih dari 2 pilihan, maka kita dapat memandang kasus tersebut sebagai kasus 2 pilihan, yaitu YA atau BUKAN. Sebagai contoh, kasus menentukan positif, nol, atau negatif; dapat dipandang sebagai kasus dengan 2 pilihan saja, yaitu apakah bilangan tersebut adalah bilangan positif atau bukan. Apabila bukan, maka bilangan tesebut hanya dimungkinkan merupakan bilangan nol atau negatif. Dalam hal ini (nol atau negatif) merupakan kasus dengan 2 pilihan. Dengan kata lain, kasus lebih dari 2 pilihan dapat dipandang menjadi kasus 2 pilihan yaitu apakah tergolong pilihan pertama atau bukan. Kemudian untuk bagian *bukan* (else), pilihan yang ada telah tereduksi (berkurang) satu. Apabila masih lebih, maka pilihan masih bisa direduksi lagi dengan cara yang sama, yaitu memandang apakah tergolong pilihan kedua atau bukan.

2. Algoritma Menentukan Positif, Nol, atau Negatif

Buatlah algoritma untuk mencetak kata "Positif", "Nol", atau "Negatif" berdasarkan jenis bilangan yang diinput oleh user. Analisis: input : sebuah bilangan (x)

Langkah pengerjaan:
1. input bilangan x
2. tentukan apakah x positif, jika benar maka cetak "Positif"
3. Jika tidak, maka tentu x mungkin nol atau negatif.
4. Tentukan apakah x sama dengan nol, jika benar maka cetak "Nol"
5. jika tidak, maka cetaklah "Negatif"

```
10 Algoritma Menentukan_Positif_Nol_Negatif
20 {Mencetak positif, nol, atau negatif sesuai dengan jenis bilangan yang diinputkan oleh user}
30 Deklarasi:
40 x : real
50 Deskripsi
60 read(x)
70 if x > 0 then
80 write(„Positif“)
90 else
100 if x = 0 then
110 write(„Nol“)
120 else
130 write(„Negatif“)
140 end if
150 end if
```

3. Algoritma Gaji Berdasarkan Golongan

Suatu perusahaan menentukan gaji pokok karyawannya menurut golongannya. Besarnya gaji pokok berdasarkan golongannya dapat dilihat pada tabel berikut.

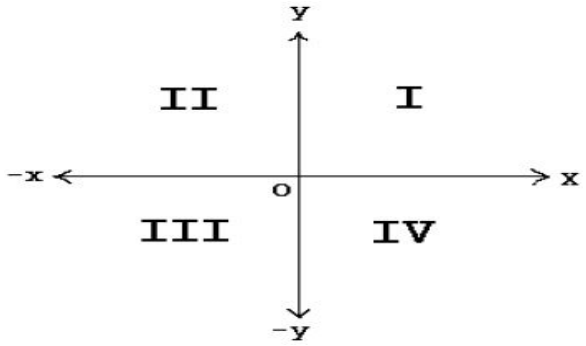
Buatlah algoritma untuk menentukan gaji pokok berdasarkan golongan yang diinput user.
Analisis:
input: Golongan (gol) bertipe **char**. Langkah pengerjaan:
1. Anggap kasus terdiri dari 2 pilihan, yaitu bergolongan A atau bukan. Apabila bukan, maka kasus tereduksi menjadi 3 pilihan, yaitu bergolongan B, C, atau D.
2. Anggap kasus terdiri dari 2 pilihan, yaitu bergolongan B atau bukan. Apabila bukan, maka kasus tereduksi menjadi 2 pilihan, yaitu bergolongan C atau D.

Golongan	Gaji Pokok
A	Rp400.000,00
B	Rp500.000,00
C	Rp750.000,00
D	Rp900.000,00

10 Algoritma Penentuan_Gaji_Pokok
20 {Menentukan gaji pokok berdasarkan golongan. Golongan A mendapat 400 ribu, golongan B mendapat 500 ribu, golongan C mendapat 750 ribu, dan golongan D mendapat 900 ribu.}
30 Deklarasi:
40 gol : char
50 gaji : integer
60 Deskripsi
70 read(gol)
80 if gol = "A"then
90 gaji←400000
100 else
110 if gol = "B" then
120 gaji ←500000
130 else
140 if gol = "C" then
150 gaji ←750000
160 else
170 gaji ←900000
180 end if
190 end if
200 end if
210 write(gaji)

4. Algoritma Penentuan Nomor Kuadran Titik Koordinat

Suatu titik dalam sistem sumbu koordinat dapat digolongan menjadi kuadran ke-1 hingga kuadran ke-4 berdasarkan letaknya dapat diperhatikan pada gambar berikut.



Ingat, bahwa apabila titik terletak pada sumbu (x maupun y) maka titik tersebut tidak terletak pada kuadran manapun. Buatlah algoritma untuk menentukan jenis kuadran berdasarkan suatu titik yang diinput oleh user. Analisis: Anda dapat memakai cara sebelumnya, yaitu memandangnya sebagai kasus dengan 5 pilihan, yaitu kuadran ke-1 hingga kuadran ke-4, ditambah "tidak di kuadran". Berdasarkan gambar di atas, maka syarat untuk masing-masing kuadran adalah Kemudian dengan teknik reduksi, kasus dengan 5 pilihan tersebut akan tereduksi menjadi 4 pilihan, dan seterusnya.

Kuadran	Syarat
I	(x > 0) and (y > 0)
II	(x < 0) and (y > 0)
III	(x < 0) and (y < 0)
IV	(x > 0) and (y < 0)

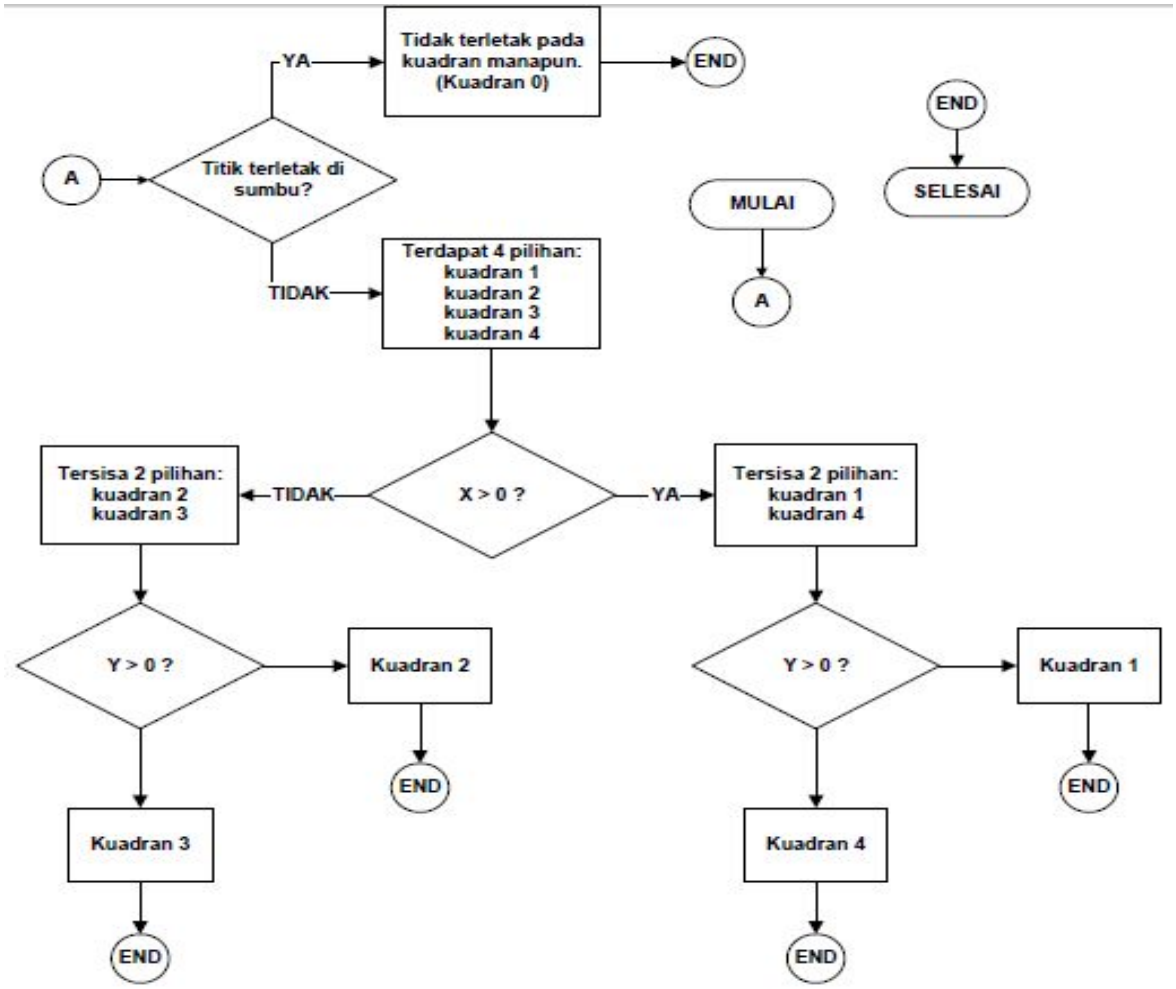
```
10 Algoritma Penentuan_Kuadran
20 {Menentukan jenis kuadran dari suatu titik koordinat}
30 Deklarasi:
40 x, y, kuadran : integer
50 Deskripsi
60 read(x,y)
70 if (x>0) and (y>0) then
80 kuadran ←1
90 else
100 if (x<0) and (y>0) then
110 kuadran ←2
120 else
130 if (x<0) and (y<0) then
140 kuadran ←3
150 else
160 if (x>0) and (y<0) then
170 kuadran ←4
180 else
190 kuadran ←0
200 end if
210 end if
220 end if
230 end if
240 Write(kuadran)
```

Apabila Anda perhatikan algoritma di atas, terlalu banyak syarat yang berulang. Perhatikan bahwa syarat (x>0) terdapat pada baris ke-07 dan baris ke-16. Hal ini mengisyaratkan bahwa syarat penentuan kuadran tersebut kurang terarah dan terstruktur. Maka kita dapat mengatur

kembali sudut pandang kita terhadap kasus ini. Urutan pengklasifikasian pilihan yang lebih terstruktur untuk kasus ini adalah

1. Anggap kasus terdiri dari 2 pilihan, yaitu titik terletak di salah satu kuadran, atau titik terletak di sumbu. Apabila terletak di sumbu, maka titik sudah pasti tidak terletak di salah satu kuadran. Syarat suatu titik terletak di salah satu sumbu adalah (x = 0) or (y = 0).
2. Jika tidak terletak di sumbu, maka tinggal 4 pilihan, yaitu terletak di kuadran ke-1, kuadran ke-2, kuadran ke-3, atau kuadran ke-4.
3. Kasus dengan 4 pilihan bisa kita kelompokkan menjadi 2 pilihan saja, yaitu apakah titik terletak di sebelah kanan sumbu-y (x>0) atau terletak di kiri sumbu-y (x<0).
4. Jika terletak di sebelah kanan sumbu-y maka tinggal 2 pilihan saja, yaitu titik terletak di kuadran ke-1 atau di kuadran ke-4, bergantung pada nilai y.
5. Jika tidak (terletak di sebelah kiri sumbu-y), maka tinggal 2 pilihan pula, yaitu titik terletak di kuadran ke-2 atau di kuadran ke-3 bergantung pada nilai y pula.

Untuk lebih jelasnya, bisa Anda perhatikan flowchart berikut ini.



```
10 Algoritma Menentukan_Kuadran
20 {Menentukan kuadran dari suatu titik koordinat}
30 Deklarasi:
40 x, y, kuadran : integer
50 Deskripsi
60 read(x,y)
70 if (x = 0) or (y = 0) then {terletak di salah satu sumbu}
80 kuadran ←0
90 else
100 if x > 0 then {di sebelah kanan sumbu-y}
110 if y > 0 then
120 kuadran ←1
130 else
140 kuadran←4
150 end if
160 else {x < 0, yaitu di sebelah kiri sumbu-y}
170 if y > 0 then
180 kuadran ←2
190 else
200 kuadran ←3
210 end if
220 end if
230 end if
240 write(kuadran)
```

5. Algoritma Tahun Kabisat

Mari kita analisa kembali syarat tahun kabisat, yaitu:

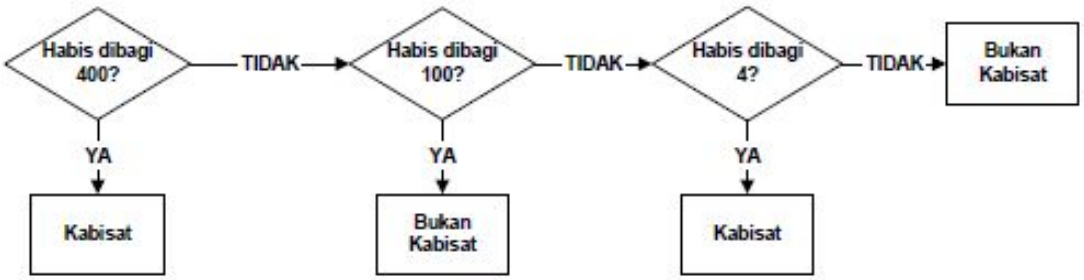
1. Tahun habis dibagi 400
2. Tahun tidak habis dibagi 100
3. Tahun habis dibagi 4

Perhatikan, bahwa syarat ke-2 lebih lemah dari syarat pertama, yaitu bahwa walaupun tahun habis dibagi 100, namun apabila tahun habis dibagi 400 maka tahun tersebut merupakan tahun kabisat. Apabila tahun habis dibagi 400 maka tahun jelas habis dibagi 4. Kesimpulan kita adalah bahwa apabila tahun habis dibagi 400, maka secara pasti tahun tersebut merupakan kabisat.

Apabila tidak habis dibagi 400, maka ada 2 kemungkinan, yaitu habis dibagi 100 atau tidak. Bila habis dibagi 100, maka secara pasti tahun tersebut **bukan** kabisat.

Apabila tidak habis dibagi 100, maka ada 2 kemungkinan, yaitu habis dibagi 4 atau tidak. Bila habis dibagi 4, maka secara pasti tahun tersebut adalah tahun kabisat. Jika tidak habis dibagi 4, maka secara pasti tahun tersebut **bukan** kabisat.

Untuk lebih jelasnya, perhatikanlah flowchart berikut ini.



10 Algoritma Penentuan_Kabisat
20 {Menentukan suatu tahun yang diinput oleh user merupakan tahun kabisat atau bukan}
30 Deklarasi:
40 n : integer {Tahun}
50 kabisat : boolean {true bila kabisat, false bila bukan}
60 Deskripsi
70 read(n)
80 if n mod 400 = 0 then
90 kabisat ←true
100 else
110 if n mod 100 = 0 then
120 kabisat ←false
130 else
140 if n mod 4 = 0 then
150 kabisat ←true
160 else
170 kabisat ←false
180 end if
190 end if
200 end if
210 if kabisat then
220 write(“kabisat”)
230 else
240 write(“bukan kabisat”)
250 end if

6. Algoritma Menentukan Grade Nilai Akhir

Buatlah algoritma untuk menentukan grade berdasarkan nilai akhir. Grade dapat dilihat pada tabel berikut.

NILAI AKHIR	GRADE
80 – 100	A
70 – <80	B
60 – <70	C
40 – <60	D
0 – <40	E
Selain di atas	K

Analisis: Kita perhatikan bahwa nilai yang valid dimulai dari 0 hingga 100. Diluar itu, nilai tidak valid dan diberi grade "K". Maka langkah pertama adalah menganggap kasus dengan 6 pilihan tersebut sebagai kasus dengan 2 pilihan, yaitu apakah nilai valid (terletak pada range 0–100) atau tidak valid. Apabila tidak valid, maka grade bernilai "K". Jika valid, maka tersisa 5 pilihan grade. Syarat untuk mendapatkan grade A adalah lebih dari sama dengan 80. **Tidak perlu mengecek apakah nilai kurang dari sama dengan 100** (mengapa?).

Bila tidak, maka tersisa 4 pilihan, yaitu grade B hingga grade E. Dengan teknik reduksi, maka akan tersisa 3 pilihan, dan seterusnya.

```
10 Algoritma Penentuan_Grade
20 {Menentukan grade berdasarkan nilai akhir. Grade A (80-100); grade B (70-80); grade C (60-
     70); grade D (40-60); grade E (0-40); selain itu diberi grade K}
30 Deklarasi:
40 nilai : real
50 grade : char
60 Deskripsi
70 read(nilai)
80 if not ((nilai>=0) and (nilai<=100)) then {nilai tidak valid}
90 grade ←"K"
100 else
110 if nilai >= 80 then
120 grade ←"A"
130 else
140 if nilai >= 70 then
150 grade ←"B"
160 else
170 if nilai >= 60 then
180 grade ←"C"
190 else
200 if nilai >= 40 then
210 grade ←"D"
220 else
```

230 grade ←"E"
240 end if
250 end if
260 end if
270 end if
280 end if
290 write(grade)

7. Algoritma Penambahan 1 Detik Pada Waktu

Diberikan suatu data waktu dalam bentuk hh:mm:ss, dengan hh adalah jam, mm adalah menit, dan ss adalah detik. Buatlah algoritma untuk menampilkan data waktu tersebut setelah ditambahkan 1 detik.

Analisis:
Misalkan data waktunya adalah 02:13:23, maka algoritma menghasilkan 02:13:24. Sepintas terlihat sederhana. Namun ada beberapa data waktu yang rumit bila ditambah 1 detik. Perhatikan tabel berikut.

Data Waktu Setelah penambahan 1 detik

Data Waktu	Setelah penambahan 1 detik
02:13:23	02:13:24
02:13:59	02:14:00
02:59:59	03:00:00
23:59:59	00:00:00

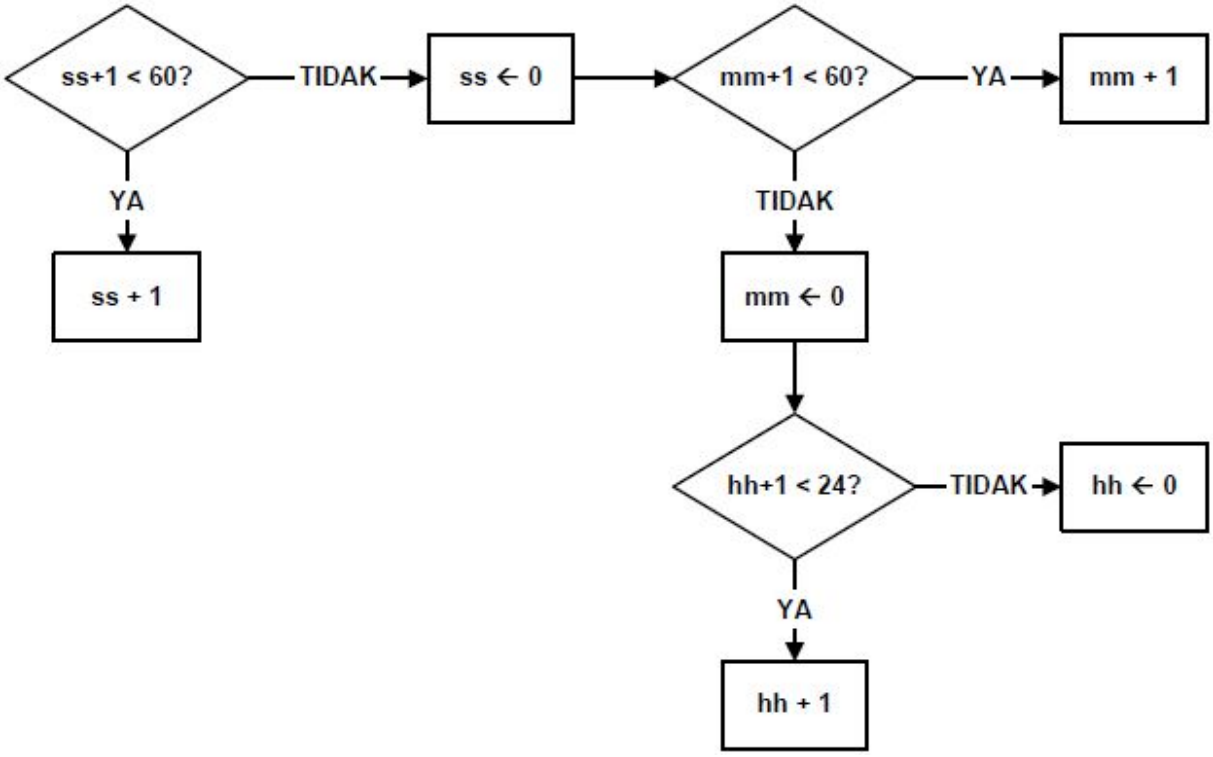
Maka sebelum detik ditambahkan, kita perlu mengecek apakah bisa ditambahkan secara langsung seperti tabel baris ke-1, yaitu dengan syarat ***ss* + 1 < 60**. Jika tidak,maka berarti *ss* bernilai 0, kemudian nilai *mm* ditambahkan 1.

Akan tetapi, sebelummenambahkan 1 ke *mm*, kita perlu mengecek apakah *mm* bisa ditambahkan secara langsung seperti tabel baris ke-2, yaitu dengan syarat ***mm* + 1 < 60**. Jika tidak, maka berarti *mm* bernilai 0, kemudian nilai *hh* ditambahkan 1.

Akan tetapi, sebelum menambahkan 1 ke *hh*, kita perlu mengecek apakah *hh* bisa ditambahkan secara langsung seperti tabel baris ke-3, yaitu dengan syarat ***hh* + 1 < 24**. Jika tidak, maka berarti *hh* bernilai 0 seperti tabel baris ke-4.

10 Algoritma Penambahan_Waktu_1_Detik
20 {Menambah 1 detik ke data waktu dalam hh:mm:ss yang diinputkan oleh user}
30 Deklarasi:
40 hh, mm, ss: integer

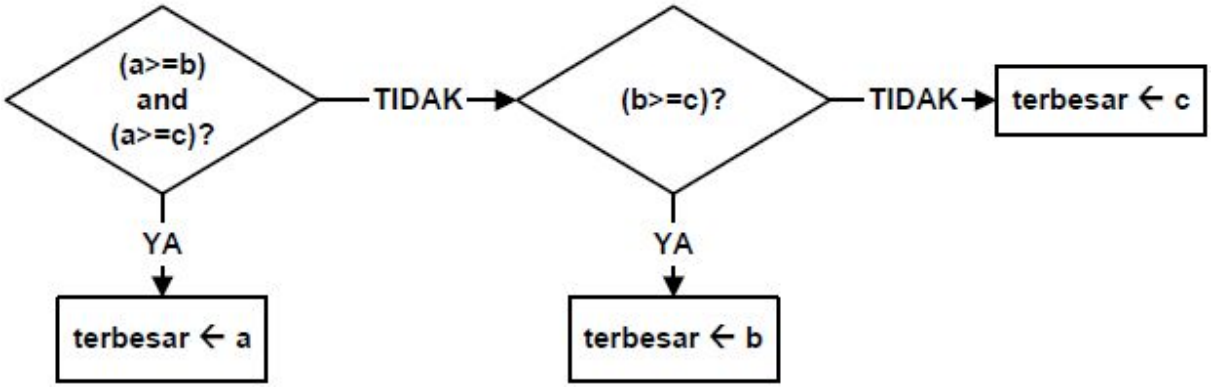
50 Deskripsi
60 read(hh,mm,ss)
70 if ss + 1 < 60 then
80 ss ←ss + 1
90 else
100 ss ←0
110 if mm + 1 < 60 then
120 mm ←mm + 1
130 else
140 mm ←0
150 if hh + 1 < 24 then
160 hh ←hh + 1
170 else
180 hh ←0
190 end if
200 end if
210 end if
220 write(hh,mm,ss)



8. Algoritma Terbesar dari 3 Bilangan (Bandingkan Satu-Satu)

Buatlah algoritma untuk menentukan yang terbesar dari 3 bilangan. Analisis: Langkah pengerjaan: Misalkan bilangan tersebut adalah bilangan *a*, *b*, dan *c*.

1. bandingkan a dengan b dan c. Bila a lebih besar dari b maupun c, maka a terbesar.
2. Jika tidak, maka tersisa 2 kemungkinan, b terbesar atau c terbesar.
3. Bandingkan b dengan c. Bila b lebih besar dari c, maka b terbesar. Jika tidak maka c yang terbesar.

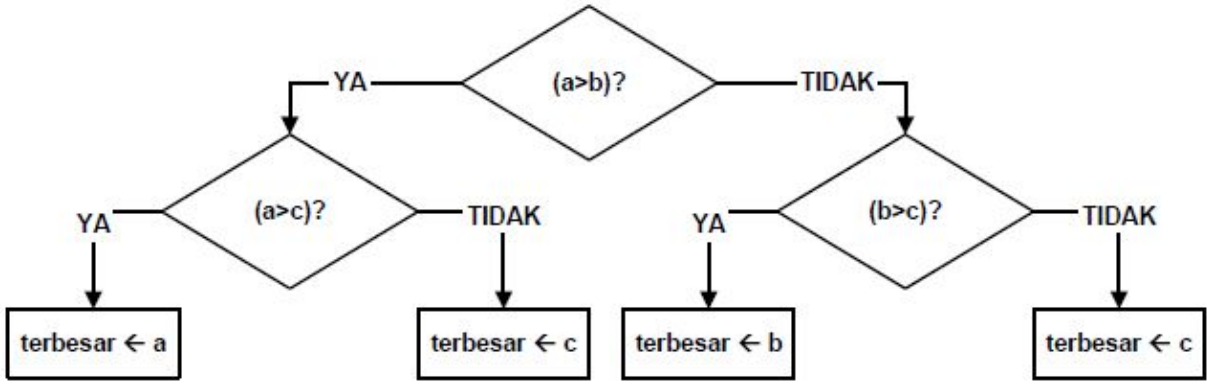



```
10 Algoritma Terbesar_dari_3_Bilangan_A
20 {Menentukan terbesar dari 3 bilangan dengan menggunakan metode compare each to all}
30 Deklarasi:
40 a, b, c, terbesar : real
50 Deskripsi
60 read(a,b,c)
70 if (a>=b) and (a>=c) then
80 terbesar ←a
90 else
100 if b >= c then
110 terbesar ←b
120 else
130 terbesar ←c
140 end if 150 end if
160 write(terbesar)
```

9. Algoritma Terbesar dari 3 Bilangan (Pohon Keputusan)

Buatlah algoritma untuk menentukan terbesar dari 3 bilangan.
Langkah Pengerjaan:
Misalnya bilangan tersebut adalah a, b, dan c.

1. Apabila a>b, maka b sudah dipastikan bukan terbesar. Tersisa 2 pilihan, yaitu a yang terbesar atau c yang terbesar. Bandingkan a dan c untuk mengetahui yang mana yang terbesar.
2. Jika tidak (a ≤ b), maka a sudah dipastikan bukan terbesar. Tersisa 2 pilihan, yaitu b yang terbesar atau c yang terbesar. Bandingkan b dengan c untuk mengetahui yang mana yang terbesar.




```
10 Algoritma Terbesar_Dari_3_Bilangan_B
20 {Menentukan terbesar dari 3 bilangan menggunakan metode pohon
keputusan}
30 Deklarasi:
40 a, b, c, terbesar: real
50 Deskripsi
60 read(a,b,c)
70 if a>b then
80 if a>c then
90 terbesar ←a
100 else
110 terbesar ←c
120 end if
130 else
140 if b>c then
150 terbesar ←b
160 else
170 terbesar ←c
180 end if
190 end if
200 write(terbesar)
```

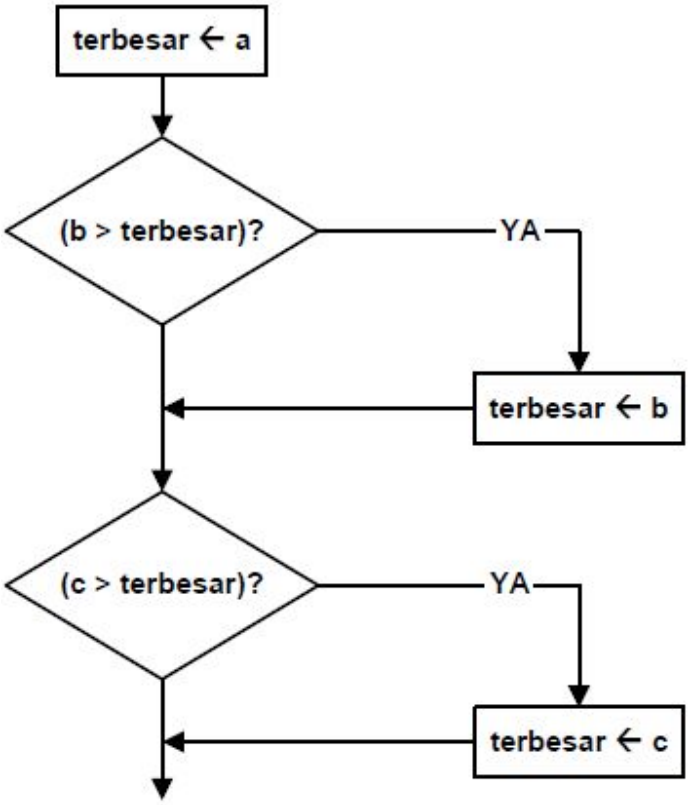
10. Algoritma Terbesar Dari 3 Bilangan (Sekuensial)

Buatlah algoritma untuk menentukan terbesar dari 3 bilangan.

Analisis:

Misalkan 3 bilangan tersebut adalah a, b, c.

1. Asumsi bahwa bilangan a yang terbesar.
2. Bandingkan asumsi dengan bilangan berikutnya, yaitu b. Jika b lebih besar, makaasumsikan terbesar adalah b. Jika tidak, maka asumsi kita bahwa a yang terbesar tidak berubah.
3. Bandingkan asumsi dengan bilangan berikutnya, yaitu c. Jika c lebih besar, maka c adalah yang terbesar. Jika tidak, maka asumsi tidak berubah, dan asumsi itulah yang benar terbesar.




```
10 Algoritma Terbesar_Dari_3_Bilangan_C
20 {Menentukan terbesar dari 3 bilangan menggunakan metode sekuensial}
30 Deklarasi:
40 a, b, c, terbesar : real
50 Deskripsi
60 read(a,b,c)
70 terbesar ←a
80 if b > terbesar then
90 terbesar←b
100 end if
110 if c > terbesar then
120 terbesar ←c
130 end if
140 write(terbesar)
```