DAFTAR ISI

BAB 1 MENGENAL ALGORITMA

Apa Itu Algoritma

Algoritma atau algorithm Dalam ilmu komputer dan matematika, pengertian algoritma adalah suatu urutan dari beberapa langkah logis dan sistematis yang digunakan untuk menyelesaikan masalah tertentu.

Pendapat lain mengatakan definisi algoritma adalah proses atau serangkaian aturan yang harus diikuti dalam perhitungan atau operasi pemecahan masalah lainnya, terutama oleh komputer. Dengan kata lain, semua susunan logis yang diurutkan berdasarkan sistematika tertentu dan digunakan untuk memecahkan suatu masalah dapat disebut dengan algoritma.

Algoritma digunakan untuk melakukan penghitungan, penalaran otomatis, serta mengolah data pada komputer dengan menggunakan software. Dalam algoritma terdapat rangkaian terbatas dari beberapa intruksi untuk menghitung suatu fungsi yang jika dieksekusi dan diproses akan menghasilkan output, lalu berhenti pada kondisi akhir yang sudah ditentukan.

Mengapa algoritma penting

Algoritma menjadi hal yang sangat penting agar pengerjaan suatu program dapat dilakukan dengan runtut dan rapi. Bisa saja kita membuat program terlebih dahulu tapi hal tersebut dapat dipastikan, dalam pengerjaannya, akan tersendat sendat.

Hal itu dikarenakan dalam proses pembuatan program, seorang programmer akan membayangkan/menghayal bagaimana aplikasi yang dibuat nantinya, yang kemudian diperparah dengan terjadimya bugs maupun error karena belum memprediksi kesalahan dan lain-lainnya sebelumnya .

Maka dari itu pembuatan algoritma harus disertai logika yang selaras karena logika dan algoritma merupakan ilmu atau seni untuk mengurutkan suatu pekerjaan seurut urutnya. Pembuat algoritma juga harus dapat mengimajinasikan aplikasi yang akan dibuat nantinya dan membuat langkah-langkah tersebut dapat dipahami oleh pembuat program(programmer).

Sebelum membuat algoritma, dijadikan dulu sebuah pemodelan atau rancangan membuat algoritma jadi pemograman disitu menjadi bagian dari rancangan-rancangan yang sudah diperhitungkan dan diracik sesempurna mungkin, karena itu semua satu kesatuan dari algoritma dan pemograman saling berkaitan.

Struktur penulisan algoritma

Penulisan dengan bahasa natural

Notasi penulisan algoritma menggunakan bahasa deskriptif biasa juga disebut dengan notasi alami. Penulisan Algoritma dengan cara ini dilakukan dengan menuliskan instruksi-instruksi yang harus dilaksanakan secara berurutan dalam bentuk uraian kalimat deskriptif dengan menggunakan bahasa yang jelas. Tidak ada aturan yang baku dalam menuliskan algoritma dengan cara deskriptif, sehingga kamu dapat membuat aturan penulisan dan notasi algoritma sendiri. Namun, agar algoritma mudah ditranslasikan ke dalam bahasa pemrograman, maka sebaiknya kamu menulisnya dengan bahasa yang koresponden dengan notasi bahasa pemrograman pada umumnya.

Notasi penulisan algoritma dengan cara ini cocok menuliskan algoritma yang pendek. Tapi untuk menuliskan algoritma dengan cara ini rasanya kurang efektif. Cara ini memang paling mudah dibuat, tetapi paling sulit untuk diterjemahkan ke dalam bahasa pemrograman. Pada dasarnya, teks algoritma yang disusun dengan bahasa deskriptif disusun atas 3 bagian utama yaitu:

1. Bagian judul (header)

Judul merupakan bagian yang terdiri atas nama algoritma dan penjelasan (spesifikasi) tentang algoritma tersebut. Dibagian ini juga digunakan untuk menentukan apakah teks algoritma yang dibuat tersebut adalah program, prosedur, atau fungsi. Nama algoritma sebaiknya singkat namun cukup menggambarkan apa yang dilalakukan oleh algoritma tersebut..

1. Bagian deklarasi

Bagian deklarasi atau kamus adalah bagian yang mendefinisikan semua data yang dipakai di dalam algoritma. Data tersebut dapat berupa variabel, konstanta, tipe, prosedur atau fungsi. Semua data tersebut dapat digunakan di dalam algoritma jika telah didefinisikan terlebih dahulu di bagian deklarasi. Penulisan sekumpulan nama dalam dalam bagian deklarasi sebaiknya dikelompokkan menurut jenisnya.

2. Bagian deskripsi

Bagian deskripsi adalah bagian inti dari suatu algoritma. Bagian ini berisi uraian langkah-langkah penyelesaian masalah. Langkah-langkah ini dituliskan dengan notasi yang lazim dalam penulisan algoritma. Setiap langkah ditulis secara terurut. Jadi, kalau urutan yang pertama diubah menjadi urutan kedua maka akan mengakibatkan algoritma tidak dapat berjalan dengan semestinya. Suatu algoritma dapat terdiri dari tiga struktur dasar, yaitu runtunan, pemilihan, dan

pengulangan. ketiga struktur dasar tersebut membentuk suatu algoritma. Pada bagian deskripsi inilah letak tiga struktur dasar algoritma tersebut.

Selain bagian-bagian tersebut, terdapat juga komentar untuk maksud memperjelas maksud teks algoritma yang ditulis. Komentar adalah kalimat yang diapit oleh pasangan tanda kurung kurawal ('{' dan '}').

Flowchart

Flowchart adalah cara penulisan algoritma dengan menggunakan notasi grafis. Flowchart merupakan gambar atau bagan yang memperlihatkan urutan atau langkah-langkah dari suatu program dan hubungan antar proses beserta pernyataannya. Penulisan algoritma dengan cara flowchart dilakukan dengan menggunakan simbol. Dengan demikian simbol menggambarkan proses tertentu. Sedangkan proses digambarkan dengan garis penghubung. Dengan menggunakan flowchart, kita akan mudah melakukan pengecekan bagian-bagian yang terlupakan dalam analisis masalah. Penulisan algoritma dengan menggunakan cara flowchart kurang efektif untuk penulisan algoritma yang panjang karena akan mengbiskan banyak tempat.Dalam pembuatan flowchart tidak ad rumus atau patokan yang bersifat mutlak. Karena flowchartmerupakan gambaran hasil pemikir dalam dalam menganalisis suatu masalah yang nantinya akan diubah menjadi program komputer. Tetapi, ada beberapa anjuran yang harus diperhatikan, yaitu:

1. Flowchart digambarkan di suatu halaman dimulai dari sisi atas ke bawah atau dari sisi kiri ke kanan.
2. Aktivitas yang digambarkan harus didefinisikan dengan menggunakan bahasa dan simbol yang tepat dan definisi ini harus dapat dimengerti oleh pembaca.
3. Kapan aktivitas dimulai dan berakhir harus ditentukan secara jelas. Hanya terdapat satu titik awal dan satu titik terakhir.
4. Setiap langkah dari aktivitas harus diuraikan dengan deskripsi kata kerja, misalkan "menghitung nilai rata-rata".
5. Setiap langkah dari aktivitas harus berada pada urutan yang benar.
6. Lingkup dan range dari aktivitas yang sedang digambarkan harus ditelusuri dengan hati-hati. Percabangan-percabangan yang memotong aktivitas yang sedang digmbarkan tidak perlu pada flowchart yang sama. Simbol konektor harus digunakan dan percabangannya diletakkan pada halaman yang terpisah atau hilangkan seluruhnya bila percabangannya tidak berkaitan dengan sistem.
7. Gunakan simbol-simbol flowchart yang standar.

Simbol-simbol flowchart yang biasanya dipakai adalah simbol-simbol flowchart standar yang dikeluarkan oleh ANSI dan ISO.

PSEUDOCODE

Pseudocode adalah cara penulisan algoritma yang menyerupai bahasa pemrograman tingkat tinggi. Pseudocode menggunakan bahasa yang hampir menyerupai bahasa pemrograman. Biasanya pseudocode menggunakan bahasa yang mudah dipahami secara universal dan juga lebih ringkas dari pada cara penulisan algoritma sebelumnya. Struktur bahasa yang digunakan dalam menulis algoritma dengan cara ini berasal dari beberapa bahasa pemrograman, tetapi bahasa tersebut hanya ditujukan agar dapat dibaca manusia. Sehingga pseudocode tidak dapat dipahami oleh komputer. Agar notasi psudocode dapat dipahami oleh komputer maka harus diterjemahkan terlebih dahulu menjadi sintaks bahasa pemrograman komputer tertentu. Tidak ada sintaks standar yang resmi dalam penulisan algoritma dengan cara penulisan pseudocode. Oleh karena itu, pseudocode ini dapat diterapkan dalam berbagai bahasa pemrograman. Tetapi, untuk memudahkan translasi ke dalam bahasa pemrograman disarankan untuk menggunakan keyword yang umum digunakan seperti: if, then, whike, do, else, repeat, for, dan lainnya. Struktur penulisan algoritma dengan notasi pesudocode sama dengan cara kalimat deskriptif yaitu:

1. Judul
2. Deklarasi
3. Deskripsi

Mengenal Operator

Operator adalah karakter khusus yang berupa simbol atau tanda yang digunakan untuk mengoperasikan atau memproses dua operand atau lebih untuk mendapatkan hasil atau output.

Operand adalah suatu nilai atau variabel yang akan dioperasikan (diproses) oleh operator untuk mengetahui hasil. Dalam Algoritma, Operator terbagi menjadi 4, berikut beserta penjelasannya.

Operator Arithmatic

Operator Arithmatic adalah operator yang digunakan untuk mengoprasikan proses matematik.

Operator Relation

Operator Relation atau relasi adalah operator perbandingan yang membandikan suatu nilai numerik atau alfa-numerik untuk menentukan kebenarannya.

Operator Logic

Operator Logic (Logika) adalah operator yang membandingkan dua atau lebih kondisi (True atau False) untuk mengeluarkan satu pernyataan (True atau False).Jenis-jenis dari operator logika adalah And, Or, Not dan XOr.

Dengan kata lain:And akan bernilai benar hanya jika semua operand bernilai benar (True).

Or akan bernilai salah hanya jika semua operand bernilai salah (False).

XOr akan bernilai benar hanya jika semua operand bernilai sama.

dan untuk operator not, hasilnya akan True jika operand nya False, dan akan bernilai False jika Operand nya True.

Operator Assignment

Operator Assignment adalah operator yang berfungsi untuk memberikan nilai pada suatu variabel dan simbolnya kita pakai (←) saja. Jadi jika ada simbol itu pada lecture selanjutnya berarti kita mendefinisikan nilai dari suatu variabel.

Percabangan

Percabangan algoritma adalah salah satu instruksi dalam algoritma yang digunakan untuk memberikan pilihan kepada program perintah mana yang harus diproses dan perintah mana yang harus dilewati sesuai dengan kondisi yang diberikan.

Algoritma percabangan terkadang diperlukan untuk kasus-kasus tertentu, karena pada kenyataannya alur pemrosesan kode program tidak selamanya berurutan dari baris instruksi satu ke baris instruksi lainnya, namun terkadang program perlu diatur agar bisa meloncat pada baris instruksi tertentu sesuai dengan kondisi yang terpenuhi.

Di dalam algoritma, instruksi percabangan dikategorikan menjadi beberapa jenis yaitu, percabangan 1 kondisi, percabangan 2 kondisi, percabangan 3 kondisi, percabangan lebih dari 3 kondisi dan percabangan bersarang.

Untuk lebih memahami logika dari algoritma percabangan, baik percabangan 1 kondisi, 2 kondisi, 3 kondisi maupun percabangan bersarang, maka di artikel kali

saya akan coba kupas tuntas mengenai algoritma percabangan disertai dengan contoh kasus lengkap, baik kasus-kasus khusus, maupun contoh kasus dalam kehidupan sehari-hari.

Algoritma percabangan biasanya menggunakan instruksi

IF (Kondisi1) THEN

pernyataan 1

ELSE IF (Kondisi 2) THEN

pernyataan 2

ELSE IF (Kondisi 3) THEN

pernyataan 3

ELSE

pernyataan 4

END IF

Untuk Kondisi ke 1 cukup menggunakan instruksi IF (Kondisi 1) THEN, sedangkan untuk kondisi 2 dan seterusnya selain kondisi terakhir, maka menggunakan ELSE IF (Kondisi N) THEN, sementara untuk kondisi terakhir cukup menggunakan ELSE saja.

Memahami Apa itu IF THEN ELSE ?

IF dapat diartikan sebagai (JIKA), sedangkan THEN dapat anda artikan dengan sebutan MAKA, sedangkan ELSE dapa anda artiken sebagai (JIKA BUKAN / SELAIN ITU).

Memahami Apa itu Kondisi ?

Kondisi, umumnya akan membandingkan 2 buah operan dengan menggunakan operator aritmatika seperti:

> (lebih besar)

< (lebih kecil)

>= (lebih besar atau sama dengan)

<= (lebih kecil atau sama dengan)

== (sama dengan)

<> (tidak sama dengan)

Contoh :

IF(nilai >=80) THEN

write("LULUS")

ELSE

write("GAGAL")

END IF

Instruksi di atas jika diterjemahkan adalah, IF="jika nilai lebih besar atau sama dengan 80, maka cetak kata LULUS", ELSE="jika tidak (artinya nilainya lebih kecil dari 80) maka GAGAL".

Yang diberikan warna merah tebal itu adalah bagian kondisi, dengan operan nilai dan angka 80, sedangkan operator yang digunakan adalah >= (lebih besar atau sama dengan), harap diingat kondisi biasanya selalu membandingkan 2 buah operan dengan operator artimatika.

Semoga dengan uraian singkat di atas anda punya gambaran apa itu percabangan.

Algoritma percabangan 1 kondisi

Algoritma percabangan 1 kondisi adalah algoritma percabangan yang hanya menggunakan1 kondisi atau ketentuan saja, jika kondisi terpenuhi maka instruksi akan diproses, jika tidak terpenuhi maka akan dilewat atau diloncati.

Contoh:

Algoritma untuk menentukan usia balita, jika usia lebih kecil atau sama dengan 5 tahun maka balita.

Pseudocode:

program cek_usia

deklarasi

var usia:integer

algoritma

read(usia)

IF(usia <5) THEN

write("BALITA")

ENDIF

Pada contoh di atas usia diinput oleh pengguna, usia kemudian dicek, jika usia lebih kecil dari lima maka cetak tulisan "BALITA" sedangkan jika tidak maka instruksi write("BALITA") akan dilewati.

Algoritma Percabangan 2 kondisi

Algoritma percabangan 2 kondisi adalah algoritma untuk memecahkan kasus yang hanya menggunakan 2 ketentuan saja.

Struktu percabangan 2 kondisi adalah sebagai berikut:

IF (Kondisi 1) THEN

pernyataan 1

ELSE

pernyataan 2

ENDIF

Contoh algoritma 2 kondisi:

Algoritma untuk menentukan Lulus dan Tidak Lulus nilai siswa, dengan ketentuan:

1. Jika nilai >=75 maka lulus,

2. Jika tidak (nilai <75 maka tidak lulus)

Jawab:

program kelulusan

deklarasi

var nilai:integer

algoritma:

read(nilai)

IF (nilai >=75) THEN

write("LULUS")

ELSE

write("TIDAK LULUS")

ENDIF

Atau Kasus di atas bisa juga ditulis seperti di bawah ini:

program kelulusan

deklarasi

var nilai:integer

algoritma:

read(nilai)

IF (nilai <75) THEN

write("TIDAK LULUS")

ELSE

write("LULUS")

ENDIF

Untuk 2 kondisi, maka algoritma akan melibatkan instruksi percabangan IF (kondisi) THEN..... ELSE....., artinya jika 2 kondisi maka instruksi kata IF hanya 1 jumlahnya lainnya menggunakan kata ELSE.

Algoritma Percabangan 3 Kondisi

Algoritma tiga kondisi merupakan algoritma yang dapat digunakan untuk memecahkan kasus yang memiliki 3 kondisi.

Untuk algoritma 3 kondisi maka strukturnya adalah:

IF (kondisi 1) THEN

pernyataan 1

ELSE IF(kondisi 2) THEN

pernyataan 2

ELSE

pernyataan 3.

ENDIF

Untuk kondisi pertama selalu menggunakan IF (Kondisi 1) THEN, sedangkan untuk kondisi kedua dan seterusnya selain kondisi terakhir, menggunakan ELSE IF (kondisi n) THEN, dan untuk kondisi terakhir menggunakan ELSE saja.

Contoh algoritma dengan 3 kondisi:

Algoritma untuk mencari nilai dalam bentuk abjad A,B atau C, dengan ketentuan

1. Jika nilai >80 maka nilai A

2. jika nilai >=70 dan <=80 maka B

3. selain itu (nilai <70) maka C

Jawab:

```
program cari_nilai

deklarasi

var nilai:integer

algoritma

read(nilai)

IF(nilai>80)THEN

write("A")

ELSE IF(nilai>=70 AND nilai <=80) THEN

write("B")

ELSEwrite("C")

ENDIF
```

Algoritma lebih dari 3 kondisi

Kondisi dalam algoritma bisa lebih dari 3 kondisi, strukturnya akan selalu sama, untuk kondisi pertama maka menggunakan IF (kondisi 1) ELSE...., sedangkan untuk kondisi ke 2 dan seterusnya selain kondisi terakhir yaitu menggunakan ELSE IF (kondisi n) THEN....., sedangkan untuk kondisi terakhir baru menggunakan ELSE....saja.

Format untuk algoritma 3 kondisi atau lebih adalah sebagai berikut:

IF(Kondisi 1) THEN

pernyataan 1

ELSE IF(Kondisi 2) THEN

pernyataan 2

ELSE IF (kondisi 3) THEN

pernyataan 3

ELSE IF(kondisi 4) THEN

pernyataan 4

...

...

...

...

...

ELSE IF(kondisi N) THEN

pernyataan N

ELSE

pernyataan terakhir

ENDIF

Contoh algoritma lebih dari 3 kondisi sama halnya dengan 3 kondisi di atas, namun ketentuannya lebih dari 3. bisa 4, 5 dan seterusya.

c. Algoritma Percabangan Bersarang

Algoritma percabangan bersarang merupakan bentuk algoritma percabangan dimana pada setiap setiap pernyataan untuk kondisi IF di dalamnya terdapat Instruksi IF Lagi.

Algoritma percabangan bersarang artinya di dalam IF terdapat IF lagi.

Struktur algoritma percabangan bersarang adalah sebagai berikut:

IF(Kondisi a) THEN

IF(kondisi x)THEN

pernyataan 1

ELSE IF

pernyataan 2

ENDIF

ELSE

pernyataan b

ENDIF

Intinya algoritma percabangan disebut percabangan bersarang jika di dalam percabangan ada percabangan lagi, banyak yang menyebut juga dengan sebutan di dalam IF ada IF lagi.

perulangan

salah satu yang dipelajari di algoritma dan pemrograman dasar adalah pengulangan atau istilah lainnya looping, bukan hanya di algoritma saja, ternyat konsep looping ini digunakan juga di berbagai bahasa pemrograman dan konspe dasarnya adalah algoritma pengulangan.

Pengulangan atau disebut sebagai looping adalah instruksi khusus dalam bahasa pemrograman dan algoritma yang digunakan untuk mengulang beberapa perintah sesuai dengan jumlah yang telah ditentukan. tujuannya adalah untuk mempermudah pengerjaan program dan untuk mempersingkat instruksi program. dengan pengulangan instruksi program yang seharunya ditulis dengan jumlah baris yang banyak bisa dipersingkat.

Instruksi Pengulangan dalam Algoritma

Ada 3 jenis bentuk instuksi format pengulangan di dalam algoritma yitu sebagai berikut:

1. Pengulangan menggunakan FOR

Pengulangan for disebut juga sebagai pengulangan di awal format instruksinya adalah sebagai berikut:

For i ← nilai_awal to nilai_akhir do

Statement

Endfor

Contoh:

Buatlah algoritma untuk mencetak tulisan "Algoritma Menyenangkan" sebanyak 100 baris maka instruknya adalah:

Jawab:

program looping_for

DEKLARASI

i:integer

ALGORITMA:

for i ← 1 to 100 do

writeln('Algoritma Menyenangkan')

endfor

2. Penggulangan menggunakan Instruksi While DO

Format:

while kondisi do

pernyataan

endwhile

Contoh Kasus:

Buatlah algoritma untuk mencetak tuliasan angka 1 sampai 100

Jawaban:

promgram looping

DEKLARASI

var i:integer

ALGORITMA:

i ← 0

while i <100 do

writeln ('angka ke', i)

i ← i+1 {pencacah naik}

endwhile

3. Pengulangan dengan Menggungakan Repeat Until

Format:

repeat

statement

pencacah naik atau pencacah turun until kondisi

contoh kasus:

Buatlah algoritma untuk mencetak tulisan Hello World sebanyak 1000 baris.

Jawab:

program cetak

DEKLARASI

i:integer

ALGORITMA:

i ← 1 {isi nilai awal variable i dengan angka 1)

repeat write ('Hello World') i ← i+1

until i<=1000

Kapan Harus menggunakan Instruksi pengolangan di dalam algoritma?

Sebenarnya untuk memecahkan masalah kasus pemrograman bisa dipecahkan dengan banyak cara tertangung logika si programmer, seperti halnya banyak jalan menuju kota jakarta, tapi tujuannya tetap saja, tapi yang terbaik adalah bagaimana membuat program dengan instruksi sedikit dan proses sangat cepat.

Programmer yang pintar akan sangat mudah sekali mencari cara yang terbaik untuk membuat program dengan instruksi yang singkat namun prosesnya cepat. salah satu instruksi yang bisa digunakan adalah pengulangan, ketika sebuah kasus memungkinkan untuk menggunakan pengulangan maka harus menggunakan pengulangan.

Kapan instruksi pengulangan harus digunakan?

Instruksi pengulangan digunakan manakala program atau bagian program terindikasi bisa menggunakan proses pengulangan.

Sebagai contoh sederhana. misalkan untuk kasus program untuk menampilkan angka 1 sampai 1000, atau program untuk mencetak tulisan tertentu dalam jumlah tertentu.

Sebenarnya bisa saja tidak menggunakan pengulangan, namun kurang efektif walaupun hasil outputnya bisa saja sama.

Pengulangan dengan Pencacah Naik

Pengulangan pencacah naik yaitu kondisi pengulangan yang dimulai dengan kondisi pencacah kecil ke besar naik sampai jumlah pengulangan yang diinginkan.

Contoh: buat algoritma untuk mencetak tulisan "Teknologi Modern" sebanyak 1000 baris.

Jika menggunakan pencacah naik instruksi algoritmanya adalah sebagai berikut:

Jawab:

algoritma pencacah_naik

DEKLARASI

i:integer

ALGORITMA:

for i ← 1 to 1000 do

writeln ('Teknologi Modern');

endfor

Pengulangan yang digunakan di algorirma di atas disebut pengulangan pencacah naik karena dimulai dari angka 1 terus naik sampai angka 1.000. bisa juga menggunakan Repeat Until atau While DO.

Pengulangan dengan Pencacah Turun

Pengulangan pencacah turun yaitu kodisi pengulangan yang dimulai dengan kondisi nilai pencacah dari besar ke kecil. sesuai dengan jumlah yang diinginkan.

Contoh: buat algoritma untuk mencetak tulisan "Teknologi HP Modern" sebanyak 1000 baris.

Jika menggunakan pencacah turun instruksi algoritmanya adalah sebagai berikut, misal menggunakan intruksi repeat until:

Jawab:

algoritma cacah_turun

DEKLARASI

i:integer

ALGORITMA:

i ← 1000 {nilai pencacah awal 1000 dimasukan ke variable i}

repeat writeln ('teknologi HP Modern')

i ← i-1 {turunkan pencacah}

until i<1.

latihan

1. Dibaca waktu tempuh seorang pelari marathon dalam jam-menit-detik (hh:mm:ss). Diminta mengkonversi waktu tempuh tersebut ke dalam detik. Tuliskan algoritmanya.

Ingatlah

1 menit = 60 detik

1 jam = 3600 detik

Misalnya waktu tempuh seorang pelari marathon adalah 1 jam, 5 menit, 40 detik. Dalam detik, waktu tempuh seluruhnya adalah (1 x 3600) + (5 x 60) + 40 = 3940 detik.

Penyelesaian :

Algoritma KONVERSI_JAM_KE_DETIK

{ dibaca jam-menit-detik (hh:mm:ss). Nilai jam-menit-detik dikonversi ke dalam detik, lalu ditampilkan ke piranti keluaran }

DEKLARASI

Type jam : record <hh : integer {0..23}, {jam}

mm : integer {0..59}, {menit}

ss : integer {0..59}, {detik}

>

J : jam

TotalDetik : integer

DESKRIPSI

read(J.hh,J.mm,J.ss))

TotalDetik ← (J.hh*3600) + (J.mm*60) + J.ss

write(TotalDetik)

Jika anda mentranslasikan algoritma KONVERSI_JAM_KE_DETIK ke dalam bahasa pascal, anda harus memperhatikan tipe bilangan bulat yang digunakan. Karena ranah nilai tipe integer terbatas, maka ada kemungkinan hasil pengubahan jam-menit-detik ke total detik bernilai negatif, sebab nilai (J.hh*3600) + (J.mm*60) + J.ss berada di luar rentang tipe integer. Tipe longint yang mempunyai ranah yang lebih besar dapat dipakai untuk masalah ini.

Jadi, program KONVERSI_JAM_KE_DETIK dalam bahasa pascal adalah sebagai berikut :

program KONVERSI_JAM_KE_DETIK;

{ dibaca jam-menit-detik (hh:mm:ss). Nilai jam-menit-detik dikonversi ke dalam detik, lalu ditampilkan ke piranti keluaran.}

uses wincrt;

(* DEKLARASI *)

type Jam = record

hh : longint; {jam}

mm : longint; {menit}

ss : longint; {detik}

```
end;
var
J : Jam;
TotalDetik : longint;
(* deskripsi *)
begin
write('Jam  :'); readln(J.hh);
write('Menit:'); readln(J.mm);
write('Detik:'); readln(J.ss);
TotalDetik:= (J.hh*3600) + (J.mm*60) + J.ss;
writeln('Total detik = ', TotalDetik);
end.
```

Referensi

1. https://www.maxmanroe.com/vid/teknologi/pengertian-algoritma.html
2. https://www.dictio.id/t/mengapa-pembuatan-algoritma-sangat-penting-dalam-pembuatan-program-komputer/13296
3. http://referensisiswa.blogspot.com/2018/09/algoritma-percabangan-12-3-kondisi.html
4. https://medium.com/codelabs-unikom/algoritma-perulangan-iteration-looping-apa-itu-df8007239f9c
5. http://noob-programer.blogspot.com/2014/10/assalamualaikum-wr.html
6. https://thenextillution.wordpress.com/2014/12/30/004-operator-pada-algoritma/


```
static int MenghitungLuasBalok(){
    Scanner Input3=new Scanner(System.in);
    double luassisi,volume,luas,akar;
   System.out.print("luassisi: ");
```

```
    luassisi = Input3.nextDouble();
    akar=Math.sqrt(luassisi);
     System.out.println("akarnya adalah: "+akar);
     volume=akar*akar*akar;
     System.out.println("volumenya adalah: "+volume);
     luas=6*akar*akar;
     System.out.println("luasnya adalah: "+luas);
   return 0;
  }
static int menghitungLuasKubus(){
Scanner Input2=new Scanner(System.in);
     System.out.println("mencari luas dan volume balok");
     double p,l,t,luaspt,luasbalok,volumebalok;
     System.out.println("masukkan panjang balok: ");
     p=Input2.nextDouble();
     System.out.println("masukkan lebar balok: ");
     l=Input2.nextDouble();
     System.out.println("masukkan luaspt: ");
     luaspt=Input2.nextDouble();
     t=luaspt/1;
     System.out.println("tingginya adalah "+t);
     luasbalok=(2*p*t);
     System.out.println("luas balok adalah "+luasbalok);
     volumebalok=p*t*l;
     System.out.println("volume balok adalah "+volumebalok);
return 0;
}
```