Front End React Development
sesi 6

Learn Algorithm & Pseudocode

Algoritma

Belajar pemrograman tentu saja memerlukan banyak berpikir secara logika, sehingga kita bisa memecahkan masalah atau melakukan kegiatan yang kita targetkan. Agar kita bisa menjelaskan alur logika kita, perlu digunakan algoritma.

Namun apa itu algoritma? Mari kita ilustrasikan terlebih dahulu.

Bagaimana cara kamu menggunakan komputer sehari-hari? Mulai dari menekan tombol on/off, menunggu proses booting, memasukkan password, membuka aplikasi yang dibutuhkan, kemudian bekerja sesuai aplikasi tersebut. Atau lain lagi, bagaimana cara menggunakan smartphone? Cukup unlock dari posisi standby, lalu pilih dan buka aplikasi yang dibutuhkan, dan seterusnya.

Nah itulah algoritma, kumpulan proses ataupun aturan untuk melakukan atau menyelesaikan sesuatu. Sesuatu ini biasanya berupa masalah atau kegiatan yang langkah-langkahnya pasti terbatas (tidak terus-menerus).

Apa itu Algoritma ?

serangkaian urutan langkah-langkah yang tepat, logis, terperinci, dan terbatas untuk menyelesaikan suatu masalah yang disusun secara sistematis.

Mengapa Algoritma ?

1. Algoritma adalah inti dari ilmu komputer
2. Algoritma adalah urutan-urutan dari instruksi atau langkah-langkah untuk menyelesaikan suatu masalah
3. Algoritma adalah blueprint dari program
4. Sebaiknya disusun sebelum membuat program

Kriteria Algoritma

- Ada input dan output
- Efektifitas dan efisien
- Terstruktur

Bagaimana cara kamu menggunakan laptop sehari-hari ?

Input	Output
1. Menekan Tombol On/Off	2. Menunggu Halaman Booting
3. Memasukkan Password	4. Menunggu validasi Password
5. Pilih Aplikasi yang mau dibuka	

Contoh lagi : Algoritma TUKAR ISI EMBER

Diberikan 2 buah ember A dan B, ember A berisi larutan berwarna merah, ember B berisi larutan berwarna biru. Tukarkan isi kedua ember itu sedemikian sehingga ember A berisi larutan warna biru dan ember B berisi larutan berwarna merah.

Deskripsi:

1. Tuangkan larutan dari ember A ke dalam ember B
2. Tuangkan larutan dari ember B ke dalam ember A

Algoritma TUKAR ISI EMBER di atas tidak menghasilkan pertukaran yang benar. Langkah di atas tidak logis, hasil pertukaran yang terjadi adalah pertukaran kedua larutan tersebut.

Supaya logis seperti apa ?

```
Deskripsi:
```

1. `Tuangkan larutan dari bejana A ke dalam ember C.`
2. `Tuangkan larutan dari bejana B ke dalam ember A.`
3. `Tuangkan larutan dari bejana C ke dalam ember B.`

MENGAPA ALGORITMA ITU PENTING ?

1. Algoritma harus berhenti setelah menjalankan sejumlah langkah terbatas.
2. Setiap langkah harus didefinisikan dengan tepat dan tidak berarti-dua (ambiguitas).
3. Algortima memiliki nol atau lebih masukan.
4. Algoritma memiliki nol atau lebih keluaran.
5. Algoritma harus efektif (setiap langkah sederhana sehingga dapat dikerjakan dalam waktu yang masuk akal).

Notasi Algoritma:

1. Penulisan algoritma tidak tergantung dari spesifikasi bahasa pemrograman dan komputer yang mengeksekusinya. Notasi algoritma bukan notasi bahasa pemrograman tetapi dapat diterjemahkan ke dalam berbagai bahasa pemrograman.
2. Notasi algoritma dapat berupa:

   Uraian kalimat deskriptif (narasi):

Algoritma kelulusan mahasiswa

Diberikan nama dan nilai mahasiswa, jika nilai tersebut lebih besar atau sama dengan 60 maka mahasiswa tersenut dinyatakan lulus, jika nilai lebih kecil dari 60 maka dinyatakan tidak lulus.

Baca nama dan nilai mahasiswa Jika nilai >= 60 maka Keterangan = lulus

Tetapi jika salah Keterangan = tidak lulus Tulis nama dan keterangan.

Latihan Berpikir Algoritma

Flowchart

1. Flowchart adalah bagan-bagan yang mempunyai arus menggambarkan langkah-langkah penyelesaian suatu masalah
2. Merupakan cara penyajian dari suatu algoritma
3. Ada 2 macam flowchart:

- System flowchart:

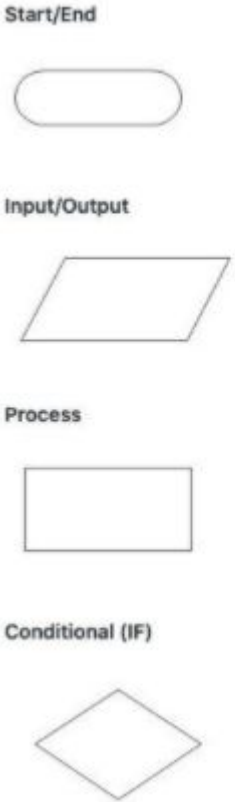
Urutan proses dalam system dengan menunjukkan alat media input, output serta jenis penyimpanan dalam proses pengolahan data.

- Program flowchart:

Urutan instruksi yang digambarkan dengan symbol tertentu untuk memecahkan masalah dalam suatu program.

Simbol-simbol Flowchart

1. Flow Direction Symbols (simbol penghubung alur)
2. Processing Symbols (simbol proses)
3. Input-Output Symbols (simbol input-output)



Pseudocode

Dalam menggunakan bahasa pemrograman, kita bisa menggunakan atau bahkan tidak perlu menggunakan algoritma. Tapi hampir 99% pastinya perlu algoritma. Misalnya saja kita sudah tahu algoritma dasar dari perulangan dan perkondisian. Mengenal Pseudocode Atau agar lebih rapi, kita gunakan pseudocode.

Pseudocode adalah konvensi terstruktur atau cara menyajikan penjelasan algoritma dengan bahasa yang deskriptif seperti kita menulis kalimat biasa sehingga mudah kita baca. Umumnya digunakan bahasa Inggris atau bahasa perantara yang mirip bahasa pemrograman. Lihatlah contoh algoritma penambahan angka sederhana dengan pseudocode berikut.

Contoh dengan Bahasa Inggris

- *READ and SAVE "first number"*
- *READ and SAVE "second number"*
- *COMPUTE "first number" added by "second number"*
- *SAVE previous computation result*
- *SHOW the computation result*

Bahasa inggris diatas nantinya akan diubah menjadi bahasa pemrograman yang kita mau.
Dibawah ini adalah contoh hasil konversi pseudocode diatas menjadi kode di bahasa lain. Saat ini kamu hanya cukup melihat hasil konversinya sekilas saja, tidak harus dipelajari, karena kita akan fokus pada pseudocode terlebih dahulu.

Python

```python
a = input("First Number? ")
b = input("Second Number? ")
c = int(a) + int(b)
print("Result", c)
```

Ruby

```ruby
puts "First Number?"
a = gets.chomp
puts "Second Number?"
b = gets.chomp
c = a.to_i + b.to_i
puts c
```

Javascript

```javascript
var a,b,c;
a = prompt("First Number?");
b = prompt("Second Number?");
c = Number(a) + Number(b);
console.log(c);
alert("Result = " + c);
```

Lebih dalam tentang Pseudocode

Berikut adalah contoh yang perlu kamu tahu saat membuat pseudocode. Jangan Terpaku 100% dengan contoh, karena dalam pseudocode tidak terpaku pada penggunakan kata tertentu. Selama pseudocode dapat dimengerti sesama pembaca, maka sudah cukup bisa digunakan. Kita bisa menggunakan huruf kapital untuk keyword yang ditekankan dari sebuah step. Misal: CALCULATE 5 plus 2, atau DISPLAY "hello".

Storing Values

Biasanya, pada saat kita belajar matematika atau fisika, kita akan bertemu dengan rumus. Paling sederhana adalah rumus luas persegi, yaitu width dikalikan height.

Kita sebagai manusia dengan natural dapat langsung mengkalkulasi nilai panjang dan lebar untuk mendapatkan luas. Tapi, komputer tidak semudah itu. Komputer harus menyimpan nilai panjang dan nilai lebar di dalam memori. Memori komputer, bayangkan saja seperti otak kita yang bisa menyimpan berbagai informasi. Sebetulnya, saat kita menghitung panjang dan lebar secara tidak sadar kita pun menampung nilai tersebut di kepala kita.

```
1  STORE "width" with any value
2  STORE "height" with any value
3  STORE "area" without any value
4
5  CALCULATE "width" times "height"
6  SET "area" value with calculation result
7  DISPLAY "area"
```

Bisa dilihat dari pseudocode diatas, ada beberapa step yang kita jalankan. Mari kita bahas tiap step ke bahasa yang lebih "manusiawi"

- Simpan "width" dengan nilai berapapun
- Simpan "height" dengan nilai berapapun
- Simpan "area" tanpa diberikan nilai. Ini akan kita isi nanti.
- Hitung hasil perkalian "width" dengan "height"
- Setelah mendapatkan hasil perhitungan, isikan hasilnya ke dalam "area"
- Tampilkan nilai dari "area"

Conditional

Saat komputer menjalankan program, seringkali komputer harus melakukan sebuah tindakan jika suatu kondisi terpenuhi. Mudahnya, di kehidupan sehari-hari misalnya, jika kita lapar, kita akan makan. "Jika kita lapar" adalah sebuah kondisi, dan "kita akan makan" adalah step yang hanya akan dijalankan apabila kondisi tersebut terpenuhi.

```
1 IF "hungry"
2   DO "eat"
3 DISPLAY "i am happy"
```

Jika kita lihat pseudocode diatas, bisa dijabarkan sebagai step berikut

Jika lapar, maka masuk ke step 2. Jika tidak, abaikan step 2 dan langsung ke step 3. lakukan proses "eat" Tampilkan "i am happy" Yang terjadi disini ada dua kemungkinan. Jika lapar, step 2 akan diabaikan. Kita bisa lihat dari pseudocode dimana "eat" kita buat menjorok ke dalam (ingat dengan identasi di HTML? ya betul! :D) untuk menunjukkan semua proses yang menjorok ke dalam setelah sebuah kondisi merupakan proses yang dijalankan hanya jika kondisi terpenuhi.

Tidak hanya sampai disana, kondisional bisa juga melakukan proses yang hanya dijalankan jika kondisi tidak terpenuhi. Misal, saat nilai ujian dibawah 70, saya harus berlajar lebih giat. Tapi jika tidak, maka saya layak memberi reward untuk diri sendiri.

```
STORE "score" to any number

IF "score" < 70
  DO "learn more"
ELSE
  DO "reward myself"
DO "continue with life..."
```

Nah disini terjadi yang biasa dinamakan percabangan. Jika score dibawah 70, maka kita akan "learn more", dan jika tidak, maka kita harus "reward myself".

Namun apapun kondisinya, kita pasti akan masuk ke step "continue with life..."

Pseudocode diatas bisa digambarkan ke step berikut:

Jika "score" dibawah 70, masuk ke step 2a. Jika tidak, masuk ke step 2b. 2a. Lakukan "learn more" 2b. Lakukan "reward myself" Lakukan "continue with life..."

Saat program berjalan, berarti hanya ada dua kemungkinan. Antara menjalankan step 1 -> 2a -> 3, atau step 1 -> 2b -> 3.

Looping

Nah sekarang kita akan masuk ke bagian terakhir yang dibutuhkan hari ini dan sekaligus yang sedikit lebih sulit dibandingkan bagian sebelumnya, yaitu perulangan.

Komputer seringkali dibutuhkan untuk melakukan sebuah proses yang sama berulang-ulang. Hal ini sering disebut sebagai looping. Tentunya, saat komputer melakukan looping, pasti ada kalanya proses tersebut akan berhenti. Sama hal nya saat kita sebagai manusia melakukan berbagai hal yang berulang, pasti ada kalanya kegiatan itu kita hentikan. Nah, looping sebetulnya melibatkan yang sudah kita pelajari sebelumnya, yaitu conditional. Looping akan terus dilakukan sampai sebuah kondisi terpenuhi. Contoh mudahnya adalah, kita akan makan hingga kenyang bukan?

```
1 WHILE "hungry"
2   DO "eat"
```

Nah, pseudocode di atas cukup simple dan mencontohkan kita proses paling sederhana dalam looping. WHILE adalah standard keyword untuk menjunukkan kondisi "selama kita masih lapar", lakukan proses makan.

Biasanya, saat kita membuat looping, ada sebuah proses yang dilakukan untuk mencapai kondisi tersebut. Kita coba perbaiki contoh pseudocode diatas, dengan asumsi kemampuan makan kita dalam sekali makan adalah 5 sendok nasi. Kita coba umpakan kemampuan makan ini sebagai "hungry level".

```
1  STORE "full level" with 0
2
3  WHILE "full level" < 5
4    ADD "full level" by 1
5
6  DISPLAY "I'm full!"
```

Nah, looping kali ini sudah lebih mendekati coding sebenarnya. Saat terjadi looping, harus ada proses apapun yang akan membuat kondisi perulangan lambat laun akan terpenuhi. Pseudocode di atas menggambarkan kita mulai dari level kenyang kita dari 0, berarti kita saat ini sangat lapar. Setiap kali kita melakukan proses makan, tingkat kenyang kita akan bertambah 1. Karena kita hanya kuat makan hingga 5 kali, maka kondisinya adalah "full level" < 5.

Kita coba ilustrasikan step pseudocode diatas:

Simpan nilai "full level" dengan angka 0

Ulangi step 3 selama "full level" masih dibawah 5. Jika "full level" sudah 5, lanjut ke step 4.

Tambah "full level" dengan 1, agar semakin mendekati batas perulangan. Kembali ke step 2.

Tampilkan "I'm full", berarti saya sudah sangat kenyang!

Ternyata, setelah dijabarkan, looping tidak begitu sulit, bukan? Nah, sebuah komputer tentu saja dapat memiliki program yang sangat memungkinkan menggabungkan penyimpanan nilai, kondisi, dan perulangan dengan sekaligus. Disini, kita sebagai calon programmer diwajibkan untuk mampu menggabungkan berbagai konsep ke dalam satu kesatuan.

Buatlah pseudocode yang menggunakan looping dan conditional sekaligus, berangkat dari kasus berikut:

Seorang anak SD sedang belajar angka genap dan ganjil. Dia ditugaskan oleh ibu guru untuk menghitung angka dari 1 sampai 10 dengan menyebut angka tersebut dan untuk setiap angka ganjil, ia harus menyebut "ODD!" dan sebaliknya jika genap, ia harus menyebut "EVEN!".

Tunggu! Angka disebut genap jika habis dibagi dua. Tapi bagaimana saya menyebutnya di pseudocode?

Mudah, caranya adalah menggunakan yang namanya mod, atau kepanjangannya modulo.

Modulo, adalah sebuah proses matematika untuk mendapatkan reminder atau sisa bagi dari sebuah proses pembagian. Misal, angka 3 jika dibagi 2 sisanya adalah 1.

Nah berarti kita bisa menyebutkan proses ini sebagai 3 mod 2.

```
1  STORE "count" to 1
2
3  WHILE "count" < 11
4    DISPLAY "count"
5    CALCULATE "count" mod 2
6    STORE "remainder" to the result of calculation
7    IF "remainder" equals to 0
8      DISPLAY "EVEN!"
9    ELSE
10     DISPLAY "ODD!"
```

CHALLENGE +

1. Algoritma Luas_Keliling_Lingkaran (ini merupakan judul algoritma)

```
{
Menghitung luas dan keliling untuk ukuran jari-jari tertentu. Algoritma menerima masukkan
jari-jari lingkaran, menghitung luas dan kelilingnya, dan mencetak  luas lingkaran ke piranti
keluaran
} (ini spesifikasi algoritma)
```

2. Deklarasi

```
const phi = 3.14 {nilai konstanta phi}  R : real {jari-jari lingkaran}
Luas : real {luas lingkaran}  Keliling : real {keliling lingkaran}
```

3. Deskripsi Read (R)

```
Luas = phi * R * R  Keliling = 2 * phi * R  Write (luas, keliling)
```