A R D I A N S Y A H

ALGORITMA PEMROGRAMAN PROSEDURAL

Algoritma Pemgoraman Prosedural

Penulis : Ardiansyah

Editor :

ISBN : xxxxxxxxx

Desain sampul : -

Penerbit: Andi Publisher/Elex Media Komputindo/Informatika

Ucapan Terima kasih

Banyak pihak terlibat dan berkontribusi selama penulisan buku ini yang persis dimulai sejak Semester Gasal 2023/2024. Oleh karena itu, halaman ini penulis dedikasikan khusus untuk menghaturkan terima kasih kepada mereka-mereka yang telah turut memberikan sumbangsihnya:

Mahasiswa Alpro 211830731 IF UAD Kelas A, H, dan I 2022/2023, Kelas A dan H 2023/2024. Mereka adalah pembaca generasi awal sewaktu buku ini sewaktu masih berupa lembaran catatan kuliah. Mereka pula yang menjadi "*proof-reader*" awal *cum* objek percobaan penerapan materi buku ini ke dalam perkuliahan dan praktikum. Doktor Rinaldi Munir (ITB). Xx, xx, xx dan xx untuk studi kasus Pemutar Lagu, xx, xx, dan xx untuk studi kasus Antrian Berprioritas.

Daftar Isi

Ucapan Terima kasih 3
Daftar Isi 4
Bab 1 Pendahuluan 8
Bab 2 Pendahuluan 12
2.1 Pemecahan Masalah 12
2.2 Apa itu Algoritma Pemrograman Prosedural? 15
2.3 Dari algoritma ke layar monitor 17
2.4 Apa itu Pemrograman 19
2.5 Tahapan Pemecahan Masalah 19
2.6 Mengapa belajar algoritma pemrograman 19
2.7 Paradigma Pemrograman 19
Bab 3 Penulisan Algoritma 20
3.1 Pseudocode 20
3.2 Konstruksi Dasar Algoritma 23
3.2.1 Membaca Sekuensial (urut) 23
3.2.2 Pemilihan (seleksi) 23
3.2.3 Pengulangan (repetisi) 24
3.3 Latihan 24
Bab 4 Data 26
4.1 Definisi Data pada algoritma 26
4.2 Tipe/Jenis data 27
4.2.1 Integer 27
4.2.2 Float/Real 27
4.2.3 Karakter 28
4.2.4 String 28
4.3 Latihan 29
Bab 5 Variabel 30
5.1 Tukar (swap) nilai 31
5.2 Latihan 32
Bab 6 Operator 32
6.1 Aritmatika 32

6.2 Logika 32
6.3 Penugasan 32
6.4 Perbandingan 33
Bab 7 Statement 34
Bab 8 Ekspresi 35
Bab 9 Blok dan indentasi 35
Bab 10 Efektivitas dan Efisiensi Algoritma 35
Bab 11 Seleksi kondisi (IF-ELSE) 35
11.1 Format umum seleksi kondisi IF 35
11.2 Format umum seleksi kondisi IF-ELSE 37
Dua kemungkinan jalur urutannya yaitu: 1-2-3-9 atau 1-4-5-6-7-8-9 37
11.3 Bentuk umum seleksi kondisi `if-else if-else` 38
11.4 If bersarang (*nested If*) 39
11.5 Latihan 39
11.6 Catatan Bibliografi 40
Bab 12 Perulangan (Loop) 41
12.1 Pengantar Loop 41
12.2 Perulangan For 42
12.2.1 Perulangan for bersarang (nested for) 46
12.3 Sejarah Loop 47
12.4 Perulangan While 47
12.5 Break-Continue 49
12.6 Latihan 49
Bab 13 Array 1-Dimensi 50
13.1 Pengertian Array 50
13.2 Sejarah Array 50
13.3 Konsep Dasar Array 50
13.4 Karakteristik array 51
13.5 Insert Elemen 52
13.5.1 Menambah elemen di ujung 52
13.6 Mengakses Elemen 52
13.6.1 Memanggil Langsung 52

13.6.2 Menggunakan Loop 52
13.7 Array Bersarang 53
13.8 Manipulasi Array 53
13.9 Latihan 54
Bab 14 Penghitungan (counting) dan Penjumlahan (summation) 55
14.1 Penghitungan (*counting*) 55
14.2 Penjumlahan (Summation) 57
14.3 Latihan 59
Bab 15 Fungsi/Prosedur 61
15.1 Sejarah Fungsi/Prosedur 61
15.2 Konsep Dasar Fungsi/Prosedur 61
15.3 Parameter 61
15.4 Argumen 61
15.5 Latihan 61
Bab 16 Sorting 62
16.1 Selection Sort 62
16.2 Insertion Sort 63
Bab 17 Searching 65
17.1 Linear Search 65
17.2 Binary Search 66
17.3 Efektivitas dan efisiensi Binary Search 67
Bab 18 Array Dua Dimensi dan Operasi Matriks 68
Bab 19 String 71
19.1 Definisi String 71
19.2 Representasi String 71
19.3 Operasi Dasar String 71
19.3.1 Penggabungan (*concatenation*) 71
19.3.2 Perbandingan String 71
19.3.3 Panjang String (*length*) 72
19.3.4 Pemotongan (Substring) string 73
19.4 Manipulasi String 74
19.4.1 Substring 74

19.4.2 Pencarian substring 75
19.4.3 Tokenisasi (pemisah string) 79
19.4.4 Pencarian (*searching*) 80
19.4.5 Penggantian (*replacement*) 80
19.4.6 Penggabungan (*joining*) 80
Bab 20 Call Stack 81
Bab 21 Rekursif 81
21.1 Definisi Rekursif 81
21.2 Sejarah Rekursif 81
21.3 Konsep Dasar Rekursif 81

Bab 1 Pendahuluan

Perhatikanlah, dalam keseharian kita nyaris selalu mengikuti yang namanya **instruksi**. Ketika memasak, sebenarnya kita mengikuti instruksi yang ditulis pada resep masakan yang dibuat oleh koki atau ahli kuliner. Rangkaian adegan yang diperankan para aktris dalam sebuah film selalu mengikuti instruksi dari sutradara yang berpedoman pada naskah cerita yang ditulis oleh penulis naskah atau skenario. Begitu pula halnya sebuah program komputer yang ditulis oleh programer, ia merupakan susunan baris instruksi yang harus dieksekusi oleh komputer untuk menyelesaikan suatu tugas tertentu. Apa saja yang menjadi persamaan dan perbedaan di antara ketiganya? Mari kita bahas satu per satu.

[ilustrasi: Problem/Tugas -> Perintah -> (1) dicerna dan dieksekusi oleh manusia, (2) dicerna dan dieksekusi oleh komputer (PC/laptop/HP/robot/dll)]

Ketika memasak, kita akan mengikut langkah-langkah yang ditulis di resep masakan. Di dalam resep masakan tertera tentang bahan-bahan yang dibutuhkan seperti air, garam, bawang, cabai, margarin, sayur, buah, dan banyak lagi yang lainnya sesuai jenis masakan yang hendak dibuat. Bahan-bahan tersebut akan diolah dalam proses yang dinamakan memasak. Selama proses tersebut, juru masak akan melakukan berbagai macam aktivitas seperti mengiris, mengaduk, menumbuk, meniris, menggoreng, menyangrai, memanggang, dan lain sebagainya. Hasil dari seluruh aktivitas tersebut nantinya adalah berupa makanan yang disajikan ke meja makan untuk siap disantap oleh penikmatnya.

Ketika *shooting*, para aktris akan patuh pada instruksi yang dijabarkan pada naskah atau skrip yang ditulis oleh penulis skenario dengan arahan sutradara. Agar adegan bisa berjalan sesuai yang diharapkan pada naskah, maka dibutuhkan berbagai bahan *shooting* yang sering disebut sebagai properti *shooting*. Banyak sekali properti yang bisa digunakan untuk keperluan adegan film sesuai peruntukannya seperti rumah, kamar, bantal, sapu, pisau, kendaraan roda, tiga, empat hingga truk atau peti kemas, sampai kendaraan lapis baja untuk film perang. Properti tersebut digunakan selama proses *shooting* dan digunakan sesuai instruksi atau naskah. Hasil dari proses *shooting* yang bisa memakan waktu berbulan-bulan ini adalah film yang bisa kita saksikan di bioskop atau layanan *streaming*.

Bagaimana dengan program komputer. Mungkin ini yang paling tidak akrab di mata dan telinga para mahasiswa bidang komputer dibanding kedua contoh sebelumnya. Padahal, wujud nyata hasil dari program-program komputer tersebut adalah berbagai aplikasi dan piranti lunak yang sangat akrab di kehidupan mereka sehari-hari.

Manusia hidup dibantu oleh teknologi baik itu perangkat keras dan perangkat lunak. Membeli makanan cukup membuka aplikasi pesan makanan. Ingin bepergian cukup membeli tiket lewat aplikasi. Nyaris semua bidang, dari perdagangan, pemerintahan hingga pertanian sudah didukung oleh aplkiasi.

Aplikasi tersusun atas perangkat lunak dan lebih spesifik lagi adalah program. Suatu program tersusun atas baris perintah untuk mengerjakan suatu tugas tertentu. Susuran urutan perintah atau instruksi tersebut adalah penerapan dari algoritma pemecahan masalah terhadap tugas yang

diberikan. Artinya, kemudahan hidup yang dirasakan manusia sekarang adalah wujud dari berjalannya instruksi-instruksi yang tertulis pada program yang dibuat oleh programer. Tiap baris pada program dieksekusi oleh CPU (*central processing unit*) untuk diproses berdasarkan inputan/masukan yang diberikan lewat piranti input seperti keyboard, mic atau kamera untuk selanjutnya memberikan output atau keluaran sesuai yang diperintahkan pada baris program. Input atau masukan yang diproses komputer biasanya berupa data yang bisa bertipe numerik atau karakter. Sedangkan aktivitas yang dilakukan oleh CPU bisa berupa operasi aritmatika (perkalian, penjumlahan, pengurangan, pembagian), operasi logika, perbandingan, cetak, dan lain sebagainya. Semua aktivitas tersebut memiliki satu tujuan utama yang memberikan output pada ke layar atau suara dan bentuk lainnya.

Nah, bila kita ambil benang merah dari ketiga contoh di atas, maka ada empat persamaan di antara ketiganya yaitu sebagai berikut:

1. Input
   Tiap proses atau aktivitas yang akan dikerjakan memerlukan masukan atau bahan atau inputan. Memasak buku bahan, adegan film butuh properti, dan program butuh data.
2. Proses/Aktivitas
   Inputan yang diberikan tidak akan bermanfaat bila tidak diproses. Dengan aktivitas atau proses inilah perlahan-lahan akan terjadi kemajuan atau *progress* untuk mencapai hasil atau output yang diharapkan. Mengiris bawang merah merupakan satu langkah maju untuk menghasilkan nasi goreng, berlari mengejar musuh merupakan satu proses penting untuk menyelesaikan sebuah adegan dari puluhan atau ratusan adegan lain, dan membagi bilangan yang diinputkan merupakan satu aktivitas penting sebelum akhirnya akan ditampilkan ke layar monitor
3. Instruksi
   Agar memberikan hasil yang berkualitas, input dan proses/aktivitas harus diperlakukan sesuai dengan instruksi yang diberikan. Jika kita menyalakan api terlalu besar tidak mengikuti batasan yang diberikan instruksi, maka masakan bisa gagal matang. Begitu pula jika adegan menangis tidak menunjukkan sisi natural seperti instruksi pada naskah, maka tentu akan diminta mengulangi oleh sutradara. Pesan *error*, atau ketidaksesuaian antara hasil yang diharapkan dengan hasil aktual akan muncul jika CPU mengeksekusi program yang dibuat programer salah dalam memberikan instruksi.
4. Output
   Rangkaian mulai dari input, diproses oleh aktivitas yang mengikuti instruksi maka akan menghasilkan keluaran. Proses memasak di dapur akan menghasilkan masakan lezat yang terhidang di meja makan, proses *shooting* yang melelahkan akan berakhir dengan tayang di berbagai layar bioskop atau layanan *streaming*. Begitu pula program, akan menampilkan solusi yang dibutuhkan oleh penggunanya.

Menjadi *chef*, penulis naskah, dan programer profesional butuh waktu bertahun-tahun. Mengapa perlu bertahun-tahun? Karena untuk menghasilkan masakan yang lezat, seorang chef harus paham bahan-bahan apa saja yang diperlukan, di mana dan bagaimana mendapatkannya, bagaimana mengolah dan meraciknya, dan seterusnya. Untuk menghasilkan film *box office*, maka naskah

cerita harus menuntun pada adegan yang membuat penontonnya larut dan hanyut membutuhkan properti pendukung serta aktris yang mumpuni. Ia tahu bagaimana merangkai cerita yang baik sesuai properti yang digunakan. Terakhir, ia paham cara megemas naskah agar mudah dipahami para aktris dan sutradara.

Begitu pula program komputer yang menjadi solusi jutaan penggunanya tentu saja si programer harus paham karakteristik dan jenis data yang diperlukan, rentang nilai tiap jenis data, ukuran tiap jenis data terpilih jika disimpan dalam *memori* komputer (RAM), memvalidasi, membuat rangkaian operasi komputasi yang dibutuhkan, hingga struktur data yang digunakan sehingga proses komputasinya menjadi efektif dan efisien dan memuaskan penggunanya. Terakhir, ia paham bagaimana menyajikan output dan menjamin kebenaran dan validitas hasilnya.

Keahlian yang mereka miliki tersebut tidak diperoleh dalam waktu singkat, namun butuh latihan dan dedikasi dalam jangka panjang. Sehingga tidak salah jika bisa dikatakan bahwa profesi mereka bertiga adalah orang-orang ahli dalam merangkai baris instruksi untuk menyelesaikan tugas dan masalah tertentu.

Sebut saja sedikit dari programer top dunia seperti Linus Torvalds yang membuat kernel sistem operasi Linux, Rasmus Lerdof yang membuat interpreter bahasa pemrograman PHP, Guido van Rossum yang membuat bahasa pemrograman Python, Dennis Ritchie yang mengembangkan sistem operasi Unix.

Tidak ketinggalan programer top berasal dari Indonesia seperti Ariya Hidayat yang mengembangkan Phantom, Steven Halim yang membina para programer untuk *competitive programming*, xxx

Pembuat Instruksi	**Produk**	**Pemroses**
Chef/ahli kuliner	Resep	Juru masak
Penulis naskah	Naskah film	Aktris/Aktor
Programer	Program komputer	Central Processing Unit (CPU)

Penulis terinspirasi dari Learnable Programming yang diperkenalkan oleh Bret Victor. Learnable Programming[1] juga menginspirasi Khan Academy untuk materi pemgrograman. Victor berargumen dua hal terhadap pembelajaran pemrograman yaitu:

1. Pemrograman adalah mengenai cara berpikir (*a way of thinking*), bukan keterampilan menghapal.
2. Orang mengerti/paham atas apa yang mereka lihat. Jika pemrogram tidak dapat melihat apa yang dilakukan oleh program yang dibuatnya, maka ia akan kesulitan untuk memahaminya.

Kedua argumen inilah yang melandasi penulis untuk menulis buku ini dengan menekankan pada apa yang terjadi di tiap baris program ketika dieksekusi seperti yang telah lama disinggung oleh Dromery pada pembukaan buku klasik yang ditulisnya tahun 19xx berjudul How to Solve It xxx.

[1] http://worrydream.com/LearnableProgramming/

Bab 2 Pendahuluan

"Ketika Einstein ditanya bagaimana cara menyelesaikan masalah dalam waktu 1 jam, Einstein menggunakan 55 menit untuk mendefinisikan masalah, dan 5 menit sisanya untuk menyelesaikan masalah"

2.1 Pemecahan Masalah

Dalam bidang algoritma dan pemrograman, masalah tidak selalu berkonotasi negatif, namun lebih kepada sebuah tugas atau tantangan yang harus diselesaikan menggunakan program. Contoh bentuk tugas atau tantangan misalnya, mencari, mengurutkan, menghitung, dsb…

Terdapat beberapa langkah utama menyelesaikan tugas atau tantangan yang diberikan.

1. Memahami masalah atau tugas yang diberikan
   Tahap pertama dan utama dalam pemecahan masalah adalah memahami masalah atau tugas dengan baik. Paham berarti tahu sedetil mungkin terhadap masalah yang hendak dipecahkan.
   Seperti pepatah Einstein di awal bab ini, 92% waktunya dihabiskan untuk mendefinisikan masalah, dan cuma 8% untuk menyelesaikan masalah. Ini menunjukkan bahwa tahap pertama ini menjadi tahap paling krusial. Oleh karena itu Anda harus fokus melatih diri untuk mengasah keterampilan mendefinisikan masalah.

   Mari kita ambil contoh, buatlah program untuk menentukan apakah sebuah bilangan yang diinputkan adalah bilangan genap atau bukan.

   Untuk memahami permasalahan tersebut kita akan rinci dan definisikan sebagai berikut:
   a. Definisikan masalah dengan jelas. Sesuai definisinya, maka kita perlu mengingat kembali definisi bilangan genap. Bilangan genap adalah suatu bilangan bulat positif yang habis dibagi 2 tanpa ada sisa. Contoh bilangan genap: 2, 4, 6, 8, 10, 100, 102, dst. Artinya bilangan genap itu ada dua ciri utama yaitu:
      - Bilangannya bulat dan positif
      - Jika dibagi 2 maka tidak ada sisa, atau sisanya adalah 0
   b. Kumpulkan informasi dengan lengkap. Artinya kita bisa juga membuat semacam simulasi awal di kertas atau media lainnya. Aktivitas ini perlu dilakukan untuk mengetahui pola yang biasanya terbentuk.

      Contoh:
      Inputkan bilangan: 5
      Maka, 5 dibagi 2 sisanya 1. Berarti 5 bukan bilangan genap, karena sisanya bukan 0

      Inputkan bilangan: 7
      Maka, 7 dibagi 2 sisanya 1. Berarti 7 bukan bilangan genap, karena sisanya bukan 0

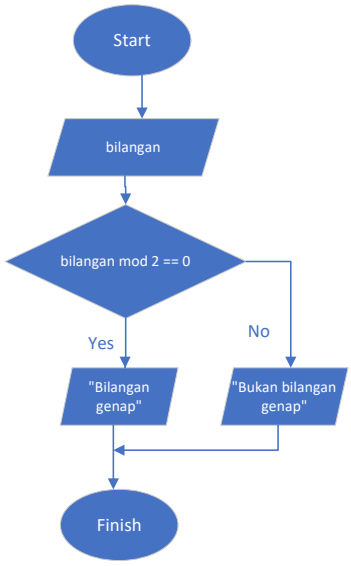
      Inputkan bilangan: 8

Maka, 8 dibagi 2 sisanya 0. Berarti 8 adalah bilangan genap, karena sisanya 0.

2. Mengidentifikasi input dan output yang diperlukan
   Pada tahap ini kita akan mengidentifikasikan input dan output pada algoritma yang akan kita rancang.
   Untunglah kita sudah mengerjakan tahap pertama. Jadi, bila kita lihat, maka inputannya adalah seluruh bilangan yang bernilai positif. Sedangkan outputnya adalah berupa string yang menampilkan dua kemungkinan yaitu:
   - "Bukan bilangan genap", atau
   - "Bilangan genap"

Input	**Output**
5	"Bukan bilangan genap"
9	"Bukan bilangan genap"
12	"Bilangan genap
82	"Bilangan genap

3. Membuat algoritma
   Pada tahap ini kita akan membuat algoritma untuk menggambarkan program yang akan dibuat. Ada tiga pilihan yang bisa dibuat yaitu menggunakan flowchart, *pseudocode* atau deskriptif. Secara lengkap ketiganya bisa dibaca pada subbab xx.




```
Kamus
bilangan: integer

Algoritma

input ("Masukkan bilangan: ", bilangan)
```

```
if bilangan modulus 2 == 0
   output ("Bilangan genap")
else
   output ("Bukan bilangan genap")
```

4. Menguji algoritma menggunakan data uji

```
input ("Masukkan bilangan: ", 5)

if 5 modulus 2 == 0 #False

else

   output ("Bukan bilangan genap")


input ("Masukkan bilangan: ", 4)
if 4 modulus 2 == 0   #True
   output ("Bilangan genap")
```

5. Menulis kode program menggunakan salah satu atau beberapa bahasa pemrograman. Aktivitas ini disebut dengan *coding*.

```
#include <iostream>
using namespace std;

int main() {
    int bilangan;

    // Meminta pengguna untuk memasukkan bilangan
    cout << "Masukkan bilangan: ";
    cin >> bilangan;

    // Memeriksa apakah bilangan tersebut genap atau tidak
    if (bilangan % 2 == 0) {
        cout << bilangan << " adalah bilangan genap." << endl;
    } else {
        cout << bilangan << " bukan bilangan genap." << endl;
    }

    return 0;
}
```

6. Menguji program menggunakan data uji

Berdasarkan urutan tersebut kita bisa lihat bahwa masalah atau tugas berada di luar komputer. Programer, yang merupakan orang yang memecahkan masalah akan menyiapkan solusi permasalahan dengan menggunakan algoritma. Algoritma haruslah bersifat generik dan tidak terikat dengan bahasa pemrograman. Untuk mengimplementasikan algoritma, programer harus menggunakan bahasa pemrograman. Mengapa demikian? Karena untuk memproses instruksi pada algoritma haruslah menggunakan bahasa yang dimengerti oleh komputer. Bahasa yang dimengerti oleh komputer adalah bahasa pemrograman.

2.2 Apa itu Algoritma Pemrograman Prosedural?

Penulis yakin, pembaca pasti pernah melihat tampilan daftar nama atau presensi kelas yang terurut berdasarkan nomor mahasiswa atau siswa atau nama atau abjad. Begitu pula pasti pernah melihat hasil pencarian ketika memasukkan kode tiket di mesin tiket mandiri KAI atau kode belanja/voucher game di aplikasi mobile, seperti halnya hasil pencarian di mesin pencari Google atau Bing, serta rute yang ditawarkan oleh Google Maps.

Tahukan Anda, bahwa ketiga hal tersebut bisa terwujud dengan bantuan algoritma. Algoritma pengurutan (*sorting*) digunakan untuk mengurutkan daftar nama presensi kelas dan algoritma pencarian (*searching*) digunakan untuk menemukan hasil pencarian berdasarkan kode voucher aplikasi game dan kata kunci tertentu di mesin pencari Google hingga rute Google Maps.

{pseudocode sorting}

Berdasarkan algoritma pengurutan tersebut, kita lihat bahwa semuanya ada awal algoritma, pendefinisian inputan, proses/operasi/aktivitas yang dilakukan terhadap inputan, lalu output atau hasil lalu pengentian atau akhir algoritma. Semua harus selesai secara hingga dan dipastikan bisa berhenti.

Banyak definisi algoritma. Menurut…

Menurut penulis, algoritma adalah keterampilan menulis rangkaian baris perintah yang akan dibaca komputer guna menyelesaikan suatu tugas spesifik secara efektif dan efisien.

Tugas spesifik yang diemban algoritma merupakan…, efektif artinya…, sedangkan efisien bermakna bahwa….

Algoritma itu seperti resep masakan. Ia tersusun atas langkah-langkah yang berurutan hingga selesai. Tiap langkah pada resep merupakan perintah yang harus dilaksanakan. Perintah tersebut disebut sebagai **pernyataan** atau ***statement***.

Pernyataan

```
Tulis "Selamat pagi!"
```

Pada manusia berarti memerintahkan manusia untuk menuliskan kalimat tersebut di kertas atau papan tulis. Sedangkan pada komputer berarti memerintahkan komputer untuk menampilkan tulisan tersebut ke layar monitor. Pada bahasa pemrograman, pernyataan tulis bisa berbeda-beda. Di Python menggunakan `print()`, Java `System.out.println()`, C++ `cout`, Javascript `console.log()`, sedangkan C# menggunakan `Console.Writeln()`.

Pernyataan lain bisa berupa perintah operasi aritmatika.

Kalikan bilanganX dengan bilanganY lalu simpan ke variabel hasil

Perlu diingat bahwa pada algoritma, simbol perkalian bukanlah ×, melainkan tanda asterik (*).

Pernyataan yang membandingkan

Jika IPK ≥ 3.5 maka tulis "Mendapat beasiswa"

Jika IPK ≥ 3.4 maka SKSMax = 22

Penyampaian algoritma bermacam-macam. Kita bisa membuat algoritma belajar seperti berikut:

1. Siapkan buku Alpro
2. Buka halaman 17-100
3. Baca halaman 17-100
4. Kerjakan soal
5. Ulangi pengerjaan soal hingga skor minimum 85
6. Selesai

Tentu agak menyulitkan bila algoritma dalam bahasa deskriptif seperti itu. Agar bisa dimengerti oleh komputer, maka bahasa atau perintah yang diberikan harus sesuai kaidah. Sehingga diperlukanlah notasi algoritma yang lebih *codeable* yang dikenal sebagai *psuodocode*.

Jadi, kalau dalam bahasa deskriptif, misalnya ada perintah atau pernyataan: "**jumlahkan** bilangan **4** dengan bilangan **6** lalu isikan ke variabel **x**", maka dalam notasi algoritmik bisa ditulis sebagai berikut:

```
x = 4 + 6
```

Dibaca: "jumlahkan 4 dan 6 lalu hasilnya isikan ke variabel x"

Contoh lain:

Deskriptif: "cetak bilangan 1 hingga 5 "

Tentu hasil eksekusinya: 1 2 3 4 5

Algoritmik:

```
for (i=1; i < 5; i++)
   cetak i
```

Dibaca:

`i=1`: Isikan 1 ke variabel *i* sebagai inisiasi awal

`i<5`: Periksa kondisi apakah *i* lebih kecil dari 5?

Jika Ya, maka

cetak *i*, lalu

i++: tambahkan *i* dengan 1

Jika tidak, maka berhenti atau selesai

Simbol atau operasi *i*, <, =, i++, adalah xxx. Peranannya sangat vital sebagai bagian komunikasi untuk membuat perintah ke komputer. Oleh karena itu, pada bab xx membahas lengkap tentang simbol, arti dan maknanya yang bisa digunakan untuk membuat algoritma.

Beberapa buku menggunakan notasi algoritmik yang cenderung ke Pascal, sedangkan di buku ini penulis menggunakan pendekatan notasi campuran antara C/C++ dan Python. Terutama ketika perulangan For yang menggunakan notasi C/C++ dan block statement yang tidak menggunakan kurawal (*bracklet*) ala Python. Sehingga harapannya pembaca tidak terpengaruh dengan bahasa pemrograman manapun dan bisa cepat memprogramnya ke dalam bahasa pemrograman tertentu.

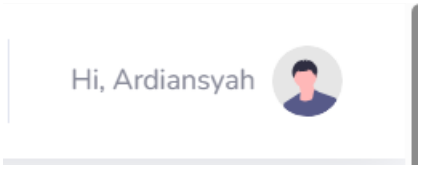
2.3 Dari algoritma ke layar monitor

Untuk mendapatkan solusi dari masalah atau tugas yang diberikan ke komputer, seorang pengguna harus "menunggu" selesainya rangkaian aktivitas/proses yaitu:

1. programer merancang atau menyiapkan algoritma yang hendak dipakai
2. programer menulis kode instruksi menggunakan bahasa pemrograman
3. Tiap baris kode dieksekusi oleh CPU
4. CPU mengalokasikan memori untuk instruksi, variabel, dan data
5. CPU menerjemahkan baris kode ke bahasa mesin
6. CPU meletakkan instruksi, variabel, data ke memori sebagai bit biner
7. CPU menjadwalkan eksekusi
8. CPU menjalankan eksekusi
9. CPU menerjemahkan balik hasil eksekusi ke bahasa tingkat tinggi
10. CPU menampilkan hasil eksekusi ke piranti output

{gambar. Alur seperti TCP/IP bentuk stack}

Perhatikanlah Gambar xx. Kita sering melihat tampilan sapaan seperti itu pada aplikasi yang kita sudah terdaftar sebagai pengguna.



Gambar.xx

Sebagai bentuk sederhana sekaligus ilustrasi, untuk menampilkan tulisan "Hi, Ardiansyah" seperti Gambar xx tersebut, seorang programer bahasa C++ akan menuliskan kodenya seperti Listing xx.

```c
#include <stdio.h>

int main() {
   printf("Hi, Ardiansyah\n");
}
```

(a). greeting.c

```python
print("Hi, Ardiansyah")
```

(b). greeting.py

Ingatlah, kedua kode program tersebut ditulis untuk manusia, bukan mesin. Artinya, kode tersebut untuk dibaca, dimengerti dan dipahami baik oleh pembuatnya maupun programer lain. Mesin komputer yaitu CPU tidak akan paham dengan bahasa tingkat tinggi yang hanya dimengerti manusia tersebut. Agar bisa menghasilkan output seperti yang diharapkan

sebelumnya, maka kode program tersebut harus diterjemahkan terlebih dahulu agar bisa dimengerti oleh mesin.

Langkah pertama untuk menerjemahkan adalah mengonversi seluruh karakter kode program ke dalam ASCII.

#	i	n	c	l	u	d	e	<spasi>	<	s
35	105	110	99	108	117	100	101	32	60	115
t	d	i	o	.	h	>	<enter>	<enter>	i	n
116	100	105	111	46	104	62	10	10	105	110
t	<spasi>	m	a	i	n	(	)	<spasi>	{	<enter>
116	32	109	97	105	110	40	41	32	123	10
<spasi>	<spasi>	<spasi>	<spasi>	p	r	i	n	t	f	(
32	32	32	32	112	114	105	110	116	102	40
"	H	i	,	<spasi>	A	r	d	i	a	n
34	72	105	44	32	65	114	100	105	97	110
s	y	a	h	\	n	"	)	;	<enter>	}
115	121	97	104	92	110	34	41	59	10	125

35	105	110	99	108	117	100	101	32	60	115
116	100	105	111	46	104	62	10	10	105	110
116	32	109	97	105	110	40	41	32	123	10
32	32	32	32	112	114	105	110	116	102	40
34	72	105	44	32	65	114	100	105	97	110
115	121	97	104	92	110	34	41	59	10	125

35	105	110	99	108	117	100	101	32	60	115
100011	1101001	1101110	1100011	1101100	1110101	1100100	1100101	100000	111100	1110011
116	100	105	111	46	104	62	10	10	105	110
1110100	1100100	1101001	1101111	101110	1101000	111110	1010	1010	1101001	1101110
116	32	109	97	105	110	40	41	32	123	10
1110100	100000	1101101	1100001	1101001	1101110	101000	101001	100000	1111011	1010
32	32	32	32	112	114	105	110	116	102	40
100000	100000	100000	100000	1110000	1110010	1101001	1101110	1110100	1100110	101000
34	72	105	44	32	65	114	100	105	97	110
100010	1001000	1101001	101100	100000	1000001	1110010	1100100	1101001	1100001	1101110
115	121	97	104	92	110	34	41	59	10	125
1110011	1111001	1100001	1101000	1011100	1101110	100010	101001	111011	1010	1111101

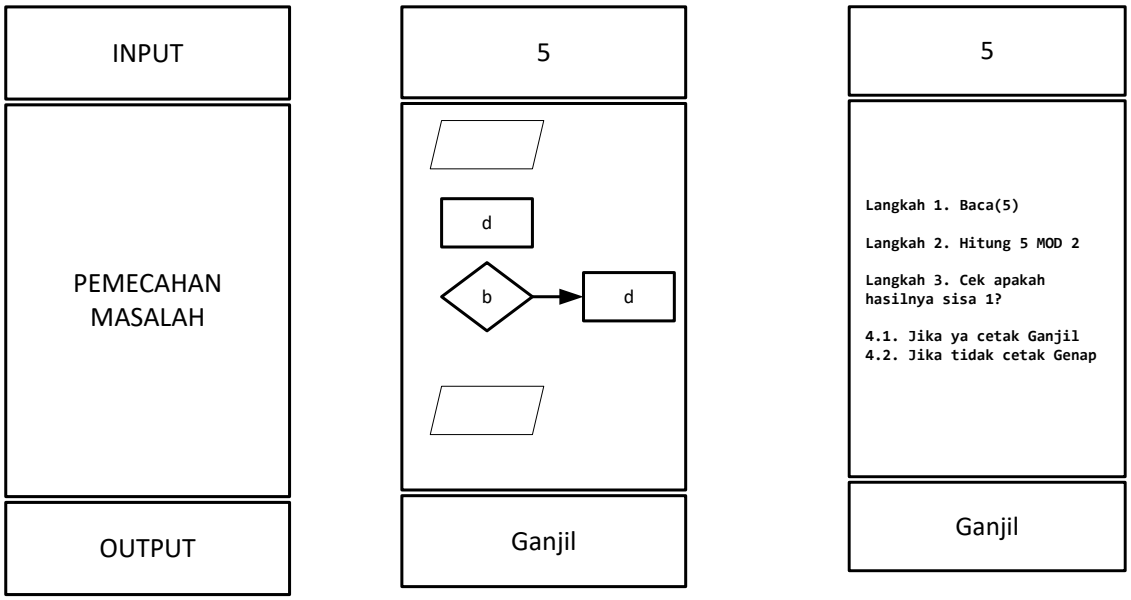
0100011	1101001	1101110	1100011	1101100	1110101	1100100	1100101	0100000	0111100	1110011
1110100	1100100	1101001	1101111	0101110	1101000	0111110	0001010	0001010	1101001	1101110
1110100	0100000	1101101	1100001	1101001	1101110	0101000	0101001	0100000	1111011	0001010
0100000	0100000	0100000	0100000	1110000	1110010	1101001	1101110	1110100	1100110	0101000
0100010	1001000	1101001	0101100	0100000	1000001	1110010	1100100	1101001	1100001	1101110
1110011	1111001	1100001	1101000	1011100	1101110	0100010	0101001	0111011	0001010	1111101

Ketika kita menyimpan file program greeting.c atau greeting.py ke *hard disk*, maka yang tersimpan adalah dalam bentuk bit-bit seperti yang ditunjukkan tabel/gambar xx. Jadi, yang kita bisa baca di layar monitor sebagai seorang programer adalah hasil konversi dari biner ke karakter alfanumerik. Begitu pun sebaliknya, ketika kita mentikkan tiap karakter satu per satu, maka yang dibaca dan tersimpan ke *hard diski* adalah hasil konversi ke biner.

ASCII

2.4 Apa itu Pemrograman

Algoritma dan pemrograman adalah suatu kegiatan analisis dan pemecahan masalah berdasarkan inputan yang diberikan berupa langkah-langkah yang sistematis dengan hasil output yang pasti. Apakah setiap program selalu butuh input? Jawabannya tidak selalu.



2.5 Tahapan Pemecahan Masalah

2.6 Mengapa belajar algoritma pemrograman

2.7 Paradigma Pemrograman

Deklaratif, proseduran, fungsional, berorientsi objek.

Bab 3 Penulisan Algoritma

Perhatikan teks singkat berikut. Itu adalah contoh sebuah tulisan. Di dalamnya terdiri dari karakter huruf, angka, hingga tanda baca berupa koma dan titik. Semuanya memiliki makna dan arti tersendiri. Misalnya koma, berarti berhenti sejenak, titik berarti berhenti.

Siapkan wadah nampan
Isi air sebanyak 500 cc
Tuang terigu

3.1 Pseudocode

Nama algoritma

Setiap algoritma yang ditulis untuk memecahkan suatu masalah atau *task* harus diberi nama. Penamaan algoritma harus mencerminkan tugas yang dia emban. Sebagai contoh, masalah untuk menghitung jarak antara dua titik bisa diberi nama **`Algoritma Hitung Jarak Dua Titik`**, tugas pengurutan data mahasiswa bisa diberi nama **`Algoritma Pengurutan Data Mahasiswa`**, begitu seterusnya.

Kamus data

Data dan inputan yang akan digunakan dalam algoritma perlu dideklarasikan terlebih dahulu. Pendeklarasian ini meliputi nama variabel dan tipe datanya. Format penulisannya sebagai berikut:

```
Kamus
<variabel>     : <tipe data>
```

Contoh:

```
Kamus
tinggiBadan           : real
jumlahData, usia      : integer
tombol                : char
namaDesa              : string
```

Berikut contoh algoritma

```
Algoritma Calculate Median

Kamus:
bodyHeight                      : array bilangan real
N, i, j, median, middleIndex    : integer
```

```
temp                              : real

Input   :
bodyHeights ← [167, 188, 158, 170, 169]
N ← 5

Output  : median

    1.   for (i ← 0  to N-1 step 1) do
    2.   |   for (j ← i+1 to N-1 step 1) do
    3.   |   |   if (bodyHeights[i] > bodyHeights[j]) then
    4.   |   |   |   temp ← bodyHeights[i]
    5.   |   |   |   bodyHeights[i] ← bodyHeights[j]
    6.   |   |   |   bodyHeights[j] ← temp
    7.   |   |   end if
    8.   |   end for
    9.   end for
    10.
    11.  if ( (N MOD 2) = 1 ) then
    12.  |   middleIndex ← (N – 1) / 2
    13.  |   median ← bodyHeights[middleIndex]
    14.  else
    15.  |   middleIndex ← N / 2
    16.  |   middleValue1 ← bodyHeights[middleIndex-1]
    17.  |   middleValue2 ← bodyHeights[middleIndex]
    18.  |   median ← (middleValue1 + middleValue2) / 2
    19.  end if
    20.
    21.  cetak median
```

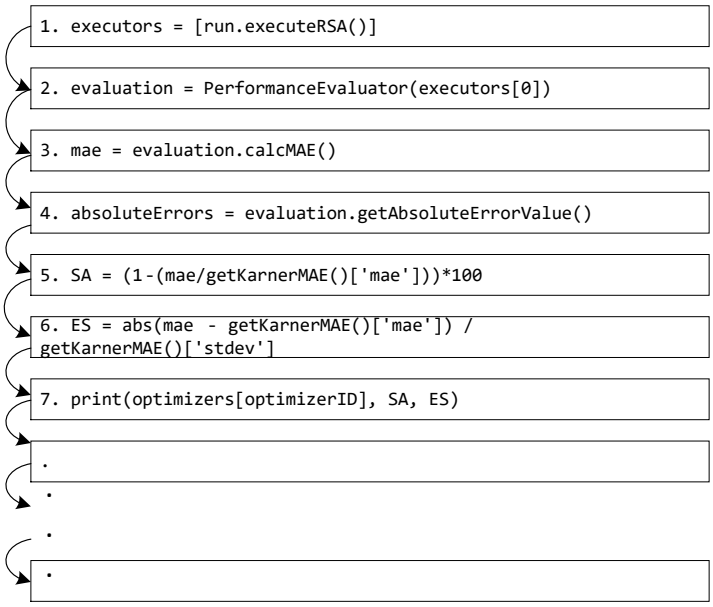
Eksekusi algoritma Calculate Median

```
1.  bodyHeights ← [167, 188, 158, 170, 169]
                    i    j
3.  numOfBodyHeights ← 5

5.  for (i ← 0 to 4 step i←0+1) do       #bodyHeights[0] = 167 #bodyHeights[1] = 188
6.     for (j ← 1 to 4 step j←1+1) do
7.        if (167 > 188) #False

6.     for (j ← 2 to 4 step j←2+1) do    #bodyHeights[2] = 158 #[167, 188, 158, 170, 169]
                                                                  i         j
7.        if (167 > 158) then
8.           temp ← 167
9.           bodyHeights[0] ← 158        #[158, 188, 158, 170, 169]
10.          bodyHeights[2] ← 167        #[158, 188, 167, 170, 169]

6.     for (j ← 3 to 4 step j←3+1) do    #bodyHeights[3] = 170 #[158, 188, 167, 170, 169]
                                                                  i             j
7.        if (167 > 170) #False

6.     for (j ← 4 to 4 step j←4+1) do    #bodyHeights[4] = 169 #[158, 188, 167, 170, 169]
                                                                  i                   j
7.        if (167 > 169) #False
```

```
6.     stop

5.  for (i ← 1 to 4 step i←1+1) do       #[158, 188, 167, 170, 169]
                                                 i    j
6.     for (j ← 2 to 4 step j←2+1) do    #bodyHeights[1] = 188 #bodyHeiths[2]=167
7.        if (188 > 167) then
8.           temp ← 188
9.           bodyHeights[1] ← 167        #[158, 167, 167, 170, 169]
10.          bodyHeights[2] ← 188        #[158, 167, 188, 170, 169]

6.      for (j ← 3 to 4 step j←3+1) do   #bodyHeights[1] = 167 #bodyHeiths[3]=170
7.         if (167 > 170) #False         #[158, 167, 188, 170, 169]
                                                 i         j
6.      for (j ← 4 to 4 step j←4+1) do   #bodyHeights[1] = 167 #bodyHeiths[4]=169
7.         if (167 > 170) #False         #[158, 167, 188, 170, 169]
                                                 i              j
6.      stop

5.   for (i ← 2 to 4 step i←2+1) do      #[158, 167, 188, 170, 169]
                                                    i    j
6.      for (j ← 3 to 4 step j←3+1) do   #bodyHeights[2] = 188 #bodyHeiths[3]=170
7.         if (188 > 170) then
8.            temp ← 188
9.            bodyHeights[2] ← 170       #[158, 167, 170, 170, 169]
10.           bodyHeights[3] ← 188       #[158, 167, 170, 188, 169]

6.      for (j ← 4 to 4 step j←4+1) do   #[158, 167, 170, 188, 169]
                                                    i         j
7.         if (170 > 169) then           #bodyHeights[2] = 170 #bodyHeiths[4] = 169
8.            temp ← 170
9.            bodyHeights[2] ← 169       #[158, 167, 169, 170, 169]
10.           bodyHeights[4] ← 170       #[158, 167, 169, 188, 170]

6.      stop

5.   for (i ← 3 to 4 step i←3+1) do      #[158, 167, 169, 188, 170]
                                                       i    j
6.      for (j ← 4 to 4 step j←4+1) do   #bodyHeights[3] = 188 #bodyHeights[4]=170
7.        if (188 > 170) then
8.           temp ← 188
9.           bodyHeights[3] ← 170              #[158, 167, 169, 170, 170]
10.          bodyHeights[4] ← 188              #[158, 167, 169, 170, 188]

5.   for (i ← 4 to t step i←4+1)
6.      stop

5.   stop       # 5 MOD 2 = 1

15.  if 1 = 1 then
16.     middleIndex ← 2                 #[158, 167, 169, 170, 188]
17.     median ← 169

25.  cetak 169
```

3.2 Konstruksi Dasar Algoritma

3.2.1 Membaca Sekuensial (urut)



3.2.2 Pemilihan (seleksi)

Pernah lihat daftar ceklist? Biasanya daftar ini bisa kita jumpati pada persyaratan mendaftar sesuatu. Di dalam ceklist tersebut terdapat hal-hal apa saja yang wajib dipenuhi dan yang tidak wajib dipenuhi atau opsional.

Untuk syarat wajib, tentu saja harus dipenuhi semuanya. Jika salah satu ada yang tidak terpenuhi, maka bisa dipastikan Anda gugur karena tidak memenuhi salah satu persyaratan atau lebih.

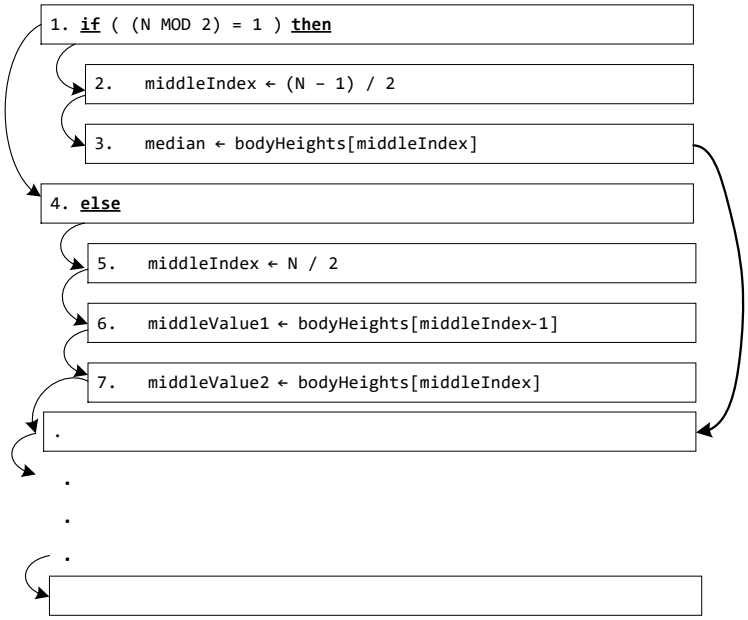
Begitu pula dengan syarat opsional, karena sifatnya boleh dipenuhi dan boleh juga tidak, maka tidak akan menggugurkan persyaratan.

Coba ingat ketika kita hendak minum dari gelas. Secara naluri kita pasti akan meneguk air dalam gelas apabila ada air di dalamnya. Sebaliknya, bila tidak ada air dalam gelas, maka kita tidak akan meneguk air di dalamnya.

Hal itu sama saja menunjukkan prasyarat wajib yang harus dipenuhi oleh gelas. Oleh karena itu dalam seleksi. Kondisi Jika bisa bernilai dua macam yaitu kondisi terpenuhi atau kondisi tidak terpenuhi. Dalam algoritma, kondisi terpenuhi sama saja True dan kondisi tidak terpenuhi adalah False. Jadi, selama kondisinya True, maka perintah di bawahnya (blok di dalam kondisi True) harus dijalankan.

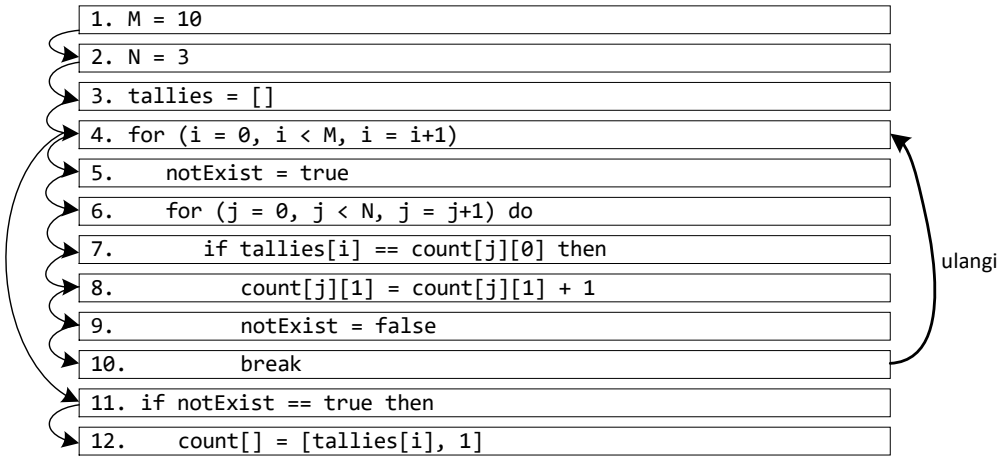
Jika gelas ada air maka:

Teguk air ke tenggorokan



3.2.3 Pengulangan (repetisi)

Latihan



Berikut contoh kode programnya dalam bahasa Dart dan Python

Pembacaan secara beruntun atau sequence

Penimpaan nilai variabel

3.3 Latihan

1. Buatlah algoritma untuk menghitung jarak antara dua titik menggunakan rumus berikut:

$$jarak = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

   Ketentuan:

- Koordinat kedua titik diisi langsung pada variabel
- Buatlah algoritmanya dalam bentuk *flow chart* dan *pseudocode*

2.

Bab 4 Data

Data adalah sekumpulan nilai yang menunjukkan kuantitas, kualitas, fakta atau simbol yang nantinya bisa diinterpretasikan. Sedangkan satu nilai tunggal dari kumpulan data tersebut disebut datum. Definisi lain menyebutkan bahwa data adalah fakta-fakta mentah yang perlu diolah lebih lanjut untuk menghasilkan informasi.

Gaji programer di Indonesia tahun 2023 adalah suatu data. Sedangkan gaji programer *backend* Rp. 13.000.000 adalah datum.

Untuk mendapatkan data biasanya menggunakan berbagai macam teknik pengumpulan data seperti, pengukuran, observasi. Sedangkan pada algoritma, sumber data bisa dari inputan yang dimasukkan oleh user melalui media input seperti keyboard, mouse, mic. Selain itu bisa juga dari data yang berasal dari hardware lain seperti piranti sensor pada IoT atau kamera CCTV.

Intinya, ketika kita sudah berhasil mendapatkan data, maka selanjutnya akan bisa diproses oleh algoritma. Namun, kadangkala kita harus melakukan praproses terlebih dahulu agar nantinya bisa dieksekusi dengan baik oleh algoritma. Misalnya data awalnya masih berupa *string*, maka harus diubah ke tipe *real* atau *integer*.

Contoh data yang dihasilkan oleh sensor suhu IoT:

{

"sensor_id": "S1",

"timestamp": "2023-12-29T12:30:00",

"temperature": 25.5,

"unit": "Celsius"

}

Misalnya kita belanja di supermarket. Total belanja kita adalah Rp. 145.000. Total belanja adalah hasil perhitungan dari item-item yang kita beli. Item-item tersebut adalah data pembelian yang diinputkan oleh kasir, baik melalui keyboard maupun alat pemindai barcode.

[ilustrasi]

Buatkan gambar keyboard yang memasukkan nilai melalui aplikasi. Setelah klik tombol enter, tunjukkan arah panah ke program yang mendeklarasikan atau menyediakan bahkan mengakomodasi inputan data tersebut.

4.1 Definisi Data pada algoritma

Pada algoritma, data adalah bahan baku yang akan diproses lebih lanjut oleh algoritma untuk menghasilkan output sesuai dengan tujuan dari algoritma tersebut.

4.2 Tipe/Jenis data

Ada 4 tipe data utama yaitu integer, karakter, string, dan real (float).

4.2.1 Integer

Integer adalah tipe data berupa bilangan bulat. Artinya tipe data integer tidak menerima data desimal. Integer memiliki ukuran yang bervariasi tergantung pada arsitektur bit komputer. Untuk arsitektur 8-bit, rentang nilai integer adalah -128 hingga 128, hingga arsitektur 64-bit yang mengakomodasi rentang dari -9,223,372,036,854,775,808 hingga 9,223,372,036,854,775,808. Daftar lengkap ukuran tipe data integer disajikan pada tabel xx.

Ukutan tipe integer berdasarkan arsitektur bit komputer

Arsitektur	Batas bawah	Batas atas
8 bit	-128	128
16 bit	-32,768	32,768
32 bit	-2,147,483,648	2,147,483,648
64 bit	-9,223,372,036,854,775,808	9,223,372,036,854,775,808

Berarti, bila kita mendeklarasikan variabel bertipe integer, maka harus mempertimbangkan terlebih dahulu arsitektur bit komputer yang kita gunakan.

```
nilaiUjian : integer
jumlahPenggunaBulanIni ← 345,000,901        [Benar]
jumlahPenggunaBulanIni ← 345,000,901.45     [Salah]


jumlahDiskon : integer
jumlahDiskon ← -150,000      [Benar]
jumlahDiskon ← -150,000.75   [Salah]
```

Kita bisa melihat arsitektur bit komputer kita pada System di Windows, xx di Linux, xx di MacOS.

Gambar xx. Arsitektur bit komputer di Windows 10, Linux, dan MacOS

4.2.2 Float/Real

Tipe data float (floating point number) digunakan untuk menyimpan angka desimal, baik positif maupun negatif.

Float direpresentasikan dengan menggunakan titik desimal (.) sebagai pemisah antara bagian bilangan bulat dan bagian desimal.

Tipe data float dapat menyimpan bilangan dengan presisi tertentu sesuai dengan standar float pada komputer yang digunakan.

Contoh:

Dalam Python: angka_desimal = 3.14

4.2.3 Karakter

Tipe data char digunakan untuk menyimpan satu karakter dalam representasi ASCII atau Unicode.

Biasanya digunakan untuk menyimpan karakter tunggal seperti huruf, angka, atau simbol.

Dalam beberapa bahasa pemrograman, seperti C atau C++, tipe data char direpresentasikan dalam tanda petik tunggal (' '), sedangkan dalam Python, karakter tunggal dapat direpresentasikan dalam tanda petik tunggal (' ') atau tanda kutip ganda (" ").

Contoh:

Dalam C/C++: char huruf = 'A';

Dalam Python: karakter = 'B' atau karakter = "B"

4.2.4 String

Tipe data string digunakan untuk menyimpan serangkaian karakter.

String dapat berisi nol atau lebih karakter, dan biasanya digunakan untuk menyimpan teks.

Dalam kebanyakan bahasa pemrograman, string direpresentasikan dalam tanda kutip tunggal (' ') atau tanda kutip ganda (" ").

Contoh:

Dalam Python: teks = "Ini adalah sebuah string."

Catatan Pemrograman

Pada saat mendeklarasikan variabel, ada bahasa pemrograman yang meminta programer untuk menentukan tipe data variabel tersebut secara statis. Bahasa pemrograman yang masuk kategori ini adalah C dan C++. Sedangkan bahasa pemrograman yang tidak meminta programer untuk menentukan tipe data variabel adalah Python, Java, PHP. Bahasa pemrograman jenis ini akan menentukan tipe data variabelnya secara dinamis ketika dieksekusi.

Ada kelebihan dan kekurangan antara kedua jenis penentuan tipe data statis dan dinamis tersebut. Bagi bahasa pemrograman yang menentukan secara statis, memiliki kelebihan eksekusi yang lebih cepat, namun kelemahannya adalah programer harus tepat dalam menentukan tipe data yang sesuai yang tentu saja membuat proses pemrograman menjadi lebih lama. Sebaliknya, bahasa pemrograman yang menentukan secara dinamis, memiliki kelebihan yaitu bisa lebih cepat diprogram, namun kelemahannya proses eksekusi tentu lebih lambat.

Kesalahan yang muncul apabila kita salah menentukan tipe data dengan nilai yang sebenarnya adalah sebagai berikut.

Python: ValueError: invalid literal for int() with base 10: '124,945.56'

Java: Exception in thread "main" java.util.InputMismatchException

C++: error: invalid conversion from 'const char*' to 'int'

Bekerja dengan uang? Jangan gunakan float atau double.

4.3 Latihan

1. Tentukanlah tipe data yang cocok untuk variabel berikut: tinggiBadan, jumlahKehadiran, rerataNilai, hargaJual, xxx

Bab 5 Variabel

Pada subbab xx kita sudah membahas mengenai data. Pada algoritma, data harus ditempatkan pada sebuah **penampung** atau **kontainer**. Penampung atau kontainer ini yang populer disebut sebagai variabel. Mengapa menggunakan istilah variabel? Karena sesuai definisinya, variabel bisa diubah-ubah nilainya. Mirip dengan waktu kita belajar matematika.

Misalnya: `x` adalah variabel tinggi badan. Pada algoritma, x disebut sebagai variabel. Artinya nilai atau isi x bisa berubah-ubah.

`x = 36` derajat celsius #matematika

`x = 45` derajat celsius #matematika

Pada algoritma:

```
x: integer
x ← 36
x ← 45
```

Kedua contoh di atas sam-sama menggunakan operator `=` (sama dengan). Namun pemaknaannya sangat berbeda. Pada matematika, tanda sama dengan (=) menunjukkan kesetaraan antara dua ekspresi atau nilai. Artinya X = 36 dibaca X setara atau sama dengan 36.

Sedangkan pada algoritma, X = 36 menunjukkan perintah atau instruksi bahwa nilai 36 diisi atau diassign ke variabel X.

[ ] => [36]

X => X

Selain diisi langsung dengan nilai, variabel juga bisa diisi secara tidak langsung melalui ekspresi tertentu. Misal:

```
x = 5 + y
```

artinya, isikan variabel X dengan hasil penjumlahan antara 5 ditambah variabel Y.

Beberapa notasi yang bisa digunakan untuk menunjukkan pengisian variabel, yaitu:

Sama dengan (=)

Panah kiri (←). Contoh: `x ← 5 + y`

Titik dua sama dengan (:=). Contoh: `x := 5 + y`

Tanda ← sering digunakan sebagai notasi algoritma seperti yang diadopsi oleh Rinaldi Munir di bukunya. Sedangkan := biasanya digunakan pada bahasa Pascal, dan tanda = adalah notasi yang paling banyak digunakan, baik untuk algoritma maupun di bahasa pemrograman.

5.1 Tukar (swap) nilai

Ketika kita membeli barang di toko, kita menukar uang kita dengan barang. Artinya uang kita pindah ke penjual, dan barang penjual berpindah ke kita. Analogi tersebut yang menjadi dasar kita untuk membahas konsep *swap* pada algoritma dan pemrograman.

Pada algoritma dan pemrograman, *swap* berarti menukar dua nilai pada dua variabel berbeda. Misalkan ada variabel bilanganX = 50 dan bilanganY = 77. Ketika kita hendak menukar keduanya, berarti kita ingin bilanganX = 77 dan bilanganY = 50.

jika hendak kita tukar, maka harus dilakukan bilanganX diisi bilanganY dan bilanganY diisi bilanganX. Apa benar hasilnya akan tertukar? Mari kita bahas:

```
1.  bilanganX = 50
2.  bilanganY = 77
3.  bilanganX = bilanganY    #bilanganX = 77 #bilanganY = 77
4.  bilanganY = bilanganX    #bilanganY = 77 #bilanganX = 77
```

ketika bilanganX diisi bilanganY, maka:

bilanganX = 77. Artinya, bilanganX yang sebelumnya bernilai 50, saat ini sudah ditimpa dan diisi 77.

bilanganX = 77

bilanganY = 77

Dengan demikian, tidak terjadi pertukaran nilai, malah terjadi penyalinan nilai sehingga bilanganX dan bilanganY bernilai sama. Pertanyaannya, mengapa bisa terjadi? Mari ingat kembali prinsip algoritma yaitu instruksi yang dieksekusi secara terurut atau *sequence*. Artinya, di baris ke-1 bilanganX bernilai 50, baris ke-2 bilanganY bernilai 50, dan di baris ke-3 bilanganX diisi bilanganY. Begitu pula di baris ke-4, karena bilanganX sudah bernilai 77, maka bilanganY ketika diisi bilanganX tetap bernilai 77, alias tidak ada perubahan! Ini sesuai dengan prinsip penulisan algoritma yang telah dibahas sebelumnya, yaitu nilai suatu variabel bisa ditimpa dengan nilai lain di baris kode berikutnya.

Jadi, bagaimana agar bisa melakukan *swap*? Jawabannya adalah kita memerlukan bantuan satu variabel bantuan, atau sering disebut variabel temporary dengan inisiasi nilai awalnya adalah null. Jadi, variabel temporary ini akan menyimpan salah satu variabel yang akan ditukar, sehingga ketika bilanganX diisi bilanganY, maka nilai awal bilanganX yaitu 50 tidak hilang atau tertimpa, karena sudah disimpan sementara di variabel temporary (baris ke-4). Selanjutnya kita dengan aman bisa mengisi bilanganX dengan bilanganY seperti sebelumnya yang untuk sementara mengakibatkan nilai bilanganX dan bilanganY adalah sama yaitu 77. Namun, seperti pada baris ke-6, giliran bilanganY diisi oleh temporary yaitu 50 yang menimpa 77. Sehingga bilanganY yang sebelumnya 77 berubah menjadi 50. Sehingga pada akhirnya penukaran kedua nilai tersebut bisa terwujud. Mari lihat:

1. temporary = null #isikan variabel temporary dengan nilai null
2. bilanganX = 50 #isi variabel bilanganX dengan nilai 50
3. bilanganY = 77 #isi variabel bilanganY dengan nilai 77

4. temporary = bilanganX #isikan variabel
5. bilanganX = bilanganY #bilanganX = 77 #bilanganY = 77
6. bilanganY = temporary #bilanganY = 50 #bilanganX = 77

Menukar nilai biasa dilakukan ketika hendak melakukan pengurutan data. Prinsip utama penukaran nilai adalah

5.2 Latihan

Bab 6 Operator

Prasyarat: Logika Proposisi

Pada algoritma, operator merupakan simbol atau tanda untuk melakukan operasi tertentu pada *operand*. Sebagaimana kita ketahui, *operand* adalah nilai atau variabel yang terlibat pada suatu operasi tertentu.

Terdapat empat jenis operator yang biasa digunakan pada algoritma yaitu operator aritmatika, logika, penugasan, dan perbandingan.

6.1 Aritmatika

Operator aritmatika terdiri dari:

Simbol	Operasi	Arti
+	penjumlahan	Menambahkan dua nilai
-	pengurangan	Mengurangkan satu nilai dari yang lain
*	perkalian	Mengalikan dua nilai
/	pembagian	Membagi satu nilai dengan nilai yang lain
%	modulus	Mengembalikan sisa dari pembagian dua nilai

6.2 Logika

Simbol	Operasi	Arti
&&	AND	Menguji apakah kedua kondisi benar
||	OR	Menguji apakah salah satu dari dua kondisi benar
!	NOT	Membalikkan nilai kebenaran suatu kondisi
^	XOR	Menguji apakah hanya salah satu dari dua kondisi yang benar

6.3 Penugasan

Simbol	Operasi	Arti	Contoh
←	Pengisian	Mengisikan nilai ke dalam variabel	number ← 56

+=	Penjumlahan dengan pengisian	Menambahkan nilai ke variabel	number ← 56 number += 3 // x ← x + 3 x ← 56 + 3 Output: 59
-=	Pengurangan dengan pengisian	Mengurangkan nilai dari variabel	number ← 56 number -= 3 // x ← x - 3 x ← 56 - 3 Output: 53
*=	Perkalian dengan pengisian	Mengalikan nilai dengan variabel	number ← 56 number *= 3 // x ← x * 3 x ← 56 * 3 Output: 168
/=	Pembagian dengan pengisian	Membagi nilai dengan variabel	number ← 56 number /= 3 // x ← x / 3 x ← 56 / 3 Output: 18.7

6.4 Perbandingan

Membandingkan kesamaan atau kesetaraan dua ekspresi atau nilai adalah menggunakan operator = yang dibaca sebagai ‘sama dengan’. Hasil perbandingan ini berupa boolean yaitu True atau False.

Misalnya: `ardi = Ardi`

Berarti kita hendak membandingkan *string* `'ardi'` apakah sama dengan *string* `'Ardi'`? Jawabannya adalah tidak atau False. Mengapa demikian? Karena keduanya berbeda pada huruf pertama, yaitu a huruf kecil dan A kapital.

Misalnya: **`Ardi = Ardi`** akan menghasilkan True.

Perbandingan juga bisa melibatkan variabel dengan variabel, variabel dengan nilai, dan nilai dengan variabel.

Contoh variabel dengan variabel

bilanganX = bilanganY

Misalnya: bilanganX ← 45 dan bilanganY ← 32, maka hasilnya adalah False

Sedangkan misalnya bilanganX dan bilanganY sama-sama bernilai 56, maka hasilnya adalah True

Contoh variabel dengan nilai

bilanganX = 56

Apabila misalnya **`bilanganX ← 55`**, maka hasilnya adalah false. Sedangkan apabila bilanganX ← 56, maka hasilnya adalah true.

Contoh perbandingan nilai dengan variabel

56 = bilanganY

Jika `bilanganY ← 44`, maka hasil perbandingannya adalah False (56 = 44) dan jika bilanganY ← 56, maka hasilnya adalah `true`, karena `56 = 56`.

Ingat!

Pada algoritma, notasi sama dengan adalah berupa simbol `=`. Sedangkan pada bahasa pemrograman, sebagian besar menggunakan simbol `==`.

Untuk pengisian atau *assign* nilai ke variabel menggunakan simbol ←. Sedangkan pada bahasa pemrograman, sebagian besar menggunakan simbol = (C/C++, Java, Kotlin, Javascript, dsb) dan := pada Pascal.

Simbol	Operasi	Arti	Contoh
==	Sama dengan	Memeriksa apakah dua nilai sama	If (x == y) then blok eksekusi
!=	Tidak sama dengan	Memerika apakah dua nilai tidak sama	If (x != y) then blok eksekusi
<	Kurang dari	Memerika apakah nilai pertama lebih kecil dari nilai kedua	If (x < y) then blok eksekusi
>	Lebih dari	Memerika apakah nilai pertama lebih besar dari nilai kedua	If (x > y) then blok eksekusi
<=	Kurang dari atau sama dengan	Memerika apakah nilai pertama lebih kecil atau sama dengan nilai kedua	If (x <= y) then blok eksekusi
>=	Lebih dari atau sama dengan	Memerika apakah nilai pertama lebih besar atau sama dengan nilai kedua	If (x >= y) then blok eksekusi

Bab 7 Statement

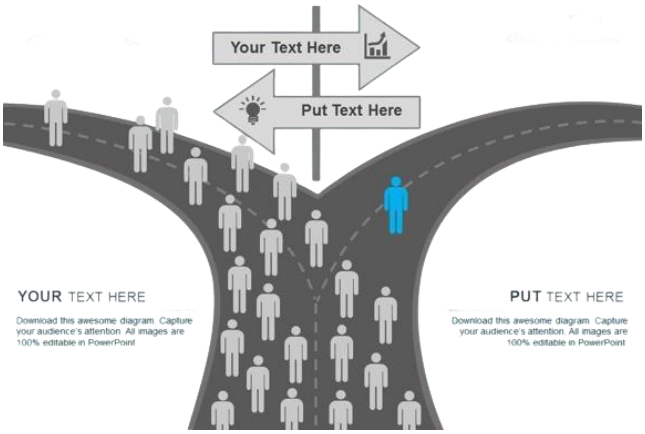
Bab 8 Ekspresi

Bab 9 Blok dan indentasi

Bab 10 Efektivitas dan Efisiensi Algoritma

Bab 11 Seleksi kondisi (IF-ELSE)

Pernah mengikuti seleksi bukan? Panitia akan menyeleksi berkas-berkas pendaftaran yang masuk sesuai dengan kriteri. Misalnya, untuk nilai SBMPTN 500 bisa masuk Prodi Informatika. Sedangkan yang nilainya 550 bisa masuk Kedokteran.



Begitu pula dalam algoritma. Kita bisa membuat sebuah statement seleksi terhadap data yang memenuhi kondisi tertentu. Sebagai contoh, ketika kita belanja barang sejumlah tertentu maka akan diberikan potongan harg. Ketika kita meletakkan barang tersebut dan ternyata memenuhi kondisi untuk mendapatkan potongan harga, maka otomasi *software* kasir di toko tersebut akan menghitung jumlah potongannya.

11.1 Format umum seleksi kondisi IF

Jika (pernyataan kondisi) maka lakukan hal berikut

Blok baris perintah

Contoh:

```
Jika (pembeli transaksi di hari sabtu) maka lakukan hal berikut
       diskon = 10%
```

Notasi algoritmanya menjadi

if (pernyataan kondisi) then

blok baris perintah

berdasarkan contoh di atas, maka notasi algoritma seleksi kondisi menjadi:

```
if (dayOfTransaction = 'Saturday') then
```

```
   discount = 0.1
```
Sebagaimana prinsip seleksi, ada prasyarat kondisi yang harus terpenuhi. Artinya, blok baris perintah di dalam IF haruslah terpenuhi terlebih dahulu. Jika tidak, maka blok baris perintah tidak akan dijalankan. Berarti, pada statement pernyataan kondisi ada penilaian benar/salah, ya/tidak, atau true/false.

Misalnya hari itu pembeli transaksi (`dayOfTransaction`) di hari Sabtu, maka bisa ditulis:

```
if ('Saturday' = 'Saturday') then
   discount = 0.1
```

Kita tahu bahwa string `'Saturday' = 'Saturday'`, maka pernyataan tersebut bernilai True. Artinya baris di dalamnya akan dieksekusi, atau si pembeli berhak mendapatkan diskon 0.1.

Sebaliknya bila `dayOfTransaction` ternyata selain hari Sabtu, misalnya Selasa, maka berarti:

```
if ('Tuesday' = 'Saturday') then
   discount = 0.1
```

Kita tahu bahwa string `'Saturday'` adalah tidak sama dengan `'Saturday'`, maka berarti pernyataan tersebut bernilai `False`. Artinya, si pembeli tidak memperoleh diskon 0.1. Dengan bahasa lain, blok baris perintah di dalam IF tersebut tidak dieksekusi atau dilompati untuk beralih ke baris perintah selanjutnya/di bawahnya. Perhatikan ilustrasi berikut:

```
1. baris sebelumnya

2. baris sebelumnya

3. if ('Tuesday' = 'Saturday') then
4.   discount = 0.1

5. baris selanjutnya

6. baris selanjutnya
```

Jika True, maka jalur eksekusi adalah: 1-2-3-4-5-6. Sebaliknya jika False, maka jalur eksekusinya adalah: 1-2-3-5-6.

Bentuk lain untuk menggambarkannya adalah menggunakan diagram alir (*flow chart*).

[diagram alir]

Kasus lain adalah memeriksa apakah sebuah array kosong atau tidak.

1. if (numOfData = 0) then
2. statement if bernilai true
3. statement

Array memang belum dibahas hingga bab ini. Untuk memahami array silakan membaca bab xx. Setelah paham mengenai array, maka Anda bisa memahami contoh menghitung jumlah elemen atau ukuran array di listing xx, xx, xx, dan xx. Pada akhirnya barulah Anda bisa memahami bagaimana numOfdata diperoleh pada contoh di atas.

11.2 Format umum seleksi kondisi IF-ELSE

Seleksi IF tunggal digunakan ketika kita hanya memiliki satu syarat kondisi. Pada situasi lain, kita bisa saja memiliki dua syarat kondisi, misalnya kalau tidak yang ini ya pasti yang itu.

Misalnya suatu bilangan, kalau tidak ganjil ya genap. Apakah ini cocok jika diterapkan menggunakan seleksi if tunggal?

1. Baris sebelumnya
2. Jika ganjil
3. Blok baris ganjil
4. Baris genap
5. Baris biasa

Jika tidak ganjil, berarti urutannya: 1-2-4-5. Sekilas benar. Namun, bila kondisi ganjil terpenuhi, maka urutannya: 1-2-3-4-5. Artinya baik ganjil dan genap dieksekusi semua. Oleh karena itulah kita perlu menggunakan **if-else**.

1. Statement
2. If ganjil
3. Statement-statement jika kondisi ganjil
4. Else
5. Statement-statement jika kondisi genap
6. Statement

Jika dieksekusi, maka ada dua kemungkinan jalur urutannya yaitu: 1-2-3-6 atau 1-2-4-5-6

Perhatikan kembali Algoritma Count Median di halaman xx.

```
1.  if (N MOD 2) = 1 then
2.  |   middleIndex ← (numOfBodyHeights – 1) / 2
3.  |   median ← bodyHeights[middleIndex]
4.  else
5.  |   middleIndex ← N / 2
6.  |   middleValue1 ← bodyHeights[middleIndex-1]
7.  |   middleValue2 ← bodyHeights[middleIndex]
8.  |   median ← (middleValue1 + middleValue2) / 2
9.  end if
```

Dua kemungkinan jalur urutannya yaitu: 1-2-3-9 atau 1-4-5-6-7-8-9

Contoh kasus.

Buatlah algoritma untuk menentukan nilai yang diperoleh lulus atau tidak lulus. Nilai bertipe real dan diinputkan lewat keyboard. Nilai ditetapkan lulus jika lebih dari atau sama dengan 65. Sedangkan nilai yang kurang dari 65 ditetapkan tidak lulus.

```
Kamus:
nilai   : real

Constants:
batasNilai ← 65
```

```
    1.  input nilai
    2.
    3.  if (nilai >= batasNilai) then
    4.     output "Anda Lulus"
    5.
    6.  else
    7.     output "Anda tidak lulus"
```

11.3 Bentuk umum seleksi kondisi `if-else if-else`

Selain satu dan dua kondisi seleksi, terkadang juga kita akan menghadapi situasi yang menuntut dua atau lebih kondisi seleksi. Misalnya pada aplikasi permainan (*game*) pertempuran. Poin pemain selalu bertambah setiap kali berhasil membunuh musuh. Di akhir *stage* atau *level*, pemain akan memperoleh bonus berdasarkan poin yang didapatkannya.

```
Kamus:
point   : integer
bonus   : string
    1.  input point
    2.
    3.  if (point >= 80) then
    4.     bonus ← "Senjata Laser"
    5.
    6.  else if (point >= 60) then
    7.     bonus ← "Armor Cambuk"
    8.
    9.  else if (point >= 40) then
    10.    bonus ← "Senjata Api Besar"
    11.
    12. else if (point >= 25) then
    13.    bonus ← "Senjata Api Kecil"
    14.
    15. else
    16.    bonus ← "You got no bonus!"
```

```
Eksekusi
    1.  point ← 80
    2.
    3.  if (80 >= 80) then
    4.     bonus ← "Senjata Laser"
Urutan: 1-2-3-4
```

```
Eksekusi
    1.  point ← 15
    2.
    3.  if (15 >= 80)         #False
    6.   else if (15 >=60)    #False
    9.   else if (15 >=40)    #False
    12. else if (15 >= 25)    #False
    15. else
    16.    bonus ← "No bonus!"
```

```
Urutan: 1-2-3-6-8-9-11-12-14-15-16
```

11.4 If bersarang (*nested If*)

IF bersarang menunjukkan bahwa di dalam blok if mengandung if lain. Perhatikan algoritma berikut:

```
1.  input suhuMotor, kecepatanRotasi
2.
3.  if (suhuMotor > 80) then
4.  |   cetak "Peringatan: Suhu motor tinggi, matikan robot untuk pendinginan"
5.  else
6.  |   if (kecepatanRotasi > 1000) then
7.  |   |    cetak "Robot beroperasi pada kecepatan tinggi"
8.  |   else
9.  |   |   cetak "Robot beroperasi pada kecepatan standar"
10. |   end if
11. end if
```

11.5 Latihan

1. Tulislah langkah-langkah eksekusinya, bila diberikan input berupa Diberikan algoritma berikut

```
Kamus:
dayOfTransaction      : string
discount              : real
```

```
1.  dayOfTransaction ← 'Thursday'
2.  discount ← 0.0
3.
4.  if (dayOfTransaction = 'Saturday') then
5.     discount ← 0.1
6.
7.  cetak discount
```

2. Panitia seleksi beasiswa akan menyeleksi mahasiswa berdasarkan: IPK minimal 3.3 dan minimal semester 3. PK dan semester diinputkan lewat keyboard. Outputnya ada dua kemungkinan: "Lolos beasiswa" atau "Tidak lolos beasiswa". Buatlah algoritmanya.
3. Kasus: Penentuan Kelulusan Siswa
   Sebuah sekolah memiliki kriteria untuk menentukan apakah seorang siswa akan lulus atau tidak. Kriteria tersebut melibatkan nilai dari beberapa mata pelajaran, absensi, dan status keikutsertaan dalam kegiatan ekstrakurikuler.
   Berikut adalah detail kriteria kelulusan:
   1. Nilai minimal 60 untuk setiap mata pelajaran.
   2. Kehadiran minimal 75% dari total hari sekolah.
   3. Siswa harus mengikuti setidaknya satu kegiatan ekstrakurikuler.

Contoh Input:
"Masukkan nilai mapel: " 70
"Masukkan jumlah pertemuan : " 16
"Masukkan jumlah kehadiran siswa : " 12
"Masukkan jumlah kegiatan kegiatan ekstra: " 1

Output: Tidak lolos

11.6 Catatan Bibliografi

Pada kondisi tertentu, banyak yang menyarankan untuk mengindari penggunaan ELSE.

Selain itu, pada dunia kerja kita akan berhadapan dengan If else yang sangat panjang atau nested if (berikan contoh code dan gambar nested if street fighter). Ada beberapa cara mengatasinya salah satunya menggunaka *guard clause*.

Pada pengujian if-else, bentuk flow graphnya menjadi xx

Bab 12 Perulangan (Loop)

12.1 Pengantar Loop

Loop adalah struktur kontrol yang digunakan untuk mengulang serangkaian instruksi/statement atau blok kode dalam program.

Coba tuliskan algoritma untuk membuat gambar segitiga seperti berikut.

```
    *
   ***
  *****
 *******
*********
```

1. cetak ' * '
2. cetak ' *** '
3. cetak ' ***** '
4. cetak ' ******* '
5. cetak '*********'

Misalnya Anda diminta membuat segitiga dengan tinggi seratus karakter, apakah bisa membuatnya? Jawabnya tentu bisa, tapi akan bertambah lama. Nah, inilah salah satu kelebihan komputer. Ia bisa melakukan aktivitas yang repetitif tanpa henti tanpa merasa lelah. Manusia, ketika diminta menulis sampai seratus karakter, ketika masuk ke dua atau tiga puluh karakter akan mulai lelah dan tidak lagi berkonsentrasi. Sebaliknya, jika kita serahkan ke komputer, ia akan melakukannya terus menerus sampai batas berhenti yang ditetapkan.

Pengerjaan suatu aktivitas yang sama secara terus menerus atau repetitif ini disebut sebagai perulangan atau loop. Loop adalah salah satu konsep penting dalam algoritma dan pemrograman. Tanpa adanya loop, komputer akan kehilangan daya magisnya.

Mari kembali ke gambar segitiga tersebut. Segitiga tersebut memiliki panjang alas sebanyak sembilan asterik (*). Kita tahu bahwa kita akan mencetak tanda * sebanyak sembilan kali. Karena yang hendak dicetak adalah sama yaitu *, maka kita bisa perintahkan komputer untuk mencetak * sebanyak sembilan kali. Artinya, kita akan mencetak * berulang-ulang hingga ia tercetak sembilan kali.

For dan While

Ada dua konsep perulangan yang dikenal dalam algoritma dan pemrograman, yaitu for dan while. Perulangan for cocok digunakan jika kita sudah tahu persis kapan perulangannya berhenti. Mencetak * sembilan kali menunjukkan bahwa kita tahu bahwa cetak * berhenti ketika mencapai sembilan. Sehingga perulangan for cocok untuk kasus ini.

12.2 Perulangan For

Struktur perulangan for adalah seperti berikut:

```
for (start, stop, step) do
   # blok kode program
```

start adalah nilai awal loop, stop adalah nilai akhir loop, dan step adalah pencacah (*counter*) agar loop bisa bergerak maju atau mundur. Baik start, stop dan step bisa berupa ekspresi atau pernyataan.

Berdasarkan kasus cetak segitiga tersebut, maka:

```
1.  for (i ← 0, i < 9, i ← i+1) do
2.     cetak '*' tanpa baris baru
```

Perhatikanlah, ada tiga hal yang perlu Anda pahami pada perulangan for tersebut:

1. i adalah bilangan atau nilai yang menjadi acuan dalam perulangan
2. 9 adalah batas akhir suatu perulangan. Artinya selama nilai i masih dalam rentang batas akhir, maka eksekusi dijalankan terus. Artinya, pada bagian ini akan ada penilaian True atau False. Perulangan akan berhenti jika i < 9 adalah False atau i bernilai lebih dari 9.
3. i ← i+1 adalah step atau langkah yang menggerakkan perulangan bisa maju atau mundur. Dengan kata lain, bagian ini menentukan nilai i selenjutnya. Tanda +1, menunjukkan bahwa langkah yang diberikan adalah naik satu per satu. Tentu saja nanti Anda bisa menggunakan +2 dan seterusnya yang menunjukkan naik tiap dua.
4. cetak '*' adalah konten atau isi yang hendak dieksekusi dalam perulangan.

Jadi, pastikan Anda bisa membedakan antara i, 9, dan cetak *

Mari kita coba eksekusi

```
1. for (i ← 0, 0 < 9, 1 ← 0+1) do
2.    cetak *
1. for (i ← 1, 1 < 9, 2 ← 1+1) do
2.    cetak **
1. for (i ← 2, 2 < 9, 3 ← 2+1) do
2.    cetak ***
1. for (i ← 3, 3 < 9, 4 ← 3+1) do
2.    cetak ****
1. for (i ← 4, 4 < 9, 4 ← 4+1) do
2.    cetak *****
1. for (i ← 5, 5 < 9, 6 ← 5+1) do
2.    cetak ******
1. for (i ← 6, 6 < 9, 7 ← 6+1) do
2.    cetak *******
1. for (i ← 7, 7 < 9, 8 ← 7+1) do
2.    cetak ********
1. for (i ← 8, 8 < 9, 9 ← 8+1) do
2.    cetak *********
1. for (i ← 9, 9 < 9, 9 ← 8+1) Stop
```

Mungkin ada yang bertanya, kenapa i dimulai dengan 0? Tidak dengan 1? Atau mengapa i < 9, kok bukan i ≤ 9?

Mari kita lihat jika eksekusinya dimulai ketika i adalah 0.

```
1. for (i ← 1, 1 < 9, 2 ← 1+1) do
2.    cetak '*' tanpa baris baru
1. for (i ← 2, 2 < 9, 3 ← 2+1) do
2.    cetak ** tanpa baris baru
1. for (i ← 3, 3 < 9, 4 ← 3+1) do
2.    cetak *** tanpa baris baru
1. for (i ← 4, 4 < 9, 5 ← 4+1) do
2.    cetak **** tanpa baris baru
1. for (i ← 5, 5 < 9, 6 ← 5+1) do
2.    cetak *****
1. for (i ← 6, 6 < 9, 7 ← 6+1) do
2.    cetak ******
1. for (i ← 7, 7 < 9, 8 ← 7+1) do
2.    cetak *******
1. for (i ← 8, 8 < 9, 9 ← 8+1) do
2.    cetak ********
1. for (i ← 9, 9 < 9, 10 ← 9+1) Stop
```

Jika dimulai dari 1, maka asterik hanya dicetak hingga 8 kali saja. Karena kondisi berhenti ditetapkan i < 9. Agar tetap bisa mencetak asterik sebanyak 9 kali, maka operator untuk kondisi berhenti diubah menjadi i ≤ 9, sehingga

```
1. for (i ← 1, 1 ≤ 9, 2 ← 1+1) do
2.    cetak *
1. for (i ← 2, 2 ≤ 9, 3 ← 2+1) do
2.    cetak **
1. for (i ← 3, 3 ≤ 9, 4 ← 3+1) do
2.    cetak ***
1. for (i ← 4, 4 ≤ 9, 5 ← 4+1) do
2.    cetak ****
1. for (i ← 5, 5 ≤ 9, 6 ← 5+1) do
2.    cetak *****
1. for (i ← 6, 6 ≤ 9, 7 ← 6+1) do
2.    cetak ******
1. for (i ← 7, 7 ≤ 9, 8 ← 7+1) do
2.    cetak *******
1. for (i ← 8, 8 ≤ 9, 9 ← 8+1) do
2.    cetak ********
1. for (i ← 9, 9 ≤ 9, 10 ← 9+1) do
2.    cetak *********
1. for (i ← 10, 10 ≤ 9, 11 ← 10+1) stop karena 10 ≤ 9 adalah False
```

Dengan demikian, kita bisa simpulkan bahwa jika mau mulai dari 0, maka berhentilah dengan menggunakan operator <. Namun, jika mau mulai dari 1, maka berhentilah dengan menggunakan ≤. Hal ini juga menunjukkan bahwa kita bisa saja tidak memulai dari 0, tapi bisa juga dari 1 dan 2.

Contoh awal inkremen dimulai dari 2.

for (i ← 2, i < 5, i←i+1)

(berikan contoh start + 1, stop I * I <= N, hingga i+2)

Perulangan for di atas adalah contoh dengan inkremental/penambahan step satu per satu alias kelipatan satu. Jika satu per satu bisa, apakah bisa sekiranya tidak satu per satu, misalnya tiap dua, tiga langkah? Jawabannya tentu bisa. Jumlah langkah atau inkremental bisa kita tentukan sesuai kebutuhan. Berikut beberapa contoh bentuk inkremental pada perulangan for:

for (i ← 1, i < 20, i←i+2) #inkremen naik kelipatan 2

for (i ← 0, i < 20, i←i+3) #inkremen naik kelipatan 3

Note!

Inkremental step pada umumnya adalah bilangan integer. Sangat jarang sekali bertipe real atau desimal.

Selain perulangan yang bersifat inkremental atau bertambah, kita juga bisa menerapkan perulangan for yang bersifat dekremental atau berkurang. Artinya, nilai awal step menjadi yang tertinggi, batas perulangan menjadi yang terendah, dan dekrementalnya dikurangi sesuai step atau kelipatannya.

Contoh:

for (i ← 10, i < 1, i←i-1)

  print i

for (i ← 10, 10 < 1, 9←10-1)

  print 10

for (i ← 9, 9 < 1, 8←9-1)

  print 9

for (i ← 8, 8 < 1, 7←8-1)

  print 8

for (i ← 7, 7 < 1, 6←7-1)

  print 7

for (i ← 6, 6 < 1, 5←6-1)

  print 6

for (i ← 5, 5 < 1, 4←5-1)

print 5

for (i ← 4, 4 < 1, 3←4-1)

print 4

for (i ← 3, 3 < 1, 2←3-1)

print 3

for (i ← 2, 2 < 1, 1←2-1)

print 2

for (i ← 1, 1 < 1, 0←1-1)

print 1

for (i ← 0, 0 < 1, -1←0-1) Stop

artinya, perulangan dimulai dengan counter 10, sampai counter bernilai < 1, dan berkurang tiap kelipatan satu.

Ingat!

Konsep penting perulangan for adalah bedakan antara counter pada loop dengan blok statement di dalamnya. Blok statement di dalam perulangan bisa berbentuk operasi aritmatika, ada seleksi kondisional, bahkan bisa juga ada perulangan lagi di dalamnya. Bahkan hanya sekedar mencetak counter saja.

Berikut contoh perulangan for yang paling sederhana untuk mencetak variabel counter.

```
1.  for (i ← 0, i < 9, i ← i+1) do
2.     cetak i
```

Agak berkembang sedikit, berikut contoh perulangan for yang di dalamnya memiliki kondisional if.

```
1.  for (i ← 0, i < 9, i ← i+1) do
2.  |   if (statement kondisi) then
3.  |   |  Statement dalam for i dan if
4.  |   |  Statement dalam for i dan if
5.  |   Statement dalam for
6.  |   Statement dalam for
```

Bisa juga di dalam perulangan for memiliki perulangan lain atau nested for (for bersarang) dan di dalamnya ada kondisional If seperti contoh berikut

```
1.  for (i ← 0, i < 9, i ← i+1) do
2.  |   statement dalam for i
3.  |   for (j ←, j < 9, j ←j+1) do
4.  |   |  statement dalam for i,j
5.  |   |  if (statement kondisi) then
```

```
6.  |   |  |      Statement dalam for i,j dan if
7.  |   |  |      Statement dalam for i,j dan if
8.  |   Statement dalam for i
9.  |   Statement dalam for i
```

Bisa juga kondisi sebaliknya dari contoh di atas yaitu:

```
1.  for (i ← 0, i < 9, i ← i+1) do
2.  |   if (statement kondisi) then
3.  |   |   statement dalam for i dan if
4.  |   |   statement dalam for i dan if
5.  |   |
6.  |   |   for (j ←, j < 9, j ←j+1) do
7.  |   |      statement dalam for i, if, dan j
8.  |   |      statement dalam for i, if, dan j
```

Pada contoh di atas sempat disinggung tentang nested for. Kondisi ini bakal banyak Anda temui dan terapkan kelak. Oleh karena itu akan kita bahas khusus di subbab xx berikut.

12.2.1 Perulangan for bersarang (nested for)

```
1.  for (i ← 0, i < 6, i ← i+1) do
2.     for (j ← 0, j < 3, j ← j+1) do
3.        cetak i j
```

```
1. for (i ← 0, 0 < 6, 1 ← 0+1) do
2.    for (j ← 0, 0 < 3, 1 ← 0+1) do
3.       cetak 0 0                          # 0 0
2.    for (j ← 1, 1 < 3, 2 ← 1+1) do
3.       cetak 0 1                          # 0 1
2.    for (j ← 2, 2 < 3, 3 ← 2+1) do
3.       cetak 0 2                          # 0 2
2.    for (j ← 3, 3 < 3, 4 ← 3+1) Stop
```

```
1. for (i ← 1, 1 < 6, 2 ← 1+1) do
2.    for (j ← 0, 0 < 3, 1 ← 0+1) do
3.       cetak 1 0                          # 1 0
2.    for (j ← 1, 1 < 3, 2 ← 1+1) do
3.       cetak 1 1                          # 1 1
2.    for (j ← 2, 2 < 3, 3 ← 2+1) do
3.       cetak 1 2                          # 1 2
2.    for (j ← 3, 3 < 3, 4 ← 3+1) stop
```

```
1. for (i ← 2, 2 < 6, 3 ← 2+1) do
2.    for (j ← 0, 0 < 3, 1 ← 0+1) do
3.       cetak 2 0                          # 2 0
2.    for (j ← 1, 1 < 3, 2 ← 1+1) do
3.       cetak 2 1                          # 2 1
2.    for (j ← 2, 2 < 3, 3 ← 2+1) do
3.       cetak 2 2                          # 2 2
```

```
2.    for (j ← 3, 3 < 3, 4 ← 3+1) stop
```

```
1. for (i ← 3, 3 < 6, 4 ← 3+1) do
2.    for (j ← 0, 0 < 3, 1 ← 0+1) do
3.       cetak 3 0                          # 3 0
2.    for (j ← 1, 1 < 3, 2 ← 1+1) do
3.       cetak 3 1                          # 3 1
2.    for (j ← 2, 2 < 3, 3 ← 2+1) do
3.       cetak 3 2                          # 3 2
2.    for (j ← 3, 3 < 3, 4 ← 3+1) stop
```

Perhatikanlah, kedua perulangan for menggunakan variabel pencacah yang berbeda yaitu i dan j. Mengapa berbeda? Mengapa demikian? Jawabannya karena jika sama, maka i pertama akan ditimpa dengan i pada for kedua. Perhatikanlah contoh berikut:

```
1.  for (i ← 0, i < 4, i ← i+1) do
2.     for (i ← 0, i < 3, i ← i+1) do
3.        cetak i i
```

```
1. for (i ← 0, 0 < 4, 1 ← 0+1) do
2.    for (i ← 0, 0 < 3, 1 ← 0+1) do
3.       cetak 0 0                          # 0 0
2.    for (i ← 1, 1 < 3, 2 ← 1+1) do
3.       cetak 1 1                          # 1 1
2.    for (j ← 2, 2 < 3, 3 ← 2+1) do
3.       cetak 2 2                          # 2 2
2.    for (j ← 3, 3 < 3, 4 ← 3+1) stop
```

Jadi, itulah alasan kenapa tiap for sebaiknya menggunakan variabel pencacah yang berbeda. Karena apabila sama maka nilai variabel pencacah sebelumnya akan ditimpa oleh nilai variabel terbawah atau terdalam. Baca kembali konsep runtunan pada variabel dan penimpaan variabel di subbab xx.

Kapan sebaiknya menggunakan perulangan for? Jika melihat penjabaran di atas, maka perulangan for cocok digunakan ketika kita tahu persis berapa kali iterasi diperlukan atau nilai batas akhir iterasi. Dengan kata lain, perulangan for yang bergantung dengan pencacah atau counter. Jadi batasIterasi yang bertipe integer tersebut memiliki peran sentral dalam perulangan for.

12.3 Sejarah Loop

12.4 Perulangan While

Ada saatnya, kita tidak bisa menentukan batas iterasi secara pasti dan spesifik. Kita hanya tahu kondisi tertentu yang terpenuhi. Jika ini yang kita alami, maka perulangan for sudah tidak memadai atau tidak cocok. Kita bisa menggunakan perulangan while.

while (kondisi) do

blok kode

Perulangan berdasarkan perhitungan. Artinya selama perhitungannya masih dalam kondisi benar (True), maka blok kode dalam while akan terus dieksekusi. Contohnya:

```
Count ← 0
While (count < 6) do
        cetak 'iterasi ke-', count
        count ← count + 1
```

Perulangan berdasarkan input pengguna. Artinya, selama input yang dimasukkan pengguna adalah sama, maka perulangan while terus dijalankan hingga inputan yang diberikan berbeda. Contohnya:

```
inputUser ← input("Lanjutkan? (ya/tidak)")
while (lower(inputUser) = "ya") do
        inputUser ← input("Lanjutkan? (ya/tidak)")
cetak 'Selesai'
```

Perulangan berdasarkan kondisi kompleks. Artinya, kondisi yang harus terpenuhi memiliki operator logika. Contohnya:

```
targetNumber ← 7
guessNumber ← 0
attempts ← 0
maxAttempts ← 3

while (guessNumber != targetNumber and attemps < maxAttempts) do
        guess ← int(input("Tebak angka: "))
        Attempts ← attempts + 1

        If (guessNumber < targetNumber) then
                cetak 'Tebakan terlalu kecil. Coba lagi'
        Else if (guessNumber > targetNumber) then
                cetak 'Tebakan terlalu besar. Coba lagi'

If guessNumber = targetNumber then
        Cetak 'Selamat! Tebakan kamu benar pada percobaan ke-', attempts
Else
        Cetak 'Maaf, kuota mencoba telah habis. Angka yang benar adalah ', targetNumber
```

Perulangan berdasarkan status objek atau variabel. Misalnya untuk menghindari bilangan acak atau primary key ganda atau bernilai sama, maka contohnya:

```
generatedKey ← random(1,10)
existingKey ← 4
while generatedKey = existingKey
        generatedKey ← random(1,10)
```

5. cetak 'generatedKey'

12.5 Break-Continue

Tips langkah-langkah memilih apakah for atau while

1. Apakah bilangan batas berhenti diketahui secara pasti ? Misalnya 4 kali, 10, kali 100 kali.
2. Jika Ya, gunakan for
3. Jika tidak, gunakan while.

12.6 Latihan

1. Diberikan input X dan Y. Hitunglah nilai dari X setelah dipangkatkan dengan Y. Contoh input:
   X = 4, Y = 3
   Output: 4 x 4 x 4 = 256
2. Diberikan input 50000. Hitunglah secara mundur tiap 10000. Di akhir munculkan 5000. Contoh output:
   50000
   40000
   30000
   20000
   10000
   5000
3.

Bab 13 Array 1-Dimensi

13.1 Pengertian Array

13.2 Sejarah Array

13.3 Konsep Dasar Array

Pada bab xx kita telah belajar tentang variabel yang merupakan penampung suatu nilai. Misalnya kita membuat lima variabel bertipe integer, maka kita deklarasikan seperti berikut:

value1 ← 12

value2 ← 9

value3 ← 10

value4 ← 4

value5 ← 90

Berbekal kelima variabel tersebut, kita b

isa melakukan bermacam operasi, seperti cetak nilai variabel, aritmatika, dan sebagainya. Misalnya kita hendak mencetak kelima variabel tersebut, berarti kita akan menuliskan:

cetak value1

cetak value2

cetak value3

cetak value4

cetak value5

Jika dieksekusi, yang tampil ke layar adalah 12, 9, 10, 4, 90. Memang sekilas tidak ada yang janggal dan salah. Namun apakah Anda bisa mengkritisi teknik yang digunakan pada algoritma tersebut? Ya, bila sebelumnya Anda sudah paham dengan baik dengan pembahasan materi perulangan di bab xx sebelumnya, maka kita bisa lihat bahwa terjadi ketidakefektifan pada teknik di atas yaitu membuat statement yang sama berulang kali. Itu saja baru lima variabel. Bagaimana jika ada 10, 20 bahkan ratusan? Tentu semakin merepotkan.

Untuk permasalahan penyimpanan banyak nilai bertipe sama, alih-alih menggunakan satu variabel tiap satu nilai, maka ada salah satu teknik yang lebih tepat kita gunakan yaitu *array*. Seperti arti katanya yang harfiah, *array* merupakan suatu wadah untuk menampung nilai-nilai bertipe sama yang disusun secara berjejer berurutan.

Pernah lihat loker di sekolah Anda? Nah, seperti itulah analogi yang paling mendekati bentuk *array*. Loker tersusun dari wadah-wadah yang saling menempel berdekatan dan digunakan untuk menyimpan barang. Biasanya tiap wadah, memiliki nomor urut untuk menandai pemiliknya. Sesuai fungsinya, tiap wadah akan memiliki isi atau barang yang berbeda-beda pula.

Nah, kita bisa samakan bahwa *array* adalah loker itu sendiri. Tiap wadah dalam loker itu disebut sel pada array. Nomor urut yang ditempel pada tiap wadah di loker, di *array* disebut sebagai indeks. Sedangkan isi dalam wadah, itu sama saja nilai, value atau elemen jika di array. Persamaan analogi antara loker dan array diperlihatkan oleh Gambar xx.

[gambar loker, wadah, nomor urut, isi]

[gambar array, sel, indeks, nilai]

Jika mau ditambahkan, analogi lain untuk array misalnya kompleks perumahan. Satu deret atau blok rumah disebut sebagai array. Tiap rumah disebut sel. Tiap rumah pasti memiliki nomor rumah misalnya B1, B2, B3 dan seterusnya yang sama saja dengan indeks di array. Tiap rumah memiliki isi dan perabot rumah tangga yang disebut nilai atau elemen jika di array. Tabel xx menunjukkan secara lengkap analogi persamaan antara array, loker sekolah, dan perumahan.

Tabel xx. Persamaan analogi array dengan dunia nyata

Array	Loker Sekolah	Perumahan
array	Loker	Satu deret atau blok rumah
Sel	Wadah	Satu rumah
Indeks	Nomor urut	Nomor rumah
Nilai atau elemen	Isi	Perabot rumah

13.4 Karakteristik array

Seperti halnya loker dan perumahan, di dalam algoritma kita harus membuat atau mendeklarasikan array terlebih dahulu sebelum digunakan. Untuk mendeklarasikan array kita harus isikan atau assign ke sebuah variabel. Namun sebelumnya kita harus kenal dulu dengan simbol sebuah variabel yang berbentuk array yaitu [].

Deklarasi array

Terdiri dari nama variabel (variabel), ukuran atau panjang array (size), dan tipe data.

```
<variabel> : array[1..<size>] of <tipe data>
```

Contoh:

```
N ← 5   #ukuran array
values : array[1..N] of integer
```

Artinya kita mendeklarasikan array ke variabel bernama values yang bertipe integer/real/char dengan ukuran 5. Ukuran array 5 itu berarti sama saja array tersebut memiliki 5 sel, atau 5 wadah loker atau 5 rumah jika di perumahan.

Jadi, dengan array kita tidak perlu lagi menulis satu variabel untuk satu nilai seperti yang dijabarkan awal subbab ini, namun cukup menampungnya ke dalam satu array. Jadi, kita bisa buatkan seperti berikut.

values ← [12, 9, 10, 4, 90]

Pengisian array bisa dilakukan melalui inputan user, maupun langsung. Jika melalui input *user* maka kita akan menggunakan perulangan for atau while. Kita akan menggunakan for bila kita sudah tahu persis ukuran *array*. Sebaliknya kita akan menggunakan while bila kita tidak terlalu tahu ukurannya.

array	[13	5	9	11	20]
indeks	0	1	2	3	4

Implementasi

Antar bahasa pemrograman memiliki cara yang berbeda untuk menghitung ukuran *array*. Python dan Go `len(myArray)`, Javascript, Ruby dan Java `myArray.length`, C++ `myArray.size()`, C# `myArray.Length`, PHP `count($myArray)`, Rust `myArray.len()`, dan Swift `myArray.count`.

13.5 Insert Elemen

Insert, penambahan di array berarti memasukkan atau meletakkan suatu bilangan, elemen atau nilai pada sel-sel array sesuai indeksnya.

Elemen bisa disimpan ke array selama masih tersedia ruang yang dialokasikan. Misalnya kita mendeklarasikan array untuk menampung 5 elemen (array[5]), apabila sudah terisi sebanyak tiga elemen, maka kita masih bisa mengisi dua elemen lagi. Namun apabila array sudah terisi penuh sebanyak lima elemen, maka tentu kita tidak bisa menambah elemen baru lagi.

Ada tiga cara menyimpan nilai ke array

1. Langsung diinisialisasi/diisi ketika pada saat deklarasi (sebagai array statis)
2. Diisi user melalui keyboard
3. Diisi langsung satu per satu sesuai indeks
   values[2] ← 10 berarti [0, 0, 10, 0, 0]
   values[4] ← 90 berarti [0, 0, 10, 0, 90]
4.

Contoh: {4, 1, 5, 0, 0}

Ingat bahwa bila array ukuran 5, baru terisi 3, maka otomatis ruang yang masih kosong bernilai 0.

13.5.1 Menambah elemen di ujung

Menambah elemen di ujung array cukup mudah dilakukan. Kita cukup tambahkan 1 (satu) pada batas atas array (upperBound) atau ukuran array lalu berikan sebuah nilai dengan catatan asumsi bahwa alokasi ruang memori untuk array masih tersedia.

13.6 Mengakses Elemen

13.6.1 Memanggil Langsung

13.6.2 Menggunakan Loop

```
Kamus
values : array[1..N] of integer
```

```
values ← [12, 9, 10, 4, 90]
numOfValues ← 5
for (i ← 0, i < numOfValues, i ← i+1)
   cetak values[i]
```

13.7 Array Bersarang

13.8 Manipulasi Array

Diberikan nilai data = [3, 2, 1, 8, 11, 7]. Baliklah agar menjadi data = [7, 11, 8, 1, 2, 3]

```
Algoritma Membalik Array

Kamus:
N, limit, i, temp     : integer
data                  : array[1..N] of integer

input:
N ← 6
data ← [3, 2, 1, 8, 11, 7]

output: [7, 11, 8, 1, 2, 3]
```

```
1.   if (N MOD 2) = 1 then
2.      limit ← ⌊(N−1)/2⌋
3.   else
4.      limit ← ⌊N/2⌋
5.   for (i ← 1, i ≤ limit, i ← i+1) do
6.      temp ← data[i-1]
7.      data[i-1] ← data[N-i]
8.      data[N-i] ← temp
```

```
Eksekusi
```

```
1.   if 0 = 1 then
3.   else
4.      limit ← 3
5.   for (i ← 1, 1 ≤ 3, 2 ← 1+1) do
6.      temp ← 3
7.      data[0] ← 7   #data = [7, 2, 1, 8, 11, 7]
8.      data[5] ← 3   #data = [7, 2, 1, 8, 11, 3]

5.   for (i ← 2, 2 ≤ 3, 3 ← 2+1) do
6.      temp ← 2
7.      data[1] ← 11  #data = [7, 11, 1, 8, 11, 3]
8.      data[4] ← 2   #data = [7, 11, 1, 8, 2, 3]

5.   for (i ← 3, 3 ≤ 3, 4 ← 3+1) do
6.      temp ← 1
7.      data[1] ← 8   #data = [7, 11, 8, 8, 11, 3]
8.      data[3] ← 1   #data = [7, 11, 8, 1, 2, 3]
```

```
5.   stop
```

Untuk diingat

Array adalah salah satu bentuk struktur data yang menyimpan data secara terurut berdekatan dengan tipe data yang sama

13.9 Latihan

1. Diberikan

```
Kamus:
dayOfTransaction      : string
days                  : array string
discount              : real
```

```
1. days   ←   ['Sunday',   'Monday',   'Tuesday',   'Wednesday',   'Thursday',
   'Friday','Thursday']
2. dayOfTransaction ← 'Thursday'
3. dayIndex ← 6
4. discount ← 0.0
5.
6. if (dayOfTransaction = days[dayIndex]) then
7.    discount ← 0.1
8.
9. cetak discount
```

2. Diberikan array 2-Dimensi berikut:

Hitunglah nilai absolute kedua diagonal array 2-Dimensi tersebut.

Contoh: diagonal pertama [x, x, x, x], diagonal kedua [y, y, y, y].

Hasilnya = $\left|\sum_{i=1}^{4} x_i + \sum_{i=1}^{4} y_i\right|$

Bab 14 Penghitungan (counting) dan Penjumlahan (summation)

14.1 Penghitungan (*counting*)

Oh ya, pada contoh di atas kita mencetak * di dalam for loop. Kita tahu bahwa variabel counternya adalah i, nah saya penasaran apakah kita bisa mencetak counter loop? Jawabannya tentu bisa!. Inilah yang mendasari materi pencacah atau counter. Pasti pernah menggunakan timer di ponsel kan? Nah prinsip counter adalah menerapkan hal tersebut.

```
1.  for (i ← 0, i < 9, i ← i+1) do
2.     cetak '*' i tanpa baris baru
```

```
1. for (i ← 0, 0 < 9, 1 ← 0+1) do
2.    cetak * 0
1. for (i ← 1, 1 < 9, 2 ← 1+1) do
2.    cetak ** 1
1. for (i ← 2, 2 < 9, 3 ← 2+1) do
2.    cetak *** 2
1. for (i ← 3, 3 < 9, 4 ← 3+1) do
2.    cetak **** 3
1. for (i ← 4, 4 < 9, 4 ← 4+1) do
2.    cetak ***** 4
1. for (i ← 5, 5 < 9, 6 ← 5+1) do
2.    cetak ****** 5
1. for (i ← 6, 6 < 9, 7 ← 6+1) do
2.    cetak ******* 6
1. for (i ← 7, 7 < 9, 8 ← 7+1) do
2.    cetak ******** 7
1. for (i ← 8, 8 < 9, 9 ← 8+1) do
2.    cetak ********* 8
1. for (i ← 9, 9 < 9, 10 ← 9+1) Stop
```

Sedangkan, bila kita hilangan tanpa baris baru maka

```
1.  for (i ← 0, i < 9, i ← i+1) do
2.     cetak i
```

```
1. for (i ← 0, 0 < 9, 1 ← 0+1) do
2.    cetak 0
1. for (i ← 1, 1 < 9, 2 ← 1+1) do
2.    cetak 1
1. for (i ← 2, 2 < 9, 3 ← 2+1) do
2.    cetak 2
1. for (i ← 3, 3 < 9, 4 ← 3+1) do
2.    cetak 3
1. for (i ← 4, 4 < 9, 4 ← 4+1) do
2.    cetak 4
1. for (i ← 5, 5 < 9, 6 ← 5+1) do
2.    cetak 5
```

```
1. for (i ← 6, 6 < 9, 7 ← 6+1) do
2.    cetak 6
1. for (i ← 7, 7 < 9, 8 ← 7+1) do
2.    cetak 7
1. for (i ← 8, 8 < 9, 9 ← 8+1) do
2.    cetak 8
1. for (i ← 9, 9 < 9, 10 ← 9+1) Stop
```

For (i ← 0, i < 10, i ← i+1)

print random(0,1)

Diberikan data penjualan produk minimarket berikut:

penjualan = [750, 400, 650, 1000, 250, 100]

Hitunglah berapa jumlah penjualan yang di atas 500.

Hasil pemilihan ketua BEM

Hasil = [1,1,2,1,2,2,2,2,1,1,3,3,1,2,1,2,2,1,1,1,3,3,2,1,2,2,3,3,3,1]

Output:

Pemilih 1: xx

Pemilh 2: xx

Pemilih 3: xx

```
Algoritma Counter Pemungutan Suara

Kamus:
i, j, M, N     : integer
tallies        : array[1..M] of integer
count          : array[1..N] of integer
notExist       : boolean

input:
M ← 10
N ← 3
tallies ← [3,1,1,2,1,2,2,2,2,1,1,3,3,1,2,1,2,2,1,1,1,3,3,2,1,2,2,3,3,3,1]

output:
[[], [], []]
```

```
1. for (i ← 0, i < M, i ← i+1)
2.
3.   notExist ← true
4.
5.   for (j ← 0, j < N, j ← j+1) do
6.
```

```
7.      if tallies[i] = count[j][0] then
8.         count[j][1] ← count[j][1] + 1
9.         notExist ← false
10.        break
11.
12.   if notExist = true then
13.      count[] ← [tallies[i], 1]
```

```
Eksekusi

1.  for (i ← 0, 0 < 10, 1 ← 0+1) do
3.     notExist ← true
5.     for (j ← 0, 0 < 3, 1 ← 0+1) do
7.        if 3 = []          #False
12.    if true = true then
13.       count[] ← [3, 1]   #count = [[3, 1]]

1.  for (i ← 1, 1 < 10, 2 ← 1+1) do
3.     notExist ← true
5.     for (j ← 0, 0 < 3, 1 ← 0+1) do
7.        if 1 = 3           #False
12.    if true = true then
13.       count[] ← [1, 1]   #count = [[3, 1], [1, 1]]

1.  for (i ← 2, 2 < 10, 3 ← 2+1) do
3.     notExist ← true
5.     for (j ← 0, 0 < 3, 1 ← 0+1) do
7.        if 1 = 3           #False
5.     for (j ← 1, 1 < 3, 2 ← 1+1) do
7.        if 1 = 1           #True
8.           count[1][1] ← 1 + 1    #count = [[3, 1], [1, 2]]
9.           notExist ← false
10.          break
12.    if false = true #False

1.  for (i ← 3, 3 < 10, 4 ← 3+1) do
3.     notExist ← true
5.     for (j ← 0, 0 < 3, 1 ← 0+1) do
7.        if 2 = 3    #False
5.     for (j ← 1, 1 < 3, 2 ← 1+1) do
7.        if 2 = 1    #False
5.     for (j ← 2, 2 < 3, 3 ← 2+1) do
7.        if 2 = []   #False
12.    if true = true then
13.       count[] ← [2, 1]   #count = [[3, 1], [1, 2], [2, 1]]
```

14.2 Penjumlahan (Summation)

$$f(x) = \sum_{i=1}^{5} x_i$$

```
1.  total ← 0
2.  for (i ← 0, i < 10, i ← i+1)
```

```
3.  |   randomValue ← random(0,1)
4.  |   total ← total + randomValue
5.  print total
```

Note

Random() merupakan notasi algoritma yang digunakan buku ini untuk menunjukkan pembangkitan bilangan acak antara dua rentang nilai, misalnya antara 0-1 (contoh: 0.14, 0.76), 0.5-4.0 (contoh: 0.9, 3.7) yang bertipe real, atau 0-100 (Contoh: 5, 44, 51), 1-10 (Contoh: 2, 4, 1, 9, 10), untuk bertipe integer dan seterusnya. Pada praktik pemrograman, perintah yang digunakan tidak jauh beda dengan random(), yang biasanya merupakan fungsi bawaan bahasa pemrograman masing-masing. Jadi harap menyesuaikan penerapannya ketika membuat program. Misalnya: Javascript menggunakan `Math.random()`, Java menggunakan `Random().nextDouble()` dan C# menggunakan `random().NextDouble()`, dan Python menggunakan `random.random()`.

Algoritma penjumlahan struk belanja

Contoh lain adalah struk belanja. Perhatikan struk belanja pada Gambar xx. Perhatikan bagian kolom jumlah. Jika kita jumlahkan maka akan menghasilkan xx.

[gambar struk belanja]

Mari kita hitung total belanjanya menggunakan summation.

```
Algoritma Penjumlahan

Kamus:
sums    : array[1..N] of real
i, N    : integer
total   : real

Input:
sums ← [5200, 5280, 1990, 2090, 765, 2700, 8655, 645, 1300, 750, 935]
```

```
total ← 0
N ← 11

Output: xx
```

```
1.   for (i ← 0, i < N, i ← i+1) do
2.      total ← total + sums[i]
3.   cetak total
```

Eksekusi

```
1.   for (i ← 0, 0 < 11, 1 ← 0+1) do
2.      5200 ← 0 + 5200
1.   for (i ← 1, 1 < 11, 2 ← 1+1) do
2.      10480 ← 5200 + 5280
1.   for (i ← 2, 2 < 11, 3 ← 2+1) do
2.      12470 ← 10480 + 1990
1.   for (i ← 3, 3 < 11, 4 ← 3+1) do
2.      14560 ← 12470 + 2090
1.   for (i ← 4, 4 < 11, 5 ← 4+1) do
2.      15325 ← 14560 + 765
1.   for (i ← 5, 5 < 11, 6 ← 5+1) do
2.      18025 ← 15325 + 2700
1.   for (i ← 6, 6 < 11, 7 ← 6+1) do
2.      26680 ← 18025 + 8655
1.   for (i ← 7, 7 < 11, 8 ← 7+1) do
2.      27325 ← 26680 + 645
1.   for (i ← 8, 8 < 11, 9 ← 8+1) do
2.      28625 ← 27325 + 1300
1.   for (i ← 9, 9 < 11, 10 ← 9+1) do
2.      29375 ← 28625 + 750
1.   for (i ← 10, 10 < 11, 11 ← 10+1) do
2.      30310 ← 29375 + 935
1.   stop
3.   cetak 30310
```

14.3 Latihan

1. Berdasarkan contoh xx, misalnya diberikan kode:
   1 – Susi, 2 – Doni, 3 – Salbi. Ubahlah hasil akhirnya agar menjadi
   Susi – 3 suara
   Doni – 11 suara
   Salbi – 8 suara

2. Buatlah algoritma untuk dot product berikut

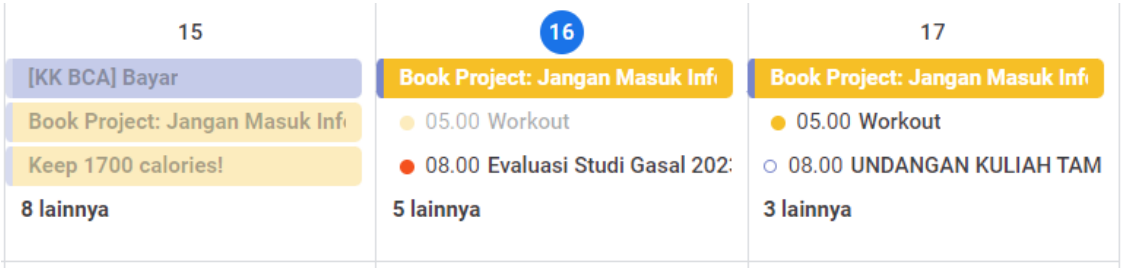
$$f(x) = \prod_{i=1}^{5} x_i$$

3. Lihat kembali struk di gambar xx. Buatlah program untuk menghasilkan struk yang sama seperti itu.
4. Seorang dosen ingin mencatat aktivitas hariannya. Input yang diberikan adalah (1) nama aktivitas/tugas, dan (2) estimasi durasi. Estimasi dibagi ke dalam jenis: ' = jam, sedangkan

" = menit. Buatlah program agar outputnya bisa seperti di bawah ini. Di baris terakhir menampilkan jumlah aktivitas dan durasi dalam format jam:menit.

Aktivitas	Durasi
Koreksi 9 soal PKPL	20"
Entri nilai akhir PKPL	30"
Koreksi 10 soal Tesi	50"
Entri nilai akhir 10 soal Tesi	30"
Periksa 3 draf skripsi	1'
Periksa 2 laporan magang	30"
Kirim contoh paper ke hendi	5"
Buat draf soal latihan Alpro berdasarkan daily report ini	2'
8 aktivitas	Xx Jam : xx menit

5. Gambar xx menunjukkan contoh tampilan agenda pada Google Calendar versi *desktop* untuk tiga hari yaitu tanggal 15, 16, dan 17. Secara baku, Google Calendar hanya menampilkan tiga agenda pertama, sedangkan sisanya ditunjukkan dalam bentuk jumlah agenda lainnya. Contohnya tanggal 15 hingga 18 tersebut masing-masing memiliki 8, 5, dan 3 sisa agenda.



Gambar xx. Contoh agenda Google Calendar

Tugas Anda, buatlah program untuk membuat calendar serupa. Inputannya berupa *array*. Anda diperbolehkan memasukkan agenda yang berbeda, yang penting harus lebih dari 3 agenda setiap hari. Berikut contoh outputnya:

```
15              16                                  17
[KK BCA] Bayar          Book Project:               Book                Project
Book Project:           05.00 Workout               05.00               Workout
Keep 1700 calories!     08.00 Evaluasi Studi Gasal 2023 08.00  Undangn  Kuliah  Tamu
8 lainnya               5 lainnya                   3 lainnya
```

Bab 15 Fungsi/Prosedur

15.1 Sejarah Fungsi/Prosedur
15.2 Konsep Dasar Fungsi/Prosedur
15.3 Parameter
15.4 Argumen
15.5 Latihan

Bab 16 Sorting

16.1 Selection Sort

Prasyarat kompetensi: tukar (*swap*) nilai antarvariabel, nested for

Algoritma SelectionSortAsc

Kamus: `values`: array of integer; `i,j,key`: integer;

```
1.  Prosedur selectionSortAsc(values)
2.     arrLimit = size(values)
3.     for (i=0, i < arrLimit, i++)
4.         for (j=i+1, j < arrLimit, j++)
5.             if values[i] > values[j]:
6.                swap(values[i], values[j])
7.     return values
```

Input : `[7, 4, 10, 2, 5]`
Output : `[2, 4, 5, 7, 10]`

Eksekusi

Baris-2	`arrlimit = 5`	
Baris-3	`for (i=0, 0<5, 1++)`	
Baris-4	**`for (j=1, j<5, 2++)`**	
Baris-5	`if 7 > 4`	`true`
Baris-6	`swap(7, 4)`	`[4,7,10,2,5]`
baris-4	`for (j=2, j<5, 3++)`	
baris-5	`if 4 > 10`	`false`
Baris-4	`for (j=3, 3<5, 4++)`	
baris-5	`if 4 > 2`	`true`
baris-6	`swap(4, 2)`	`[2,7,10,4,5]`
baris-4	`for (j=4, 4<5, 5++)`	
baris-5	`if 2 > 5`	`false`
baris-3	**`for (i=1, 1<5, 2++)`**	
baris-4	`for (j=2, 2<5, 3++)`	
baris-5	`if 7 > 10`	`false`
baris-4	`for (j=3, 3>5, 4++)`	
Baris-5	`if 7 > 4`	`true`
Baris-6	`swap(7, 4)`	`[2,4,10,7,5]`
Baris-4	`for (j=4, 4<5, 5++)`	
Baris-5	`if 4 > 5`	`false`
Baris-3	**`for (i=2, 2<5, 3++)`**	
Baris-4	`for (j=3, 3<5, 4++)`	
Baris-5	`if 10 > 7`	`true`
Baris-6	`swap(10, 7)`	`[2,4,7,10,5]`
Baris-4	`for (j=4, 4<5, 5++)`	
Baris-5	`if 7 > 5`	`true`
Baris-6	`swap(7, 5)`	`[2,4,5,10,7]`
Baris-3	`for (i=3, 3<5, 4++)`	
Baris-4	`for (j=4, 4<5, 5++)`	
Baris-5	`if 10 > 7`	`true`
Baris-6	`swap(10, 7)`	`[2,4,5,7,10]`

Baris-3	`for (i=4, 4<5, 5++)`	
Baris-7	`return [2,4,5,7,10]`	`selesai`

16.2 Insertion Sort

Algoritma `InsertionSortAsc`
Berfungsi untuk mengurutkan bilangan integer dengan urut menaik atau *ascending*
Kamus: `values`: array of integer; `i,j,key`: integer;
Input : `[7, 4, 10, 2, 5]`
Output : `[2, 4, 5, 7, 10]`

```
1.  function insertionSortAsc(values)
2.      for (i = 1, i < size(values), i++)
3.          key = values[i]
4.          j = i - 1
5.
6.          while (j >= 0) and (values[j] > key)
7.              values[j + 1] = values[j]
8.              j = j - 1
9.
10.         values[j + 1] = key
11.
12.     return values
```

Eksekusi

Baris-2	**`for (i=1, 1<5, i++)`**	
Baris-3	`key = 4`	
Baris-4	`j = 0`	
Baris-6	**`while 0 >= 0 and 7 > 4`**	**`true and true`**
Baris-7	`values[1] = 7`	`[7,7,10,2,5]`
Baris-8	`j = -1`	
Baris-6	**`while -1 >= 0 and 5 > 4`**	**`false and true`**
Baris-10	`values[0] = 4`	`[4,7,10,2,5]`
Baris-2	**`for (i=2, 2<5, i++)`**	
Baris-3	`key = 10`	
Baris-4	`j = 1`	
Baris-6	**`while 1 >= 0 and 7 > 10`**	**`true and false`**
Baris-10	`values[2] = 10`	`[4,7,10,2,5]`
Baris-2	**`for (i=3, 3<5, i++)`**	
Baris-3	`key = 2`	
Baris-4	`j = 2`	
Baris-6	**`while 2 >= 0 and 10 > 2`**	**`true and true`**
Baris-7	`values[3] = 10`	`[4,7,10,10,5]`
Baris-8	`j = 1`	
Baris-6	**`while 1 >= 0 and 7 > 2`**	**`true and true`**
Baris-7	`values[2] = 7`	`[4,7,7,10,5]`
Baris-8	`j = 0`	
Baris-6	**`while 0 >= 0  and 4 > 2`**	**`true and true`**
Baris-7	`values[1] = 4`	`[4,4,7,10,5]`

Baris-8	`j = -1`	
Baris-6	**`while -1 >= 0 and 5 > 2`**	**`false and true`**
Baris-10	`values[0] = 2`	`[2,4,7,10,5]`
Baris-2	**`for (i=4, 4<5, i++)`**	
Baris-3	`key = 5`	
Baris-4	`j = 3`	
Baris-6	**`while 3 >= 0 and 10 > 5`**	**`true and true`**
Baris-7	`values[4] = 10`	`[2,4,7,10,10]`
Baris-8	`j = 2`	
Baris-6	**`while 2 >= 0 and 7 > 5`**	**`true and true`**
Baris-7	`values[3] = 7`	`[2,4,7,7,10]`
Baris-8	`j = 1`	
Baris-6	**`while 1 >= 0 and 4 > 5`**	**`true and false`**
Baris-10	`values[2] = 5`	`[2,4,5,7,10]`
Baris-12	**`return [2,4,5,7,10]`**	**`selesai`**

Bab 17 Searching

17.1 Linear Search

Prasyarat kompetensi: for loop, array, seleksi kondisi

Topik: kelebihan, kelemahan, kapan sebaiknya digunakan, konsep umum.

Algoritma LinearSearch

Deskripsi
-
Kamus
-

```
1.  function linearSearch(values, searchKey)
2.     for (i = 0, i < size(values), i++)
3.         if values[i] == searchKey
4.           return i+1
```

Input
```
values    = [7, 4, 10, 2, 5, 1, 3]
searchKey = 2
```

Output
```
4
```

Eksekusi

Baris-2	`for (i=0, 0<7, 0++)`	
Baris-3	`   if 7 == 2`	`false`
Baris-2	`for (i=1, 1<7, 1++)`	
Baris-3	`   if 4 == 2`	`false`
Baris-2	`for (i=2, 2<7, 2++)`	
Baris-3	`   if 10 == 2`	`false`
Baris-2	`for (i=3, 3<7, 3++)`	
Baris-3	`   if 2 == 2`	`true`
Baris-4	**`      return 4`**	

Ketika dipanggil

```
1.  values = [4, 2, 3, 1, 5, 10, 7]
2.  found = sequentialSearch(values, 4)
3.  if found
4.     print('indeks ke-',found-1)
5.  else
6.     print('not found')
```

Eksekusi

Baris-2	`Found = 4`	
Baris-3	`If 4`	`True`
Baris-4	`   indeks ke-3`	

17.2 Binary Search

Kompetensi prasyarat: algoritma *sorting*, while loop, array

Topik: sejarah, penemu, konsep umum (membagi menjadi 2 bagian), kapan sebaiknya digunakan, kelebihan, kelemahan

Algoritma `BinarySearch`

Deskripsi
Berfungsi untuk menemukan indeks dalam *array* berdasarkan pada kunci pencarian yang diberikan

Kamus
`values`: array of integer
`lowIndex, highIndex, searchKey`: integer
[middleIndex]: hasil pembagian yang dibulatkan ke bawah

```
1.  function binarySearch(values, searchKey)
2.     sortedValues = SortAscending(values)
3.     lowIndex     = 0
4.     highIndex    = size(sortedValues)-1
5.
6.     while lowIndex <= highIndex
7.        ⌊middleIndex⌋ = (lowIndex + highIndex) / 2
8.        if searchKey == sortedValues[middleIndex]
9.           return middleIndex + 1
10.       if searchKey < sortedValues[middleIndex]
11.          highIndex = middleIndex - 1
12.       if searchKey > sortedValues[middleIndex]
13.          lowIndex = middleIndex + 1
```

Input
```
values    = [7, 4, 10, 2, 5, 1, 3]
searchKey = 2
```

Output
```
2
```

Eksekusi

Baris-2	`sortedvalues = [1, 2, 3, 4, 5, 7, 10]`	
Baris-3	`lowindex = 0`	
Baris-4	`highindex = 6`	
Baris-6	**`while 0 <= 6`**	**`true`**
Baris-7	`   middleindex = 3`	
Baris-8	`   if 2 == 4`	`false`
Baris-10	`   if 2 < 4`	`true`
Baris-11	`      highindex = 2`	
Baris-14	`   if 2 > 4`	`false`
Baris-6	**`while 0 <= 2`**	**`true`**
Baris-7	`   ⌊middleindex⌋ = 1`	
Baris-8	`   if 2 == 2`	`True`

Baris-9	**return 2**	

Input

```
values    = [7, 4, 10, 2, 5, 1, 3]
searchKey = 6
```

Output

```
False
```

Eksekusi

Baris-2	sortedvalues = [1, 2, 3, 4, 5, 7, 10]	
Baris-3	lowindex = 0	
Baris-4	highindex = 6	
Baris-6	**while 0 <= 6**	**true**
Baris-7	middleindex = 3	
Baris-8	if 6 == 4	false
Baris-10	if 6 < 4	false
Baris-12	if 6 > 4	true
Baris-13	lowindex = 4	
Baris-6	**while 4 <= 6**	**true**
Baris-7	middleindex = 5	
Baris-8	if 6 == 7	false
Baris-10	if 6 < 7	true
Baris-11	highindex = 4	
Baris-12	if 6 > 7	false
Baris-6	**while 4 <= 4**	**true**
Baris-7	middleindex = 4	
Baris-8	if 6 == 5	false
Baris-10	if 6 < 5	false
Baris-12	if 6 > 5	true
Baris-13	lowindex = 5	
Baris-7	**while 5 <= 4**	**false**

17.3 Efektivitas dan efisiensi Binary Search

{big O}

Bab 18 Array Dua Dimensi dan Operasi Matriks

Kompetensi prasyarat: array, for loop, nested loop

Penjumlahan Matriks

Algoritma CetakMatriks

```
1.  function printmatriks(matriks){
2.     sizerow = size(matriks)
3.     sizecol = size(matriks[0])
4.     for (i = 0, i < sizerow, i++)
5.        for (j = 0, j < sizecol, j++)
6.           print matriks[i][j]
7.        pindah baris
```

Input

```
Matriks = [
   [2,6,7,3],
   [1,5,2,2],
   [4,10,9,8]
]
```

Output

```
   2 6 7 3
   1 5 2 2
   4 10 9 8
```

Eksekusi

Baris-2	sizeRow = 3	
Baris-3	sizeCol = 4	
Baris-4	For (i=0, 0<3, 1++)	
Baris-5	For (j=0, 0<4, 1++)	
Baris-6	Print matriks[0][0]	2
Baris-5	For (j=1, 1<4, 2++)	
Baris-6	Print matriks[0][1]	2 6
Baris-5	For (j=2, 2<4, 3++)	
Baris-6	Print matriks[0][2]	2 6 7
Baris-5	For (j=3, 3<4, 4++)	
Baris-6	Print matriks[0][3]	2 6 7 3
Baris-4	For (i=1, 1<3, 2++)	
Baris-5	For (j=0, 0<4, 1++)	
Baris-6	Print matriks[1][0]	1
Baris-5	For (j=1, 1<4, 2++)	
Baris-6	Print matriks[1][1]	1 5
Baris-5	For (j=2, 2<4, 3++)	
Baris-6	Print matriks[1][2]	1 5 2
Baris-5	For (j=3, 3<4, 4++)	
Baris-6	Print matriks[1][3]	1 5 2 2
Baris-4	For (i=2, 2<3, 3++)	
Baris-5	For (j=0, 0<4, 1++)	
Baris-6	Print matriks[2][0]	4
Baris-5	For (j=1, 1<4, 2++)	
Baris-6	Print matriks[2][1]	4 10

Baris-5	For (j=2, 2<4, 3++)	
Baris-6	Print matriks[1][2]	1 5 2
Baris-5	For (j=3, 3<4, 4++)	
Baris-6	Print matriks[1][3]	1 5 2 2

Penjumlahan Matriks

```
1.  function matrixSummation(rowSize, colSize, matriks)
2.     colSum = []
3.     for (col = 0, col < colSize, col++)
4.        sum = 0
5.        for (row = 0, row < rowSize, row++)
6.           sum = sum + matriks[row][col]
7.        colSum[col] = sum
8.
9.     for (i = 0, i < colSize, i++)
10.       print colSum[i] " "
```

Input

```
matriks = [
   [2,6,7,3],
   [1,5,2,2],
   [4,10,9,8]
]
```

Output

```
   7 21 18 13
```

Eksekusi

Baris-1	function matrixsummation(3, 4, [[2,6,7,3],[1,5,2,2],[4,10,9,8]])	
Baris-2	colsum = []	[]
Baris-3	for (col = 0, 0 < 4, 1++)	
Baris-4	sum = 0	0
Baris-5	for (row = 0, 0 < 3, 1++)	
Baris-6	sum = 0 + 2	2
Baris-5	for (row = 1, 1 < 3, 2++)	
Baris-6	sum = 2 + 1	3
Baris-5	for (row = 2, 2 < 3, 3++)	
Baris-6	sum = 3 + 4	7
Baris-7	colsum[0] = 7	[7]
Baris-3	for (col = 1, 1 < 4, 2++)	
Baris-4	sum = 0	0
Baris-5	for (row = 0, 0 < 3, 1++)	
Baris-6	sum = 0 + 6	6
Baris-5	for (row = 1, 1 < 3, 2++)	
Baris-6	sum = 6 + 5	11
Baris-5	for (row = 2, 2 < 3, 3++)	
Baris-6	sum = 11 + 10	21
Baris-7	colsum[1] = 21	[7, 21]
Baris-3	for (col = 2, 2 < 4, 3++)	

Baris-4	sum = 0	0
Baris-5	for (row = 0, 0 < 3, 1++)	
Baris-6	sum = 0 + 7	7
Baris-5	for (row = 1, 1 < 3, 2++)	
Baris-6	sum = 7 + 2	9
Baris-5	for (row = 2, 2 < 3, 3++)	
Baris-6	sum = 9 + 9	18
Baris-7	colsum[2] = 18	[7, 21, 18]
Baris-3	for (col = 3, 3 < 4, 4++)	
Baris-4	sum = 0	0
Baris-5	for (row = 0, 0 < 3, 1++)	
Baris-6	sum = 0 + 3	3
Baris-5	for (row = 1, 1 < 3, 2++)	
Baris-6	sum = 3 + 2	5
Baris-5	for (row = 2, 2 < 3, 3++)	
Baris-6	sum = 5 + 8	13
Baris-7	colsum[3] = 13	[7, 21, 18, 13]
Baris-9	for (i = 0, i < 4, 1++)	
Baris-10	print 7	7
Baris-9	for (i = 1, 1 < 4, 2++)	
Baris-10	print 21	7 21
Baris-9	for (i = 2, 2 < 4, 3++)	
Baris-10	print 18	7 21 18
Baris-9	for (i = 3, 3 < 4, 4++)	
Baris-10	print 13	7 21 18 13

Bab 19 String

19.1 Definisi String

Selain tipe data numerik seperti integer dan real kita akan kerap berjumpa dengan data-data di luar itu seperti huruf, angka (‘3244’,’1’,’001’, dst), simbol, bahkan spasi yang semuanya disebut sebagai karakter. Himpunan karakter tersebut dinamakan **string**.

Dalam praktiknya, string terdiri dari ururan karakter yang bisa berjumlah puluhan, ratusan, ribuan hingga jutaan. Salah satu ciri khas string biasanya ditulis di antara tanda petik, seperti “hai, ardiansyah” atau “ardiansyah@tif.uad.ac.id”, bahkan “1982844555”.

19.2 Representasi String

Terdapat beberapa cara untuk merepresentasikan string, tergantung kemampuan bahasa pemrograman yang digunakan yaitu antara lain:

1. Berupa array dari karakter
   Contoh:
   myString = [‘a’,’r’,’d’,’i’,’a’,’n’]
   myString = [‘ardi’,’ridi’,’doni’,’isyana’,’alan’,’nuari’]
   myString = [‘ardi@uad.ac.id’,’ridi’,’doni’,’isyana’,’alan’,’nuari’]
2. Berupa tipe data string
   Contoh:
   String1 = “hello”
   String2 = “how are you?”

19.3 Operasi Dasar String

19.3.1 Penggabungan (*concatenation*)

Menggabungan 2 atau lebih string menjadi satu. Misal:
firstName = “Aliudin”
middleName = “Cipta”
lastName = “Semesta”
fullName = firstName + ' ' + middleName + ' '+ lastName
print (fullName)
Output: Aliudin Cipta Semesta

Names = [“Aliudin”, “cipta”, “semesta”]
fullName = names[0] + names[1] + names[2]
print(fullName)
Output: Aliudin Cipta Semesta

atau menggunakan loop

```
names = ['Aliudin', 'cipta', 'semesta']
for (name=0, name < len(names), name++)
    print(names[name] " ")
Output: Aliudin Cipta Semesta
```

19.3.2 Perbandingan String

```
str1 = "Hai"
str2 = "hai"
result = str1 == str2
print(result)
```

```
Output: False
```

```
str1 = "Hai"
str2 = "Hai"
result = str1 == str2
print(result)
Output: True
```

19.3.3 Panjang String (*length*)

Operasi ini akan menghitung berapa jumlah karakter dalam sebuah string yang diberikan. Contoh

```
1.  function stringLength(string)
2.     length = 0
3.     while(string[length] != null)
4.        length = length + 1
5.     return length
```

Input: string = ‘Ardi’
Output: 4

Eksekusi

Baris-1	`function stringlength(‘ardi’)`	
Baris-2	`length = 0`	`0`
Baris-3	`while (‘a’ != null)`	`true`
Baris-4	`length = 0 + 1`	`1`
Baris-3	`while (‘r’ != null)`	`true`
Baris-4	`length = 1 + 1`	`2`
Baris-3	`while (‘d’ != null)`	`true`
Baris-4	`length = 2 + 1`	`3`
Baris-3	`while (‘i’ != null)`	`true`
Baris-4	`length = 3 + 1`	`4`
Baris-3	`while (‘’ != null)`	`false`
Baris-5	`return 4`	

Karena kita tidak tahu berapa panjang string atau banyaknya karakter, maka perulangan yang cocok digunakan adalah `while`. Jadi, selama karakter yang dibaca tidak sama dengan `null`, maka length akan ditambah dengan nilai 1.
Perlu diperhatikan bahwa yang dimaksud dengan null adalah akhir dari string yang tidak memiliki karakter lain. Dalam bahasa C atau C++ istilah ini dikenal sebagai *null-terminating* dengan simbol `'\0'`. Sedangkan dalam bahasa pemrograman lain seperti Python, Java, Go, C# dan banyak lagi lainnya tidak mengenal konsep *null-terminating*. Oleh karena itu perlu dilakukan penyesuaian dengan bahasa pemrograman tersebut.

Berikut contoh bila menggunakan Python

```
def getStringLength(string):
    length = 0
    index = 0
    try:
        while True:
            string[index]
            length += 1
            index += 1
    except IndexError:
```

```
        pass
    return length

strings = "Kala itu di musim yang lalu"
result = getStringLength(strings)
print(strings, 'terdiri ', result, 'karakter')
# Output: Kala itu di musim yang lalu, terdiri, 27 karakter
```

Latihan

Diberikan input:
["Jakarta", "Yogyakarta", "Palembang Darussalam"]
Buatlah program untuk menghitung panjang karakter tiap elemen array berupa string tersebut.
Contoh output:
[7, 10, 20]

Buatlah program menggunakan bahasa java/kotlin, javascript, dart, dll untuk menghitung panjang string, tanpa menggunakan library/fungsi bawaan sama sekali.

19.3.4 Pemotongan (Substring) string

1. Pemotongan berdasarkan indeks

```
1.  function delString(startIndex, lastIndex, strings)
2.     deletedString = ''
3.     for (char = startIndex, char <= lastIndex, char++)
4.        deletedString = deletedString + strings[char]
5.     return deletedString
```

Input
String = "BG 70 VA"
startIndex = 3
lastIndex = 4

Output
70

Eksekusi

Baris-1	function delstring(3, 4, "BG 70 VA")	
Baris-2	deleteString = ''	''
Baris-3	For (char = 3, 3 <= 4, 4++)	True
Baris-4	deleteString = '' + '7'	'7'
Baris-3	For (char = 4, 4 <= 4, 5++)	True
Baris-4	deleteString = '7' + '0'	'70'
Baris-3	For (char = 5, 5 <= 4, 6++)	false
Baris-5	Return '70'	'70'

2. Pemotongan dengan langkah (step) tertentu
3. Pemotongan dengan indeks negatif (dimulai dari akhir string)
4. Pemotongan dengan split

5. Pemotongan dengan partisi
6. Pemotongan dengan pencarian dan *slicing*

19.4 Manipulasi String

19.4.1 Substring

Substring mulai indeks awal dan seterusnya alitas tanpa indeks batas akhir.

```
1.  function substringWithoutEnd(strings, startIndex, lenString)
2.     for (char = startIndex, char < lenString, char++)
3.        Print strings[char]
```

Input:

```
strings = "Hello Ardiansyah"
lenString = sizeArrayOf(strings)
startIndex = 7
```

Output

```
rdiansyah
```

Eksekusi

Baris-1	function subStringWithoutEnd("Hello Ardiansyah", 7, 16)	
Baris-2	For (char = 7, 7 < 16, 8++)	
Baris-3	Print strings[7]	"r"
Baris-2	For (char = 8, 8 < 16, 9++)	
Baris-3	Print strings[8]	"rd"
Baris-2	For (char = 9, 9 < 16, 10++)	
Baris-3	Print strings[9]	"rdi"
Baris-2	For (char = 10, 9 < 16, 11++)	
Baris-3	Print strings[10]	"rdia"
Baris-2	For (char = 11, 11 < 16, 12++)	
Baris-3	Print strings[11]	"rdian"
Baris-2	For (char = 12, 12 < 16, 13++)	
Baris-3	Print strings[12]	"rdians"
Baris-2	For (char = 13, 13 < 16, 14++)	
Baris-3	Print strings[13]	"rdiansy"
Baris-2	For (char = 14, 14 < 16, 15++)	
Baris-3	Print strings[14]	"rdiansya"
Baris-2	For (char = 15, 15 < 16, 16++)	
Baris-3	Print strings[15]	"rdiansyah"

Substring dengan indeks awal dan akhir.

```
1.  function substringWithEnd(strings, indexRange, lenString)
2.     for (char = indexRange[0], char ≤ indexRange[1], char++)
3.           Print strings[char]
```

Input:
```
strings = "Hello Ardiansyah"
lenString = sizeArrayOf(strings)
indexRange = [2, 10]
```

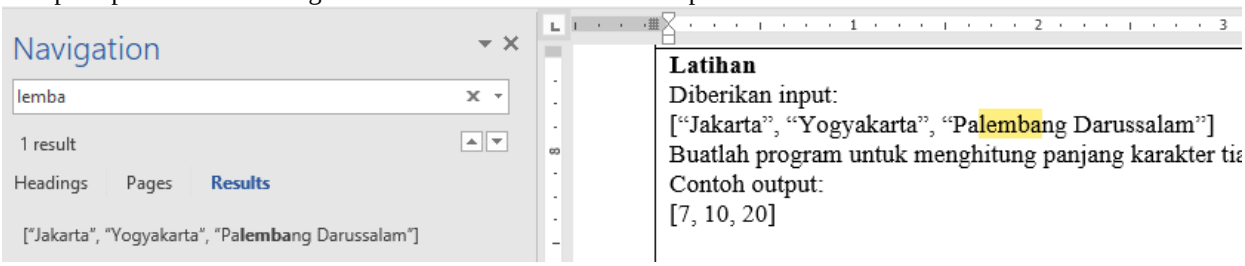
Output
```
llo Ardia
```

Eksekusi

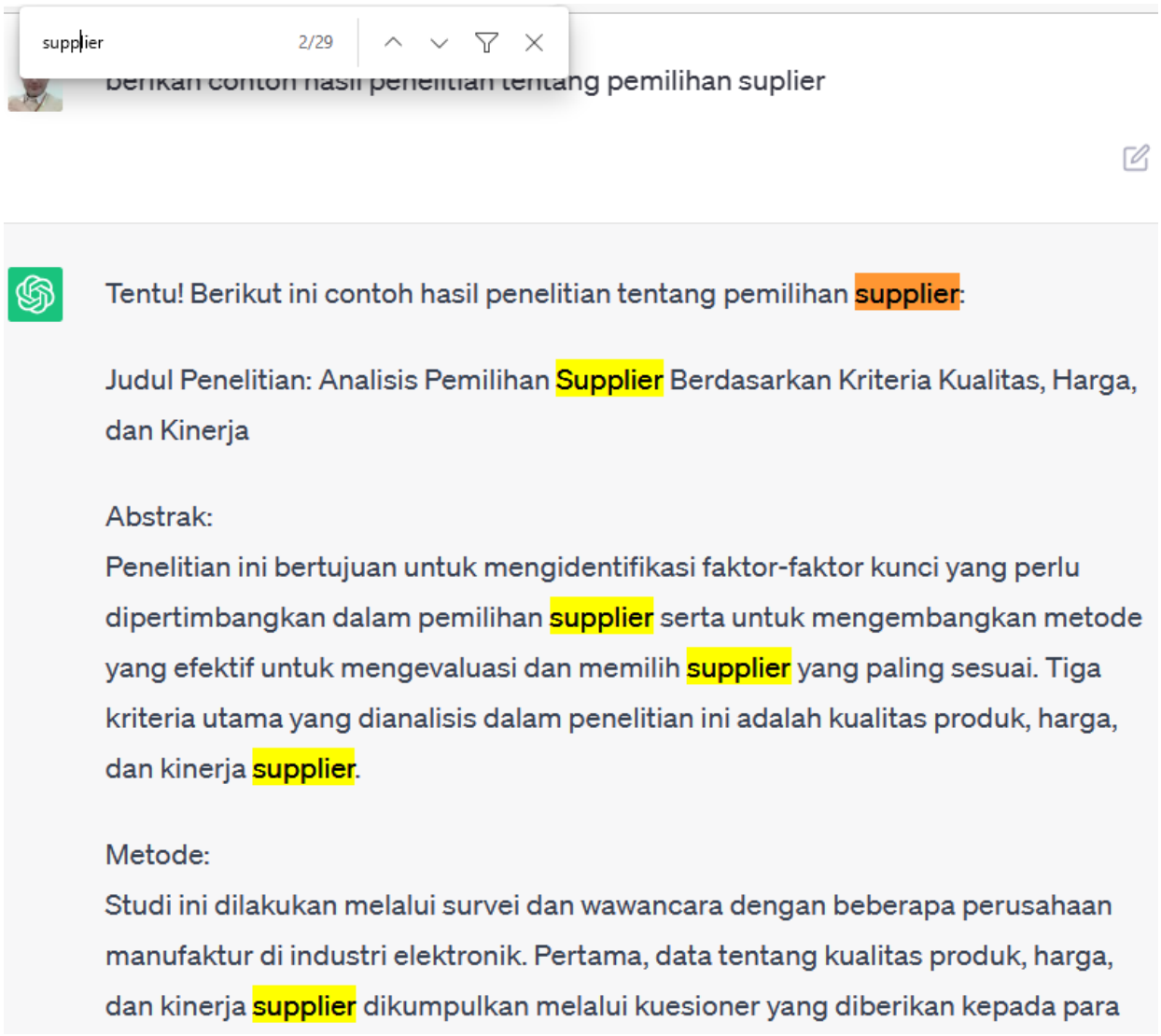
Baris-1	`function subStringWithEnd("Hello Ardiansyah", [2, 10], 16)`	
Baris-2	`   For (char = 2, 2 <= 10, 3++)`	
Baris-3	`      Print strings[2]`	`"l"`
Baris-2	`   For (char = 3, 3 <= 10, 4++)`	
Baris-3	`      Print strings[3]`	`"ll"`
Baris-2	`   For (char = 4, 4 <= 10, 5++)`	
Baris-3	`      Print strings[4]`	`"llo"`
Baris-2	`   For (char = 5, 5 <= 10, 6++)`	
Baris-3	`      Print strings[5]`	`"llo "`
Baris-2	`   For (char = 6, 6 <= 10, 7++)`	
Baris-3	`      Print strings[6]`	`"llo A"`
Baris-2	`   For (char = 7, 7 <= 10, 8++)`	
Baris-3	`      Print strings[7]`	`"llo Ar"`
Baris-2	`   For (char = 8, 8 <= 10, 9++)`	
Baris-3	`      Print strings[8]`	`"llo Ard"`
Baris-2	`   For (char = 9, 9 <= 10, 10++)`	
Baris-3	`      Print strings[9]`	`"llo Ardi"`
Baris-2	`   For (char = 10, 10 <= 10, 11++)`	
Baris-3	`      Print strings[10]`	`"llo Ardia"`

19.4.2 Pencarian substring

Kita pasti pernah mencari bagian dari kata di Microsoft Word seperti ini



Atau juga melakukan hal yang sama di browser web seperti ini



Kedua aktivitas itu biasanya disebut sebagai pencarian substring atau subteks, atau dengan kata lain kita hendak mencari bagian atau potongan string dari satu atau lebih kata.

```
1.  function delstring(string)
2.     strlen = getstringlength(string)
3.     deletedstring = ''
4.     for (char = 1, char < strlen, char++)
5.        deletedstring = deletedstring + string[char]
6.     return deletedstring
7.
8.  function findsubstring(str, substr)
9.     slices = []
10.    found = false
11.    lastSlice = ''
12.    strlength = getstringlength(str)
13.    substrlength = getstringlength(substr)
```

```
14.    for (i = 0, i < strlength, i++)
15.       lastslice = lastslice + str[i]
16.       lastslicelength = getstringlength(lastslice)
17.       if lastslicelength == substrlength and lastslice != substr
18.          slices.append(lastslice)
19.          lastslice = delstring(lastslice)
20.       if lastslicelength == substrlength and lastslice == substr
21.          startindex = (i-substrlength)+1
22.          lastindex = i
23.          found = [startindex, lastindex]
24.          break
25.    return found
```

Catatan
Fungsi `getStringLength` yang dipanggil di fungsi `delString` dan `findSubstring` menggunakan fungsi yang telah dibahas pada panjang string sebelumnya.

Input:
string = “Ardiansyah”
subString = “dian”
Output:
[2, 5]
Artinya, subString “dian” ditemukan mulai indeks ke-3 hingga indeks ke-6 pada string “Ardiansyah” tersebut.

Eksekusi fungsi delString

Baris-1	`Function delString(‘Ardi’)`	
Baris-2	`strLen = getStringLength(‘Ardi’)`	`4`
Baris-3	`deleteString = ‘’`	`‘’`
Baris-4	`For (char = 1, 1 < 4, 2++)`	
Baris-5	`deleteString = ‘’ + ‘r’`	`‘r’`
Baris-4	`For (char = 2, 2 < 4, 3++)`	
Baris-5	`deleteString = ‘r’ + ‘d’`	`‘rd’`
Baris-4	`For (char = 3, 3 < 4, 4++)`	
Baris-5	`deleteString = ‘rd’ + ‘i’`	`‘rdi’`
Baris-1	`Function delString(‘rdia’)`	
Baris-2	`strLen = getStringLength(‘rdia’)`	`4`
Baris-3	`deleteString = ‘’`	`‘’`
Baris-4	`For (char = 1, 1 < 4, 2++)`	
Baris-5	`deleteString = ‘’ + ‘d’`	`‘d’`
Baris-4	`For (char = 2, 2 < 4, 3++)`	
Baris-5	`deleteString = ‘d’ + ‘i’`	`‘di’`
Baris-4	`For (char = 3, 3 < 4, 4++)`	
Baris-5	`deleteString = ‘di’ + ‘a’`	`‘dia’`

Eksekusi fungsi findSubstring

Baris-8	`function findSubtring(“Ardiansyah”, “dian”)`	
Baris-9	`Slices = []`	`[]`
Baris-10	`Found = False`	`0`
Baris-11	`lastSlice = ‘’`	`‘’`
Baris-12	`strLength = getStringLength(‘Ardiansyah’)`	`10`
Baris-13	`subStrLength = getStringLength(‘dian’)`	`4`
Baris-14	`For (I = 0, 0 < 10, 1++)`	
Baris-15	`lastSlice = ‘’ + A`	`‘A’`

Baris-16	lastSliceLength = getStringLength(‘A’)	1
Baris-17	If 1 == 4 and ‘A’ != ‘dian’	False and True
Baris-20	If 1 == 4 and ‘A’ == ‘dian	False and false
Baris-14	For (I = 1, 1 < 10, 2++)	
Baris-15	lastSlice = ‘A’ + ‘r’	‘Ar’
Baris-16	lastSliceLength = getStringLength(‘Ar’)	2
Baris-17	If 2 == 4 and ‘Ar’ != ‘dian’	False and True
Baris-20	If 2 == 4 and ‘A’ == ‘dian	False and false
Baris-14	For (I = 2, 2 < 10, 3++)	
Baris-15	lastSlice = ‘Ar’ + ‘d’	‘Ard’
Baris-16	lastSliceLength = getStringLength(‘Ard’)	3
Baris-17	If 3 == 4 and ‘Ard’ != ‘dian’	False and True
Baris-20	If 3 == 4 and ‘Ard’ == ‘dian	False and false
Baris-14	For (I = 3, 3 < 10, 4++)	
Baris-15	lastSlice = ‘Ard’ + ‘i’	‘Ardi’
Baris-16	lastSliceLength = getStringLength(‘Ardi’)	4
Baris-17	If 4 == 4 and ‘Ardi’ != ‘dian’	true and True
Baris-18	Slices.append(‘Ardi’)	[‘Ardi’]
Baris-19	lastSlice = delstring(‘Ardi’)	‘rdi’
Baris-20	If 4 == 4 and ‘rdi’ == ‘dian’	True and false
Baris-14	For (I = 4, 4 < 10, 5++)	
Baris-15	lastSlice = ‘rdi’ + ‘a’	‘rdia’
Baris-16	lastSliceLength = getStringLength(‘Ardi’)	4
Baris-17	If 4 == 4 and ‘rdia’ != ‘dian’	true and True
Baris-18	Slices.append(‘rdia’)	[‘Ardi’,’rdia’]
Baris-19	lastSlice = delstring(‘rdia’)	‘dia’
Baris-20	If 4 == 4 and ‘dia’ == ‘dian’	True and false
Baris-14	For (I = 5, 5 < 10, 6++)	
Baris-15	lastSlice = ‘dia’ + ‘n’	‘dian’
Baris-16	lastSliceLength = getStringLength(‘dian’)	4
Baris-17	If 4 == 4 and ‘dian’ != ‘dian’	true and false
Baris-20	If 4 == 4 and ‘dian’ == ‘dian’	True and true
Baris-21	startIndex = (5 – 4) + 1	2
Baris-22	lastIndex = 5	5
Baris-23	Found = [2, 5]	[2, 5]
Baris-24	break	
Baris-25	Return [2, 5]	[2, 5]

Latihan

1. Modifikasilah contoh substring dengan menggunakan perulangan **`while`**.
2. Modifikasilah program di atas agar bisa menambahkan fungsi untuk menghitung jumlah substring yang ditemukan.

Palembang adalah ibu kota Provinsi Sumatra Selatan, Indonesia. Kota ini merupakan kota tertua yang ada di Indonesia, yang berasal dari abad ke-7. Palembang pernah menjadi ibu kota Sriwijaya, sebuah kerajaan Melayu yang memerintah sebagian dari Nusantara bagian barat dan menguasai rute perdagangan maritim, khususnya di Selat Malaka. Palembang digabungkan ke dalam Hindia Belanda pada tahun 1825 setelah penghapusan Kesultanan Palembang. Palembang diberi status sebagai kota pada 1 April 1906 pada zaman Hindia Belanda. Palembang kini merupakan salah satu kota terbesar di Sumatra dan Indonesia. Kota ini telah menjadi tuan rumah dari beberapa acara internasional, termasuk Pesta Olahraga Asia Tenggara 2011 dan Pesta Olahraga Asia 2018.

```
Kata yang dicari: "lemba"
Output: Ditemukan 6 kali
```

3. Modifikasilah agar bisa membedakan atau tidak membedakan antara karakter huruf kapital dan non-kapital

19.4.3 Tokenisasi (pemisah string)

Tokenisasi adalah proses pemecahan string menjadi bagian-bagian kecil atau juga dikenal sebagai pemisah (*splitter*). Contoh paling jamak ditemukan adalah memecah suatu string ke dalam kata-kata atau pemisah kata.

```
1.  function wordSplitting(strings)
2.     word = ''
3.     words = []
4.     strLen = getStringLength(strings)
5.     for char = 0, char < strLen, char++)
6.        if strings[char] != " "
7.           word = word + strings[char]
8.        else
9.           words.append(word)
10.          word = ''
11.    if word
12.       words.append(word)
13.    return words
```

Input:
Strings = “Kala itu”
Output:
[“kala”, “itu”]

Eksekusi

Baris-1	Function wordSplitting(‘Kala itu’)	
Baris-2	Word = ‘’	‘’
Baris-3	Words = []	[]
Baris-4	strLen = getStringLength(‘Kala itu’)	8
Baris-5	For (char = 0, 0 < 8, 1++)	
Baris-6	If ‘K’ != “ “	True
Baris-7	Word = ‘’ + ‘K’	‘K’
Baris-5	For (char = 1, 1 < 8, 2++)	
Baris-6	If ‘a’ != “ “	True
Baris-7	Word = ‘K’ + ‘a’	‘Ka’
Baris-5	For (char = 2, 2 < 8, 3++)	
Baris-6	If ‘l’ != “ “	True
Baris-7	Word = ‘Ka’ + ‘l’	‘Kal’
Baris-5	For (char = 3, 3 < 8, 4++)	
Baris-6	If ‘a’ != “ “	True
Baris-7	Word = ‘Kal’ + ‘a’	‘Kala’
Baris-5	For (char = 4, 4 < 8, 5++)	
Baris-6	If “ “ != “ “	False
Baris-8	Else	
Baris-9	Words.append(‘Kala’)	[“Kala”]
Baris-10	Word = ‘’	
Baris-5	For (char = 5, 5 < 8, 6++)	

Baris-6	`If “i“ != “ “`	`True`
Baris-7	`Word = ‘’ + ‘I’`	`‘I’`
Baris-5	`For (char = 6, 6 < 8, 7++)`	
Baris-6	`If ‘t’ != “ “`	`True`
Baris-7	`Word = ‘i’ + ‘t’`	`‘it’`
Baris-5	`For (char = 7, 7 < 8, 8++)`	
Baris-6	`If ‘u’ != “ “`	`True`
Baris-7	`Word = ‘it’ + ‘u’`	`‘itu’`
Baris-11	`If ‘itu’`	`True`
Baris-12	`Words.append(‘itu’)`	`[“Kala”, “itu”]`
Baris-13	`Return [“kala”, “itu”]`	

Latihan

Hitunglah jumlah kata
Buatlah tokenisasi untuk memecah atau memisah tiap kalimat.
Hitunglah berapa jumlah kalimat
Ambil token tiap kata dalam sebuah paragraf/kalimat

19.4.4 Pencarian (*searching*)

19.4.5 Penggantian (*replacement*)

19.4.6 Penggabungan (*joining*)

Pembacaan dan penulisan string dan dari dan ke file

Bab 20 Call Stack

Setiap kali suatu fungsi atau prosedur dipanggil, rangkaian informasi penting mengenai fungsi yang sedang dieksekusi tersebut akan akan dipush ke atas tumpukan (*stack*). Informasi tersebut bisa berupa alamat memori instruksi berikutnya yang akan dieksekusi setelah fungsi selesai, parameter yang dilewatkan ke fungsi, variabel lokal, dan beberapa informasi tambahan lainnya yang diperlukan untuk melacak eksekusi fungsi.

Karena menggunakan tumpukan, maka akan melibatkan push dan pop. Kedua proses ini beroperasi secara otomatis oleh mesin eksekusi program dan ini biasanya dilakukan dalam urutan LIFO (Last-In, First-Out). Artinya fungsi terakhir yang dimasukkan ke dalam tumpukan akan menjadi yang pertama kali dikeluarkan atau dihapus setelah dieksekusi.

Bab 21 Rekursif

21.1 Definisi Rekursif

21.2 Sejarah Rekursif

21.3 Konsep Dasar Rekursif

Secara prinsip dasar, algoritma rekursif menggunakan statemen kondisi `if` seperti berikut

```
if (basis terpenuhi)
   selesaikan
else
```

```
lakukan rekursi
```

Salah satu contoh kasus yang memiliki pola adalah faktorial. Misalnya 5! Berarti 5x4x3x2x1 = 120

Faktorial Menggunakan Iterasi For

```
1.  function faktorial(N)
2.     result = 0
3.     if N < 0
4.        result = "Input harus non-negatif"
5.     if N == 0 or N == 1
6.        result = 1
7.     if N > 1
8.        result = 1
9.        for (i=2, i <= N, i++)
10.          result = result * i
11.    return result
```

Input: 5
Output: 120

Eksekusi

Baris-1	Function faktorial(5)	
Baris-2	Result = 0	0
Baris-3	If 5 < 0	False
Baris-5	If 5 == 0 or 5 == 1	False or False
Baris-7	If 5 > 1	True
Baris-8	Result = 1	1
Baris-9	For (I = 2, 2 <= 5, 3++)	True
Baris-10	Result = 1 * 2	2
Baris-9	For (I = 3, 3 <= 5, 4++)	true
Baris-10	Result = 2 * 3	6
Baris-9	For (i = 4, 4 <= 5, 5++)	true
Baris-10	Result = 6 * 4	24
Baris-9	For (i = 5, 5 <= 5, 6++)	true
Baris-10	Result = 24 * 5	120
Baris-9	For (i = 6, 6 <= 5, 7++)	false
Baris-11	Return 120	120

Faktorial menggunakan rekursif sederhana (linear)

```
1.  function faktorial(N)
2.     if (N == 0) or (N < 0)
3.        return 1
4.     if N > 0
5.        return N * faktorial(N-1)
```

Input: 5
Output: 120

Eksekusi

Baris-1	Function faktorial(5)	
Baris-2	If (5 == 0) or (5 < 0)	False or false
Baris-4	If 5 > 0	True
Baris-5	Return 5 * faktorial(5-1)	Faktorial(4)
Baris-1	Function faktorial(4)	
Baris-2	If (4 == 0) or (4 < 0)	False or false
Baris-4	If 4 > 0	True
Baris-5	Return 4 * faktorial(4-1)	Faktorial(3)
Baris-1	Function faktorial(3)	
Baris-2	If (3 == 0) or (3 < 0)	False or false
Baris-4	If 3 > 0	True
Baris-5	Return 3 * faktorial(3-1)	Faktorial(2)
Baris-1	Function faktorial(2)	
Baris-2	If (2 == 0) or (2 < 0)	False or false
Baris-4	If 2 > 0	True
Baris-5	Return 2 * faktorial(2-1)	Faktorial(1)
Baris-1	Function faktorial(1)	
Baris-2	If (1 == 0) or (1 < 0)	False or false
Baris-4	If 1 > 0	True
Baris-5	Return 1 * faktorial(1-1)	Faktorial(0)
Baris-1	Function faktorial(0)	
Baris-2	If (0 == 0) or (0 < 0)	True or false
Baris-3	Return 1	1
Baris-5	Return 1 * 1	1
Baris-5	Return 2 * 1	2
Baris-5	Return 3 * 2	6
Baris-5	Return 4 * 6	24
Baris-5	Return 5 * 24	120

Faktor Persekutuan Terbesar Tanpa Rekursif

```
1.  function gcd(A, B)
2.     while B != 0
3.        temp = A
4.        A = B
5.        B = temp mod B
6.     return A
```

Input: A = 24, B = 36
Output: 12

Eksekusi

Baris-1	Function gcd(24, 36)	
Baris-2	While 36 != 0	true
Baris-3	Temp = 24	24
Baris-4	A = 36	36
Baris-5	B = 24 mod 36	24
Baris-2	While 24 != 0	true
Baris-3	Temp = 36	36

Baris-4	A = 24	24
Baris-5	B = 36 mod 24	12
Baris-2	While 12 != 0	true
Baris-3	Temp = 24	24
Baris-4	A = 12	12
Baris-5	B = 24 mod 12	0
Baris-2	While 0 != 0	False
Baris-6	Return 12	12

Faktor Persekutuan Terbesar Menggunakan Rekursif

```
1.  function gcd(A, B)
2.     if B == 0
3.        return A
4.     else
5.        return gcd(B, A % B)
```

Input: A = 24, B = 36
Output: 12

Eksekusi

Baris-1	Function gcd(24, 36)	
Baris-2	If 36 == 0	false
Baris-4	Else	True
Baris-5	Return gcd(36, 24 mod 36)	Gcd(36, 24)
Baris-1	Function gcd(36, 24)	
Baris-2	If 24 == 0	false
Baris-4	Else	True
Baris-5	Return gcd(24, 36 mod 24)	Gcd(24, 12)
Baris-1	Function gcd(24, 12)	
Baris-2	If 12 == 0	false
Baris-4	Else	True
Baris-5	Return gcd(12, 24 mod 12)	Gcd(12, 0)
Baris-1	Function gcd(12, 0)	
Baris-2	If 0 == 0	True
Baris-3	Return 12	12
Baris-7	Return gcd(12, 0)	12
Baris-7	Return gcd(24, 12)	12
Baris-7	Return gcd(36, 24)	12

Latihan Soal

1. Buatlah algoritma (*pseudocode*) agar bisa terjadi pertukaran nilai seperti berikut
   $a \rightarrow b \rightarrow c \rightarrow d \rightarrow a$
   Bila dibaca, *b* diisi nilai *a*, *c* diisi nilai *b*, *d* diisi nilai *c*, dan *a* diisi nilai *d*.
   Contoh input: [5, 3, 10, 6]
   Contoh output: [6, 5, 3, 10]
2. Buatlah algoritma (*pseudocode*) agar bisa terjadi pertukaran nilai seperti berikut
   $a \leftarrow b \leftarrow c \leftarrow d \leftarrow a$
   Bila dibaca, *a* diisi nilai *b*, *b* diisi nilai *c*, *c* diisi nilai *d*, dan *d* diisi nilai *a*.
   Contoh input: [5, 3, 10, 6]
   Contoh output: [3, 10, 6, 5]
3. Buatlah algoritma untuk menukarkan nilai variabel *a* dan *b* tanpa menggunakan variabel sementara (*temporary*)
4.