KONSEP DASAR ALGORITMA PEMROGRAMAN

Buku ini merupakan luaran penelitian dengan judul " Pengembangan Media Pembelajaran Penalaran Matematika Pada Pembuatan Algoritma Komputasi Berbasis Gaya Belajar Dan Pendidikan Karakter di POLTEK Surabaya". Dibiayai oleh DRPM Kemenristek dikti Sesuai dengan Kontrak Penelitian Tahun Anggaran 2019 Nomor:113/SP2H/LT/DRPM/019 Tanggal 11 Maret 2019. Setiap bab mengulas tentang penggunaan pernyataan matematika, intruksi komputer, serta logika matematika untuk pembuatan algoritma komputasi. Sangat direkomendasikan sebagai pendamping belajar Algoritma Pemrograman. Khususnya untuk meningkatkan kemampuan penalaran matematis, keterampilan belajar mandiri, penyelesaian masalah, berpikir kritis, serta kreatif peserta didik. Buku ini juga direkomendasikan untuk Pendidik yang mayoritas peserta didiknya belum trampil dalam bernalar secara matematis.

Sulis Janu Hartati, adalah Dosen FKIP Universitas Dr. Soetomo Surabaya. Mengampu mata kuliah Logika dan Algoritma mulai tahun 1998 – 2014 di STIKOM Surabaya. Karya publikasinya dapat dilihat di url: https://scholar.google.co.id/citations?user=tPLOvk0AAAAJ&hl=id.

Neny Kurniati, mengajar Algoritma di Poltek Surabaya sejak tahun 1999. Mulia tahun 2018, mengajar Algoritma Pemrograman di Fakultas Ilmu Komputer UNUSIDA Sidoarjo.

Anik Vega Vitianingsih, menjadi Dosen Algoritma Pemrograman di FTI Universitas Dr. Soetomo Surabaya sejak tahun 2009. Karya publikasinya dapat dilihat di url: https://scholar.google.co.id/citations?user=acUhDoIAAAAJ&hl=id.

Endra Rahmawati, adalah Dosen di Fakultas Teknologi dan Informatika Universitas Dinamika Surabaya. Membina mata kuliah Logika dan Algoritma sejak tahun 2012. Karya publikasinya dapat dilihat di url: https://scholar.google.com/citations?user=iy7kn4UAAAAJ&hl=en&authuser=1.

Norma Ningsih, adalah Dosen Pemrograman Dasar di Fakultas Teknologi dan Informatika Universitas Dinamika Surabaya, sejak tahun 2016. Karya publikasinya dapat dilihat di url: https://scholar.google.com/citations?hl=en&authuser=1&user=21JhTZ8AAAAJ.

ISBN: 978-623-228-850-8

GRAHA ILMU

Sulis Janu Hartati; dkk

KONSEP DASAR ALGORITMA PEMROGRAMAN

GRAHA ILMU

KONSEP DASAR ALGORITMA PEMROGRAMAN

Sulis Janu Hartati
Neny Kurniati
Anik Vega Vitianingsih
Endra Rahmawati
Norma Ningsih

Pembaca Buku yang tercinta.

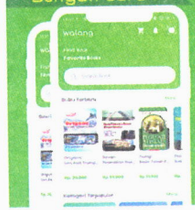
Terimakasih kami ucapkan atas pilihan Anda membeli buku asli terbitan kami. Sebagai penghargaan kami atas kebaikan Anda, kami memberikan kesempatan terbatas memiliki buku digital dari buku kertas yang Anda beli ini.

Ketentuannya sbb:

1. Buku digital ini hanya bisa dibaca di gadget Anda tidak bisa di share ke orang lain.
2. Buku digital ini hanya bisa dibaca menggunakan apps **WALANG**.
   Cara download dan cara mengunduh filenya ikuti petunjuk.
3. Mohon perhatian, ada batasan waktu pengunduhan secara gratis.
   Selebih dari tanggal tersebut file tak bisa lagi diunduh secara gratis.
4. Jangan berikan Kode Voucher tersebut ke orang lain. Satu kode voucher hanya berlaku 1 kali unduh dan tidak bisa dipindahkan ke gadget lainnya.
5. Silakan mencobanya dan menikmati **membaca secara digital** buku koleksi Anda.

1. UNDUH APLIKASI

Dengan cara :



1. Buka Playstore, Ketik pencarian " Walang "
2. jika tidak ditemukan ketik lengkap " Walang PT Enam Kubuku Indonesia "
3. Install Aplikasi
4. Lakukan Registrasi

Logo Aplikasi Walang

walang

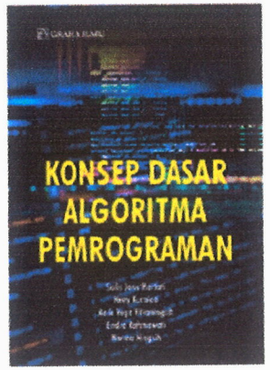
GET IT ON Google Play

2. GUNAKAN KODE VOUCHER

Dengan cara :

1. Pilih menu scan dikanan atas
2. Scan QR Code Voucher atau ketik Kode Voucher yang anda dapatkan pada Tombol Scan
3. Pilih tombol TUKARKAN untuk mendapatkan buku digital gratis

INFORMASI
0812-2976-5166



Kode Voucher
1ed2d66cffdc

*Kode terbatas untuk 1 pengguna.

*Berlaku sampai 12 April 2021.

KONSEP DASAR ALGORITMA PEMROGRAMAN

KONSEP DASAR ALGORITMA PEMROGRAMAN

Sulis Janu Hartati
Neny Kurniati
Anik Vega Vitianingsih
Endra Rahmawati
Norma Ningsih

GRAHA ILMU

KONSEP DASAR ALGORITMA PEMROGRAMAN

oleh Sulis Janu Hartati, Neny Kurniati, Anik Vega Vitianingsih, Endra Rahmawati, Norma Ningsih

Hak Cipta © 2021 pada penulis

Edisi Pertama; Cetakan Pertama ~ 2021

GRAHA ILMU

Ruko Jambusari 7A Yogyakarta 55283
Telp: 0274-889398; 0274-882262; email: info@grahailmu.co.id

Hak Cipta dilindungi undang-undang. Dilarang memperbanyak atau memindahkan sebagian atau seluruh isi buku ini dalam bentuk apa pun, secara elektronis maupun mekanis, termasuk memfotokopi, merekam, atau dengan teknik perekaman lainnya, tanpa izin tertulis dari penerbit.

ISBN: 978-623-228-850-8

Buku ini tersedia sumber elektronisnya

DATA BUKU:
Format: 17 x 24 cm; Jml. Hal.: xxii + 266; Kertas Isi: HVS 70 gram; Tinta Isi: BW/Colour; Kertas Cover: Ivori 260 gram; Tinta Cover: Colour; Finishing: Perfect Binding: Laminasi Doff.

KATA PENGANTAR

Syukur *Alhamdullilah* kami panjatkan kehadirat Tuhan Yang Maha Esa atas bimbingan dan ridlo-Nya, sehingga TIM kami dapat menuntaskan penulisan buku dengan judul *"Konsep Dasar Algoritma Pemrograman"*. Buku ini disusun sebagai pengantar siswa atau mahasiswa yang akan belajar Algoritma Pemrograman. Namun demikian, buku ini juga bisa dijadikan suplemen bagi siswa atau mahasiswa yang sedang mengambil mata kuliah Algoritma Pemrograman atau sejenisnya. Khususnya bagi siswa atau mahasiswa dengan kemampuan matematika di bawah 60 namun mempunyai keinginan kuat atau harus belajar algoritma pemrograman.

Buku ini disusun sebagai pelengkapan media pembelajaran Penalaran Matematika pada Pembuatan Algoritma Pemrograman yang diunggah pada alamat https://penalaran.com. Oleh karena itu, disarankan kepada pemakai buku ini untuk berkunjung ke alamat tersebut supaya mendapat hasil belajar yang maksimal. Kebanyakan buku Algoritma Pemrograman tidak membahas dasar penalaran matematika, yang meliputi membuat pernyataan matematika dan logika matematika. Oleh karena itu, buku ini layak dibaca bagi siapapun yang belajar Algoritma namun belum trampil melakukan penalaran matematika yang merupakan kebutuhan dalam pembuatan algoritma pemrograman. Buku ini juga direkomendasikan bagi Dosen atau

Pendidik yang mengajar Algoritma Pemrograman, namun mayoritas peserta didik belum trampil dalam bernalar secara matematis.

Buku ini terdiri dari 7 BAB, yang semuanya merupakan kemampuan dasar yang seharusnya dimiliki oleh setiap siswa atau mahasiswa yang akan belajar algoritma pemrogaman. Adapun sistematika pembabanya adalah: BAB 1, membahas tentang pentingnya memahami paradigm berpikir komputasi sebelum belajar algoritma pemrograman. Selain itu, pada BAB 1 juga juga diberikan ulasan keterkaitan antara berpikir komputasi dengan komponen terkecil dalam pengolahan data. Kemudian dilanjutkan dengan BAB 2, membahas tentang pernyataan matematika dan logika matika, serta kaitannya dengan berpikir komputasi dan pengolahan data. Sedang BAB 3 sampai dengan BAB 7 sudah mulai masuk pada diskusi tentang dasar-dasar pembuatan algoritma pemrograman. Diskusi dimulai dengan pengenalan konsep variabel dan data, serta keterkaitannya dengan pernyataan matematika dan logika matematika. Kemudian dilanjutkan dengan konsep dasar proses pengolahan data meliputi percabangan dan perulangan. Kemudian diakhiri dengan pengantar pembuatan algoritma pemrograman dalam bentuk *pseudocode* dan *flowchart*.

Kami menyadari bahwa buku ini masih banyak kekurangan. Untuk itu, pada kesempatan ini kami membuka diri atas masukan yang membangun guna meningkatkan kebermanfaatannya. Tak lupa kami sampaikan penghargaan dan ucapan terima kasih kepada TIM, baik tim penyusun buku maupun tim penelitian. Ucapan terima kasih juga kepada para dosen mata kuliah algoritma pemrograman di Politeknik Surabaya selaku mitra dalam penelitian, Universitas Dr. Soetomo Surabaya, Universitas Muhammadiyah Sidoardjo, Universitas NU Sidoarjo, Institut Bisnis dan Informatika Stikom Surabaya, Universitas Merdeka Pasuruan, serta STMIK YADIKA Bangil Pasuruan. Semoga dukungan semua tim dapat menjadi amal jariyah tim penyusun.

Akhir kata, semoga dokumen ini bermanfaat untuk peningkatan kualitas proses pembelajaran mata kuliah algoritma pemrograman, Aamiin.

Surabaya, November 2019

TIM Penyusun

DAFTAR ISI

KATA PENGANTAR		v
DAFTAR ISI		ix
DAFTAR GAMBAR		xiii
DAFTAR TABEL		xvii
BAB 1	PARADIGMA BERPIKIR KOMPUTASI: KOMPONEN PENGOLAHAN DATA	1
	1.1 Pendahuluan	2
	1.2 Komponen Masukan	12
	1.3 Komponen Proses	18
	1.4 Komponen Keluaran	25
	1.5 Pengguna	29
	1.6 Rangkuman	29
	1.7 Latihan Soal	30
BAB 2	PERNYATAAN MATEMATIKA DAN LOGIKA MATEMATIKA	35
	2.1 Pendahuluan	36
	2.2 Pernyataan Matematika	44
	2.3 Logika Matematika	59
	2.4 Rangkuman	68
	2.5 Latihan Soal	69

BAB 3 VARIABEL DAN DATA 75
3.1 Pendahuluan 76
3.2 Variabel dan Pernyataan Matematika 81
3.3 Variabel dan Logika Matematika 86
3.4 Variabel dan Instruksi Komputasi 90
3.5 Variabel dan Tipe Data 98
3.6 Identifikasi Variabel Berdasarkan Komponen Pengolahan Data 100
3.7 Rangkuman 108
3.8 Latihan Soal 109

BAB 4 PERCABANGAN 115
4.1 Pendahuluan 116
4.2 Percabangan dan Pernyataan Matematika 120
4.3 Percabangan dan Logika Matematika 122
4.4 Percabangan dan Instruksi Komputasi 126
4.5 Percabangan dan Variabel 131
4.6 Klasifikasi Percabangan pada Pengolahan Data 132
4.7 Rangkuman 140
4.8 Latihan Soal 141

BAB 5 PERULANGAN 1 145
5.1 Pendahuluan 146
5.2 Perulangan dan Pernyataan Matematika 149
5.3 Perulangan dan Logika Matematika 154
5.4 Perulangan dan Instruksi Komputasi 157
5.5 Perulangan dan Variabel 162
5.6 Perulangan dan Percabangan 165
5.7 Rangkuman 170
5.8 Latihan Soal 171

BAB 6 PERULANGAN 2 177
6.1 Pendahuluan 178
6.2 Jenis Perulangan 180
6.3 Masalah Perulangan Sederhana 184
6.4 Akumulator 184

6.5 Multiplikator 190
6.6 Faktorial 196
6.7 Rangkuman 201
6.8 Latihan Soal 201

BAB 7 MEMBUAT *PSEUDOCODE* DARI *FLOWCHART* DAN VARIABEL ARRAY 1 DIMENSI 209
7.1 Pendahuluan 210
7.2 Penyajian Instruksi dalam Komputasi 213
7.3 Pengenalan *Pseudocode* 219
7.4 Pengenalan *Flowchart* 231
7.5 Variabel Array 1 Dimensi 238
7.6 Rangkuman 241
7.7 Latihan Soal 241

DAFTAR PUSTAKA 247
GLOSARIUM 251
INDEKS 261
TENTANG PENULIS 263

-oo0oo-

DAFTAR GAMBAR

Gambar 1.1	Contoh Pengolahan Data Sederhana	7
Gambar 1.2	Contoh Komponen Masukan dan Keluaran	8
Gambar 1.3	Contoh Keluaran Hasil Proses pada Gambar 1.2	8
Gambar 1.4	Elemen Terkecil Pengolahan Data	9
Gambar 1.5	Pseudo Code Transformasi dari Gambar 1.2. Menjadi Gambar 1.3.	10
Gambar 1.6	Contoh Persamaan dan Ketaksamaan pada Algoritma	11
Gambar 1.7	Tampilan Awal Aplikasi penalaran.com (Komponen Masukan)	12
Gambar 1.8	Perubahan Memasukan *USERNAME* dan *PASSWORD* Salah atau Belum Register	14
Gambar 1.9	Perubahan Memasukan *USERNAME* dan *PASSWORD* Setelah Registrasi	15
Gambar 1.10	Perubahan yang Terjadi untuk Data Masukan dengan Kondisi Nama, user_name, dan password Belum Pernah Ada.	16
Gambar 1.11	Tampilan untuk NAMA Belum Pernah Ada, namun *USERNAME* dan atau *PASSWORD* Sama dengan Data Sebelumnya	16
Gambar 1.12	Tampilan untuk NAMA Sama dengan Data Sebelumnya, namun *USERNAME* dan *PASSWORD* yang Belum Pernah Ada	17

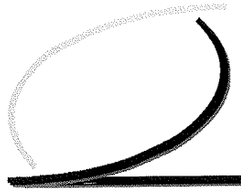
Gambar 1.13	Tampilan untuk NAMA, *USERNAME* dan *PASSWORD* Sama dengan Data Sebelumnya	17
Gambar 1.14	Perubahan Tampian Awal Aplikasi penalaran.com Setelah Pengguna Memasukan *Username* dan *Password* Namun Belum Registrasi atau *Username* atau dan *Password* yang Dimasukan Salah	18
Gambar 1.15	Perubahan Gambar 1.7 Menjadi Gambar 1.9	19
Gambar 1.16	Tampilan Awal Aplikasi penalaran.com Sebelum dan Setelah Memilih Fungsi REGISTER	21
Gambar 1.17	Belajar Proses Membuat Kounter Dasar-2 pada Aplikasi penalaran.com	23
Gambar 1.18	Proses Perubahan dari Gambar 1.17.b menjadi 1.17.c.	23
Gambar 1.19	Contoh Keluaran dari Aplikasi Presensi Mahasiswa	26
Gambar 1.20	Perubahan Gambar 1.7 Menjadi Gambar 1.9	26
Gambar 2.1	Contoh Posisi A dan B pada Diagram Kartesius	38
Gambar 2.2	Contoh *Pseudocode*	40
Gambar 2.3	Materi Pernyataan Matematika pada penalaran.com	45
Gambar 2.4	Pseudo Code Aritmetika Dasar	47
Gambar 2.5	Pseudo Code Aplikasi Aritmetika	51
Gambar 2.6	*Pseudocode* untuk Menghitung Luas Lingkaran dan Akar Persamaan Kuadrat	55
Gambar 2.7	Operasi Pembagian	61
Gambar 2.8	Kemungkinan Hasil Pasangan (TAMPIL=TRUE dan BIL2!=0)	64
Gambar 2.9	Operasi Penjumlahan Otomatis	65
Gambar 2.10	Kemungkinan Hasil Logika Matematika TAMPIL=TRUE	67
Gambar 3.1	Tampilan Menu Utama penalaran.com	76
Gambar 3.2	Menu Utama penalaran.com	77
Gambar 3.3	Menu Aritmetika Dasar Penjumlahan	78
Gambar 3.4	Tampilan Operasi Aritmetika Dasar Penjumlahan	78
Gambar 3.5	Masukan Operasi Aritmetika Dasar Penjumlahan	78
Gambar 3.6	Keluaran Operasi Aritmetika Dasar Penjumlahan	79
Gambar 3.7	Tampilan Hasil Operasi Penjumlahan	79

Gambar 3.8	Menu Aplikasi Aritmetika Submenu Menghitung Luas Lingkaran	80
Gambar 3.9	Tampilan Operasi Hitung Luas Lingkaran	80
Gambar 3.10	Masukan Operasi Menghitung Luas Lingkaran. Hasil luas Lingkaran Akan Tampil pada Bagian Keluaran Seperti Gambar 3.11.	80
Gambar 3.11	Tampilan Keluaran Operasi Menghitung Luas Lingkaran	80
Gambar 3.12	Contoh Operasi Menghitung Luas Lingkaran	81
Gambar 3.13	Contoh Pernyataan Matematika	82
Gambar 3.14	Variabel dalam Operasi Penjumlahan	83
Gambar 3.15	Tampilan Data dalam Operasi Penjumlahan	84
Gambar 3.16	Tampilan Operator dalam Operasi Penjumlahan	84
Gambar 3.17	Variabel pada Operasi Menghitung Luas Lingkaran	85
Gambar 3.18	Data pada Operasi Luas Lingkaran	86
Gambar 3.19	Tampilan Menu Logika Dasar	88
Gambar 3.20	Masukan, Keluaran, dan Syarat/Kondisi pada Operasi Logika Dasar	89
Gambar 3.21	Hasil Operasi Logika Dasar untuk Nilai Masukan 0 Sekarang Lihat Gambar 3.22	89
Gambar 3.22	Hasil Operasi Logika Dasar untuk Nilai Masukan 3	90
Gambar 3.23	Tampilan Keluaran pada Operasi Penjumlahan	92
Gambar 3.24	*Flowchart* Operasi Penjumlahan	92
Gambar 3.25	*Pseudocode* Operasi Penjumlahan	93
Gambar 3.26	Kode Program Operasi Penjumlahan	93
Gambar 3.27	Operator Proses Menghitung Luas Lingkaran	95
Gambar 3.28	*Flowchart* Operasi Menghitung Luas Lingkaran	95
Gambar 3.29	*Pseudocode* Operasi Menghitung Luas Lingkaran	96
Gambar 3.30	Kode Program Operasi Menghitung Luas Lingkaran	96
Gambar 3.31	Masukan Operasi Menghitung Rata-rata Dua Variabel	101
Gambar 3.32	Contoh Operasi Menghitung Rata-rata Dua Variabel	101
Gambar 3.33	Keluaran Operasi Menghitung Rata-rata Dua Variabel	101
Gambar 3.34	Variabel pada Operasi Menghitung Rata-rata Dua Variabel	102
Gambar 3.35	Data pada Operasi Menghitung Rata-rata Dua Variabel	103

Gambar 3.36	Konstanta pada Operasi Menghitung Rata-Rata Dua Variabel	103
Gambar 3.37	Operator pada Operasi Menghitung Rata-rata Dua Variabel	104
Gambar 3.38	Penalaran pada Operasi Menghitung Rata-rata Dua Variabel	104
Gambar 3.39	*Flowchart* Operasi Menghitung Rata-rata Dua Variabel	105
Gambar 3.40	*Pseudocode* Operasi Menghitung Rata-rata Dua Variabel	106
Gambar 3.41	Kode Program Operasi Menghitung Rata-rata Dua Variabel	107
Gambar 4.1	Simbol *Flowchart* Percabangan	116
Gambar 4.2	Simbol *Pseudocode* Percabangan	117
Gambar 4.3	Percabangan Bentuk Tunggal	117
Gambar 4.4	Percabangan Bentuk Ganda	117
Gambar 4.5	Percabangan Bentuk Jamak	118
Gambar 4.6	Menu Operasi Aritmetika Pembagian	119
Gambar 4.7	Masukan Operasi Aritmetika Pembagian	119
Gambar 4.8	Keluaran Operasi Aritmetika Pembagian	119
Gambar 4.9	Operasi Deteksi Bilangan	120
Gambar 4.10	Variabel Operasi Aritmetika Pembagian	122
Gambar 4.11	Variabel dalam Operasi Logika Dasar	123
Gambar 4.12	Penalaran Operasi Logika Dasar	124
Gambar 4.13	Logika Matematika untuk Deteksi Bilangan Genap/ Ganjil Tabel 4.2 Contoh Operasi Deteksi Bilangan Genap/Ganjil	125
Gambar 4.14	*Flowchart* Operasi Aritmetika Pembagian	127
Gambar 4.15	*Pseudocode* Operasi Aritmetika Pembagian	127
Gambar 4.16	Kode Program Operasi Aritmetika Pembagian	128
Gambar 4.17	*Flowchart* Deteksi Bilangan Genap/Ganjil	130
Gambar 4.18	*Pseudocode* Deteksi Bilangan Genap/Ganjil	131
Gambar 4.19	Kode Program Deteksi Bilangan Ganjil/Genap	131
Gambar 4.20	Operasi Kombinasi Dua Buah Relasi	133

Gambar 4.21	Masukan Kombinasi Dua Buah Relasi	133
Gambar 4.22	Keluaran Kombinasi Dua Variabel	133
Gambar 4.23	Variabel Kombinasi Dua Buah Relasi	134
Gambar 4.24	Data Kombinasi Dua Buah Relasi	134
Gambar 4.25	Operator Kombinasi Dua Buah Relasi	135
Gambar 4.26	Operasi Relasi pada Kombinasi Dua Buah Relasi	135
Gambar 4.27	Penalaran Kombinasi Dua Buah Relasi	136
Gambar 4.28	*Flowchart* Kombinasi Dua Buah Relasi	137
Gambar 4.29	*Pseudocode* Kombinasi Dua Buah Relasi	138
Gambar 4.30	Kode Program dan Keluaran Kombinasi Dua Buah Relasi	138
Gambar 5.1	*Flowchart* Dasar Perulangan For	149
Gambar 5.2	Hasil Membuat Kounter Dasar di Penalaran.com	150
Gambar 5.3	Penggunaan Variabel pada Menu Kounter Dasar 1.	150
Gambar 5.4	Model Penalaran Pembuatan Kounter Dasar 1.	151
Gambar 5.5	Tabel Kebenaran Pembuatan Kounter Dasar 1.	151
Gambar 5.6	*Flowchart* Pembuatan Kounter Dasar 1.	152
Gambar 5.7	Contoh Logika Matematika	154
Gambar 5.8	Menu Konsep Perulangan	155
Gambar 5.9	*Flowchart* Konsep Perulangan pada Menu Logika Matematika	155
Gambar 5.10	Menu Membuat Akumulator pada Penalaran.com (1)	158
Gambar 5.11	Menu Membuat Akumulator pada Penalaran.com (2)	158
Gambar 5.12	Instruksi Komputasi pada Akumulator	159
Gambar 5.13	Menu Multiplikator pada Penalaran.com (1)	159
Gambar 5.14	Menu Multiplikator pada Penalaran.com (2)	161
Gambar 5.15	Kebutuhan Variabel pada Menu Pembuatan Akumulator pada Penalaran.com	163
Gambar 5.16	Jenis Data yang Digunakan pada Akumulator Menu Penalaran.com.	163
Gambar 5.17	Kebutuhan Variabel pada Menu Pembuatan Multiplikator pada Penalaran.com	164
Gambar 5.18	Jenis Data yang digunakan pada Akumulator Menu Penalaran.com.	165

Gambar 5.19	(A) *Flowchart* Alur Percabangan (B) *Flowchart* Alur Perulangan.	166
Gambar 5.20	Contoh *Flowchart* Percabangan	167
Gambar 5.21	Contoh *Flowchart* Perulangan	169
Gambar 6.1	*Flowchart* Proses Perulangan.	178
Gambar 6.2	*Flowchart* Mencetak Kata "Indonesia Raya" Sebanyak 5 kali.	179
Gambar 6.3	*Flowchart* Mencetak Deret Angka 1-10 secara Ascending (FOR).	180
Gambar 6.4	*Flowchart* mencetak deret angka 1-10 secara Ascending (DO WHILE)	181
Gambar 6.5	*Flowchart* Mencetak Deret Angka 1-10 secara Ascending (WHILE).	182
Gambar 6.6	*Flowchart* Mencetak Deret Perkalian Angka 1-5 Secara Urut	183
Gambar 6.7	Menu Membuat Akumulator pada Penalaran.com	186
Gambar 6.8	*Flowchart* Menu Akumulasi Penjumlahan (Akumulator)	187
Gambar 6.9	Menu Multiplikator pada Penalaran.com	191
Gambar 6.10	*Flowchart* Menu Akumulasi Perkalian (Multiplikator)	193
Gambar 6.11	Menu Faktorial pada Penalaran.com	197
Gambar 6.12	*Flowchart* Faktorial pada Penalaran.com.	198
Gambar 7.1	Indeks pada Array	211
Gambar 7.2	Deklarasi Array	212
Gambar 7.3	Inisialisasi Aray	212
Gambar 7.4	Input Elemen Array	213
Gambar 7.5	Inputan Matriks Penjumlahan	214
Gambar 7.6	*Output* Matriks Penjumlahan	214
Gambar 7.7	Hasil Matriks Penjumlahan	215
Gambar 7.8	Data pada Matriks Pnjumlahan	215
Gambar 7.9	Operator Pada Matriks Penjumlahan	216
Gambar 7.10	Hasil Pengurangan Matriks	216
Gambar 7.11	Operator pada Matriks Pengurangan.	217
Gambar 7.12	Masukan Perulangan Array	217
Gambar 7.13	Keluaran Perulangan Array	217



DAFTAR TABEL

Tabel 1.1	Proses Perubahan Gambar 1.7 Menjadi Gambar 1.8	19
Tabel 1.2	Proses Perubahan Gambar 1.7 Menjadi Gambar 1.8 dan Gambar 1.9	20
Tabel 1.3	Proses Perubahan Gambar 1.7 Menjadi Gambar 1.8 dan Gambar 1.9 dengan Melibatkan Fungsi LOGIN	21
Tabel 1.4	Proses Perubahan Gambar 1.16.a. Menjadi Gambar 1.16.b. dengan Melibatkan Fungsi REGISTER	22
Tabel 2.1	Penggunaan Pernyataan Matematika, Logika Matematika, serta Perintah atau Instruksi pada Algoritma	40
Tabel 2.2	Contoh Penggunaan Operasi Aritmetika Dasar	46
Tabel 2.3	Aturan Tabel Kebenaran	60
Tabel 2.4	Contoh Kemungkinan Nilai Logika Matematika Gambar 2.7(a)	63
Tabel 3.1	Prioritas Penggunaan Operator Aritmatika	82
Tabel 3.2	Operator Relasi	87
Tabel 3.3	Operator Logika	87
Tabel 3.4	Contoh Operasi Logika Matematika	88
Tabel 3.5	Tabel Kebenaran Penalaran Sebab Akibat	105
Tabel 4.1	Syarat dalam Operasi Logika Dasar	124
Tabel 4.2	Operator Pembagian	132

-oo0oo-

Gambar 7.14 Data pada Perulangan Array 218
Gambar 7.15 Operator pada Perulangan Array 218
Gambar 7.16 Penalaran pada Perulangan Array 218
Gambar 7.17 *Flowchart* Penjumlahan Matriks Bagian 1 231
Gambar 7.18 *Flowchart* Penjumlahan Matriks Bagian 2 232
Gambar 7.19 *Flowchart* Penjumlahan Matriks Bagian 3 233
Gambar 7.20 *Flowchart* Penjumlahan Matriks Bagian 4 233
Gambar 7.21 *Flowchart* Pengurangan Matriks Bagian 1 234
Gambar 7.22 *Flowchart* Pengurangan Matriks Bagian 2 235
Gambar 7.23 *Flowchart* Pengurangan Matriks Bagian 3 236
Gambar 7.24 *Flowchart* Pengurangan Matriks Bagian 4 237
Gambar 7.25 *Flowchart* Perulangan Array 1 Dimensi 238
Gambar 7.26 Variabel Penjumlahan Matriks 239
Gambar 7.27 Variabel Pengurangan Matriks 240
Gambar 7.28 Variabel Perulangan Array 1 Dimensi 240

-oo0oo-

BAB 1

PARADIGMA BERPIKIR KOMPUTASI: KOMPONEN PENGOLAHAN DATA

Deskripsi Singkat

Bab 1 menjelaskan pentingnya kemampuan berpikir komputasi bagi semua orang dalam menangani permasalahan atau menyelesaikan tugas rutin di era digital saat ini dan beberapa tahun ke depan. Diskusi ini sejalan dengan Kurikulum 2013, yang memasukan materi berpikir komputasi pada siswa sekolah menengah. Selain itu, sejalan juga dengan Kepmendikbud tahun 2018, yang memutuskan untuk melakukan implementasi kurikulum 2013 di semua sekolah menengah di Indonesia.

Diskusi dimulai dengan pendahuluan, kemudian berturut-turut komponen masukan, komponen proses, komponen keluaran, serta komponen pengguna. Sebelum akhir bab ada rangkuman dan latihan.

Pendahuluan, diawali dengan beberapa contoh peralatan yang menggunakan konsep berpikir komputasi dalam kehidupan sehari-hari beserta cara penggunaannya. Baik peralatan yang menggunakan mesin digital maupun tidak. Kemudian dilanjutkan dengan uraian tentang definisi berpikir komputasi dari beberapa pakar. Berdasarkan uraian tersebut, pembaca digiring untuk menemukan keterkaitan antara berpikir komputasi dengan pengolahan data secara komputasi dan peranan penalaran matematika pada keduanya.

Kemudian dilanjutkan dengan berpikir komputasi dalam setiap komponen pengolahan data, yang meliputi: masukan, proses, dan keluaran. Dalam diskusi ini ditambahkan satu komponen lagi yaitu pengguna. Artinya, pembaca juga harus mempertimbangkan bahwa hasil akhir dari berpikir komputasi maupun pengolahan data akan digunakan oleh manusia dalam menyelesaikan masalah atau menyelesaikan tugas rutinnya.

Tujuan Pembelajaran

Akhir pembelajaran atau diskusi, diharapkan pembaca mempunyai beberapa keterampilan berikut ini.

1. Jika diberikan soal beserta jawabannya, pembaca dapat memilih jawaban yang diselesaikan menggunakan paradigm berpikir komputasi dan bukan.
2. Jika diberikan soal beserta jawabannya, pembaca dapat memilih jawaban yang diselesaikan menggunakan paradigm pengolahan data secara komputasi dan bukan.
3. Jika diberikan soal beserta jawabannya, pembaca dapat menyebutkan dengan benar komponen masukan, proses, dan keluaran dari jawaban yang diberikan.

1.1 Pendahuluan

Penggunaan perangkat lunak mulai menjamur dalam kehidupan sehari-hari. Sebagai contoh, untuk belanja kebutuhan sehari-hari, seperti beli kue, es krim, bahkan bumbu dapur bisa dilakukan menggunakan aplikasi perangkat lunak. Berangkat sekolah, kerja, bahkan jika ada berkas yang ketinggalanpun dapat diselesaikan dengan aplikasi perangkat lunak. Mengantar makanan ke kerabat juga bisa dilakukan dengan aplikasi perangkat lunak. Beberapa contoh tersebut menunjukkan bahwa aplikasi perangkat lunak dapat digunakan untuk mempermudah atau membantu pekerjaan manusia. Sekaligus, dengan perangkat tersebut dapat membuka lapangan pekerjaan yang bersifat kolaborasi. Contoh-contoh tersebut merupakan contoh-contoh peralatan yang dibuat

berdasarkan kemampuan berpikir komputasi menggunakan mesin komputer, yaitu mesin pengolahan data berbasis digital.

Peralatan lain yang banyak dijumpai untuk memudahkan atau membantu pekerjaan manusia diantaranya adalah: mesin cuci, setrika listrik, pemanas air, kompor, penyedot debu, kendaraan bermotor, penanak dan pemanas nasi, alat pendingin, lemari es, dan lain- lain peralatan yang meringankan pekerjaan manusia. Peralatan tersebut juga termasuk contoh peralatan yang dibuat menggunakan konsep berpikir komputasi.

Dengan sekumpulan contoh tersebut dapat dikatakan bahwa berpikir komputasi merupakan kemampuan dasar untuk memudahan atau meringankan pekerjaan. Atau dengan kata lain, berpikir komputasi merupakan kemampuan dasar untuk menyelesaian semua pekerjaan. Baik pekerjaan itu bersifat rutin, maupun tidak. Dari uraian di atas, contoh pekerjaan rutin adalah menyelesaikan pekerjaan di rumah, misal: mencuci, setrika, belanja. Contoh pekerjaan yang tidak rutin adalah menyelesaikan masalah yang timbul dari pekerjaan rutin, misal: saat menanak nasi tiba-tiba listrik mati, sementara nasi belum matang dan jam sudah menunjukkan waktu makan.

Tidak berlebihan jika banyak pakar berpendapat bahwa, berpikir komputasi adalah kemampuan dasar yang diperlukan oleh setiap anak jaman sekarang dan masa depan untuk melakukan analisa, selain kemampuan membaca, menulis, serta aritmetika atau berhitung. Terdapat banyak definisi berpikir komputasi sejak dikenalkan pertama oleh Papert pada tahun 1990. Diantaranya adalah definisi yang diajukan Wing pada tahun 2010. Menurut dia, berpikir komputasi adalah proses pemecahan masalah, sehingga agen pemrosesan pesan dapat dieksekusi secara efektif dan masalah dapat diselesaikan. Dengan demikian, berpikir komputasi tidak hanya fokus pada penyelesaian masalah, tetapi juga mengembangkan dan mengidentifikasi masalah. Oleh karena itu, menurut Grover & Pea berpikir komputasi menjadi keterampilan universal yang dibutuhkan manusia dalam hidup, khususnya mengadopsi mode

berpikir mirip dengan ilmuwan komputer saat menghadapi masalah. Pendapat itu disampaikannya pada tahun 2013.

Pada uraian sebelumnya dicontohkan bahwa, akibat listrik mati, kegiatan menanak nasi yang tadinya merupakan pekerjaan rutin berubah menjadi masalah. Kecuali, orang tersebut sudah mempunyai beberapa pengetahuan, diantaranya adalah sebagai berikut.

1. Pengetahuan tentang menanak nasi tanpa menggunakan peralatan yang membutuhan listrik dan peralatan tersebut juga sudah disediakan di rumah.
2. Pengetahuan menggunakan aplikasi perangkat lunak untuk membeli nasi.
3. Pengetahuan tentang warung dekat rumah yang menjual nasi.
4. Dan seterusnya.

Apa keterkaitan dari ketiga contoh alternatif solusi di atas dengan cara berpikir seorang ilmuwan komputer? Jawabanya adalah: seorang ilmuwan komputer terbiasa berpikir dengan konsep pengolahan data. Lebih tepatnya mengolah data dengan menggunakan mesin komputasi atau komputer.

Mengolah data secara komputasi bertujuan mengubah data tertentu (bukan sembarang data) yang dimasukan ke dalam mesin komputer menjadi keluaran tertentu (bukan sembarang) pula. Pada contoh di atas, seorang ilmuwan komputer sudah tahu bahwa keluaran yang diinginkan adalah nasi. Dia juga tahu bahwa data masukan untuk menghasilkan nasi adalah: beras yang sudah dicuci bersih dan air dengan ukuran tertentu (bukan sembarang). Sedangkan mesin untuk mengubah data masukan menjadi keluaran diantaranya adalah: (1) mesin penanak yang membutuhkan listrik, (2) alat penanak nasi sederhana yang membutuhkan kompor dan bukan listrik,

Berdasarkan uraian tersebut, kebiasaan ilmuwan komputer adalah selalu mengenali tiga hal, yaitu: (1) apa keluaran yang diinginkan, atau apa hasil yang diinginkan, (2) berdasarkan keluaran yang sudah ditetapkan pada nomer (1), selanjutnya dilakukan identifikasi masukan

yang dibutuhkan untuk menghasilkan keluaran, dan (3) ilmuwan computer berpikir bagaimana menyusun proses transformasi untuk mengubah masukan tertentu menjadi keluaran tertentu pula. Ketiga hal tersebut dalam ilmu computer disebut dengan komponen pengolahan data.

Dengan demikian, pengolahan data secara komputasi merupakan proses dengan komponen terkecilnya adalah masukan, proses, serta keluaran. Berikut ini adalah contoh sederhana proses pengolahan data yang saat ini biasa dilakukan oleh siswa atau mahasiswa.

1. Transportasi online.
   Pada contoh penggunaan aplikasi *online*, urutan atau prosesnya adalah sebagai berikut.
   a. Pengguna menghidup aplikasi.
   b. Memasukan tujuan, tepatnya alamat tujuan dengan lengkap. Terkadang, juga diminta memasukan alamat atau posisi awal.
   c. Menekan tombol pesan.
   d. Setelah beberapa saat, aplikasi akan menampilkan kendaraan yang akan menjemput beserta biayanya.

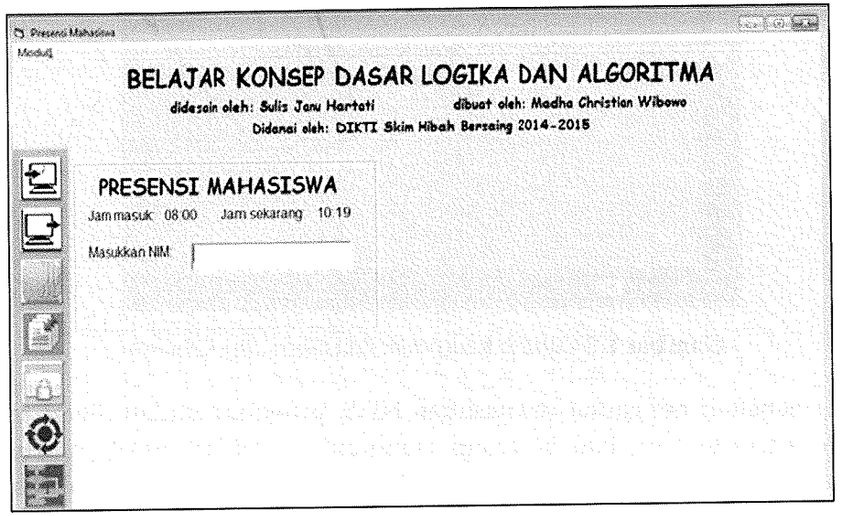
   Penjelasan proses berdasarkan komponen pengolahan data adalah sebagai berikut.
   a. Komponen masukan : membutuhkan data berupa alamat tujuan dengan lengkap, alamat atau posisi awal, menekan tombol pesan.
   b. Komponen proses: aplikasi memilihkan kendaraan dan menghitungkan biaya perjalanan.
   c. Komponen keluaran: informasi nama pengemudi, nomer kendaraan, dan biaya.
2. Pembelian makanan secara online.
   Pada contoh penggunaan aplikasi online, urutan atau prosesnya adalah sebagai berikut.
   a. Pengguna menghidup aplikasi.
   b. Memasukan makanan yang akan dibeli.
   c. Aplikasi akan menampilkan beberapa alternative penjual.

d. Memilih penjual.
e. Aplikasi akan menampilkan beberapa alternative makanan.
f. Memilih makanan yang akan dibeli.
g. Menekan tombol pesan.
h. Aplikasi akan menampilkan nama pengemudi dan nomer kendaraan yang akan mengantar makanan yang dipesan beserta biayanya.

Penjelasan proses berdasarkan komponen pengolahan data adalah sebagai berikut.

a. Komponen masukan: membutuhkan beberapa data masukan berikut ini.
   1) Pertama jenis makanan.
   2) Penjual yang dipilih.
   3) Makanan yang dipilih
b. Komponen proses: ada beberapa proses yang dilakukan, dengan urutan seperti berikut ini.
   1) Aplikasi mencari dan menampilkan semua penjual makanan yang terdaftar.
   2) Aplikasi mencari dan menampilkan semua makanan yang disediakan oleh penjual yang dipilih.
   3) Aplikasi mencarikan dan menampilkan nama pengemudi, nomer kendaraan dan total yang harus dibayar oleh pengguna aplikasi.
c. Komponen keluaran: ada beberapa informasi yang ditampilkan oleh aplikasi, dengan urutan seperti berikut ini.
   1) Daftar penjual yang menyediakan makanan yang dimasukan oleh pengguna.
   2) Daftar semua makanan yang dapat dibeli dari penjual yang dipilih oleh pengguna beserta harga masing-masing makanan.
   3) Informasi nama pengemudi, nomer kendaraan, dan biaya.

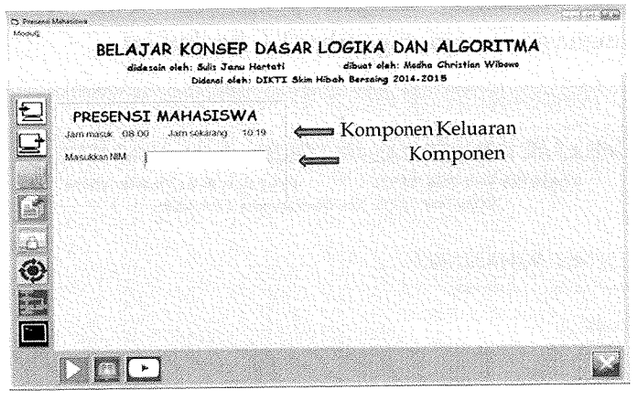
3. Contoh lain berkaitan dengan pengolahan data adalah mesin presensi mahasiswa seperti gambar 1.1. berikut ini.



Gambar 1. 1 *Contoh Pengolahan Data Sederhana*

Tampilan pada gambar 1.1. merupakan contoh aplikasi untuk mengolah data kehadiran mahasiswa. Tampilan di atas di atas disebut **komponen masukan**. Dalam contoh ini, peralatan masukan yang terlihat adalah monitor. Ada peralatan lain yang termasuk komponen masukan dalam contoh tersebut, yaitu: papan ketik, tetikus, layar sentuh atau sejenisnya. Fungsi peralatan tersebut adalah untuk memasukan data yang diminta oleh aplikasi perangkat lunak, yaitu: data nomer induk mahasiswa (NIM).

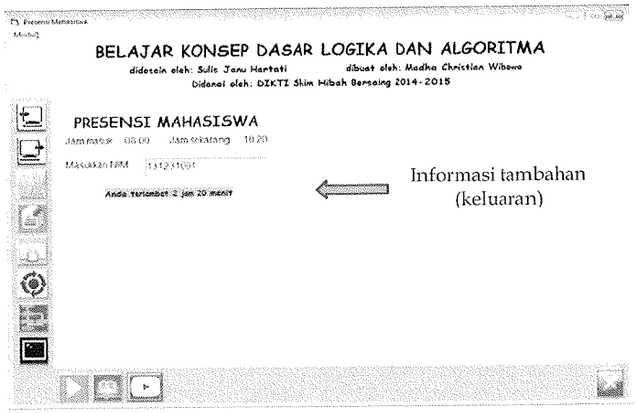
Komponen proses tidak bisa dilihat oleh pemakai, karena komponen proses berada dalam mesin komputer, disebut dengan *Central Processing Unit* (CPU). Proses yang terjadi hanya diketahui oleh pembuat program. Gambar 1.1.di atas ditampilkan lagi pada gambar 1.2. berikut ini dengan menambahkan penjelasan bagian komponen masukan dan keluaran.



Gambar 1.2 *Contoh Komponen Masukan dan Keluaran*

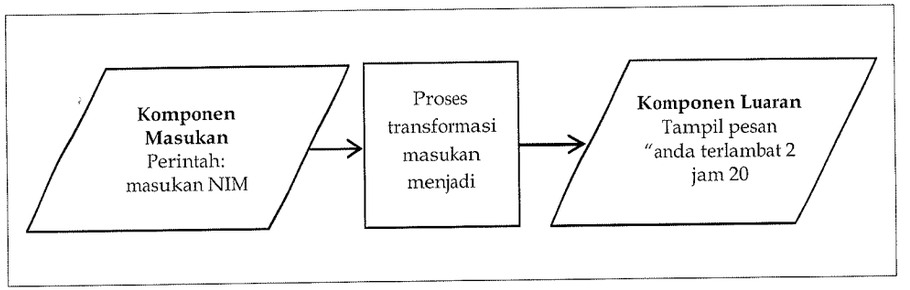
Sebelum pengguna memasukan NIM, pengguna sudah diberi tahu oleh sistem tentang jam sekarang menunjukkan pukul 10.19, informasi tersebut termasuk keluaran karena nilai tersebut akan berubah sesuai dengan jam di sistem komputer. Ini berarti ada **PROSES** menampilkan jam pada sistem komputer ke monitor begitu aplikasi dijalankan oleh pengguna. Sedangkan informasi jam masuk adalah 08.00 bukan hasil proses, karena jam berapapun saat ini, informasi tersebut tetap.

Setelah pengguna memasukan NIM pada komponen masukan, maka gambar 1.2. berubah menjadi gambar 1.3. berikut ini.



Gambar 1.3 *Contoh Keluaran Hasil Proses pada Gambar 1.2*

Pada monitor ada informasi tambahan yaitu: Anda terlambat 2 jam 20 menit. Informasi ini disebut dengan keluaran atau berfungsi sebagai **komponen keluaran**. Proses di atas dapat disederhanakan menjadi seperti pada Gambar 1.4 berikut:

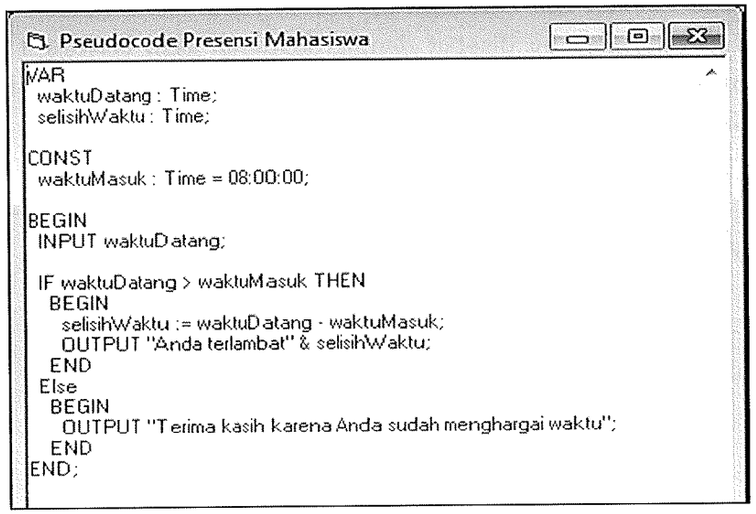


Gambar 1.4 *Elemen Terkecil Pengolahan Data*

Data NIM dimasukkan pada posisi tertentu (bukan sembarang). Demikian juga informasi keluaran ditampilkan pada posisi tertentu (bukan sembarang) pula, dengan proses yang tertentu pula. Pada contoh di atas, proses yang dilakukan oleh mesin komputer adalah menghitung selisih antara jam sekarang dengan jam masuk. Kemudian menampilkan informasi dengan ketentuan sebagai berikut.

1. Jika selisih sama dengan nol (0) atau lebih kecil, maka informasi yang ditampilkan adalah terima kasih anda telah menghargai waktu.
2. Jika sebaliknya, maka informasi yang ditampilkan adalah anda terlambat sekian jam dan sekian menit sesuai selisih antara jam sekarang dengan jam masuk.

Proses mentransformasikan masukan menjadi keluaran disusun menggunakan *pseudocode* bahasa Pascal, seperti disajikan pada gambar 1.5. berikut ini.

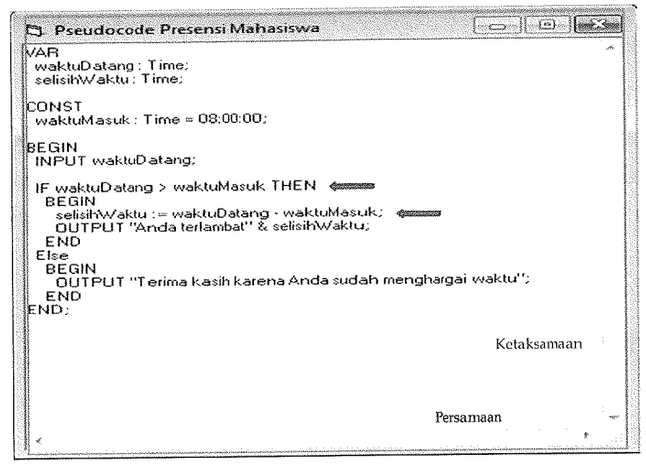


Gambar 1.5 *Pseudocode Transformasi dari Gambar 1.2. Menjadi Gambar 1.3.*

Suatu rangkaian proses dimulai dari masukan sampai menjadi luaran disebut pengolahan data. Proses transformasinya, pada contoh di atas disajikan dalam bentuk *pseudocode*. Beberapa penulis, seperti Farrel, memasukan *pseudocode* ke dalam metode pembuatan algoritma komputasi. algoritma komputasi adalah proses transformasi yang berfungsi mengubah masukan tertentu menjadi luaran tertentu pula menggunakan peralatan komputer sebagai mesin pengolah data.

Uraian di atas menunjukkan bahwa pengolahan data secara komputasi merupakan implementasi dari konsep berpikir komputasi. Selain itu, ditunjukkan pula bahwa membuat algoritma komputasi merupakan bagian dari kemampuan berpikir komputasi. Sebagaimana pendapat Wing (2010), Weintrop *et al.* (2016) dan lainnya.

Hartati (2017) berpendapat bahwa pembelajaran algoritma komputasi dibutuhkan penalaran matematika kreatif bukan imitasi. Penalaran matematika tersebut digunakan untuk membuat sekumpulan proposi yang mewakili proses transformasi masukan menjadi keluaran. Perhatikan kembali gambar 1.5, yang ditampilkan lagi pada gambar 1.6. berikut ini.



Gambar 1.6 *Contoh Persamaan dan Ketaksamaan pada Algoritma*

Gambar 1.6. di atas adalah contoh sekumpulan proposisi dalam algoritma. Pada gambar tersebut, dicontohkan bentuk ketaksamaan adalah waktuDatang > waktuMasuk. Contoh persamaan adalah: selisihWaktu = waktuDatang - waktuMasuk. Keterampilan atau kemampuan membuat proposisi tersebut harus dilatihkan sejak pertama pembelajaran sampai dengan selesai. Oleh karena itu, pembahasan dalam buku ini selalu menghubungkan dengan proses pembuatan proposisi. Dalam buku ini, proposisi dibedakan menjadi 3, yaitu: pernyataan matematika, logika matematika, dan instruksi komputasi. Buku ini juga dilengkapi dengan perangkat lunak pembelajaran penalaran matematika, yang diunggah di alamat https://penalaran.com.

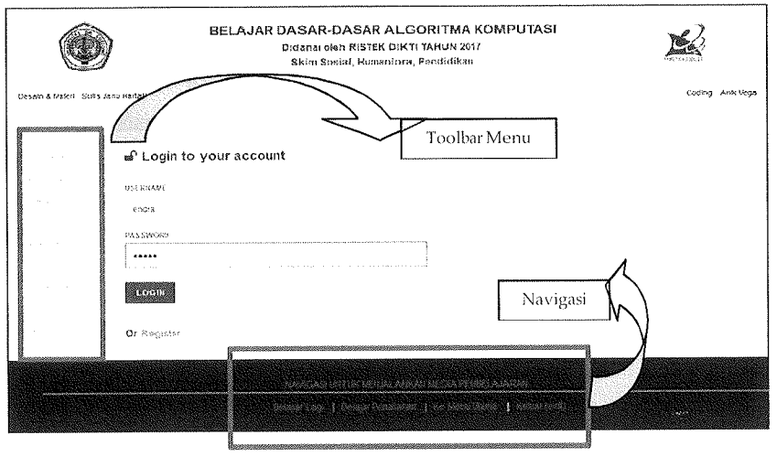
Buku ini ditujukan bagi siswa sekolah menengah atas dan sederajat dan mahasiswa yang sedang atau akan belajar algoritma pemrograman namun kemampuan penalaran matematikanya masih butuh ditingkatkan atau diasah lagi. Oleh karena itu, dalam diskusi pada buku ini, definisi operasional yang digunakan untuk menjelaskan berpikir komputasi adalah sekumpulan keterampilan yang terukur, berkaitan dengan pengolahan data secara komputasi. Keterampilan tersebut, meliputi: (1) kemampuan mengidentifikasi masalah komputasi, (2) kemampuan mengidentifikasi komponen terkecil pengolahan data, yaitu masukan,

proses, dan keluaran, (3) kemampuan membuat proposisi, (4) kemampuan membuat algoritma dalam bentuk *flowchart* dan *pseudocode*.

Definisi operasional tersebut menunjukkan bahwa, membuat algoritma merupakan bagian dari berpikir komputasi. Demikian juga pengolahan data, merupakan bagian dari berpikir komputasi. Termasuk di dalamnya kemampuan melakukan dekomposisi ke unsur pengolahan data terkecil: masukan, proses, dan keluaran.

1.2 Komponen Masukan

Komponen masukan merupakan komponen pengolahan data yang pertama kali terlihat di layar begitu aplikasi perangkat lunak dijalankan. Kita mulai pembahasan dengan berkunjung ke https://penalaran.com. Cara berkunjung adalah: (1) yakinkan bahwa komputer sudah terhubung dengan internet, (2) buka browser, disarankan mozila, dan (3) ketikan alamat aplikasi penalaran.com diunggah. Maka akan muncul tampilan seperti gambar 1.7. berikut ini.



Gambar 1.7 *Tampilan Awal Aplikasi penalaran.com (Komponen Masukan)*

Masukan yang bisa diterima oleh komponen masukan pada gambar 1.7. terdiri dari 3 kelompok, yaitu: (1) kelompok *tools bar*, (2) kelompok navigasi, dan (3) kelompok isian login atau registrasi.

Kelompok *tools bar* memuat 10 pilihan, meliputi: MASUKAN, KELUARAN, VARIABEL, DATA, KONSTANTA, OPERATOR, PENALARAN, FLOWCHART,

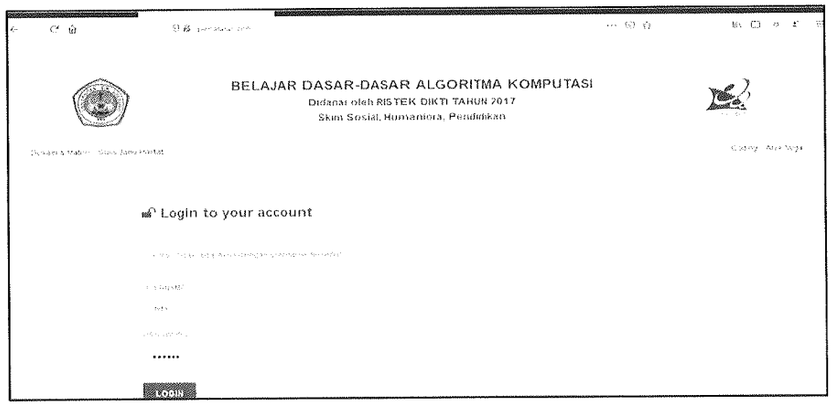
PSEUDOCODE, CODE. Kelompok navigasi memuat 4 pilihan, yaitu: Belajar Lagi, Belajar Penalaran, Ke Menu Utama, Keluar (*exit*). Sedangkan kelompok isian login atau registrasi dibagi menjadi 2, yaitu: login atau registrasi. Kedua-duanya minta masukan yang sama, yaitu: USERNAME dan PASSWORD. Pilihan registrasi digunakan untuk mendaftarkan diri menggunakan aplikasi penalaran.com. sedangkan pilihan LOGIN digunakan untuk menggunakan aplikasi penalaran.com.

Setiap kelompok masukan mempunyai fungsi yang berbeda satu sama lainnya. Fungsi masing-masing kelompok diuraikan seperti berikut ini.

1. Kelompok *tools bar*
   a. *Tool bar* masukan berfungsi untuk belajar komponen masukan.
   b. *Tool bar* keluaran berfungsi untuk belajar komponen keluaran
   c. *Tool bar* variabel berfungsi untuk belajar tentang variabel yang ada pada komponen masukan dan keluaran, serta proses.
   d. *Tool bar* data berfungsi untuk belajar tentang data yang ada pada komponen masukan dan keluaran, serta proses.
   e. *Tool bar* konstanta berfungsi untuk belajar tentang konstanta yang ada pada komponen masukan dan keluaran, serta proses.
   f. *Tool bar* operator berfungsi untuk belajar tentang operator yang ada pada komponen proses.
   g. *Tool bar* penalaran berfungsi untuk belajar tentang penalaran yang ada pada proses transformasi masukan menjadi luaran.
   h. *Tool bar flowchar* berfungsi untuk belajar tentang *flowchart* yang mentransformasikan masukan menjadi keluaran.
   i. *Tool bar pseudocode* berfungsi untuk belajar tentang pseudo code yang mentransformasikan masukan menjadi keluaran.
   j. *Tool bar* code berfungsi untuk belajar *coding* atau allgoritma yang sudah dikonversikan ke dalam Bahasa pemrograman.

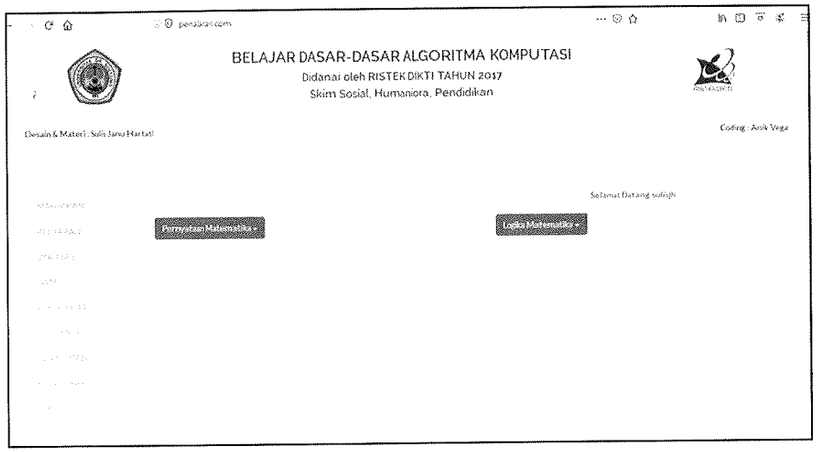
Kelompok *tool bar* membatasi nilai masukan. Pengguna sudah diarahkan untuk memilih data masukan yang sudah disediakan oleh aplikasi.

2. Kelompok navigasi
   a. Navigasi belajar lagi, berfungsi untuk mengarahkan pengguna jika menginginkan belajar lagi untuk materi pada *tool bar*.
   b. Navigasi belajar penalaran, berfungsi untuk mengarahkan pengguna jika menginginkan belajar penalaran pada proses transformasi.
   c. Navigasi kembali ke menu utama, berfungsi untuk mengarahkan pengguna kembali ke menu utama.
   d. Navigasi *exit*, berfungsi untuk mengarahkan pengguna keluar dari aplikasi.
3. Kelompok isian login atau registrasi.
   a. Pilihan login berfungsi untuk masuk ke aplikasi dan menggunakannya sebagai media pembelajaran. Pilihan login bisa dijalankan jika pengguna sudah registrasi melalui pilihan register. Gambar 1.8. berikut ini adalah contoh pengguna yang belum register atau salah memasukan *username* dan atau *password* saat melakukan pilihan login.



Gambar 1.8 *Perubahan Memasukan USERNAME dan PASSWORD Salah atau Belum Register*

Namun demikian, jika pengguna memasukan USERNAME dan PASSWORD setelah mendaftarkan diri sebagai pengguna melalui pilihan register dan nilai masukan keduanya benar, maka tampilan akan seperti pada gambar 1.9. berikut ini.



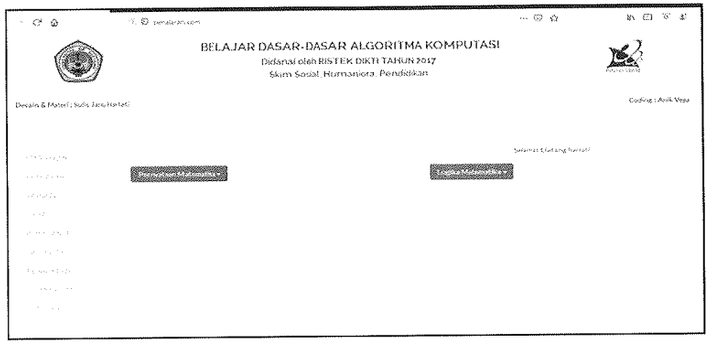
Gambar 1.9 *Perubahan Memasukan USERNAME dan PASSWORD Setelah Registrasi*

Dari contoh kejadian di atas menunjukkan bahwa komponen masukan didesain secara bersamaan dengan komponen keluaran dan proses. Artinya, cara berpikir ilmuwan komputer bersifat menyeluruh, karena solusi atas permasalahan merupakan satu kesatuan yang komponen terkecilnya adalah masukan, keluaran, serta proses.

Perubahan gambar 1.7. menjadi 1.8. atau 1.9. merupakan satu kesatuan. Desain kemungkinan-kemungkinan perubahan yang akan terjadi disusun pada komponen proses. Uraian di atas menunjukkan bahwa setiap masukan akan memberikan keluaran yang mungkin berbeda sesuai dengan desain pada komponen proses.

a. Pilihan register, berfungsi untuk mendaftar sebagai pengguna aplikasi penalaran.com. Berpikir komputasi yang berkaitan dengan komponen masukan pada pilihan register adalah sebagai berikut.

1) Nama, user_name, dan password belum pernah ada. Gambar 1.10. adalah perubahan yang terjadi untuk data masukan dengan kondisi Nama, user_name, dan *password* belum pernah ada.



Gambar 1.10 *Perubahan yang Terjadi untuk Data Masukan dengan Kondisi Nama, user_name, dan password Belum Pernah Ada*

2) Nama pengguna berbeda, namun user_name, dan atau password ada yang sama dengan data sebelumnya. Gambar 1.11. adalah perubahan yang terjadi untuk data masukan dengan kondisi Nama pengguna berbeda, namun user_name, dan atau password ada yang sama dengan data sebelumnya.

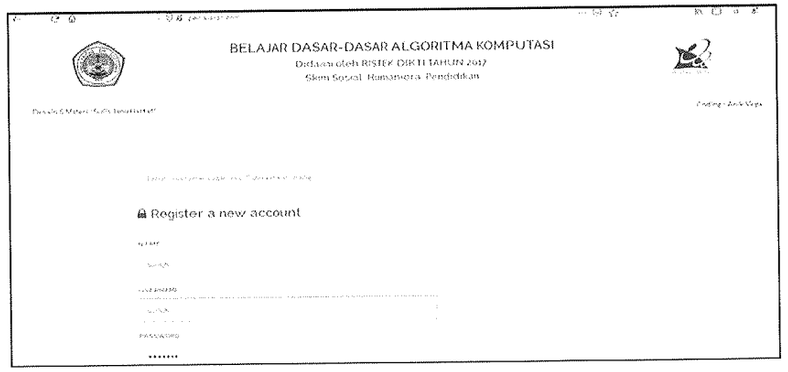
Gambar 1.11 *Tampilan untuk NAMA Belum Pernah Ada, namun USERNAME dan atau PASSWORD Sama dengan Data Sebelumnya*

3) Nama pengguna sudah ada, namun user_name, dan atau *password* berbeda dari sebelumnya, baik keduanya berbeda,

atau hanya satu yang berbeda data sebelumnya. Gambar 1.12. adalah perubahan yang terjadi untuk Nama pengguna sudah ada, namun user_name, dan atau *password* berbeda dari sebelumnya, baik keduanya berbeda, atau hanya satu yang berbeda data sebelumnya.

Gambar 1.12 *Tampilan untuk NAMA Sama dengan Data Sebelumnya, namun USERNAME dan PASSWORD yang Belum Pernah Ada*

4) Nama pengguna sudah ada, user_name, dan *password* juga sama dengan data sebelumnya. Gambar 1.13. adalah perubahan yang terjadi untuk Nama pengguna sudah ada, demikian juga user_name, dan atau password sama dengan data sebelumnya.



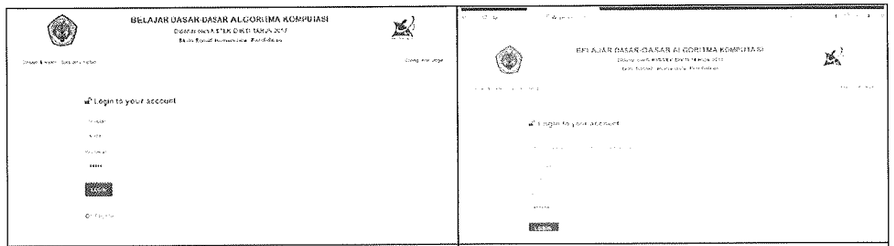
Gambar 1.13 *Tampilan untuk NAMA, USERNAME dan PASSWORD Sama dengan Data Sebelumnya*

Uraian di atas menunjukkan bahwa, untuk membuat komponen masukan dibutuhkan kemampuan berpikir komputasi secara menyeluruh. Seorang ilmuwan komputer dituntut mempunyai kemampuan memperkirakan beberapa kemungkinan data yang akan dimasukan oleh pengguna. Pada contoh di atas, diberikan 4 contoh kemungkinan data yang dimasukkan untuk registrasi oleh pengguna. Maka, seorang ilmuwan harus mengetahui ada berapa kemungkinan data masukan yang berbeda yang mungkin akan dilakukan oleh pengguna.

1.3 Komponen Proses

Komponen proses tidak dapat dilihat oleh pengguna secara langsung, sebagaimana yang dijelaskan pada uraian sebelumnya. Komponen proses tidak bisa diketahui oleh orang yang tidak terbiasa berpikir komputasi. Bagian ini menjelaskan tentang apa yang dilakukan oleh komponen proses dengan mencari hubungan yang terjadi antara komponen masukan dan keluaran.

Mari perhatikan kembali komponen masukan pada gambar 1.7 dan perubahannya yang terjadi pada gambar 1.8 sebagaimana gambar 1.14 berikut ini.



a. Sesuai Gambar 1.7 b. Sesuai Gambar 1.8

Gambar 1.14 *Perubahan Tampian Awal Aplikasi* ***penalaran.com*** *Setelah Pengguna Memasukan Username dan Password Namun Belum Registrasi atau Username atau dan Password yang Dimasukan Salah*

Dari perubahan tampilan yang disajikan pada gambar 1.14, **PROSES** yang dibuat pada komponen proses dapat diperkirakan oleh

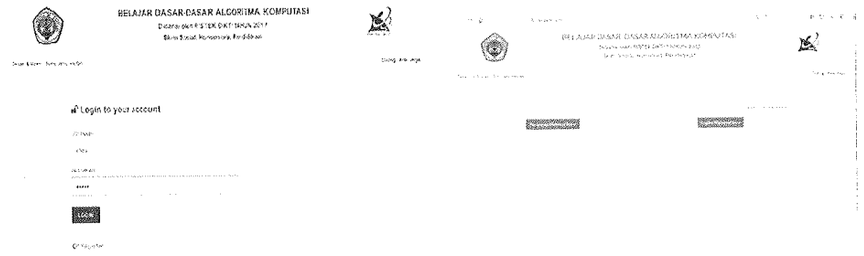
orang yang punya kemampuan berpikir komputasi walaupun dia bukan pembuatnya. Prosesnya adalah sebagai berikut.

Tabel 1.1 *Proses Perubahan Gambar 1.7 menjadi Gambar 1.8*

Urutan Proses Perubahan Gambar 1.7 menjadi Gambar 1.8.	Algoritma Perubahan Gambar 1.7 menjadi Gambar 1.8.

1. Cari username dan password yang dimasukan dengan yang ada di basis data.
2. Jika username dan password tidak ada di basis data, tampilkan gambar 1.8. Pesan yang ditampilkan adalah "Error tidak ada akun dengan username tersebut"
3. Jika username atau password tidak ada di basis data, tampilkan gambar 1.8. Pesan yang ditampilkan adalah "Error tidak ada akun dengan username tersebut".
	Input (username) Input (password) Int i = 0
	For i=0 to tbluser.count() do
	If (username != tbluser[i] AND password != tblpass[i]) then
	Print ("Error tidak ada akun dengan username tersebut")
	Else if (username != tbluser[i] OR password == tblpass[i]) then
	Print ("Error tidak ada akun dengan username tersebut")
	End If End for

Kemudian, perhatikan perubahan Gambar 1.7 menjadi Gambar 1.9 sebagaimana digambarkan ulang pada Gambar 1.15 berikut ini.



a. Sesuai Gambar 1.7 b. Sesuai Gambar 1.9

Gambar 1.15 *Perubahan Gambar 1.7 Menjadi Gambar 1.9*

Perubahan dari Gambar 1.7 Menjadi Gambar 1.9 terjadi pada saat *username* dan *password* yang dimasukan oleh pengguna benar. Jika dikaitkan dengan perubahan gambar 1.7 menjadi 1.8, maka perubahan gambar 1.7 menjadi 1.9 tidak bisa dipisahkan dengan proses sebelumnya. Dengan kata lain, pada saat pengguna memasukan *username*

dan *password* melalui tampilan pada gambar 1.7 terdapat 3 kondisi yang harus dipikirkan, yaitu:

1. Kondisi 1, jika *username* **dan** *password* tidak ada pada basis data.
2. Kondisi 2, jika *username* **atau** *password* tidak ada pada basis data.
3. Kondisi 3, jika *username* dan *password* ada pada basis data.

Oleh karena itu proses yang diperkirakan terjadi sebagaimana pada tabel 1.1 di atas berubah menjadi seperti pada tabel 1.2 berikut ini.

Tabel 1.2 *Proses Perubahan Gambar 1.7 Menjadi Gambar 1.8 dan Gambar 1.9*

Urutan Proses Perubahan Gambar 1.7 Menjadi Gambar 1.8.	Algoritma Perubahan Gambar 1.7 Menjadi Gambar 1.8.

1. Cari username dan password yang dimasukan dengan yang ada di basis data.
2. Jika username dan password tidak ada di basis data, tampilkan gambar 1.8. Pesan yang ditampilkan adalah "Error tidak ada akun dengan username tersebut"
3. Jika username atau password tidak ada di basis data, tampilkan gambar 1.8. Pesan yang ditampilkan adalah "Error tidak ada akun dengan username tersebut".
4. Jika username dan password ada pada basis data, tampilkan gambar 1.9.
	Input (username) Input (password) Int i = 0
	For i=0 to tbluser.count() do
	If (username != tbluser[i] AND password != tblpass[i]) then
	Print ("Error tidak ada akun dengan username tersebut")
	Else if (username != tbluser[i] OR password == tblpass[i]) then
	Print ("Error tidak ada akun dengan username tersebut")
	Else
	TampilkanMenuUtama() End If
	End for

Uraian di atas menjelaskan bahwa proses transformasi dari masukan menjadi keluaran dapat diperkirakan urutan dan aturannya oleh orang lain (bukan pembuatnya) jika orang tersebut mempunyai kemampuan berpikir secara komputasi. Gambar 1.7 di atas adalah komponen masukan yang harus dilewati oleh setiap pengguna yang akan menjalankan aplikasi penalaran.com dengan menjalankan fungsi LOGIN. Dengan kata lain, perkiraan proses pada tabel 1.2 akan dijalankan jika fungsi LOGIN dijalankan dengan cara menekan pilihan LOGIN. Berarti, perkiraan proses pada tabel 1.2 harus direvisi menjadi seperti pada tabel 1.3 berikut ini.

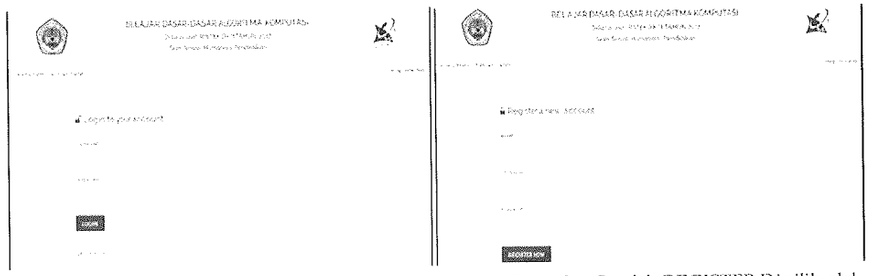
Tabel 1.3 *Proses Perubahan Gambar 1.7 Menjadi Gambar 1.8 dan Gambar 1.9 dengan Melibatkan Fungsi LOGIN*

Urutan Proses Perubahan Gambar 1.7 Menjadi Gambar 1.8.	Algoritma Perubahan Gambar 1.7 Menjadi Gambar 1.8.

1. Tekan atau pilih tombol LOGIN. Jika LOGIN tidak ditekan oleh pengguna, maka proses 2 sampai 5 tidak dilakukan oleh system.
2. Cari username dan password yang dimasukan dengan yang ada di basis data.
3. Jika username dan password tidak ada di basis data, tampilkan gambar 1.8. Pesan yang ditampilkan adalah "Error tidak ada akun dengan username tersebut"
4. Jika username atau password tidak ada di basis data, tampilkan gambar 1.8. Pesan yang ditampilkan adalah "Error tidak ada akun dengan username tersebut".
5. Jika username dan password ada pada basis data, tampilkan gambar 1.9.
	If (LOGIN == true) then
	Input (username) Input (password) Int i = 0
	For i=0 to tbluser.count() do
	If (username != tbluser[i] AND password != tblpass[i]) then
	Print ("Error tidak ada akun dengan username tersebut")
	Else if (username != tbluser[i] OR password == tblpass[i]) then
	Print ("Error tidak ada akun dengan username tersebut")
	Else
	TampilkanMenuUtama() End If
	End for End If

Uraian di atas menegaskan bahwa hanya pengguna yang sudah terdaftar saja yang bisa menggunakan atau menjalankan aplikasi pada penalaran.com. Ini berarti, setiap pengguna harus melakukan pendaftaran dulu. Aplikasi menyiapkan fasilitas pendaftaran dengan menyediakan tombol REGISTER.

Uraian berikut ini membuat prediksi **PROSES** melalui tombol REGISTER. Pengamatan dimulai dengan memilih atau menekan tombol register gambar 1.7. Perubahan ditunjukan pada 1.16.a. menjadi gambar 1.16.b. berikut ini.



a. Tampilan Awal

b. Tampilan Setelah REGISTER Dipilih oleh Pengguna

Gambar 1.16 *Tampilan Awal Aplikasi penalaran.com Sebelum dan Setelah Memilih Fungsi REGISTER*

Perubahan dari gambar 1.16.a. menjadi gambar 1.16.b. disajikan pada tabel 4 berikut ini.

Tabel 1.4 *Proses Perubahan Gambar 1.16.a. Menjadi Gambar 1.16.b. dengan Melibatkan Fungsi REGISTER*

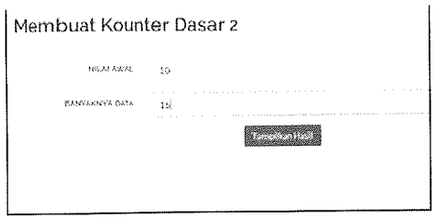
Urutan Proses Perubahan Gambar 1.16.a. Menjadi Gambar 1.16.b.	Algoritma Perubahan Gambar 1.16.a. Menjadi Gambar 1.16.b.
1. Tekan atau pilih tombol REGISER. Jika REGISER tidak ditekan oleh pengguna, maka proses 2 tidak dilakukan oleh system. 2. Tampilkan gambar 1.16.b.	If(Register == true) then TampilkanFormRegister() End If

Uraian di atas adalah contoh proses yang dapat diprediksi dengan mengamati perubahan yang terjadi. Proses tranformasi dapat dinyatakan dalam bentuk urutan proses transformasi atau algoritma. Kemampuan ini disebut dengan kemampuan berpikir komputasi secara dasar atau kemampuan membuat algoritma dasar. Walaupun tidak semua **PROSES** bisa diprediksi, khususnya proses yang membutuhkan pengetahuan lain tidak sekedar perubahan yang terlihat, seperti contoh perubahan pada gambar 1.17 berikut ini.

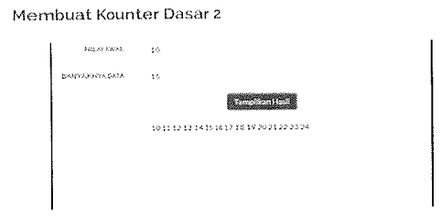
a. Tampilan Awal Contoh Membuat Kounter Dasar-2 Pada Aplikasi penalaran.com

b. Contoh Masukan-1 sebelum TAMPILKAN HASIL Dipilih oleh Pengguna

c. Perubahan Masukan-1 setelah TAMPILKAN HASIL Dipilih oleh Pengguna



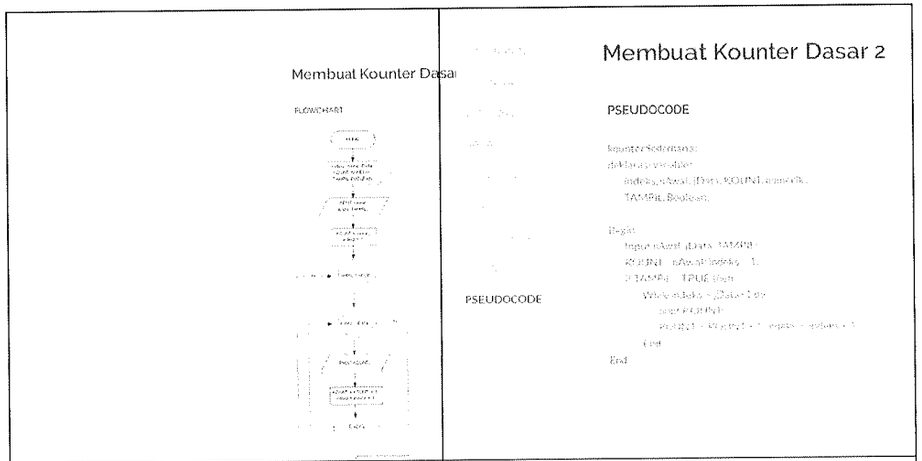
d. Contoh Masukan-1 sebelum TAMPILKAN HASIL Dipilih oleh Pengguna



e. Perubahan Masukan-1 setelah TAMPILKAN HASIL Dipilih oleh Pengguna

Gambar 1.17 *Belajar Proses Membuat Kounter Dasar-2 pada Aplikasi penalaran.com*

Mengamati perubahan yang terjadi pada gambar 1.17, mulai dari gambar 1.17.a. sampai dengan gambar 1.17.e. tidak bisa dipahami secara langsung oleh pengguna. Hanya pengguna yang pernah belajar dan paham konsep penjumlahan secara berulang dengan nilai penambahan yang tetap yang bisa memprediksi proses. Untuk mengetahui proses, **penalaran.com** menyediakan fasilitas melalui tools bar pseudocode atau flowchat seperti disajikan pada gambar 1.18. berikut ini.



a. Proses Disajikan dalam Bentuk Flowchart

b. Proses Disajikan dalam Bentuk Pseudocode

Gambar 1.18 *Proses Perubahan dari Gambar 1.17.b menjadi 1.17.c.*

Setelah *Flowchart* selesai dibuat, langkah selanjutnya adalah membuat *pseudocode*. Adapun bentuk pseudocode dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk pembuatan Kounter Dasar 2 dapat dilihat pada tabel di bawah ini.

Pseudocode	Bahasa Pascal

kounterSederhana; deklarasi
variable;
indeks, nAwal, jData, KOUNT:
numerik;
TAMPIL: Boolean;
 	
Begin
Input nAwal, jData, TAMPIL; KOUNT =
nAwal; indeks = 1; if TAMPIL = TRUE
then
While indeks < jData+1 do print
KOUNT;
KOUNT = KOUNT + 1;
indeks = indeks + 1;
End;
End.
	var
	indeks: integer; nAwal: integer;
	jData: integer; kount: integer;
	
	begin indeks := 1;
	nAwal := 5;
	jData := 6;
	kount := 0; kount := nAwal;
	while (indeks < jData+1) begin
	writeln (' ' , kount); kount := kount
	+ 1; indeks := indeks + 1; end;
	writeln(' '); end;

Pseudocode	Bahasa C++

kounterSederhana; deklarasi
variable;
indeks, nAwal, jData, KOUNT:
numerik;
TAMPIL: Boolean;
 	
Begin
Input nAwal, jData, TAMPIL; KOUNT =
nAwal; indeks = 1; if TAMPIL = TRUE
then
While indeks < jData+1 do print
KOUNT;
KOUNT = KOUNT + 1;
indeks = indeks + 1;
End;
End.
	#include <stdio.h> using namespace
	std; int main() {
	int indeks = 1; int nAwal = 5; int
	jData = 6; int kount = 0; kount =
	nAwal;
	while (indeks < jData+1) { cout <<
	" " << kount<< endl; kount = kount +
	1;
	indeks = indeks + 1;
	}
	cout << "" << endl; return 0;
	}

Pseudocode	Bahasa Java

kounterSederhana; deklarasi
variable;
indeks, nAwal, jData, KOUNT:
numerik;
TAMPIL: Boolean;
	public class Kounter_2 {
	public static void main(String[]
	args)
	{
	int indeks = 1; int nAwal = 5;

Pseudocode	Bahasa Java

Begin
Input nAwal, jData, TAMPIL; KOUNT
= nAwal; indeks = 1; if TAMPIL =
TRUE then
While indeks < jData+1 do print
KOUNT;
KOUNT = KOUNT + 1;
indeks = indeks + 1;
End;
End.
	int jData = 6; int kount = 0; kount =
	nAwal;
	while (indeks < jData+1) {
	System.out.println(" " + kount); kount
	= kount + 1;
	indeks = indeks + 1;
	}
	System.out.println("");
	}
	}

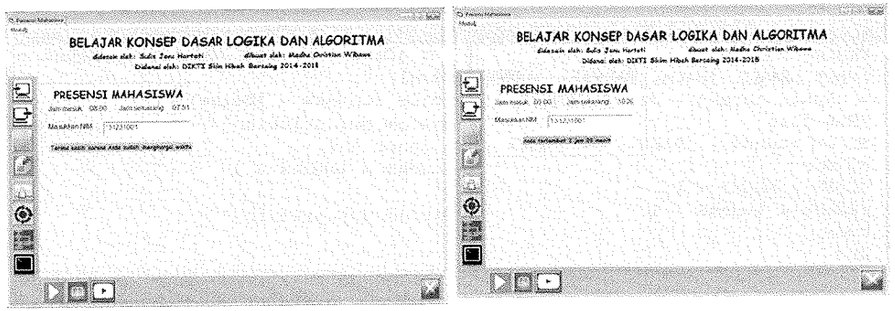
Uraian di atas menunjukan bahwa untuk memahami proses dibutuhkan kemampuan berpikir komputasi lebih tinggi, namun demikian dapat dilatih secara bertahap. Nampak jelas bahwa dalam berpikir komputasi, desain komponen terkecil dari pengolahan data harus dibuat secara bersama, tidak bisa secara parsial. Oleh sebab itu, aplikasi **penalaran.com** memberikan penjelasan secara bertahap, dimulai dari yang paling dasar.

Dengan demikian, kemampuan berpikir komputasi dapat dilatihkan melalui hal-hal yang sederhana. Pada saat yang sama, latihan tersebut mengantarkan siswa yang sedang berlatih untuk belajar pengolahan data secara komputasi. Selain itu, dia juga berlatih membuat algoritma komputasi mulai materi yang paling dasar.

1.4 Komponen Keluaran

Sebelum membahas komponen keluaran, perhatikan kembali beberapa contoh di atas darai sudut pandang komponen keluaran berikut ini.

1. Mesin cuci, berfungsi untuk membersihkan pakaian. Keluaran dari mesin tersebut adalah pakain bersih. Pakaian bersih inilah yang disebut dengan keluaran.
2. Aplikasi presensi mahasiswa, berfungsi untuk mencatat kehadiran mahasiswa dan memberikan informasi serta penghargaan bagi mahasiswa yang disiplin atau tidak terlambat hadir. Pada aplikasi presensi mahasiswa keluaran yang diharapkan ada 2, yaitu: (1) waktu sekarang yang diambil dari sistem komputer, dan (2) informasi berupa peringatan atau penghargaan. Model keluaran tersebut disajikan pada gambar 1.19. berikut ini.



a. Mahasiswa Datang Tidak Terlambat b. Mahasiswa Datang Terlambat

Gambar 1.19 *Contoh Keluaran dari Aplikasi Presensi Mahasiswa*

Gambar 1.19.a. adalah keluaran yang ditampilkan jika mahasiswa tidak terlambat hadir dalam perkuliah. Sedangkan gambar 1.19.b. adalah keluaran yang ditampilkan jika mahasiswa terlambat hadir pada perkuliahan. Pada contoh ini terdapat dua jenis keluaran yang diinginkan oleh pemilik aplikasi.

1. Aplikasi **penalaran.com**, berfungsi untuk belajar algoritma pemrograman tingkat dasar. Penggunanya tidak dibatasi oleh pembuat system, namun setiap pengguna harus registrasi terlebih dahulu. Oleh sebab itu, pengguna diminta melakukan LOGIN untuk menjalankan **penalaran.com.** Gambar 1.20. berikut ini adalah contoh keluaran dari fungsi LOGIN.

a. Pengguna Tidak Berhasil LOGIN b. Pengguna Berhasil LOGIN

Gambar 1.20 *Perubahan Gambar 1.7 Menjadi Gambar 1.9*

Fungsi LOGIN mempunyai 2 macam keluaran,yaitu: (1) gambar 1.20.a. adalah tampilan LOGIN yang tidak berhasil karena username dan

atau *password* ada yang salah, (2) gambar 1.20.b. adalah tampilan LOGIN yang berhasil. Ini menunjukkan bahwa relasi masukan dan keluaran tidak satu-satu. Satu fungsi LOGIN menghasilkan dua buah luaran sesuai dengan aturan yang ditetapkan oleh pemilik sistem.

Dari ketiga contoh di atas, dapat dikatakan bahwa komponen keluaran merupakan bagian inti dari pengolah data. Artinya, jika tidak ada keluaran yang diinginkan maka tidak ada masukan dan proses. Dengan kata lain, berpikir tentang pengolahan data secara komputasi akan dilakukan setelah ada keluaran yang diinginkan oleh pemilik sistem atau pengguna. Oleh sebab itu, berpikir tentang masukan dan proses semua mengarah kepada keluaran.

Keluaran dapat juga didefinisikan sebagai kejadian atau hasil perubahan yang diharapkan oleh pengguna. Perhatikan kembali ketiga contoh di atas, yaitu: tentang mesin cuci, aplikasi presensi mahasiswa, dan aplikasi **penalaran.com.**

1. Mesin cuci. Mencuci pakaian merupakan kegiatan rutinitas yang membutuhkan tenaga dan waktu. Manusia akhirnya berpikir, bagaimana membuat alat pencuci pakaian yang dapat meringankan pekerjaan rutinitas tersebut. Dari pernyataan tersebut, terlihat ada kebutuhan alat yang meringankan pekerjaan mencuci pakaian. Keluaran yang diinginkan adalah pakaian bersih. Alat pemroses adalah mesin cuci. Masukan yang dibutuhkan supaya mesin dapat bekerja meliputi: (1) aliran listrik, (2) pasukan air bersih dan detergen, serta (3) operator yang menjalankan mesin cuci. Jadi, yang muncul pertama kali adalah keluaran yang diinginkan. Setelah itu berpikir spesifikasi alat atau mesin dan cara kerjanya. Kemudian baru berpikir masukan yang diperlukan supaya mesin dapat bekerja sesuai desainnya. Keterampilan berpikir yang demikian ini tergolong sebagai berpikir komputasi.
2. Aplikasi presensi mahasiswa. Presensi mahasiswa awalnya dilakukan dengan mencatat setiap mahasiswa yang hadir di formulir daftar kehadiran. Ada juga presensi dicatat melalui form kehadiran yang ditandatangani oleh mahasiswa. Data yang diperoleh hanya sebatas

hadir atau tidak. Informasi lain, seperti jam kehadiran mahasiswa tidak bisa terekam pada form hadir. Dengan berkembangnya teknologi informasi dan kebutuhan data yang berkaitan dengan peningkatan kualitas pelayanan dan mutu pendidikan, maka dipandang perlu merekam data aktivitas pembelajaran mulai dari kehadiran mahasiswa sampai dengan nilai mahasiswa. Kebutuhan akan peningkatan kualitas pembelajaran mengakibatkan kebutuhan informasi kedisiplinan mahasiswa. Kemudian muncul pertanyaan, informasi apa yang perlu disimpan dari data kehadiran. Kebutuhan ini berkaitan dengan proses. Pada saat mendesain proses inilah muncul kebutuhan data apa saja yang akan dimasukan ke dalam proses untuk menghasilkan data kehadiran mahasiswa. Contoh kedua juga menunjukan bahwa berpikir komputasi dimulai dari keinginan memperbaiki keadaan saat ini. Perubahan yang diinginkan tersebut dalam pengolahan data komputasi disebut dengan keluaran yang terdapat pada komponen keluaran. Kemudian mulai dipikirkan proses untuk melakukan perubahan guna mencapai keluaran yang diinginkan. Dalam pengolahan data komputasi, proses perubahan untuk mendapatkan keluaran yang diinginkan dimasukan ke dalam komponen proses. Berdasarkan proses perubahan selanjutnya diidenfikasi masukan yang dibutuhkan supaya proses perubahan menghasilkan keluaran yang benar.

3. Aplikasi **penalaran.com** dibuat untuk memperpendek kesenjangan antara materi kuliah algoritma pemrograman dengan kemampuan mahasiswa membuat algoritma komputasi. Berdasrkan kurikulum nasional, prasyarat belajar algoritma pemrograman adalah mempunyai kemampuan penalaran matematika yang baik. Indicator baik adalah nilai matematika di atas 70.

Dengan ilustrasi tiga contoh di atas, maka komponen keluaran berisi keinginan untuk memperbaiki kondisi saat ini. Dalam pengolahan data, yang merupakan bagian dari berpikir komputasi, keluaran merupakan komponen yang harus ditentukan dulu sebelum berpikir komponen proses dan masukan.

1.5 Pengguna

Pada diskusi sebelumnya sudah dijelaskan bahwa, komponen terkecil dari pengolahan data adalah masukan, proses, dan keluaran. Beberapa pakar, diantaranya, Dix, Finlay, Abowd, and Beale mengeksplorasi kebiasaan pengguna sebagai hal yang perlu diperhatikan dalam pengolahan data. Hal tersebut, dalam ilmu komputer dipelajari secara tersendiri di bidang ilmu interaksi manusia dan komputer. Manusia dikelompokan dalam pengguna pada sistem pengolahan data.

Pengguna penting dipertimbangkan sebagai komponen pengolahan data, khususnya bagi mereka yang bukan tingkat dasar. Oleh karena itu, pada bab ini pembaca juga dikenalkan komponen lain dari pengolahan data selain masukan, proses, dan keluaran. Dengan tujuan, sejak awal para pemula yang belajar algoritma pemrograman mengetahui bahwa komponen pengolahan data yang perlu diperhatikan tidak hanya tiga. Walaupun, dalam pembahasan selanjutnya komponen pengguna diabaikan dalam buku ini.

Bahasan tentang pengguna tidak dibahas secara rinci karena membutuhkan kemampuan berpikir komputasi lebih komplek. Namun demikian tidak diabaikan sama sekali. Pada buku ini, pengguna diwakili oleh keluaran yang diinginkan atau ditetapkan. Tanpa memperhatikan karakteristik pengguna atau kenyamanan pengguna dalam memakai aplikasi.

1.6 Rangkuman

Berpikir komputasi tidak hanya dibutuhkan dalam pengolahan data secara komputasi. Namun, berpikir komputasi dibutuhkan dalam kehidupan sehari-hari dengan permasalahan yang jauh lebih kompleks karena ada factor ketidakpastian. Tujuan dari membiasakan berpikir komputasi adalah untuk mengasah kepekaan seseorang untuk berpikir tuntas atau menyeluruh dalam menyelesaikan tugas dan tanggung jawabnya. Muara berpikir komputasi selalu ke arah perbaikan dan penyempurnaan keadaan saat ini. Namun demikian, kemampuan

berpikir komputasi dapat dilakukan dengan cara melatih siswa atau mahasiswa menyelesaikan persoalan pengolahan data komputasi secara sistematis. Mereka dilatih sistematika pengolahan data yang terdiri dari: komponen masukan, proses, dan keluaran.

1.7 Latihan Soal

Perhatikan uraian permasalahan beserta solusinya pada soal nomer 1 -10 berikut ini. Kemudian kerjakan dengan ketentuan seperti berikut.

a. Kelompokan menjadi dua, yaitu: masalah yang diselesaikan dengan paradigm berpikir komputasi dan bukan.
b. Berikan penjelasan secara sistematis di setiap jawaban.

1. Uraian_Masalah_Solusi_1.
   Firda diberi tanggung jawab membersihkan rumah oleh ibunya. Oleh karena itu, setiap pagi dia mengerjakan tanggung jawabnya sebelum berangkat kuliah. Setiap hari dia bangun jauh lebih awal supaya tanggung jawab tersebut bisa ditunaikannya dengan baik sekalipun ada hari tertentu dia harus sudah di kampus sebelum jam 07.00, karena ada jadwal kuliah yang dimulai dari jam 07.00.
2. Uraian_Masalah_Solusi_2.
   Yuritsuna pulang kuliah jam 13.00. Sampai di rumah dia dapati meja makan hanya ada nasi, tidak ada lauk. Saat itu ia sangat lapar. Kemudian, dia membuka kulkas. Dia menemukan ada telur ayam mentah, daging ayam mentah, tempe mentah, serta beberapa jenis sayuran juga masih mentah. Dia berpikir, sebaiknya dia buat telur ceplok untuk makan siangnya. Setelah itu barulah dia memasak lauk dan sayuran untuk kedua orang tuanya yang 3 jam lagi sampai rumah.
3. Uraian_Masalah_Solusi_3.
   Pada hari rabu siang, udara sangat panas. Alifan sedang menempuh perjalanan ke rumah neneknya yang jaraknya kurang lebih 10 km dengan bersepeda. Diperjalanan, ia melihat penjual es tebu dan makanan. Sebelum berangkat, dia tidak berpikir membawa minum walaupun dia tau bahwa udara sangat panas dan dia harus menempuh perjalanan yang cukup jauh dengan bersepeda. Karena merasa haus dan melihat penjual es tebu maka dia langsung mampir ke penjual tersebut dan menikmati es tebu beserta jajanan. Setelah selesai, dia membayar minuman dan makanan tersebut senilai Rp. 5.000,00. Dia membawa uang sebesar Rp. 5.000,00. Maka, habislah uangnya. Kemudian dia melanjutkan perjalanan. Belum sampai tujuan, ban sepedanya meletus. Karena dia sudah tidak punya uang untuk menambal ban, maka dia melanjutkan perjalanan dengan berjalan kaki sambil membawa sepeda di bawah udara yang sangat panas.
4. Uraian_Masalah_Solusi_4.
   Seorang dosen menyiapkan materi kuliah dalam bentuk power point. Dia berpikir bahwa setiap ruang kelas ada LCD. Kemudian sesampai di kelas dia menghidupkan computer dan LCD. Ternyata laptopnya baterainya habis. Dosen tersebut tidak membawa alat pengisi baterai dan juga tidak membawa spidol. Akhirnya, supaya kuliah tetap berlangsung dia memberikan kuliah dengan ceramah walaupun materinya tidak bisa dipahami tanpa ada visualisasi.
5. Uraian_Masalah_Solusi_5.
   Seorang mahasiswa tekun belajar dan rajin mengerjakan tugas kuliah. Dia juga tidak pernah terlambat mengikuti perkuliahan. Dia selalu hadir di ruang kuliah sebelum dosennya hadir. Tidak heran dia dapat menyelesaikan kuliahnya dengan indeks prestasi 3.9.
6. Uraian_Masalah_Solusi_6.
   Sulaiman adalah anak orang kaya. Dia kuliah di Perguruan Tinggi ternama di Kota Surabaya. Orang tuanya memberikan semua fasilitas yang dibutuhkan. Tanggung jawab yang diberikan oleh orang tuanya adalah menempuh studi saja. Dengan kondisi tersebut Sulaiman mengisi waktunya dengan menikmati semua fasilitas yang diberikan orang tuanya. Karena sakunya cukup banyak, dia sering meminta teman kuliahnya untuk mengerjakan tugas-tugas kuliah dengan imbalan uang atau lainya sesuai kesepakatan. Demikian juga pada saat ujian, dia selalu berusaha untuk mendapatkan nilai bagus dengan segala cara supaya indeks prestasinya di atas 3.

7. Uraian_Masalah_Solusi_7.
   Seorang guru matematika SD meminta siswa menyelesaikan soal berikut ini.

   1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20=?

   Dia mengamati siswanya selama mereka menyelesaikan soal tersebut. Semua siswa menghitung penjumlah dari awal sampai selesai secara berurutan. Kemudian guru tersebut mengajarkan menjumlahkan bilangan tersebut dengan cara yang lebih singkat seperti berikut.

   1+19=20
   2+18=20
   3+17=20
   4+16=20
   5+15=20
   6+14=20
   7+13=20
   8+12=20
   9+11=20

   Ada sebanyak 9 bilangan 20. Secara matematika dapat ditulis dengan 9 × 20 = 180. Dan tinggal dua bilangan yang belum dijumlahkan, yaitu: 10+20=30. Maka hasil penjumlahan keseluruhan adalah: 180+30= 210.

8. Uraian_Masalah_Solusi_8.
   Sebuah supermarket akan memberikan hadiah pada pelanggannya yang loyal. Selama ini transaksi pembelian menggunakan mesin kasir kecil seperti gambar berikut ini.

   Untuk itu, manager memutuskan akan memberikan 1 buah stamp pada setiap konsumen yang melakukan transaksi minimal senilai Rp. 200.000,00. Konsumen yang berhasil mengumpulkan stamp 10 buah akan mendapatkan bingkisan cantik. Ketentuan tersebut diberlakukan mulai bulan depan.

9. Uraian_Masalah_Solusi_9.
   Pemilik biro travel akan memberikan hadiah pada pelanggannya yang loyal. Selama ini transaksi menggunakan aplikasi perangkat lunak sederhana. Data **yang dicatat pada transaksi meliputi data pribadi konsumen sesuai de**ngan informasi yang terekam pada KTP, kota asal, kota tujuan, dan nilai transaksi. Untuk memenuhi keingiannya, dia memanggil kembali vendor yang membuatkan aplikasi tersebut. Kemudian minta ditambahkan laporan tentang konsumen dengan transaksi paling sedikit 10 kali dalam satu tahun. Berdasarkan laporan tersebut, pemilik biro travel akan memberikan hadiah pada konsumen tersebut pada transaksi ke-11.

10. Uraian_Masalah_Solusi_10.
    Guru matematika SD kelas 1 akan mengajar berhitung pada siswa yang baru masuk sekolah. Ibu Guru meminta siswa membawa biji sawo sebanyak 10 butir, tutup botol air minum kemasan 10 buah, dan biji salak 10 butir.

Perhatikan kembali uraian permasalahan beserta solusinya pada soal nomer 1 - 10 di atas. Kemudian kerjakan dengan ketentuan seperti berikut.

a. Pilihlah masalah dan solusi yang diselesaikan dengan paradigm pengolahan data secara komputasi.
b. Berikan penjelasan setiap jawaban secara sistematis menurut komponen masukan, proses, dan keluaran.

-oo0o-

BAB 2

PERNYATAAN MATEMATIKA DAN LOGIKA MATEMATIKA

Deskripsi Singkat

Bab 2 menjelaskan tentang pentingnya memahami pernyataan matematika dan logika matematika dalam algoritma komputasi. Pada bab 1, sudah dijelaskan definisi operasional berpikir komputasi, meliputi: (1) kemampuan mengidentifikasi masalah komputasi, (2) kemampuan mengidentifikasi komponen terkecil pengolahan data, yaitu masukan, proses, dan keluaran, (3) kemampuan membuat proposisi, (4) kemampuan membuat algoritma dalam bentuk *flowchart* dan *pseudocode*.

Untuk belajar tentang proposisi dan membuat algoritma dibutuhkan kemampuan membuat pernyataan matematika dan logika matematika. Diskusi pada bab ini menitikberatkan pada peningkatan kemampuan menulis pernyataan matematika dan logika matematika dalam algoritma.

Materi diskusi meliputi: pernyataan matematika dan logika matematika secara umum pada bidang ilmu lain, bahasan ini ditempatkan pada pendahuluan. Kemudian dilanjutkan dengan diskusi pernyataan matematika yang dipelajari pada aplikasi pembelajaran penalaran.com, serta logika matematika. Kemudian berturut-turut didiskusikan tentang: pernyataan matematika dalam komputasi, logika

matematika dalam komputasi. Diskusi ditutup dengan rangkuman, soal latihan dan daftar pustaka.

Pendahuluan, diawali dengan beberapa contoh pernyataan matematika dan logika matematika yang berkaitan dengan contoh pada bab 1. Kemudian dilanjutkan tentang pernyataan matematika dan logika matematika pada bidang ilmu lain. Pendahuluan diakhiri dengan menilai pernyataan matematika dan logika matematika.

Diskusi tentang pernyataan matematika dan logika matematika diambil dari aplikasi penalaran.com. Sistematika diskusi mengikuti komponen terkecil pengolah data, yaitu: masukan, proses, serta keluaran. Demikian juga diskusi untuk sub materi selanjutnya.

Tujuan Pembelajaran

Akhir pembelajaran atau diskusi, diharapkan pembaca mempunyai beberapa keterampilan berikut ini.

1. Jika diberikan pernyataan matematika, pembaca dapat menilai kebenaran pernyataan matematika tersebut.
2. Jika diberikan logika matematika, pembaca dapat memberikan contoh data sedemikian hingga logika matematika tersebut mempunyai nilai, benar atau salah.
3. Jika diberikan penggalan *pseudocode*, pembaca dapat memilih dengan benar pernyataan matematika atau logika matematika yang termuat dalam penggalan *pseudocode* tersebut.
4. Jika diberikan soal beserta jawabannya, pembaca dapat meng-identifikasi komponen masukan, proses, serta keluaran.

2.1 Pendahuluan

Hartati pada tahun 2017, menyatakan bahwa kemampuan dasar untuk belajar algoritma komputasi adalah penalaran matematika kreatif. Kemampuan dasar tersebut digunakan untuk menyusun atau membuat pernyataan matematika dan logika matematika sebagai bentuk lain atau model permasalahan yang sedang diselesaikan oleh pembuat

algoritma komputasi. penyajian pernyataan matematika dan logika matematika secara matematis mempunyai kesamaan. Namun keduanya mempunyai makna yang berbeda dalam pembuatan algoritma.

Supaya bisa membedakan kedua pernyataan matematika dan logika matematika dalam pembuatan algoritma, berikut ini diberikan contoh untuk hal tersebut.

Perhatikan kembali contoh yang disajikan pada bab 1, didiskusikan kembali seperti berikut ini.

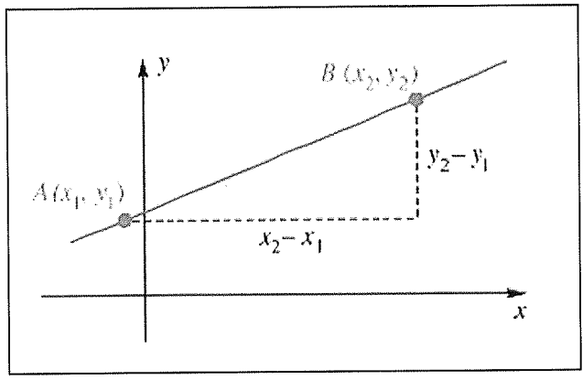
1. Transportasi *online*
   Urutan proses menggunakan transportasi *online* adalah sebagai berikut.
   a. Pengguna menghidup aplikasi.
   b. Memasukan tujuan, tepatnya alamat tujuan dengan lengkap. Terkadang, juga diminta memasukan alamat penjemputan.
   c. Menekan tombol pesan.
   d. Setelah beberapa saat, aplikasi akan menampilkan kendaraan yang akan menjemput beserta biayanya.

   Dari empat langkah yang diuraikan di atas, pernyataan matematika terletak pada urutan- b dan urutan-d. Pernyataan matematika pada urutan-b dituliskan sebagai berikut.

   A = posisi_tujuan.

   B = posisi_penjemputan.

   Makna pernyataan matematika di atas adalah: (1) posisi tujuan disimpan pada variable A, (2) posisi penjemputan disimpan pada variable B. Dalam pelajaran matematika, posisi dapat dinyatakan dengan koordinat kartesius seperti gambar 2.1. berikut ini.



Gambar 2.1 *Contoh Posisi A dan B pada Diagram Kartesius*

Secara diagram kartesius, nilai variabel A dan B adalah pasangan nilai (x,y) sebagaimana yang ditunjukan pada gambar 2.1. di atas. Pada gambar 2.1. di atas dicontohkan posisi A pada (x_1, y_1). Jika nilai x_1 adalah -1 dan nilai y_1 adalah 20, maka nilai variable A adalah (-1.20). demikian juga untuk variable B. Jika nilai x_2 adalah 30 dan nilai y_1 adalah 35, maka nilai variable B adalah (30,35). Dalam matematika, variable A dan B disebut dengan titik A dan titik B, masing-masing mempunyai koordinat (-1.20) dan (30,35).

Pernyataan matematika pada urutan-d dituliskan sebagai berikut.

Jarak = panjang_posisi_penjemputan_ke_tujuan. Biaya = tarif * Jarak.

Merujuk pada gambar 2.1. di atas, maka makna pernyataan di atas adalah sebagai berikut.

a. Jarak adalah nama variabel yang digunakan untuk menyimpan nilai dari panjang garis yang menghubungkan variable A dan B. Makna secara matematis, jarak adalah garis yang menghubungkan dua buah titik, dalam hal ini menghubungkan titik A dan titik B. Panjang garis dapat dihitung dengan menggunakan persamaan berikut.
$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Nilai d yang merupakan jarak antara titik A dan titik B disimpan pada variabel Jarak.

b. Biaya adalah nama variable yang digunakan untuk menyimpan nilai hasil perkalian antara variable jarak dan tarif.

Sedangkan logika matematika ada pada urutan-a dan urutan-c. Dalam Bahasa manusia, logika matematika tersebut dituliskan dengan kalimat "Menghidupkan aplikasi" dan "Menekan tombol pesan". Dalam penalaran matematika dua kalimat tersebut diganti dengan symbol tertentu yang menyatakan variable, misal variable "*a*" mewakili "Menghidupkan aplikasi" dan variable "*p*" mewakili "Menekan tombol pesan". Dua variable tersebut hanya punya dua nilai benar atau salah. Dalam algoritma, logika matematika tersebut dapat disajikan seperti berikut ini.

a = benar, atau a = salah. b = benar, atau b = salah.

2. Pembelian makanan secara online.
Urutan proses pembelian makanan secara online adalah sebagai berikut.

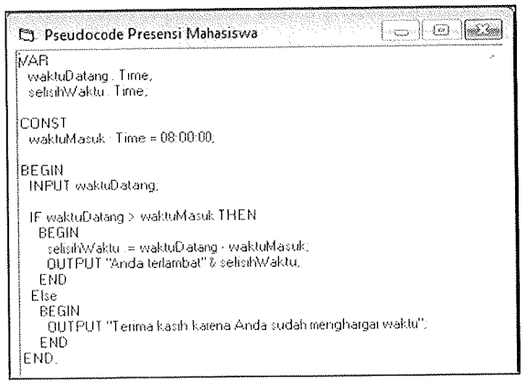
a. Pengguna menghidup aplikasi.
b. Memasukan makanan yang akan dibeli.
c. Aplikasi akan menampilkan beberapa alternative penjual.
d. Memilih penjual.
e. Aplikasi akan menampilkan beberapa alternative makanan.
f. Memilih makanan yang akan dibeli.
g. Menekan tombol pesan.
h. Aplikasi akan menampilkan nama pengemudi dan nomer kendaraan yang akan mengantar makanan yang dipesan beserta biayanya.

Tabel 2.1. berikut ini menunjukan penggunaan pernyataan matematika, logika matematika, serta perintah atau instruksi pada urutan langkah pembelian makanan online, serta cara penyajiannya dalam algoritma.

Tabel 2.1 *Penggunaan Pernyataan Matematika, Logika Matematika, serta Perintah atau Instruksi pada Algoritma*

Urutan	Aktivitas	Keterangan	Penyajian
a.	Pengguna menghidup aplikasi	logika matematika	Misal, nama variable adalah a. a = benar atau a = salah
b.	Memasukan makanan yang akan dibeli	Perintah atau instruksi	Misal, nama variable m[i]. Masukan m[i]
c.	Aplikasi akan menampilkan	Perintah atau instruksi	Misal, nama variable pj[i]. Tampilkan pj[i]
d.	Memilih penjual	logika matematika	Misal, nama variable adalah tpj. tpj = benar atau tpj = salah
e.	Aplikasi akan menampilkan beberapa alternative makanan	Perintah atau instruksi	Misal, nama variable mp[i]. Tampilkan mp[i]
f.	Memilih makanan yang akan dibeli	logika matematika	Misal, namavariable adalah tm. tm = benar atau tm = salah
g.	Menekan tombol pesan	logika matematika	Misal, nama variable adalah tpn. tpn = benar atau tpn = salah
h.	Aplikasi akan menampilkan nama pengemudi dan nomer kendaraan yang akan mengantar makanan yang	Instruksi, Pernyataan matematika	**Instruksi**: Tampilkan nama_pengemudi, dan nomer_kendaraan, **Pernyataan matematika.** Biaya = harga_makanan[i]*jml

1. Contoh lain disajikan dalam bentuk pseudo code pada gambar 2.2. berikut ini.



Gambar 2.2 *Contoh Pseudocode*

Pada gambar 2.2. di atas, yang merupakan pernyataan matematika, logika matematika, serta instruksi adalah sebagai berikut.

a. Instruksi, ada 3 buah instruksi, yaitu: *INPUT* waktuDatang, *OUTPUT* "Anda Terlambat", *OUTPUT* "Terimakasih karena Anda sudah menghargai waktu". Makna setiap instruksi di atas adalah sebagai berikut.
   1) *INPUT* waktuDatang, maknanya adalah sebuah perintah untuk memasukan nilai atau memberi nilai pada variable waktuDatang menggunakan media atau alat masukan, misal papan ketik atau layar sentuh atau lainnya.
   2) *OUTPUT* "Anda Terlambat", maknanya adalah tampilkan tulisan "Anda Terlambat" pada media atau alat keluaran, misal layar komputer.
   3) *OUTPUT* "Terimakasih karena Anda sudah menghargai waktu", maknanya adalah tampilkan tulisan "Terimakasih karena Anda sudah menghargai waktu" pada media atau alat keluaran, misal layar komputer
b. Logika matematika, ada 1 logika matematika: waktuDatang > waktuMasuk. Pada *pseudocode* di atas (gambar 2.2.), waktuDatang adalah variabel yang digunakan untuk menyimpan nilai jam kedatangan mahasiswa. Sedangkan variabel waktuMasuk adalah konstanta yang digunakan untuk menyimpan jam masuk mahasiswa dan diberi nilai 8. Nilai variabel waktuMasuk diambil dari jam pada sistem komputer. Berikut ini diberikan contoh kemungkinan nilai logika matematika waktuDatang > waktuMasuk.
   1) Jika waktuDatang diberi nilai 8, maka logika matematika waktuDatang > waktuMasuk bernilai SALAH. karena tidak benar 8 > 8.
   2) Jika waktuDatang diberi nilai 7, maka logika matematika waktuDatang > waktuMasuk bernilai SALAH. karena tidak benar 7 > 8.

3) Jika waktuDatang diberi nilai 9, maka logika matematika waktuDatang > waktuMasuk bernilai BENAR. karena benar bahwa 9 > 8.
4) Jika waktuDatang diberi nilai 8.1, maka logika matematika waktuDatang > waktuMasuk bernilai BENAR. karena benar bahwa 8.1 > 8

c. Pernyataan matematika, ada 1 pernyataan matematika yaitu: selisihWaktu = waktuDatang - waktuMasuk. Maknanya adalah variable selisihWaktu digunakan untuk menyimpan nilai selisih antara variable waktuDatang dan variabel waktuMasuk.

Dari tiga contoh di atas, dapat dibedakan antara pernyataan matematika, logika matematika, serta instruksi dalam algoritma komputasi. Pernyataan matematika disajikan dalam bentuk persamaan eksplisit, dan dinyatakan sebagai pernyataan tertutup. Berikut ini diberikan

a. $x = 5 + 2 * b$.
b. $y = 3*a - 5 * b + c$.
c. selisihWaktu = waktuDatang - waktuMasuk.
d. bayar = 10 * harga_cake + 5 * harga_esKrim + ongkos_kirim.

Logika matematika adalah pernyataan matematika yang hanya mempunyai dua nilai, yaitu benar atau salah. logika matematika fokus pada hasil perbandingan antara 2 nilai. Berikut ini diberikan beberapa contoh pernyataan matematika secara umum.

a. $x = 5 + 2 * b$. b. $x > 5 + 2 * b$
c. $x + y = 5 + 2 * b$
d. $y = 3*a - 5 * b + c$.
e. $y > 3*a - 5 * b + c$
f. $y + 5 * b = 3 * a + c$.
g. selisihWaktu = waktuDatang - waktuMasuk.
h. selisihWaktu < waktuDatang - waktuMasuk
i. bayar = 1000.
j. bayar >1000

Instruksi merupakan pernyataan atau kalimat perintah. Oleh karena itu pada algoritma komputasi instruksi hanya punya dua macam, yaitu: masukan dan tampilkan. Beberapa contoh instruksi pada algoritma komputasi adalah sebagai berikut.

a. INPUT waktuDatang.
b. OUTPUT ''Anda Terlambat''.
c. OUTPUT ''Terimakasih karena Anda sudah menghargai waktu''.
d. INPUT x, harga.
e. INPUT nim, nama, alamat.

Pernyataan matematika dan logika matematika keduanya bisa bernilai salah atau benar, namun tidak bisa bernilai keduanya pada saat yang bersamaan. Sedangkan instruksi tidak punya nilai. Berikut ini adalah contoh nilai dari pernyataan matematika dan logika matematika.

a. $x = 5 + 2 * b$.
   1) Nilai x sama dengan 10. Nilai b sama dengan 2. Maka $x = 5 + 2 * b$ bernilai SALAH. Pada kondisi ini, persamaan $x = 5 + 2 * b$ dipandang sebagai logika matematika, bukan pernyataan matematika.
   2) Nilai b sama dengan 2. Maka nilai x= 9. Pada kondisi ini, persamaan $x = 5 + 2 * b$ dipandang sebagai pernyataan matematika dan selalu bernilai BENAR untuk x dan b anggota bilangan riil.
b. $x > 5 + 2 * b$.
   1) Nilai x sama dengan 10. Nilai b sama dengan 2. Maka $x > 5 + 2 * b$ bernilai BENAR. Pertidaksamaan $x > 5 + 2 * b$ termasuk logika matematika.
   2) Nilai b sama dengan 2. Sedangkan x tidak diberi nilai, maka pertidaksamaan $x > 5 + 2 * b$ tidak bisa diberi nilai.
   3) Demikian juga untuk nilai x sama dengan 5 sedangkan nilai b tidak diketahui, maka pertidaksamaan $x > 5 + 2 * b$ tidak bisa diberi nilai.

c. $x + y = 5 + 2 * b$.
   1) Nilai x sama dengan 10. Nilai y sama dengan 30. Nilai b sama dengan 2. Maka persamaan $x + y = 5 + 2 * b$ bernilai SALAH. Pada contoh tersebut persamaan $x + y = 5 + 2 * b$ termasuk logika matematika.
   2) Nilai x sama dengan 9. Sedangkan y tidak diberi nilai. Nilai b sama dengan 2. Maka persamaan $x + y = 5 + 2 * b$ tidak bisa diberi nilai.

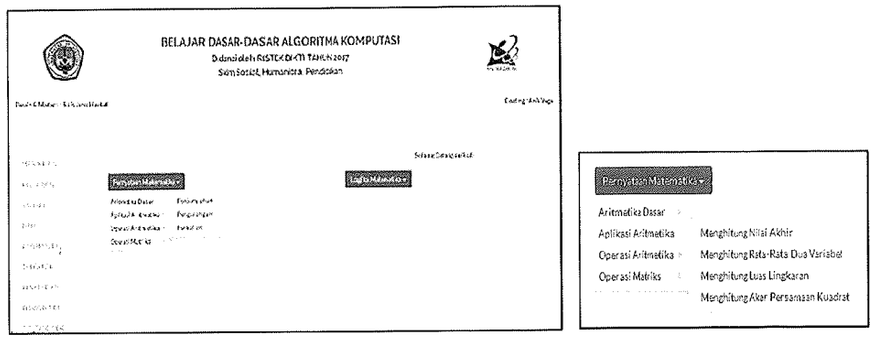
d. $y = 3*a - 5 * b + c$.
   1) Nilai y sama dengan 20. Nilai a sama dengan 10. Nilai b sama dengan 2. Dan nilai c=20 maka persamaan $y = 3*a - 5 * b + c$ bernilai salah. Pada contoh ini, persamaan $y = 3*a - 5 * b + c$ termasuk logika matematika.
   2) Nilai a sama dengan 10. Nilai b sama dengan 2. Dan nilai c = 20, Maka nilai $y = 40$. Dalam keadaan demikian persamaan $y = 3*a - 5 * b + c$ termasuk pernyataan matematika dan bernilai BENAR untuk y, a, b, dan c anggota bilangan riil.

e. Bayar = 1000. Persamaan ini tidak bisa digolongkan ke dalam pernyataan matematika atau logika matematika secara langsung. Untuk mengetahuinya dibutuhkan persamaan atau pertidaksamaan tambahan. Perhatikan beberapa keadaan berikut ini.
   1) Bayar = 1000, bayar = bayar + 10. Persamaan bayar = 1000 adalah pernyataan matematika.
   2) Jika bayar = 1000, maka bayar = 0.01 * bayar. Persamaan bayar = 1000 adalah logika matematika.

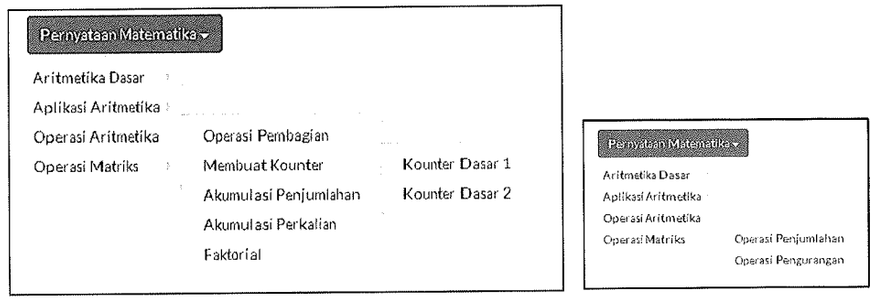
f. Bayar >1000. Nilai bayar sama dengan 200. Pertidaksamaan bayar >1000 bernilai SALAH. Pertidaksamaan bayar >1000 termasuk logika matematika.

2.2 Pernyataan Matematika

Berdasarkan pada uraian sebelumnya sudah diberikan beberapa contoh pernyataan matematika dan logika matematika. Uraian berikut ini menjelaskan pernyataan matematika yang ada pada aplikasi penalaran.com. Perhatikan gambar 2.3. berikut ini.



(a) Menu Utama **penalaran.com** dan Pernyataan Matematika untuk Aritmetika Dasar (b) Pernyataan Matematika untuk Aplikasi Aritmetika



(c) Pernyataan Matematika untuk Operasi Aritmetika (d) Pernyataan Matematika untuk Operasi Matriks

Gambar 2.3 *Materi Pernyataan Matematika pada* ***penalaran.com***

Pernyataan matematika untuk aritmetika dasar, terdapat pada gambar 3.(a), ada 3 materi yang dipelajari, yaitu: penjumlahan, pengurangan, serta perkalian. Dalam matematika, aritmetika dasar ada 4 materi, yaitu: penjumlahan, pengurangan, perkalian, serta pembagian. Terdapat perbedaan mendasar antara kepentingan pembelajaran matematika dan algoritma komputasi dasar. Untuk melakukan operasi penjumlahan, pengurangan, serta perkalian secara matematis tidak diperlukan persyaratan apapun. Artinya, pada ketiga operasi tersebut bilangan berapapun diperbolehkan untuk dilakukan operasi aritmetika. Perhatikan contoh yang disajikan pada tabel 2.2. berikut ini.

Tabel 2.2 *Contoh Penggunaan Operasi Aritmetika Dasar*

NO	OPERASI ARITMETIKA DASAR	CONTOH
1.	Penjumlahan	X = Y + Z a. Untuk Y = 5, Z = 10, maka X = 15. b. Untuk Y = 0, Z = 0, maka X = 0. c. Untuk Y = 0, Z = 10, maka X = 10. d. Untuk Y = -500, Z = 10, maka X = -490 e. Dan seterusnya silahkan dibuat contoh sendiri. Berapapun nilai variable Y dan Z dengan syarat Y dan Z bilangan riil maka nilai X pasti dapat dihitung.
2.	Pengurangan	X = Y - Z a. Untuk Y = 5, Z = 10, maka X = -5. b. Untuk Y = 0, Z = 0, maka X = 0. c. Untuk Y = 0, Z = 10, maka X = -10. d. Untuk Y = -500, Z = 10, maka X = -510 e. Dan seterusnya silahkan dibuat contoh sendiri. Berapapun nilai variable Y dan Z dengan syarat Y dan Z bilangan riil maka nilai X pasti dapat dihitung.
3.	Perkalian	X = Y * Z a. Untuk Y = 5, Z = 10, maka X = 50. b. Untuk Y = 0, Z = 0, maka X = 0. c. Untuk Y = 0, Z = 10, maka X = 0. d. Untuk Y = -500, Z = 10, maka X = -5000 e. Dan seterusnya silahkan dibuat contoh sendiri. Berapapun nilai variable Y dan Z dengan syarat Y dan Z bilangan riil maka nilai X pasti dapat dihitung.
4.	Pembagian	X = Y / Z a. Untuk Y = 5, Z = 10, maka X = 0.5. b. Untuk Y = 0, Z = 0, maka X = tidak terdefinisi atau tidak dapat dihitung. c. Untuk Y = 0, Z = 10, maka X = 0. d. Untuk Y = -500, Z = 10, maka X = -50 e. Dari keempat contoh tersebut terlihat contoh-b nilai X tidak terdefinisi. Artinya, untuk operasi pembagian tidak semua nilai variable Y dan Z dengan syarat Y dan Z bilangan riil maka nilai X pasti dapat dihitung. Nilai X tidak bisa dihitung jika Z = 0, atau nilai pembagi sama dengan 0.

Itu sebabnya, dalam pembelajaran algoritma komputasi dasar, operasi pembagian diperlakukan secara berbeda dengan operasi yang lainya. Dengan pertimbangan tersebut, maka operasi pembagian dimasukan ke dalam kelompok pernyatanyaan matematika yang ketiga, yaitu: operasi aritmetika.

Selanjutnya, bagaimana membuat pernyataan matematika dalam algoritma komputasi?. Untuk menjawab pertanyaan tersebut, diskusi dimulai dengan pernyataan matematika untuk kelompok aritmetika dasar. Perhatikan gambar 2.4. berikut ini, yang merupakan pseudo code aritmetika dasar. pseudo code berikut diambil dari **penalaran.com** dengan memilih tolls bar PSEUDOCODE.

```
PSEUDOCODE

penjumlahan;
deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;

Begin
Input BIL1, BIL2, TAMPIL;
HASIL = BIL1 + BIL2;
If TAMPIL = TRUE then print HASIL;
End.
```

```
PSEUDOCODE

pengurangan;
deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;

Begin
Input BIL1, BIL2, TAMPIL;
HASIL = BIL1 - BIL2;
If TAMPIL = TRUE then print HASIL;
End.
```

```
PSEUDOCODE

perkalian;
deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;

Begin
Input BIL1, BIL2, TAMPIL;
HASIL = BIL1 * BIL2;
If TAMPIL = TRUE then print HASIL;
End.
```

(a) Pseudo code penjumlahan (b) Pseudo code pengurangan (c) Pseudo code perkalian

Gambar 2.4 *Pseudo Code Aritmetika Dasar*

Adapun bentuk pseudocode dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Penjumlahan, Pengurangan, dan Perkalian dapat dilihat pada tabel di bawah ini.

1. Menu Penjumlahan

Pseudocode	Bahasa Pascal

penjumlahan; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
Begin
Input BIL1, BIL2, TAMPIL; HASIL =
BIL1 + BIL2;
If TAMPIL = TRUE then print HASIL;
End.
	var
	BIL1 : integer; BIL2 : integer;
	HASIL : integer;
	begin
	readln (BIL1); readln (BIL2);
	HASIL := BIL1 + BIL2;
	writeln(HASIL); end;

Pseudocode	Bahasa C++

penjumlahan; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; HASIL =
BIL1 + BIL2;
If TAMPIL = TRUE then print HASIL;
End.
	#include <stdio.h> using namespace
	std; int main() {
	int BIL1 = 10; int BIL2 = 20; int
	HASIL = 0;
	HASIL = BIL1 + BIL2;
	cout<<"Hasil : " << HASIL<<endl;
	return 0;
	}

Pseudocode	Bahasa Java

penjumlahan; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; HASIL =
BIL1 + BIL2;
If TAMPIL = TRUE then print HASIL;
End.
	public class penjumlahan {
	public static void main(String[] args)
	{ int BIL1 = 10;
	int BIL2 = 20; int HASIL = 0;
	HASIL = BIL1 + BIL2;
	System.out.println("Hasil : " +
	HASIL);
	}
	}

2. Menu Pengurangan

Pseudocode	Bahasa Pascal

pengurangan; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; HASIL =
BIL1 - BIL2;
If TAMPIL = TRUE then print HASIL;
End.
	var
	BIL1 : integer; BIL2 : integer;
	HASIL : integer;
	
	begin
	readln (BIL1); readln (BIL2);
	HASIL := BIL1 - BIL2;
	writeln(HASIL); end;

Pseudocode	Bahasa C++

pengurangan; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; HASIL =
BIL1 - BIL2;
If TAMPIL = TRUE then print HASIL;
End.
	#include <stdio.h> using namespace
	std; int main() {
	int BIL1 = 10; int BIL2 = 20; int
	HASIL = 0;
	HASIL = BIL1 - BIL2;
	cout<<"Hasil : " << HASIL<<endl;
	return 0;
	}

Pseudocode	Bahasa Java

pengurangan; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; HASIL =
BIL1 - BIL2;
If TAMPIL = TRUE then print HASIL;
End.
	public class pengurangan {
	public static void main(String[]
	args) { int BIL1 = 10;
	int BIL2 = 20; int HASIL = 0;
	HASIL = BIL1 - BIL2;
	System.out.println("Hasil : " +
	HASIL);
	}
	}

3. Menu Perkalian

Pseudocode	Bahasa Pascal
perkalian; deklarasi variable; BIL1, BIL2, HASIL: numerik; TAMPIL: Boolean; Begin Input BIL1, BIL2, TAMPIL; HASIL = BIL1 * BIL2; If TAMPIL = TRUE then print HASIL; End.	var BIL1 : integer; BIL2 : integer; HASIL : integer; begin readln (BIL1); readln (BIL2); HASIL := BIL1 * BIL2; writeln(HASIL); end;

Pseudocode	Bahasa C++

perkalian; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; HASIL =
BIL1 * BIL2;
If TAMPIL = TRUE then print HASIL;
End.
	#include <stdio.h> using namespace
	std; int main() {
	int BIL1 = 10; int BIL2 = 20; int
	HASIL = 0;
	HASIL = BIL1 * BIL2;
	cout<<"Hasil : " << HASIL<<endl;
	return 0;
	}

Pseudocode	Bahasa Java

perkalian; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; HASIL =
BIL1 * BIL2;
If TAMPIL = TRUE then print HASIL;
End.
	public class perkalian {
	public static void main(String[]
	args) { int BIL1 = 10;
	int BIL2 = 20; int HASIL = 0;
	HASIL = BIL1 * BIL2;
	System.out.println("Hasil : " +
	HASIL);
	}
	}

Pada gambar 2.4.(a) ditunjukkan bahwa pernyataan matematika untuk penjumlahan adalah: HASIL = BIL1 + BIL2;

Makna dari pernyataan matematika tersebut adalah, simpan hasil penjumlahan dari variable BIL1 dan BIL2 pada variabel HASIL. Demikian juga pernyataan matematika untuk pengurangan dan perkalian yang disajikan pada gambar 2.4.(b) dan gambar 2.4.(c), seperti berikut ini.

a. Pernyataan matematika untuk pengurangan adalah: HASIL = BIL1 - BIL2.
b. Pernyataan matematika untuk perkalian adalah: HASIL = BIL1 * BIL2.

Gambar 2.4. tersebut jika diidentifikasi ke dalam komponen pengolahan data menjadi seperti berikut ini.

a. Komponen masukan.
   Pada komponen masukan ada tiga instruksi yang dituliskan menjadi satu, yaitu: Input BIL1, BIL2, TAMPIL.

   Maknanya, pengguna diminta memasukan atau memberi nilai pada 3 buah variable, yaitu: BIL1, BIL2, TAMPIL melalui peralatan masukan.
b. Komponen proses.
   Pada komponen proses ada 1 pernyataan matematika di setiap pseudocode berikut ini.

   1) Penjumlahan: HASIL = BIL1 + BIL2.
   2) Pengurangan: HASIL = BIL1 - BIL2.
   3) Perkalian: HASIL = BIL1 * BIL2.
c. Komponen keluaran.
   Pada komponen keluaran ada satu instruksi, yaitu: Print HASIL.

   Maknanya adalah, mesin computer diminta menampilkan nilai yang disimpan atau ditampung pada variable HASIL.

Menghitung Nilai Akhir

PSEUDOCODE

```
nilai_akhir;
deklarasi variable;
nHadir, nAktif, nJujur, nTgs, nUts, nUas, nAkhir: numerik;
TAMPIL: Boolean;

Begin
Input nHadir, nAktif, nJujur, nTgs, nUts, nUas, TAMPIL;
nAkhir = 0.1*nHadir + 0.1*nAktif + 0.5*nJujur + 0.1*nTgs + 0.1*nUts +0.1* nUas;
If TAMPIL = TRUE then print nAkhir;
End.
```

Rata-Rata Dua Variabel

PSEUDOCODE

```
rata_dua_variable;
deklarasi variable;
BIL1, BIL2, RATA: numerik;
TAMPIL: Boolean;

Begin
Input BIL1, BIL2, TAMPIL;
RATA = (BIL1 + BIL2)/2;
If TAMPIL = TRUE then print RATA;
End.
```

(1) Pseudo code Menghitung Nilai Akhir (2) Pseudo code Rata-rata Dua Variabel

Gambar 2.5 *Pseudo Code Aplikasi Aritmetika*

Adapun bentuk pseudocode dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Menghitung Nilai Akhir dan Rata-rata Dua Variabel dapat dilihat pada tabel di bawah ini.

1. Menghitung Nilai Akhir

Pseudocode	Bahasa Pascal
nilai_akhir; deklarasi variable; nHadir, nAktif, nJujur, nTgs, nUts, nUas, nAkhir: numerik; TAMPIL: Boolean; Begin Input nHadir, nAktif, nJujur, nTgs, nUts, nUas, TAMPIL; nAkhir = 0.1*nHadir + 0.1*nAktif + 0.5*nJujur + 0.1*nTgs + 0.1*nUts +0.1* nUas; If TAMPIL = TRUE then print nAkhir; End.	var nHadir: real; nAktif: real; nJujur: real; nTgs: real; nUts: real; nUas: real; nAkhir: real; begin readln (nHadir); readln (nAktif); readln (nJujur); readln (nTgs); readln (nUts); readln (nUas); nAkhir := 0.1*nHadir + 0.1*nAktif + 0.5*nJujur + 0.1*nTgs + 0.1*nUts +0.1* nUas; writeln(nAkhir); end;

Pseudocode	Bahasa C++

nilai_akhir; deklarasi variable;
nHadir, nAktif, nJujur, nTgs, nUts,
nUas, nAkhir: numerik;
TAMPIL: Boolean;
 	
Begin
Input nHadir, nAktif, nJujur, nTgs,
nUts, nUas, TAMPIL;
nAkhir = 0.1*nHadir + 0.1*nAktif +
0.5*nJujur + 0.1*nTgs + 0.1*nUts +0.1*
nUas;
If TAMPIL = TRUE then print nAkhir;
End.
	#include <stdio.h> using
	namespace std; int main() {
	float nHadir = 100; float nAktif
	= 100; float nJujur = 70; float
	nTgs = 100; float nUts = 90;
	float nUas = 80; float nAkhir =
	0;
	nAkhir= (float)
	((0.1*nHadir)+
	(0.1*nAktif) +
	(0.5*nJujur)+(0.1*nTgs)+
	(0.1*nUts)+(0.1*nUas));
	cout<<"Hasil : " <<
	nAkhir<<endl; retun 0;

Pseudocode	Bahasa Java

nilai_akhir; deklarasi variable;
nHadir, nAktif, nJujur, nTgs, nUts,
nUas, nAkhir: numerik;
TAMPIL: Boolean;
 	
Begin
Input nHadir, nAktif, nJujur, nTgs,
nUts, nUas, TAMPIL;
nAkhir = 0.1*nHadir + 0.1*nAktif +
0.5*nJujur + 0.1*nTgs + 0.1*nUts +0.1*
nUas;
If TAMPIL = TRUE then print nAkhir;
End.
	public class NilaiAkhir {
	public static void main(String[]
	args)
	{
	float nHadir = 100; float nAktif =
	100; float nJujur = 70; float nTgs
	= 100; float nUts = 90; float nUas
	= 80; float nAkhir = 0;
	nAkhir = (float)
	((0.1*nHadir)+ (0.1*nAktif)
	+
	(0.5*nJujur)+(0.1*nTgs)+(0.1*nUts)+(
	0.1*nUas));
	System.out.println("Hasil :
	" + nAkhir);
	}
	}

2. Menghitung Rata-rata dua variabel

Pseudocode	Bahasa Pascal

rata_dua_variable; deklarasi
variable;
BIL1, BIL2, RATA: numerik; TAMPIL:
Boolean;
Begin
Input BIL1, BIL2, TAMPIL; RATA =
(BIL1 + BIL2)/2;
If TAMPIL = TRUE then print RATA;
End.
	var
	BIL1 : integer; BIL2 : integer;
	RATA : integer;
	begin
	readln (BIL1); readln (BIL2);
	RATA := (BIL1 + BIL2)/2;
	writeln(RATA); end;

Pseudocode	Bahasa C++

rata_dua_variable; deklarasi
variable;
BIL1, BIL2, RATA: numerik; TAMPIL:
Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; RATA =
(BIL1 + BIL2)/2;
If TAMPIL = TRUE then print RATA;
End.
	#include <stdio.h> using
	namespace std; int main() {
	int BIL1 = 10; int BIL2 = 20; int
	RATA = 0;
	RATA = (BIL1 + BIL2)/2;
	cout<<"Rata-rata : " <<
	RATA<<endl; return 0;
	}

Pseudocode	Bahasa Java

rata_dua_variable; deklarasi
variable;
BIL1, BIL2, RATA: numerik; TAMPIL:
Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; RATA =
(BIL1 + BIL2)/2;
If TAMPIL = TRUE then print RATA;
End.
	public class RataRata {
	public static void main(String[]
	args) { int BIL1 = 10;
	int BIL2 = 20; int RATA = 0;
	RATA = (BIL1 + BIL2)/2;
	System.out.println("Rata-rata : "
	+ RATA);
	}
	}

Pernyataan matematika pada kedua *pseudocode* yang ditunjukkan pada gambar 2.5.(a) dan gambar 2.5.(b) masing-masing adalah sebagai berikut.

a. Pernyataan matematika untuk Menghitung Nilai Akhir adalah:
nAkhir = 0.1*nHadir + 0.1*nAktif + 0.5*nJujur + 0.1*nTugas + 0.1*nUts + 0.1*nUas.

Maknanya adalah, variabel nAkhir diperoleh dari perhitungan di atas. Fungsi variabel nAkhir adalah menampung nilai akhir. Persamaan di atas diperoleh dari pengguna. Dalam hal ini adalah aturan yang diberlakukan pada institusi pendidikan tertentu. Contoh hasil perhitungan nAkhir adalah sebagai berikut.

1) Jika nHadir = 100, nAktif = 70, nJujur = 100, nTugas = 100, nUts = 85, dan Uas = 90, maka nAkhir = 86.
2) Jika nHadir = 100, nAktif = 90, nJujur = 0, nTugas = 100, nUts = 90, dan Uas= 90, maka nAkhir = 38.

b. Pernyataan matematika untuk Menghitung Rata-rata Dua Variabel adalah: RATA = (BIL1 + BIL2)/2.
Maknanya adalah, variabel RATA digunakan untuk menampung hasil perhitungan nilai rata-rata dua buah variabel. Contoh hasil pernyataan matematika untuk menghitung nilai rata-rata adalah sebagai berikut.

1) Jika BIL1 = 100, BIL2 = 50, maka RATA = 75.
2) Jika BIL1 = 90, BIL2 = 90, maka RATA = 90.

Hasil identifikasi gambar 2.5.(a) di atas ke dalam komponen pengolahan data adalah seperti berikut ini.

a. Komponen masukan.
Pada komponen masukan ada 7 buah instruksi yang dituliskan menjadi satu, yaitu: Input nHadir, nAktif, nJujur, nTgs, nUts, nUas, dan TAMPIL.

Maknanya, pengguna diminta memasukan atau memberi nilai pada 7 buah variabel tersebut, yaitu: nHadir, nAktif, nJujur, nTgs, nUts, nUas, dan TAMPIL melalui peralatan masukan.

b. Komponen proses
Pada komponen proses ada satu pernyataan matematika, yaitu:

nAkhir = 0.1*nHadir + 0.1*nAktif + 0.5*nJujur + 0.1*nTugas + 0.1*nUts + 0.1*nUas. Maknanya, mesin komputer diperintah menghitung nilai akhir dan ditampung pada variabel nAkhir dengan aturan yang disajikan pada pernyataan matematika.

c. Komponen keluaran
Pada komponen keluaran ada satu instruksi, yaitu: Print nAkhir.

Maknanya adalah, mesin komputer diminta menampilkan nilai yang disimpan atau ditampung pada variabel nAkhir pada layar atau peralatan keluaran lainnya.

Sedangkan, hasil identifikasi Gambar 2.5.(b) ke dalam komponen pengolahan data adalah sebagai berikut.

a. Komponen masukan.
Pada komponen masukan ada tiga instruksi yang dituliskan menjadi satu, yaitu: Input BIL1, BIL2, TAMPIL.

Maknanya, pengguna diminta memasukan atau memberi nilai pada 3 buah variabel, yaitu: BIL1, BIL2, TAMPIL melalui peralatan masukan.

b. Komponen proses.
Pada komponen proses ada satu pernyataan matematika, yaitu: RATA = (BIL1 + BIL2)/2.

Maknanya, mesin komputer diperintah menghitung nilai rata-rata yang ditampung pada variabel RATA dengan aturan perhitungan adalah nilai RATA diperoleh dari menjumlahkan nilai BIL1 dan BIL2, kemudian hasilnya dibagi dengan bilangan 2.

c. Komponen keluaran.
Pada komponen keluaran ada satu instruksi, yaitu: Print RATA.

Maknanya adalah, mesin komputer diminta menampilkan nilai yang disimpan atau ditampung pada variabel RATA.

Luas Lingkaran

PSEUDOCODE

```
luas_lingkaran;
deklarasi variable;
  jariLingkaran, luasLingkaran: numerik;
  TAMPIL: Boolean;
  KONSTANTA: phi=3.14;
Begin
  Input jariLingkaran, TAMPIL;
  luasLingkaran = phi* jariLingkaran^2;
  If TAMPIL = TRUE then print luasLingkaran;
End.
```

(a) pseudo code menghitung luas lingkaran

Akar Persamaan Kuadrat

$ax^2+bx+c=0$

PSEUDOCODE

```
akar_persamaan_kuadrat;
deklarasi variable;
  a, b, c, x1, x2: numerik;
  TAMPIL: Boolean;

Begin
  Input a, b, c, TAMPIL;
  x1 = (-b +( b^2 – ((4*a*c)^0.5)))/ (2*a)
  x2 = (-b -( b^2 – ((4*a*c)^0.5)))/ (2*a)

  If TAMPIL = TRUE then print x1, x2;
End.
```

(b) pseudo code akar persamaan kuadrat

Gambar 2.6 *Pseudocode untuk Menghitung Luas Lingkaran dan Akar Persamaan Kuadrat*

Adapun bentuk *pseudocode* dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Menghitung Luas Lingkaran dan Akar Persamaan Kuadrat dapat dilihat pada tabel di bawah ini.

1. Menghitung Luas Lingkaran

Pseudocode	Bahasa Pascal
luas_lingkaran; deklarasi variable; jariLingkaran, luasLingkaran: numerik; TAMPIL: Boolean; KONSTANTA: phi=3.14; Begin Input jariLingkaran, TAMPIL; luasLingkaran = phi* jariLingkaran^2; If TAMPIL = TRUE then print luasLingkaran; End.	var jariLingkaran: real; luasLingkaran: real; begin phi := 3.14; readln (jariLingkaran); luasLingkaran = phi* jariLingkaran * jariLingkaran; writeln(luasLingkaran); end;

Pseudocode	Bahasa C++
luas_lingkaran; deklarasi variable; jariLingkaran, luasLingkaran: numerik; TAMPIL: Boolean; KONSTANTA: phi=3.14;	#include <stdio.h> using namespace std; int main() { double jariLingkaran = 10; double luasLingkaran = 0;

Pseudocode	Bahasa C++
Begin Input jariLingkaran, TAMPIL; luasLingkaran = phi* jariLingkaran^2; If TAMPIL = TRUE then print luasLingkaran; End.	luasLingkaran = Math.PI*(Math.pow (jariLingkaran, 2)); cout << "Luas Lingkaran : " << luasLingkaran << endl; retun 0; }

Pseudocode	Bahasa Java

luas_lingkaran; deklarasi variable;
jariLingkaran, luasLingkaran:
numerik;
TAMPIL: Boolean; KONSTANTA:
phi=3.14;
Begin
Input jariLingkaran, TAMPIL;
luasLingkaran = phi*
jariLingkaran^2;
If TAMPIL = TRUE then print
luasLingkaran;
End.
	public class Luas Lingkaran {
	public static void main(String[]
	args) { double jariLingkaran =
	10; double luasLingkaran = 0;
	luasLingkaran =
	Math.PI*(Math.pow(jariLingkaran,
	2)); System.out.println("Luas
	Lingkaran : " + luasLingkaran);
	}
	}

2. Menghitung Akar Persamaan Kuadrat

Pseudocode	Bahasa Pascal
akar_persamaan_kuadrat; deklarasi variable; a, b, c, x1, x2: numerik; TAMPIL: Boolean; Begin Input a, b, c, TAMPIL; x1 = (-b +(b^2 - ((4*a*c)^0.5)))/ (2*a) x2 = (-b -(b^2 - ((4*a*c)^0.5)))/ (2*a) If TAMPIL = TRUE then print x1, x2; End.	var a: real; b: real; c: real; x1 :real; x2 : real begin readln (a); readln (b); readln (c); x1 := (-(b) + sqrt((b*b) - (4*a*c)))/ (2*a); x2 := (-(b) - sqrt((b*b) - (4*a*c)))/ (2*a); writeln ('X1 : ' , x1); writeln ('X2 : ' , x2); End;

Pseudocode	Bahasa C++

akar_persamaan_kuadrat; deklarasi
variable;
a, b, c, x1, x2: numerik; TAMPIL:
Boolean;
Begin
Input a, b, c, TAMPIL;
x1 = (-b +(b^2 - ((4*a*c)^0.5)))/
(2*a)
x2 = (-b -(b^2 - ((4*a*c)^0.5)))/
(2*a)
If TAMPIL = TRUE then print x1, x2;
End.
	#include <stdio.h> using
	namespace std; int main() {
	double a = 2; double b = 3;
	double c = -2; double x1, x2;
	x1 = (-(b) +(Math.sqrt((Math.pow
	(b,2))
	- ((4*a*c))))/ (2*a)
	x2 = (-(b) - (Math.sqrt((Math.pow
	(b,2))
	- ((4*a*c))))/ (2*a) cout<<"X1 : "
	<< x1<<endl; cout<<"X2 : " <<
	x2<<endl; return 0;
	}

Pseudocode	Bahasa Java

akar_persamaan_kuadrat; deklarasi
variable;
a, b, c, x1, x2: numerik; TAMPIL:
Boolean;
 	
Begin
Input a, b, c, TAMPIL;
x1 = (-b +(b^2 - ((4*a*c)^0.5)))/
(2*a)
x2 = (-b -(b^2 - ((4*a*c)^0.5)))/
(2*a)
 	
If TAMPIL = TRUE then print x1, x2;
End.
	public class AkarPersamaanKuadrat
	{
	public static void main(String[]
	args)
	{
	double a = 2; double b = 3; double
	c = -2; double x1, x2;
	x1 = (-(b) +(Math.sqrt((Math.pow
	(b,2))
	- ((4*a*c))))/ (2*a)
	x2 = (-(b) - (Math.sqrt((Math.pow
	(b,2))
	- ((4*a*c))))/ (2*a)
	System.out.println("X1 : " + x1);
	System.out.println("X2 : " + x2);
	}
	}

Materi pernyataan matematika untuk aplikasi aritmetika yang lain pada **penalaran.com** adalah menghitung luas lingkaran dan akar persamaan kuadrat. *Pseudocode* dari kedua materi tersebut disajikan pada gambar 2.6.

Pernyataan matematika pada gambar 2.6.(a) hanya ada satu, yaitu: luasLingkaran = phi * jariLingkaran^2.

Makna dari pernyataan matematika tersebut adalah, variabel luasLingkaran diperoleh dengan cara melakukan perkalian antara konstanta phi dan nilai jariLigkaran dikuadratkan. Pernyataan matematika tersebut diperoleh dari ilmu matematika, khususnya materi geometri. Contoh hasil perhitungan luasLingkaran adalah sebagai berikut.

a. Nilai konstanta phi sifatnya tetap, yaitu 3.14. Jika nilai variabel jariLingkaran = 10, maka nilai variabel luasLingkaran = 314.
b. Jika nilai variabel jariLingkaran = 1, maka nilai variabel luasLingkaran = 3.14..

Sedangkan pernyataan matematika untuk gambar 2.6.(b) ada dua persamaan, yaitu:

a. x1 = (-b+(b^2 – ((4*a*c)^0.5)))/(2*a), dan
b. x2 = (-b-(b^2 – ((4*a*c)^0.5)))/(2*a).

Makna dari pernyataan matematika tersebut adalah, variabel $x1$ adalah akar pertama dari persamaan kuadrat $ax^2 + bx + c = 0$. Sedang variabel $x2$ adalah akar keduanya. Untuk mendapatkan nilai $x1$ dan $x2$ digunakan rumus abc sebagaimana pernyataan matematika tersebut. Contoh hasil perhitungan pernyataan matematika tersebut adalah sebagai berikut.

a. Jika nilai variabel a = 1, b = 2, dan c = 1, maka diperoleh nilai variabel x1 = x2 = -1.
b. Jika nilai variabel a = 1, b = 0, dan c = -1, maka diperoleh nilai variabel x1 = 1 dan x2 = -1 Atau sebaliknya, yaitu x1 = -1 dan x2 = 1.

Selanjutnya melakukan identifikasi gambar 2.6.(a) di atas ke dalam komponen pengolahan data. Hasilnya adalah seperti berikut ini.

a. Komponen masukan.
   Komponen masukan mempunyai 2 buah instruksi yang dituliskan menjadi satu, yaitu: Input jariLingkaran, dan TAMPIL.

   Maknanya, nilai variabel jariLingkaran, dan TAMPIL dimasukan melalui peralatan masukan.
b. Komponen proses.
   Pada komponen proses ada satu pernyataan matematika, yaitu: luasLingkaran = phi * jariLingkaran^2.

   Maknanya, nilai variabel luasLingkaran dihitung oleh mesin computer dengan aturan perhitungan sebagaimana yang disajikan pada pernyataan matematika tersebut.
c. Komponen keluaran.
   Pada komponen keluaran ada satu instruksi, yaitu: Print luasLingkaran.

   Maknanya adalah, mesin komputer diminta menampilkan nilai yang disimpan atau ditampung pada variable luasLingkaran pada layar.

Sedangkan, hasil identifikasi Gambar 2.6.(b) ke dalam komponen pengolahan data adalah sebagai berikut.

a. Komponen masukan.
   Pada komponen masukan ada empat instruksi yang dituliskan menjadi satu, yaitu: Input a, b, c, dan TAMPIL.

   Maknanya, pengguna diminta memasukan nilai pada 4 buah variable, yaitu: a, b, c, dan TAMPIL melalui peralatan masukan.
b. Komponen proses.
   Pada komponen proses ada dua pernyataan matematika, yaitu persamaan (1) dan (2) berikut, (1) x1 = (-b+(b^2 - ((4*a*c) ^ 0.5)))/(2*a), dan (2) $x2$ = (-b-(b^2 – ((4*a*c) ^ 0.5)))/(2*a). Maknanya adalah nilai akar persamaan kuadrat ke-1 dihitung oleh mesin komputer menggunakan persamaan (1) dan nilai akar persamaan kuadrat ke-2 menggunakan persamaan (2). Masing-masing disimpan dalam variabel $x1$ dan $x2$.
c. Komponen keluaran.
   Pada komponen keluaran terdapat dua instruksi, yaitu: Print $x1$, dan $x2$.

   Maknanya adalah, mesin komputer diminta menampilkan nilai yang ditampung pada variabel $x1$ dan $x2$.

Contoh di atas menunjukkan bahwa, dalam algoritma komputasi, pernyataan matematika untuk aritmetika dasar berada pada komponen proses. Demikian juga, untuk aplikasi aritmetika, pernyataan matematika berada pada komponen proses. Sedangkan penyajian pernyataan matematika dalam algoritma komputasi bentuk persamaan aljabar secara eksplisit.

2.3 Logika Matematika

Pada diskusi sebelumnya sudah diberikan beberapa contoh logika matematika. Berdasarkan contoh-contoh tersebut dapat diketahui bahwa penyajian logika matematika bisa sama dengan penyajian pernyataan matematika. Namun demikian, ada juga yang berbeda penyajiannya.

Logika matematika, ada yang memberi istilah logika proposisi. Artinya menilai kebenaran proposisi. Di atas sudah disampaikan bahwa proposisi adalah kalimat deklaratif yang hanya bernilai benar atau salah, dan tidak bisa keduanya. Contoh, jika benar hari ini adalah hari Senin, maka pasti hari ini bukan hari Selasa. Contoh yang lain adalah, anggota himpunan bilangan genap adalah semua bilangan yang punya sisa hasil bagi 0 (nol) jika dibagi dengan bilangan 2. Secara matematis ditulis dengan symbol seperti berikut ini.

$e = 2 * x$; untuk semua x anggota bilangan bulat.

e adalah bilangan genap, proposisi ini bernilai benar. $o = 2 * x + 1$; untuk semua x anggota bilangan bulat. 0 adalah bilangan genap, proposisi ini bernilai salah.

Tabel 2.3.berikut ini adalah tabel kebenaran logika matematika sesuai dengan operator yang digunakan.

Tabel 2.3 *Aturan Tabel Kebenaran*

NO	x(misal x adalah a=5)	NOT x
1.	a-5 (TRUE)	FALSE
2.	A-11 (FALSE)	TRUE

(a) Operator NOT

NO	x	y	xy and	NOTx	NOTy	NOTx *and* NOT y
1.	T	T	T	F	F	F
2.	T	F	F	F	T	F
3.	F	T	F	T	F	F
4.	F	F	F	T	T	T

(c) Operator and

NO	y(misal x adalah TAMPIL	NOT y
1.	(TRUE)	FALSE
2.	(FALSE)	TRUE

(b) Contoh lain penggunaan operator NOT

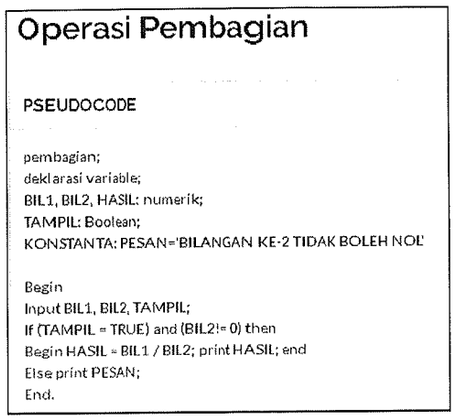
NO	x	y	x or y	NOTx	NOTy	NOTx *or* NOT y
1.	T	T	T	F	F	F
2.	T	F	T	F	T	T
3.	F	T	T	T	F	T
4.	F	F	F	T	T	T

(d) Operator or

Dari tabel 2.3. di atas, pembaca diberikan pedoman bahwa operator *and* akan menghasilkan nilai TRUE jika semua proposisi yang direlasikan bernilai *true*. Sebaliknya, untuk operator *or* hasil proposisi gabungan akan bernilai TRUE kalua satu diantara sekian proposisi yang digabungkan ada yang bernilai TRUE. Sehingga tabel 2.3 di atas dapat dikembangkan penggunaannya untuk gabungan 3 buah proposisi atau lebih.

Sebuah persamaan termasuk logika matematika atau pernyataan matematika dalam algoritma pemrograman ada sedikit perbedaan

dengan dalam pelajaran matematika. Untuk meningkatkan pemahaman logika matematika berikut ini dibahas dua contoh tentang logika matematika dengan stimulus **penalaran.com.** Untuk itu contoh kesatu, perhatikan gambar 2.7 berikut ini.



Gambar 2.7 *Operasi Pembagian*

Adapun bentuk pseudocode dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Menghitung Pembagian dapat dilihat pada tabel di bawah ini.

1. Operasi Pembagian

Pseudocode	Bahasa Pascal

pembagian; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
KONSTANTA: PESAN='BILANGAN KE- 2
TIDAK BOLEH NOL'
 	
Begin
Input BIL1, BIL2, TAMPIL;
If (TAMPIL = TRUE) and (BIL2!= 0)
then
Begin HASIL = BIL1 / BIL2; print
HASIL; end
Else print PESAN; End.
	var
	BIL1 : integer; BIL2 : integer;
	HASIL : integer;
	
	begin
	readln (BIL1); readln (BIL2);
	PESAN : String; begin
	PESAN := 'BILANGAN KE-2 TIDAK
	BOLEH NOL';
	if(BIL2 = 0)
	begin
	writeln (PESAN); end
	Else begin
	HASIL := BIL1 / BIL2;
	writeln('HASIL : ' , HASIL);
	end; end;

Pseudocode	Bahasa C++ 2.3

pembagian; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
KONSTANTA: PESAN='BILANGAN KE- 2
TIDAK BOLEH NOL'
Begin
Input BIL1, BIL2, TAMPIL;
If (TAMPIL = TRUE) and (BIL2!= 0)
then
Begin HASIL = BIL1 / BIL2; print
HASIL; end
Else print PESAN; End.
	#include <stdio.h> using namespace
	std; int main() {
	int BIL1 = 400; int BIL2 = 20; int
	HASIL = 0;
	String PESAN = "BILANGAN KE-2
	TIDAK BOLEH NOL";
	If(BIL2 == 0) cout<<PESAN<<endl;
	Else {
	HASIL = BIL1 / BIL2; Cout<<"HASIL :
	" << HASIL<<endl;
	}
	return 0;
	}

Pseudocode	Bahasa Java

pembagian; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
KONSTANTA: PESAN='BILANGAN KE-2
TIDAK BOLEH NOL'
 	
Begin
Input BIL1, BIL2, TAMPIL;
If (TAMPIL = TRUE) and (BIL2!= 0)
then
Begin HASIL = BIL1 / BIL2; print
HASIL; end
Else print PESAN; End.
	public class Pembagian {
	public static void main(String[]
	args) { int BIL1 = 400;
	int BIL2 = 20; int HASIL = 0;
	String PESAN = "BILANGAN KE-2
	TIDAK BOLEH NOL";
	If(BIL2 == 0)
	System.out.println(PESAN); Else {
	HASIL = BIL1 / BIL2;
	System.out.println("HASIL :
	" + HASIL);
	}
	}
	}

Gambar 2.7 adalah pseudo code operasi pembagian. Hasil identifikasi pada Gambar 2.7 di atas ke dalam komponen pengolahan data, adalah sebagai berikut.

a. Komponen masukan.
   Pada komponen masukan terdapat tiga buah instruksi yang dituliskan menjadi satu, yaitu: Input BIL1, BIL2, dan TAMPIL.

   Maknanya, pengguna diminta memasukan nilai variable BIL1, BIL2, dan TAMPIL melalui peralatan masukan.
b. Komponen proses.
   Pada komponen proses ada satu logika matematika dan satu pernyataan matematika, yaitu:

1) Logika matematika adalah: **if** (TAMPIL=TRUE) and (BIL2 != 0) **then.**
2) Pernyataan matematika adalah: HASIL = BIL1/BIL2.

Makna dari kedua hal di atas adalah sebagai berkut.

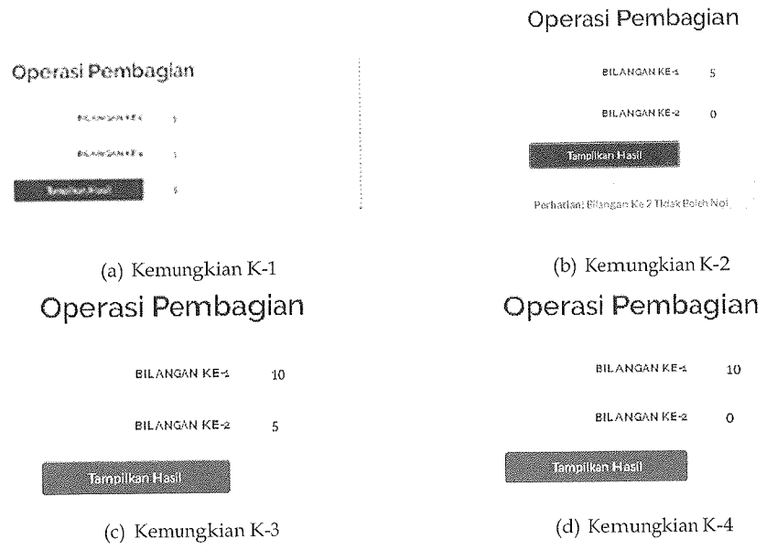
1) Logika matematika adalah: **if** (TAMPIL=TRUE) and (BIL2 != 0) **then.**
   Pada logika matematika di atas, memuat 2 buah proposisi, yaitu: (1) TAMPIL = TRUE, (2) BIL2 != 0. Penghubung kedua proposisi adalah operator **and**. Penyajian dalam algoritma, selalu diawali dengan kata **if** dan diakhiri dengan kata **then**. Aturan pemberian nilai pada logika matematika tersebut disajikan pada tabel 2.4. berikut ini.

Tabel 2.4 *Contoh Kemungkinan Nilai Logika Matematika Gambar 2.7(a)*

KEMUNGKINAN	TAMPIL=TRUE	BIL2!=0	(TAMPIL=TRUE) and (BIL2!=0)
K-1	TRUE	1 (TRUE)	TRUE
K-2	TRUE	0 (FLASE)	FALSE
K-3	FALSE	5 (TRUE)	FALSE
K-4	FALSE	0 (FLASE)	FALSE

   Logika matematika di atas akan menjalankan perintah atau istruksi yang terletak setelah kata **then.** Instruksi atau perintah setelah kata then ditulis dalam bentuk pernyataan matematika, yaitu: HASIL = BIL1/BIL2.

   Aturan pemberian nilai pada tabel 2.4 di atas mengikuti aturan pada logika matematika, sebagaimana pada tabel 2.3. di atas.
2) Pernyataan matematika adalah: HASIL = BIL1/BIL2.
   Pernyataan matematika pada gambar 2.7.(a) dinyatakan dalam bentuk proposisi pembagian dua buah bilangan,yaitu: HASIL = BIL1/BIL2.



(a) Kemungkian K-1 (b) Kemungkian K-2

(c) Kemungkian K-3 (d) Kemungkian K-4

Gambar 2.8 *Kemungkinan Hasil Pasangan (TAMPIL=TRUE dan BIL2!=0)*

Gambar 2.8. di atas menunjukan 4 kemungkinan pasangan hasil proposisi gabungan (TAMPIL= TRUE **and** BIL2!=0). Proposisi ini akan dijalankan jika logika matematika bernilai benar. Pada tabel 2.5 di atas di contohkan jika TAMPIL=TRUE dan BIL2=1, maka logika matematika tersebut bernilai benar. Hasilnya disajikan pada gambar 2.8.

Pertanyaannya adalah, apakah ada kemungkinan pasangan hasil yang kelima? Jawabannya adalah tidak ada. Karena dari dua buah prosisi maka kemungkinan hasil pasangannya adalah 4 (2^2). Kalau pasangan data bisa lebih dari 4, namun pasangan kemungkinan hasil hanya ada 4. Jika proposisi yang digabung ada 3, maka kemungkinan pasangan hasil sebanyak 8 (2^3).

Gambar 2.8.(a) merupakan kemungkinan kesatu (K-1) dari hasil gambar 2.7.(a) dengan kemungkinan pasangan: TAMPIL=TRUE (TRUE); BIL2=1, yang bermakna BIL2 tidak sama dengan 0 ditulis dengan symbol BIL2!=0 (TRUE). Sehinga TRUE and TRUE, maka hasilnya adalah TRUE. Disajikan pada gambar 2.8.(a).

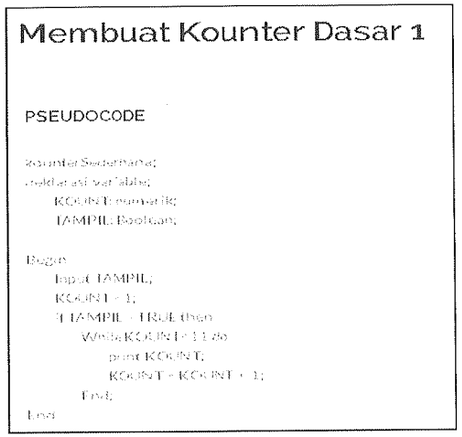
Gambar 2.8.(b) merupakan kemungkinan kedua (K-2) dari gambar 2.7.(a) dengan kemungkinan pasangan: TAMPIL=TRUE (TRUE); BIL2=0, bermakna symbol BIL2!=0 bernilai salah (FALSE). Sehinga TRUE and FALSE, maka hasilnya adalah FALSE. Disajikan pada gambar 2.8.(b). Ditampilkan pesan "Bilangan Kedua Tidak Boleh Nol" terjadi jika TAMPIL bernilai TRUE.

Itú sebanya, jika TAMPIL bernilai FALSE, baik BIL2!=0 bernilai TRUE maupun FALSE tidak ada perubahan apapun pada layar. Mesin computer menunggu TAMPIL dipilih oleh pengguna supay bernilai TRUE.

c. Komponen keluaran

Terdapat dua buah perintah pada komponen keluaran, yaitu: print HASIL dan print pesan dalam bentuk tulisan, yaitu: "Bilangan Kedua Tidak Boleh Nol". Print HASIL dikerjakan jika TAMPIL=TRUE dan BIL2!=0 juga bernilai TRUE. Jika TAMPIL=TRUE dan BIL2!=0 juga bernilai FALSE maka intruksi yang dikerjakan di komponen keluaran adalah print "Bilangan Kedua Tidak Boleh Nol".

Sekarang perhatikan contoh kedua, yang disajikan pada gambar 2.9. berikut ini.



Gambar 2.9 *Operasi Penjumlahan Otomatis*

Adapun bentuk pseudocode dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Menghitung Kounter Dasar 1 dapat dilihat pada tabel di bawah ini.

Kounter Dasar 1

Pseudocode	Bahasa Pascal

kounterSederhana; deklarasi variable;
KOUNT: numerik; TAMPIL: Boolean;
 	
Begin
Input TAMPIL; KOUNT = 1;
if TAMPIL = TRUE then While KOUNT<11 do
print KOUNT;
KOUNT = KOUNT + 1;
End;
End.
	var
	kount : integer;
	
	begin kount:=1;
	while (kount < 11) begin
	writeln(kount, " "); kount := kount + 1;
	end; writeln (' '); end;

Pseudocode	Bahasa C++

kounterSederhana; deklarasi variable;
KOUNT: numerik; TAMPIL: Boolean;
 	
Begin
Input TAMPIL; KOUNT = 1;
if TAMPIL = TRUE then While KOUNT<11 do
print KOUNT;
KOUNT = KOUNT + 1;
End;
End.
	#include <stdio.h> using namespace std; int main() {
	int kount = 1;
	while (kount < 11) {
	cout << kount<<" "<<endl; kount = kount + 1;
	}
	cout << "" << endl; return 0;
	}

Pseudocode	Bahasa Java

kounterSederhana; deklarasi variable;
KOUNT: numerik; TAMPIL: Boolean;
 	
Begin
Input TAMPIL; KOUNT = 1;
if TAMPIL = TRUE then While KOUNT<11 do
print KOUNT;
KOUNT = KOUNT + 1;
End;
End.
	public class Kounter_1 {
	public static void main(String[] args) { int kount = 1;
	while (kount < 11) {
	System.out.println(kount+" ");
	Kount = kount + 1;
	}
	System.out.println("");
	}
	}

Gambar 2.9 adalah pseudo code operasi penjumlahan yang diulang secara otomatis. Dalam istilah algoritma disebut dengan

kounter. Hasil identifikasi pada Gambar 2.9 di atas ke dalam komponen pengolahan data, adalah sebagai berikut.

a. Komponen masukan.
   Pada komponen masukan terdapat satu buah instruksi yang dituliskan, yaitu: *Input* TAMPIL.

   Maknanya, pengguna diminta memasukan nilai variable TAMPIL melalui peralatan masukan.

b. Komponen proses.
   Pada komponen proses ada dua logika matematika dan dua pernyataan matematika, yaitu:

   1) Logika matematika yang kesatu adalah: **if** (TAMPIL=TRUE) **then.** Dan yang kedua **while** KOUNT < 11 **do.** Makna perintah atau instruksi tersebut adalah: lakukan perintah atau instruksi setelah do jika nilai KOUNT kurang dari 11. Sehingga Logika matematika yang kedua ini dilakukan berulang sampai nilai KOUNT = 11. Pada saat nilai KOUNT=11 maka nilai proposisi KOUNT<11 adalah FALSE. Dalam algoritma pemrograman, logika matematika kedua ini disebut dengan perintah perulangan. Sedangkan logika matematika kesatu disebut dengan perintah percabangan.

      Hasilnya logika matematika kesatu disajikan pada gambar 2.10. berikut ini.

(a) TAMPIL=FALSE

(b) TAMPIL=TRUE

Gambar 2.10 *Kemungkinan Hasil Logika Matematika TAMPIL=TRUE*

Gambar 2.10. di atas menunjukan terdapat 2 kemungkinan nilai untuk proposisi tunggal, yaitu: TAMPIL=TRUE bernilai benar atau salah.

Kemungkinan nilai hanya dua karena jumlah proposisi hanya 1 (2^1).

2) Pernyataan matematika yang kesatu adalah KOUNT=1. Proposisi ini dikelompokan ke dalam pernyataan matematika karena tidak disertai dengan perintah **if ... then**. Pernyataan matematika yang kedua adalah KOUNT= KOUNT + 1. Proposisi ini dikelompokan ke dalam pernyataan matematika juga, karena tidak disertai dengan perintah **if ... then** .... atau **while** ... **do** ....

c. Komponen keluaran.
Terdapat satu buah perintah pada komponen keluaran, yaitu: print KOUNT. Tidak ada perintah lain pada komponen keluaran selain tersebut di atas..

Berdasarkan penjelasan dengan menggunakan dua contoh diatas terlihat jelas bahwa penyajian logika matematika dalam komputasi berbeda dengan pernyataan matematika. Walaupu pada kondisi tertentu logika matematika disajikan dalam bentuk persamaan eksplisit. Yang membedakan penyajian dari keduanya adalah penyajian logika matematika selalu disertai instruksi seperti **if** ... **then** ... atau lainnya.

2.4 Rangkuman

Berdasarkan diskusi di atas, terlihat jelas perbedaan antara pernyataan matematika dan logika matematika pada algoritma komputasi. Pernyataan matematika nilai kebenarannya focus pada kebenaran persamaan matematika secara keilmuan. Misal, persamaan untuk luas bidang segi empat adalah panjang dikali lebar. Jika variabel panjang diberi notasi p dan variabel lebar diberi notasi l, serta luas bidang segiempat diberi notasi a, maka persamaan matematika yang mewakili luas bidang segiempat adalah: a = p * l, bukan yang lain. Kebenaran pernyataan matematika adalah mutlak karena berfungsi mentransformasikan masukan menjadi keluaran. Jika pernyataan matematika salah, maka akan mengakibatkan proses transformasik masukan menjadi keluaran juga akan salah. Dalam algoritma komputasi disebut kesalahan logika berpikir.

Logika matematika mempunyai nilai kebenaran relative, bergantung nilai yang diberikan pada variabel penyusunnya. Misal logika matematika dituliskan dalam persamaan $x = y$, maka nilai tersebut akan benar jika nilai variabel x sama dengan nilai variabel y. Contoh, jika x diberi nilai 4 dan y jugadiberi nilai 4, maka persamaan $x = y$ bernilai benar. Sedangkan jika jika x diberi nilai 4 dan y diberi nilai 5, maka persamaan $x = y$ bernilai salah.

2.5 Latihan Soal

Untuk soal nomer 1 – 10, pertanyaannya adalah: (a) apakah pernyataan matematika yang disajikan benar?, (b) jika salah, maka tuliskan yang benar.

1. Persamaan untuk penjumlahan 3 buah bilangan adalah: c = a + b.
2. Persamaan untuk nilai rata-rata dari 4 buah bilangan adalah: a = b + c + d + e /4.
3. Persamaan untuk menghitung volume tabung adalah: v = 3.14 * t * r ^ 2; dengan t adalah tinggi, r adalah jari-jari tabung.
4. Persamaan untuk membuat bilangan genap adalah: x = 2 * a; dengan a adalah bilangan bulat.
5. Persamaan untuk membuat bilangan ganjil adalah: $y = 2 * a - 1$; dengan a adalah bilangan bulat.
6. Persamaan untuk membuat bilangan yang habis dibagi 6 adalah: z = 6 * a + 12; dengan a adalah bilangan bulat.
7. Persamaan untuk menghitung ppn adalah: t = 1.1 * a; dengan a adalah nilai total pembayaran.
8. Persamaan untuk menghitung gaji bersih pegawai adalah: gajiBersih = gajiKotor – ptkp * gajiKotor
9. Persamaan untuk membuat akumulator adalah: ak = ak + nilaiBaru.

10. Persamaan untuk menghitung selisih dua buah bilangan adalah: a = b - c * d

Untuk soal nomer 11 - 20, buatlah dua contoh data supaya logika matematika berikut ini: (a) bernilai benar, (b) bernilai salah.

11. $a = b + 10 * c$.
12. $x < y - 1$.
13. $x = y$ dan $a = 5$.
14. $a < 5$.
15. $x <= y$ atau $y - x = 0$
16. $x \wedge 2 > 0$.
17. $a <$ b dan $b < 5$.
18. $a < b$ atau $b < 5$.
19. $a \wedge 2 = b + c$ atau $c = 1$.
20. ($d < e$ dan $e = a$) atau $e = 3$.

Perhatikan pseudocode pada nomer 21 - 30, tuliskan dengan benar: (a) pernyataan matematika, (b) logika matematika yang termuat dalam pseudo code tersebut.

21.
```
Penjumlahan; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
Begin
Input BIL1, BIL2, TAMPIL; HASIL = BIL1 + BIL2;
If TAMPIL = TRUE then print HASIL; End.
```

22.
```
nilai_akhir; deklarasi variable;
nHadir, nAktif, nJujur, nTgs, nUts, nUas, nAkhir: numerik; TAMPIL:
Boolean;
Begin
Input nHadir, nAktif, nJujur, nTgs, nUts, nUas, TAMPIL;
nAkhir = 0.1*nHadir + 0.1*nAktif + 0.5*nJujur + 0.1*nTgs + 0.1*nUts
+0.1* nUas; If TAMPIL = TRUE then print nAkhir;
End.
```

23.
```
rata_dua_variable; deklarasi variable;
BIL1, BIL2, RATA: numerik; TAMPIL: Boolean;

Begin
Input BIL1, BIL2, TAMPIL; RATA = (BIL1 + BIL2)/2;
If TAMPIL = TRUE then print RATA; End.
```

24.
```
rata_dua_variable; deklarasi variable;
BIL1, BIL2, RATA: numerik; TAMPIL: Boolean;
```



```
Begin
Input BIL1, BIL2, TAMPIL; RATA = (BIL1 + BIL2)/2;
If TAMPIL = TRUE then print RATA; End.
```

25.
```
akar_persamaan_kuadrat; deklarasi variable;
a, b, c, x1, x2: numerik; TAMPIL: Boolean;

Begin
Input a, b, c, TAMPIL;
x1 = (-b +( b^2 - ((4*a*c)^0.5)))/ (2*a)
x2 = (-b -( b^2 - ((4*a*c)^0.5)))/ (2*a)

If TAMPIL = TRUE then print x1, x2; End.
```

26.
```
pembagian; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
KONSTANTA: PESAN='BILANGAN KE-2 TIDAK BOLEH NOL'

Begin
Input BIL1, BIL2, TAMPIL;
If (TAMPIL = TRUE) and (BIL2!= 0) then
Begin HASIL = BIL1 / BIL2; print HASIL; end Else print PESAN;
End.
```

27.
```
kounterSederhana; deklarasi variable;
KOUNT: numerik; TAMPIL: Boolean;

Begin
Input TAMPIL; KOUNT = 1;
if TAMPIL = TRUE then While KOUNT<11 do
print KOUNT;
KOUNT = KOUNT + 1;
End;
End.
```

28.
```
kounterSederhana; deklarasi variable;
indeks, nAwal, jData, KOUNT: numerik; TAMPIL: Boolean;

Begin
Input nAwal, jData, TAMPIL; KOUNT = nAwal; indeks = 1; if TAMPIL = TRUE
then
While indeks < jData+1 do print KOUNT;
KOUNT = KOUNT + 1; indeks = indeks + 1;
End;
End.
```

29.
```
akumulator; deklarasi variable;
nTrans, totTrans: numerik; LAGI, tampilHasil: Boolean;

Begin
totTrans =0;
input nTrans, LAGI, tampilHasil;
While (LAGI = TRUE) or (tampilHasil=true) do totTrans= totTrans +nTrans;
Print nTrans.
```

```
If tampilHasil = true then Begin
Print hasil;
tampilHasil=false; LAGI=false; End;
Else begin
input nTrans; LAGI, tampilHasil;
End;
EndEnd;
```

30. multiplikator;

```
deklarasi variable;
nTrans, hasil: numerik; LAGI, tampilHasil: Boolean;

Begin
hasil=1;
input nTrans, LAGI, tampilHasil;
While (LAGI = TRUE) or (tampilHasil=true) do hasil=hasil*nTrans;
Print nTrans.
If tampilHasil = true then Begin
Print hasil;
tampilHasil=false; LAGI=false;
End;
Else begin
input nTrans; LAGI, tampilHasil;
End;
EndEnd;
```

Untuk soal beserta jawabannya pada nomer 31 - 33, lakukan identifikasi untuk komponen:

(a) masukan, (b) proses, serta (c) keluaran.

31. Berikut ini adalah pseudocode untuk membuat luaran factorial

```
faktorial;  deklarasi variable;
n, fakto: numerik; tampilHasil: Boolean;

Begin
fakto=1;
input n, tampilHasil;
While (tampilHasil=true) and (n>0) do fakto=fakto*n; n=n-1;
Print n,"*";
End;

Print "=", fakto;
End.
```

32. Berikut ini adalah pseudocode untuk membuat luaran penjumlahan matriks

```
penjumlahanMatriks; deklarasi variable;
banyakKol_1, banyakKol_2, I: numerik;
tampil_1, tampil_2, simpan_1, simpan_2, tampilkanHasil: boolean;
A[1..4], B[1..4], C[1..4]: numerik;
```



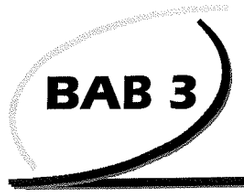
```
Begin
INPUT banyakKol_1; REPEAT
INPUT tampil_1; UNTIL tampil_1 = TRUE;

IF (banyakKol_1 = 2) THEN (PRINT matriks_Arr1_2)
ELSE IF(banyakKol_1 = 3) THEN (PRINT matriks_Arr1_3) ELSE (PRINT
matriks_Arr1_4);
FOR i=1 TO banyakKol_1 DO BEGIN
IF(banyakKol_1 = 2)THEN INPUT matriks_Arr1_2[i]
ELSE IF(banyakKol_1 = 3) THEN INPUT matriks_Arr1_2[i] ELSE INPUT
matriks_Arr1_4[i];
END FOR

IF simpan_1 = TRUE
FOR i=1 TO banyakKol_1 DO
IF(banyakKol_1 = 2) THEN A[i]=matriks_Arr1_2[i]
ELSE IF(banyakKol_1=3) THEN A[i]=matriks_Arr1_2[i] ELSE
A[i]=matriks_Arr1_2[i];

INPUT banyakKol_2; REPEAT
INPUT tampil_2; UNTIL tampil_2 = TRUE;

IF(banyakKol_2 = 2) THEN (PRINT matriks_Arr1_2)
ELSE IF (banyakKol_2=3) THEN (PRINT matriks_Arr1_3) ELSE (PRINT
matriks_Arr1_4);
FOR i=1 TO banyakKol_2 DO BEGIN
IF (banyakKol_2 = 2) THEN INPUT matriks_Arr1_2[i]
ELSE IF(banyakKol_2 = 3) THEN INPUT matriks_Arr1_3[i] ELSE INPUT
matriks_Arr1_4[i];
END FOR

IF simpan_2 = TRUE

FOR i=1 TO banyakKol_2 DO
IF(banyakKol_1 = 2) THEN B[i]=matriks_Arr1_2[i]
ELSE IF(banyakKol_1 = 3)THEN B[i]=matriks_Arr1_2[i] ELSE B[i] =
matriks_Arr1_2[i];

IF banyakKol_1 = banyakKol_2 THEN BEGIN
FOR i=1 TO banyakKol_1 DO C[i] = A[i] + B[i]; INPUT tampilkanHasil;
REPEAT
FOR i=1 TO banyakKol_1 DO PRINT C[i] UNTIL tampilkanHasil=TRUE;
END IF
ELSE PRINT "Matriks A tidak bisa dijumlahkan dengan Matriks B";

END
```

33. Berikut ini adalah pseudocode untuk membuat luaran pengurangan matriks

```
penjumlahanMatriks; deklarasi variable;
banyakKol_1, banyakKol_2, I: numerik;
tampil_1, tampil_2, simpan_1, simpan_2, tampilkanHasil: boolean;
A[1..4], B[1..4], C[1..4]: numerik;

Begin
INPUT banyakKol_1; REPEAT
INPUT tampil_1; UNTIL tampil_1 = TRUE;
```

```
IF (banyakKol_1 = 2) THEN (PRINT matriks_Arr1_2)
ELSE IF(banyakKol_1 = 3) THEN (PRINT matriks_Arr1_3) ELSE (PRINT
matriks_Arr1_4);
FOR i=1 TO banyakKol_1 DO BEGIN
IF(banyakKol_1 = 2)THEN INPUT matriks_Arr1_2[i]
ELSE IF(banyakKol_1 = 3) THEN INPUT matriks_Arr1_2[i] ELSE INPUT
matriks_Arr1_4[i];
END FOR

IF simpan_1 = TRUE
FOR i=1 TO banyakKol_1 DO
IF(banyakKol_1 = 2) THEN A[i]=matriks_Arr1_2[i]
ELSE IF(banyakKol_1=3) THEN A[i]=matriks_Arr1_2[i] ELSE
A[i]=matriks_Arr1_2[i];

INPUT banyakKol_2;
REPEAT
INPUT tampil_2; UNTIL tampil_2 = TRUE;

IF(banyakKol_2 = 2) THEN (PRINT matriks_Arr1_2)
ELSE IF (banyakKol_2=3) THEN (PRINT matriks_Arr1_3) ELSE (PRINT
matriks_Arr1_4);
FOR i=1 TO banyakKol_2 DO BEGIN
IF (banyakKol_2 = 2) THEN INPUT matriks_Arr1_2[i]
ELSE IF(banyakKol_2 = 3) THEN INPUT matriks_Arr1_3[i] ELSE INPUT
matriks_Arr1_4[i];
END FOR

IF simpan_2 = TRUE
FOR i=1 TO banyakKol_2 DO
IF(banyakKol_1 = 2) THEN B[i]=matriks_Arr1_2[i]
ELSE IF(banyakKol_1 = 3)THEN B[i]=matriks_Arr1_2[i] ELSE B[i] =
matriks_Arr1_2[i];

IF banyakKol_1 = banyakKol_2 THEN BEGIN
FOR i=1 TO banyakKol_1 DO C[i] = A[i] + B[i]; INPUT tampilkanHasil;
REPEAT
FOR i=1 TO banyakKol_1 DO PRINT C[i] UNTIL tampilkanHasil=TRUE;
END IF
ELSE PRINT "Matriks A tidak bisa dijumlahkan dengan Matriks B";

END
```

-oo0o-



VARIABEL DAN DATA

Deskripsi Singkat

Bab 3 menjelaskan pentingnya kemampuan berpikir komputasi bagi semua orang dalam menangani permasalahan, yaitu dapat mengidentifikasi bagian masukan, proses, dan keluaran. Dalam masukan, proses, dan keluaran diperlukan variabel dan data.

Diskusi pada bab 3 diawali dengan pendahuluan, kemudian diikuti dengan variabel dan pernyataan matematika, variabel dan logika matematika, variabel dan instruksi komputasi, variabel dan tipe data, dan identifikasi variabel berdasarkan komponen pengolahan data. Bab ini ditutup dengan rangkuman, latihan soal, dan daftar referensi yang digunakan untuk menyusun bab ini.

Pada bagian pendahuluan diberikan contoh-contoh operasi aritmatika dasar dan aplikasi aritmatika yang terdapat pada **penalaran.com.** Berdasarkan contoh-contoh ini, dalam subbab berikutnya dijelaskan macam-macam variabel, konstanta, tipe data, dan operator yang digunakan untuk menyusun instruksi komputasi dalam bentuk *flowchart*, *pseudocode*, dan kode bahasa pemrograman.

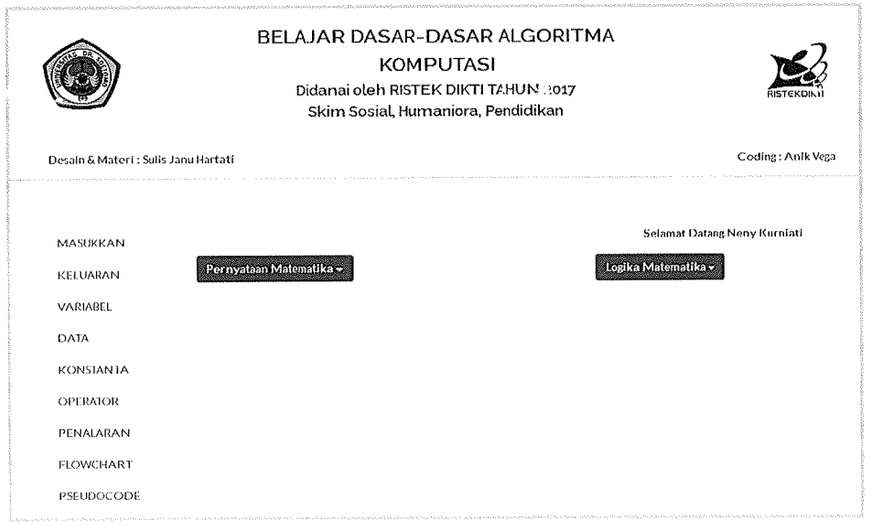
Tujuan Pembelajaran

Akhir pembelajaran atau diskusi, diharapkan pembaca mempunyai beberapa keterampilan berikut:

1. Jika diberikan soal, pembaca dapat mengidentifikasi bagian masukan, keluaran, variabel, dan data.
2. Jika diberikan soal, pembaca dapat mengidentifikasi operator dan pernyataan matematika pada bagian proses.
3. Jika diberikan soal, pembaca dapat melakukan penalaran induksi sebab akibat untuk mengidentifikasi logika matematika.
4. Berdasarkan keterampilan 1 sampai dengan 3 di atas diharapkan pembaca dapat menyusun instruksi komputasi dalam bentuk *flowchart*, *pseudocode*, dan *code* programnya.

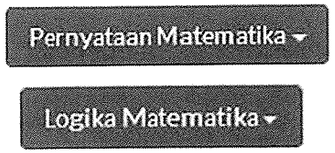
3.1 Pendahuluan

Untuk lebih memahami tentang Variabel, Data, dan Konstanta silahkan berkunjung ke laman https://penalaran.com. Lakukan login dengan username dan password yang sudah Anda daftarkan. Setelah Anda berhasil login, akan muncul Menu Utama seperti Gambar 3.1.



Gambar 3.1 *Tampilan Menu Utama* ***penalaran.com***

Menu Utama **penalaran.com** mempunyai 2(dua) menu, yaitu **Pernyataan Matematika** dan **Logika Matematika** sesuai Gambar 3.2..

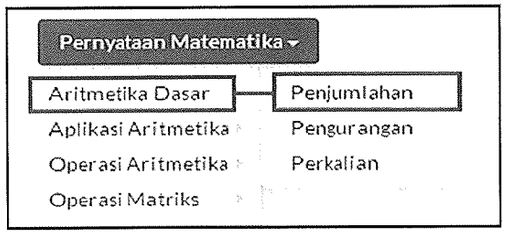


Gambar 3.2 *Menu Utama* ***penalaran.com***

Menu tersebut mempunyai struktur sebagai berikut:

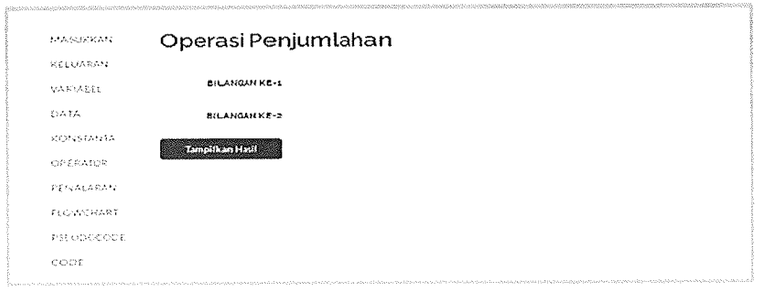
1. Pernyataan Matematika
   a. Aritmetika Dasar
      1) Penjumlahan
      2) Pengurangan
      3) Perkalian
   b. Aplikasi Aritmetika
      1) Menghitung Nilai Akhir
      2) Menghitung Rata-Rata Dua Variabel
      3) Menghitung Luas Lingkaran
      4) Menghitung Akar Persamaan Kuadrat
   c. Operasi Aritmetika
      1) Operasi Pembagian
      2) Membuat Kounter
         a) Kounter Dasar 1
         b) Kounter Dasar 2
      3) Akumulasi Penjumlahan
      4) Akumulasi Perkalian
      5) Faktorial
   d. Operasi Matriks
      1) Operasi Penjumlahan
      2) Operasi Pengurangan
2. Logika Matematika
   a. Logika Dasar
   b. Kombinasi Dua Relasi
   c. Konsep Perulangan
   d. Konsep Perulangan Array

Kita akan belajar tentang Penjumlahan pada **penalaran.com.** Klik menu **Pernyataan Matematika - Aritmetika Dasar**, kemudian klik **Penjumlahan** seperti terlihat pada Gambar 3.3.



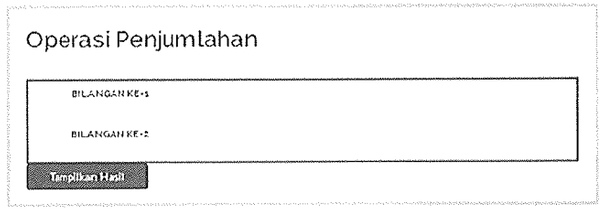
Gambar 3.3 *Menu Aritmetika Dasar Penjumlahan*

Muncul tampilan operasi aritmetika dasar Penjumlahan seperti pada Gambar 3.4.



Gambar 3.4 *Tampilan Operasi Aritmetika Dasar Penjumlahan*

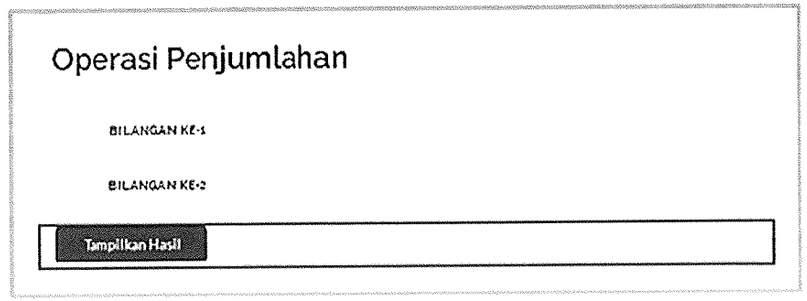
Klik toolbar menu **MASUKKAN** di bagian kiri layar. Operasi Aritmetika Dasar Penjumlahan mempunyai 2(dua) kotak masukan, yaitu **BILANGAN KE-1** dan **BILANGAN KE-2** ditunjukkan dengan kotak merah di Gambar 3.5,



Gambar 3.5 *Masukan Operasi Aritmetika Dasar Penjumlahan*

Klik toolbar menu **KELUARAN**. Ada 1 (satu) kotak keluaran yang ditunjukkan oleh Gambar 3.6. Contoh pengoperasian Operasi Penjumlahan sebagai berikut:

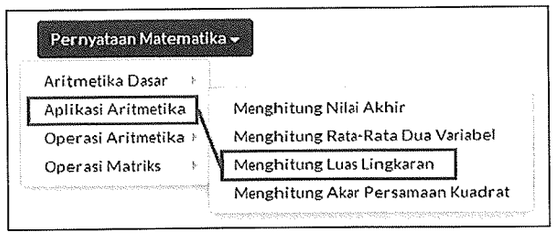
1. Ketikkan nilai **27** dalam kotak masukan **BILANGAN KE-1**.
2. Ketikkan nilai **14** dalam kotak masukan **BILANGAN ke-2**.
3. Klik tombol **Tampilkan Hasil** pada bagian keluaran.
4. Dalam kotak keluaran tampil nilai **41** yang merupakan hasil penjumlahan bilangan **27** ditambah dengan bilangan **14**, ditunjukkan pada Gambar 3.7



Gambar 3.6 *Keluaran Operasi Aritmetika Dasar Penjumlahan*

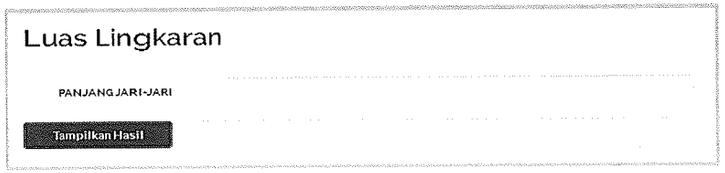
Gambar 3.7 *Tampilan Hasil Operasi Penjumlahan*

Contoh lain untuk lebih memahami variabel, data, dan konstanta kita coba program **Menghitung Luas Lingkaran** dalam **penalaran.com.** Program Menghitung Luas Lingkaran terdapat pada menu **Pernyataan Matematika - Aplikasi Aritmetika**. Klik **Menghitung Luas Lingkaran** sesuai Gambar 3.8. Muncul tampilan seperti Gambar 3.9.

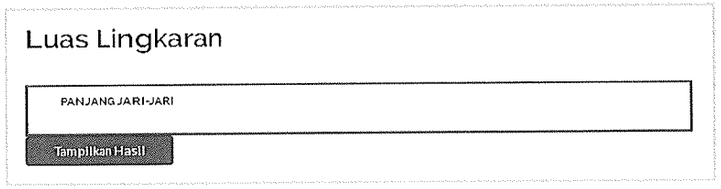


Gambar 3.8 *Menu Aplikasi Aritmetika Submenu Menghitung Luas Lingkaran*

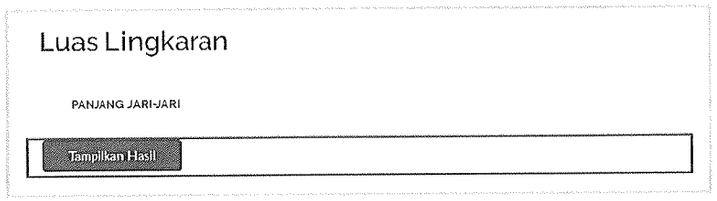
Sesuai rumus di matematika untuk menghitung luas lingkaran adalah **3.14** kali panjang jari-jari lingkaran, sehingga operasi menghitung luas lingkaran diperlukan masukan **PANJANG JARI-JARI** yang ditunjukan oleh Gambar 3.10.



Gambar 3.9 *Tampilan Operasi Hitung Luas Lingkaran*



Gambar 3.10 *Masukan Operasi Menghitung Luas Lingkaran. Hasil luas Lingkaran Akan Tampil pada Bagian Keluaran Seperti Gambar 3.11.*



Gambar 3.11 *Tampilan Keluaran Operasi Menghitung Luas Lingkaran*

Selanjutnya kita coba menghitung luas lingkaran dengan panjang jari-jari 5. Ketikkan nilai **5** pada bagian masukan PANJANG JARI-JARI, kemudian klik tombol **Tamplkan Hasil**. Pada bagian keluaran akan muncul nilai **79** yang artinya luas lingkaran dengan jari-jari 5 adalah 79. Hal ini bisa dilihat pada Gambar 3.12.

Gambar 3.12 *Contoh Operasi Menghitung Luas Lingkaran*

Bagaimana cara menentukan langkah-langkah penyelesaian masalah (algoritma) dalam bentuk *Flowchart* dan *Pseudocode* untuk tiga contoh di atas (Operasi Penjumlahan, Menghitung Luas Lingkaran) yang kemudian algoritma tersebut diterjemahkan ke dalam bahasa pemrograman? Ikuti pembahasannya pada sub pokok bahasan selanjutnya.

3.2 Variabel dan Pernyataan Matematika

Pernyataan matematika adalah sebuah proses yang mentransformasikan nilai (masukan) menjadi hasil (keluaran) yang diinginkan melalui proses perhitungan (komputasi). Pernyataan matematika terdiri dari **operand** dan **operator** aritmatika. Operand adalah nilai yang diberikan, dapat berupa variabel, konstanta, nilai, dan nilai balik dari fungsi. Sedangkan operator merupakan penghubung antar operand.

Pernyataan matematika mempunyai ciri:

1. **Operator yang digunakan** : ^, *, /, DIV, MOD, +, -, =, ←
2. **Operand yang digunakan** : numerik (angka)
3. **Hasil (Keluaran) berupa** : numerik (angka)
4. **Contoh** : HASIL = BIL1 + BIL2
   Luas = 3.14 * radius^2 Sisa = BIL MOD 2

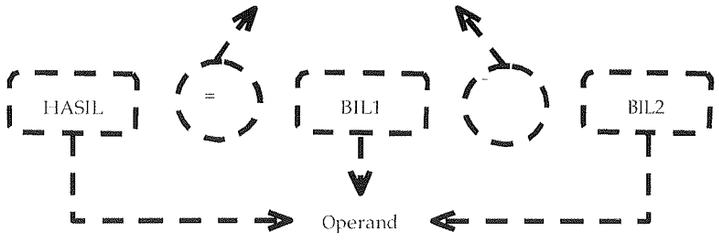
Operator aritmatika mempunyai prioritas yang berbeda. Operator yang mempunyai prioritas lebih tinggi akan dioperasikan lebih dulu (lihat Tabel 3.1).

Tabel 3.1 *Prioritas Penggunaan Operator Aritmatika*

Prioritas	Operator	Operasi	Tipe Operand	Tipe Hasil
1	^	Pangkat	Real, Real Integer, Real Real, Integer	Real Integer Real
2	*	Perkalian	Real, Real Integer, Integer Real, Integer	Real Integer Real
	/	Pembagian Real	Real, Real Integer, Integer Real, Integer	Real Real Real
	DIV	Pembagian Bulat	Integer, Integer	integer
	MOD	Sisa Pembagian	Integer, Integer	Integer
3	+	Penjumlahan	Real, Real Integer, Integer Real, Integer	Real Integer Real
	-	Pengurangan	Real, Real Integer, Integer Real, Integer	Real Integer Real
4	= atau ←	Pemuatan Nilai	Integer Real	Integer Real

Perhatikan Gambar 3.13.

Operator Aritmatika

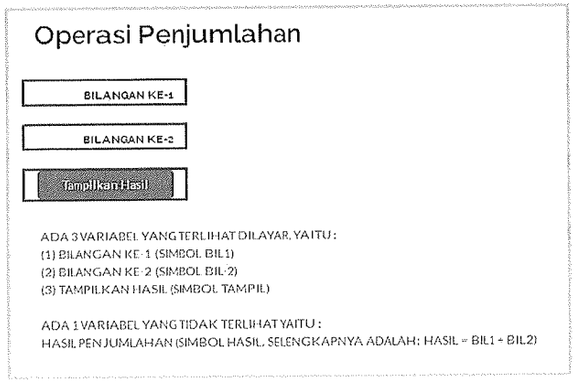


Gambar 3.13 *Contoh Pernyataan Matematika*

Contoh pernyataan matematika pada Gambar 3.13 mempunyai operand berupa variabel **BIL1**, **BIL2**, dan **HASIL**. Variabel BIL1 dan BIL2 berisi nilai bilangan yang akan dioperasikan, dan variabel HASIL berisi nilai hasil operasi. Operator yang digunakan adalah operator penjumlahan (+) untuk melakukan operasi penjumlahan terhadap nilai dalam BIL1 dan BIL2 serta operator pemuatan nilai (=) untuk memasukkan hasil operasi penjumlahan dalam variabel HASIL.

Kita akan belajar tentang variabel dan pernyataan matematika menggunakan contoh-contoh dalam **penalaran.com** yang sudah dibahas di bagian PENDAHULUAN. Contoh pertama tentang operasi aritmetika dasar penjumlahan (lihat Gambar 3.7). Lakukan operasi penjumlahan pada **penalaran.com** seperti Gambar 3.7.

Pada bagian kiri layar terdapat tampilan *toolbar* menu (MASUKKAN, KELUARAN, VARIABEL, DATA, dan seterusnya). Klik toolbar menu **VARIABEL**, Di layar akan muncul tampilan seperti Gambar 3.14.



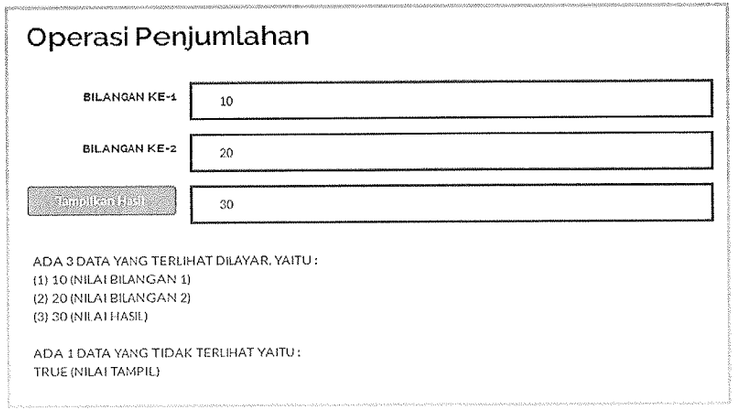
Gambar 3.14 *Variabel dalam Operasi Penjumlahan*

Gambar 3.14 menunjukkan bahwa ada 3 (tiga) variabel yang terlihat di layar untuk operasi penjumlahan, yaitu:

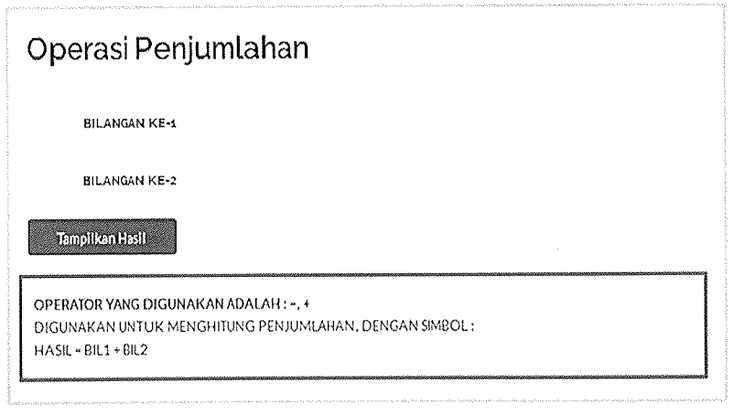
1. BILANGAN KE-1, dinyatakan dengan nama variabel atau simbol **BIL1**.
2. BILANGAN KE-2, dinyatakan dengan nama variabel atau simbol **BIL2**.

3. Tampilkan Hasil, dinyatakan dengan nama variabel atau simbol **TAMPIL**. Selain itu ada 1 (satu) variabel yang tidak terlihat di layar, yaitu **HASIL**.

Klik *toolbar* menu **DATA** dan tampil seperti Gambar 3.15. Gambar tersebut menunjukkan bahwa data adalah bilangan yang akan dijumlahkan dan juga hasil penjumlahannya. Berdasarkan Gambar 3.14 dan Gambar 3.15 dapat disimpulkan bahwa variabel **BIL1** berisi nilai **10**, **BIL2** berisi nilai **20**, dan variabel **HASIL** berisi nilai **30** yang merupakan hasil penjumlahan antara 10 dan 20.



Gambar 3.15 *Tampilan Data dalam Operasi Penjumlahan*



Gambar 3.16 *Tampilan Operator dalam Operasi Penjumlahan*

Klik *toolbar* menu **OPERATOR**. Untuk menjumlahkan BIL1 dan BIL2 diperlukan operator aritmatika penjumlahan (+) dan operator pemuatan nilai (=) untuk memasukkan nilai hasil penjumlahan ke dalam variabel HASIL. Operasi penjumlahan ini dinyatakan dengan pernyataan matematika **HASIL = BIL1 + BIL2** (lihat Gambar 3.16).

Bagaimana variabel dan pernyataan matematika pada operasi menghitung luas lingkaran? Aktifkan Menu Utama di **penalaran.com.** klik menu **Pernyataan Matematika - Aplikasi Aritmetika**, klik **Menghitung Luas Lingkaran**. Lakukan operasi dengan memasukkan nilai **10** pada masukan **PANJANG JARI-JARI** kemudian klik tombol **Tampilkan Hasil**. Klik toolbar menu **VARIABEL**. Hasilnya bisa dilihat pada Gambar 3.17.

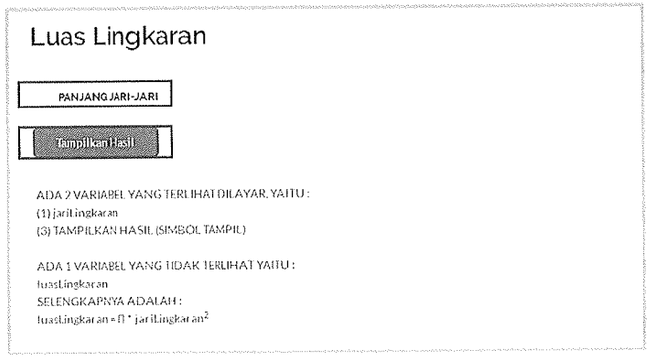
Ada 2(dua) variabel yang terlihat di layar, yaitu:

1. Panjang jari-jari, dinyatakan dengan nama variabel atau simbol **jariLingkaran**
2. Tampilkan Hasil, dinyatakan dengan nama variabel atau simbol **TAMPIL**.

Ada 1 (satu) variabel yang tidak terlihat di layar, yaitu **luasLingkaran** dinyatakan dengan simbol:

$$\texttt{jariLingkaran} = \pi * \texttt{jariLingkaran}^{2}$$

π (dibaca *phi*) adalah suatu KONSTANTA yang mempunyai nilai tetap **3.14**.

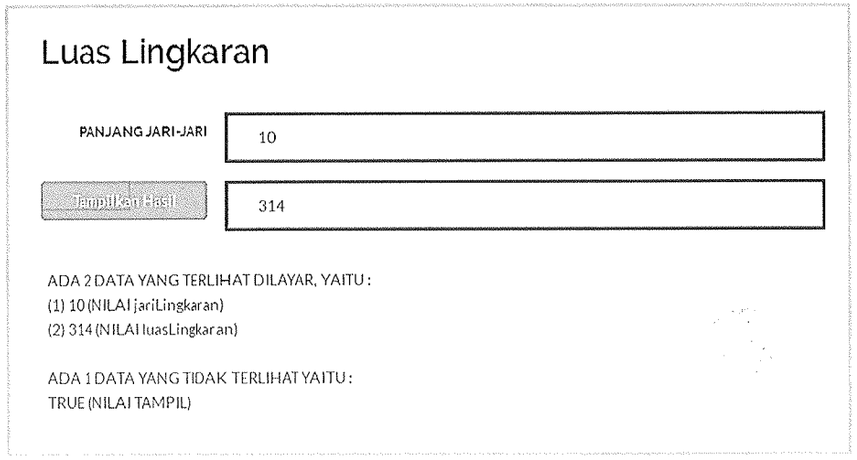


Gambar 3.17 *Variabel pada Operasi Menghitung Luas Lingkaran*

Klik *toolbar* menu **DATA**. Dari Gambar 3.18 diketahui bahwa variabel **jariLingkaran** mempunyai nilai **10**. Operasi menghitung luas lingkaran mempunyai pernyataan matematika **luasLingkaran = phi * jariLingkaran^2**. Dari pernyataan matematika tersebut diketahui operator aritmatika yang digunakan adalah perkalian (*), pangkat (^), dan operator pemuatan (=). Variabel **luasLingkaran** mempunyai nilai **314** yang diperoleh dari operasi pernyataan matematika:

luasLingkaran = phi * jariLingkaran^2
= 3.14 * 10^2
= 3.14 * 100
= 314

Perhitungan **jariLingkaran^2** dioperasikan lebih dulu karena operator pangkat (^) mempunyai prioritas lebih tinggi dibandingkan operator perkalian (*) (lihat Tabel 3.1).



Gambar 3.18 *Data pada Operasi Luas Lingkaran*

3.3 Variabel dan Logika Matematika

Logika Matematika merupakan pernyataan yang digunakan pada algoritma yang mengandung proses percabangan, tepatnya pada kondisi

yang berlaku sebagai syarat/aturan dari sebuah permasalahan. Logika matematika menggunakan Operator Relasi dan Operator Logika.

1. **Operator yang digunakan:** <, >, <>, = , >=, <=, NOT, AND, OR
   Logika matematika digunakan untuk membandingkan hubungan antara 2 buah operand dan hasil yang didapatkan merupakan keluaran bertipe boolean, yaitu TRUE atau FALSE.
2. Operand yang digunakan : numerik, string
3. Hasil (Keluaran) berupa : Boolean (TRUE, FALSE)
4. Contoh: A = 10
   B = 30
   C = 10
   A <> C = FALSE

Macam-macam operator relasi dan operator logika dapat dilihat pada Tabel 3.2 dan Tabel 3.3.

Tabel 3.2 *Operator Relasi*

Operator	Arti
=	sama dengan
<>	tidak sama dengan
>	lebih besar
>=	lebih besar atau sama dengan
<	lebih kecil
<=	lebih kecil atau sama dengan

Tabel 3.3 *Operator Logika*

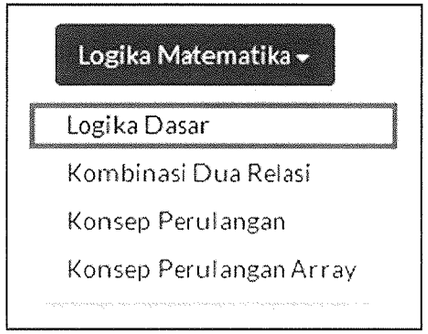
X1	X2	NOT X	NOT Y	X1 OR X2	X1 AND X2
FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
FALSE	TRUE	TRUE	FALSE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
TRUE	TRUE	FALSE	FALSE	TRUE	TRUE

Misalkan ada 3 (tiga) variabel X, Y, dan Z yang dikenai operasi logika matematika. Jika nilai variabel X=16, Y=20, dan Z=16, maka hasil /keluaran operasi logika terhadap ketiga variabel tersebut dapat dilihat dalam Tabel 3.4.

Tabel 3.4 *Contoh Operasi Logika Matematika*

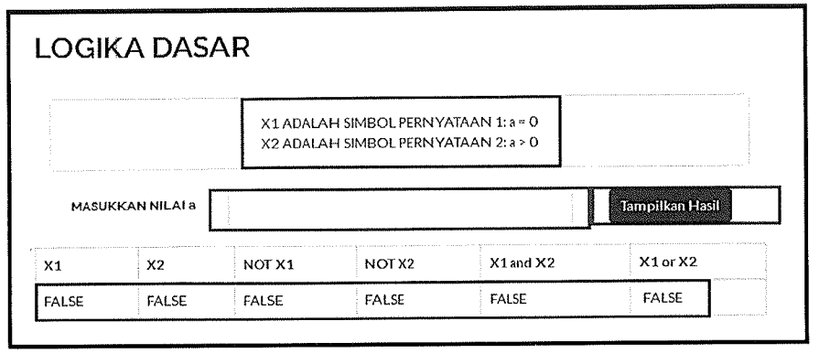
Logika Matematika	Arti		Keluaran
X = Y	Apakah isi variabel X sama dengan isi variabel Y?	(16 = 20)?	FALSE
X <> Z	Apakah isi variabel X tidak sama dengan isi variabel Z?	(16 <> 16)?	FALSE
Y > X	Apakah isi variabel Y lebih besar dari isi variabel X?	(20 > 16)?	TRUE
Y < Z	Apakah isi variabel Y lebih kecil dari isi variabel Z?	(20 < 16)?	FALSE
X >= Z	Apakah isi variabel X lebih besar atau sama dengan isi variabel Z?	(16 >= 16)?	TRUE
X <= Y	Apakah isi variabel X lebih kecil atau sama dengan isi variabel Y?	(16 <= 20)?	TRUE

Selanjutnya kita pelajari logika matematika ini pada aplikasi penalaran.com. Aktifkan **Menu Utama**. Klik menu **Logika Matematika**, kemudian klik **Logika Dasar** (lihat Gambar 3.19).



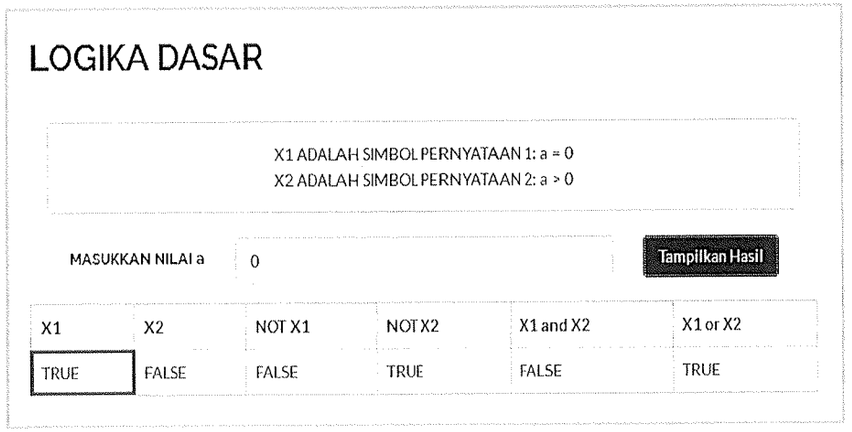
Gambar 3.19 *Tampilan Menu Logika Dasar*

Operasi logika dasar mempunyai masukan, keluaran, dan syarat /kondisi seperti yang ditunjukkan oleh Gambar 3.20. Tulisan dalam kotak hijau menyatakan syarat/kondisi untuk operasi logika. Kotak merah sebagai area masukan, dan kotak biru sebagai area keluaran.

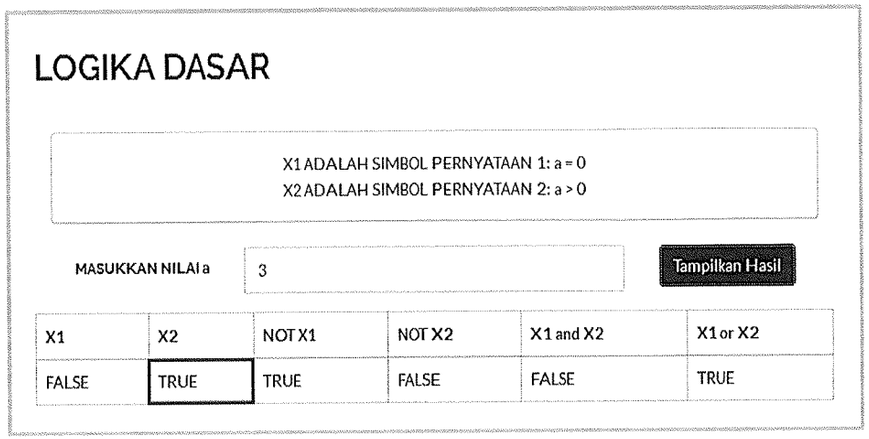


Gambar 3.20 *Masukan, Keluaran, dan Syarat/Kondisi pada Operasi Logika Dasar*

Selanjutnya coba ketikkan nilai **0** pada bagian **MASUKKAN NILAI a**. Berdasarkan syarat/kondisi yang sudah ditentukan maka akan menghasilkan nilai keluaran seperti pada Gambar 3.21. Keluaran **X1** bernilai **TRUE** karena Nilai a=0 memenuhi syarat Pernyataan 1, yaitu **a=0**. Keluaran **X2** bernilai **FALSE** karena Masukan Nilai a=0 tidak memenuhi syarat Pernyataan 2, yaitu **a>0**. Keluaran **NOT X1**, **NOT X2**, **X1 and X2**, dan **X1 or X2** mengikuti aturan Tabel 3.3.



Gambar 3.21 *Hasil Operasi Logika Dasar untuk Nilai Masukan 0 Sekarang lihat Gambar 3.22*



Gambar 3.22 *Hasil Operasi Logika Dasar untuk Nilai Masukan 3*

Coba ketikkan nilai **3** pada bagian **MASUKKAN NILAI a**. Berdasarkan syarat/kondisi yang sudah ditentukan, maka keluaran **X1** bernilai **FALSE** karena nilai 3 tidak memenuhi syarat Pernyataan 1, yaitu **a=0**. Keluaran **X2** bernilai **TRUE** karena nilai 3 memenuhi syarat Pernyataan 2, yaitu **a>0**. Keluaran **NOT X1, NOT X2, X1 and X2,** dan **X1 or X2** mengikuti aturan yang terdapat pada Tabel 3.3.

Pengetahuan tentang Logika Matematika akan menjadi dasar untuk materi PERCABANGAN yang akan dibahas pada BAB 4.

3.4 Variabel dan Instruksi Komputasi

Di **penalaran.com** terdapat menu **PERNYATAAN MATEMATIKA** dan menu **LOGIKA MATEMATIKA**. Setiap menu mempunyai beberapa pilihan contoh aplikasi. Untuk setiap aplikasi yang dipilih (Aritmetika Dasar - Penjumlahan, Aplikasi Aritmetika - Menghitung Luas Lingkaran, Logika Matematika - Logika Dasar, dan seterusnya) kita bisa mengetahui /belajar tentang Masukan, Keluaran, Variabel, Data, dan seterusnya sampai dengan Code programnya. Fasilitas ini terdapat pada *toolbar* menu di bagian kiri layar.

Pada subbab ini kita akan belajar tentang **Instruksi Komputasi,** yaitu algoritma dalam bentuk *flowchart* dan *pseudocode* sekaligus kode programnya. Instruksi komputasi tidak bisa dipisahkan dari **variabel**. Dalam BAB 1 dibahas tentang Komponen Dasar Pengolahan Data yang meliputi Masukan (*Input*) - Proses - Keluaran (*Output*). Variabel menjadi tempat untuk menampung data masukan. Variabel yang sudah berisi data diproses dan hasil prosesnya dimasukkan dalam variabel. Variabel yang berisi data hasil proses inilah yang dinyatakan sebagai keluaran. Masukan - Proses - Keluaran akan dinyatakan dalam instruksi komputasi.

Kita awali dengan contoh operasi Penjumlahan di menu Aritmetika Dasar. Jalankan Operasi **Penjumlahan**, kemudian klik toolbar menu **VARIABEL**. Akan muncul tampilan Gambar 3.14.

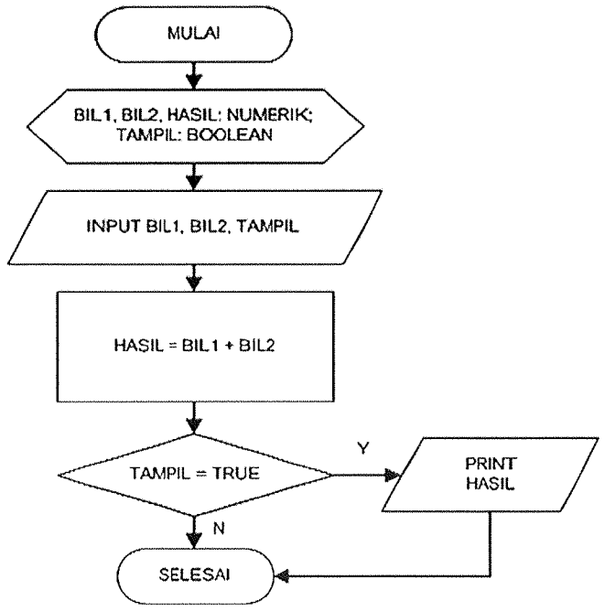
Pada gambar tersebut ditunjukkan bahwa Operasi Penjumlahan mempunyai 3 (tiga) variabel yang terlihat di layar, yaitu BILANGAN-1 dinyatakan dengan nama variabel BIL1, variabel BILANGAN-2 dinyatakan dengan nama variabel BIL2, dan variabel TAMPIL dengan nama variabel TAMPIL. Ketiga variabel ini sebagai variabel masukan. BIL1 dan BIL2 berisi dua bilangan yang akan dijumlahkan sehingga BIL1 dan BIL2 harus bertipe NUMERIK. Variabel TAMPIL berisi nilai TRUE atau FALSE untuk menentukan apakah hasil operasi ditampilkan atau tidak, maka variabel TAMPIL harus bertipe BOOLEAN.

Selanjutnya klik toolbar menu OPERATOR, akan tampil seperti Gambar 3.16. Operator diperlukan dalam Proses pengolahan data. Operator yang digunakan dalam Operasi Penjumlahan adalah operator pemuatan nilai (=) dan operator penjumlahan (+). Pernyataan matematika untuk operasi ini adalah **HASIL = BIL1 + BIL2**.

Operasi penjumlahan menghasilkan sebuah nilai hasil penjumlahan antara nilai BIL1 dengan nilai BIL2 dan dimasukkan dalam variabel HASIL. Jika variabel TAMPIL bernilai TRUE maka nilai HASIL ditampilkan pada bagian Keluaran (Gambar 3.23).

Gambar 3.23 *Tampilan Keluaran pada Operasi Penjumlahan*

Semua aktivitas masukan - proses - keluaran pada operasi penjumlahan dinyatakan dalam instruksi komputasi berupa algoritma berbentuk *Flowchart* (Gambar 3.24) dan *Pseudocode* (Gambar 3.25).



Gambar 3.24 *Flowchart Operasi Penjumlahan*

Dalam **penalaran.com,** *FLOWCHART* dan *PSEUDOCODE* operasi penjumlahan bisa dilihat dengan melakukan klik *toolbar* menu *FLOWCHART* dan *PSEUDOCODE*. Instruksi komputasi berbentuk algoritma tidak bisa dijalankan oleh pengguna (*user*). Agar pengguna bisa menjalankan operasi penjumlahan, harus dinyatakan dalam kode program. Kode program yang digunakan adalah bahasa **Java**. Klik *toolbar* CODE akan muncul tampilan seperti Gambar 3.35.


```
penjumlahan;
deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;

Begin
Input BIL1, BIL2, TAMPIL;
HASIL = BIL1 + BIL2;
If TAMPIL = TRUE then print HASIL;
End.
```

Gambar 3.25 *Pseudocode Operasi Penjumlahan*

```
public class Penjumlahan {
    public static void main(String[] args) {
        int BIL1 = 10;
        int BIL2 = 20;
        int HASIL = 0;

        HASIL = BIL1 + BIL2;
        System.out.println("HASIL : "+HASIL);
    }
}
```

Yang tampil adalah code dalam bahasa java.

```
run:
HASIL : 30
BUILD SUCCESSFUL (total time: 1 second)
```

Jika code dikompile dan dijalankan akan muncul tampilan seperti pada simulasi (area pembelajaran) media pembelajaran ini.

Gambar 3.26 *Kode Program Operasi Penjumlahan*

Adapun bentuk *pseudocode* dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Menghitung Penjumlahan dapat dilihat pada tabel di bawah ini.

1. Menu Penjumlahan

Pseudocode	Bahasa Pascal

penjumlahan; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; HASIL =
BIL1 + BIL2;
If TAMPIL = TRUE then print HASIL;
End.
	var
	BIL1 : integer; BIL2 : integer;
	HASIL : integer;
	
	begin
	readln (BIL1); readln (BIL2);
	HASIL := BIL1 + BIL2;
	writeln(HASIL); end;

Pseudocode	Bahasa C++

penjumlahan; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; HASIL =
BIL1 + BIL2;
If TAMPIL = TRUE then print HASIL;
End.
	#include <stdio.h> using namespace
	std; int main() {
	int BIL1 = 10; int BIL2 = 20; int
	HASIL = 0;
	HASIL = BIL1 + BIL2;
	cout<<"Hasil : " << HASIL<<endl;
	return 0;
	}

Pseudocode	Bahasa Java

penjumlahan; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; HASIL =
BIL1 + BIL2;
If TAMPIL = TRUE then print HASIL;
End.
	public class penjumlahan {
	public static void main(String[]
	args) { int BIL1 = 10;
	int BIL2 = 20; int HASIL = 0;
	HASIL = BIL1 + BIL2;
	System.out.println("Hasil : " +
	HASIL);
	}
	}

Contoh berikutnya **Menghitung Luas Lingkaran**. Klik menu **Pernyataan Matematika - Aplikasi Aritmetika**. Klik **Menghitung Luas Lingkaran**. Klik toolbar menu VARIABEL dan muncul tampilan Gambar 3.17.

Ada 2 (dua) variabel yang terlihat di layar, yaitu **PANJANG JARI-JARI** dinyatakan dalam variabel **jariLingkaran** dan **Tampilkan Hasil** dengan nama variabel **TAMPIL**. Variabel jariLingkaran dan TAMPIL sebagai variabel masukan. jariLingkaran berisi nilai panjang jari-jari lingkaran sehingga harus bertipe numerik. Untuk menghitung luas lingkaran diperlukan konstanta bernama phi dengan nilai ketetapan **3.14**. Seperti dalam operasi penjumlahan, variabel TAMPIL juga bertipe BOOLEAN karena berisi nilai TRUE atau FALSE.

Klik *toolbar* menu OPERATOR, tampil Gambar 3.27. Proses menghitung luas lingkaran memerlukan operator pemuatan (=), perkalian (*), dan operator pangkat (^). Pernyataan matematika untuk menghitung luas lingkaran:

luasLingkaran = phi * jariLingkaran^2

Urutan operasi atau prioritas sesuai dengan aturan dalam Tabel 3.1. Operator pangkat mempunyai prioritas lebih tinggi daripada operator perkalian, sehinga operasi perpangkatan harus dilakukan lebih dulu. Hasil operasi pangkat dikalikan dengan nilai konstanta phi.

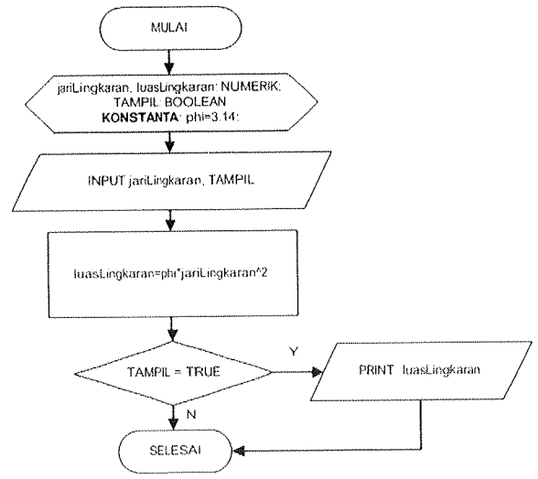
Keluaran berupa nilai luas lingkaran dimasukkan dalam variabel **luasLingkaran** dan ditampilkan ke layar. Semua aktivitas masukan - proses - keluaran dituangkan dalam instruksi komputasi algoritma *Flowchart* dan *Pseudocode* dan diterjemahkan dalam Kode bahasa Java. Klik *toolbar* menu *FLOWCHART, PSEUDOCODE,* dan CODE untuk mendapatkan tampilan seperti Gambar 3.28, 3.29, dan 3,30.

OPERATOR YANG DIGUNAKAN ADALAH : =, +, ^
DIGUNAKAN UNTUK MENGHITUNG NILAI LUAS LINGKARAN DENGAN SIMBOL :
luasLingkaran = phi * jariLingkaran^2

PAHAMI MAKNA SIMBOL BERIKUT INI

SIMBOL	MAKNA
luasLingkaran = phi * jariLingkaran^2	1) jariLingkaran dikuadratkan 2)hasil kuadratnya kemudian dikalikan konstanta phi yang nilainya 3.14

Gambar 3.27 *Operator Proses Menghitung Luas Lingkaran*



Gambar 3.28 *Flowchart Operasi Menghitung Luas Lingkaran*

```
luas_lingkaran;
deklarasi variable;
  jariLingkaran, luasLingkaran: numerik;
  TAMPIL: Boolean;
  KONSTANTA: phi=3.14;
Begin
  Input jariLingkaran, TAMPIL;
  luasLingkaran = phi* jariLingkaran^2;
  If TAMPIL = TRUE then print luasLingkaran;
End.
```

Gambar 3. 29 *Pseudocode Operasi Menghitung Luas Lingkaran*

```
public class LuasLingkaran {
    public static void main(String[] args) {
        double jariLingkaran = 10;
        double luasLingkaran = 0;

        luasLingkaran = Math.PI*(Math.pow(jariLingkaran, 2));
        System.out.println("Luas Lingkaran : "+luasLingkaran);
    }
}
```

Yang tampil adalah code dalam bahasa pemrograman JAVA.

```
run:
Luas Lingkaran : 314.1592653589793
BUILD SUCCESSFUL (total time: 0 seconds)
```

Jika code dikompile dan dijalankan akan muncul tampilan seperti pada simulasi (area pembelajaran) media pembelajaran ini

Gambar 3.30 *Kode Program Operasi Menghitung Luas Lingkaran*

Adapun bentuk *pseudocode* dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Menghitung Luas Lingkaran dapat dilihat pada tabel di bawah ini.

Menghitung Luas Lingkaran

Pseudocode	Bahasa Pascal

luas_lingkaran; deklarasi variable;
jariLingkaran, luasLingkaran: numerik;
TAMPIL: Boolean; KONSTANTA:
phi=3.14;
Begin
Input jariLingkaran, TAMPIL;
luasLingkaran = phi* jariLingkaran^2;
If TAMPIL = TRUE then print luasLingkaran;
End.
	var
	jariLingkaran: real;
	luasLingkaran: real; begin
	phi := 3.14;
	readln (jariLingkaran);
	luasLingkaran = phi* jariLingkaran * jariLingkaran;
	writeln(luasLingkaran);
	end;

Pseudocode	Bahasa C++

luas_lingkaran; deklarasi variable;
jariLingkaran, luasLingkaran: numerik;
TAMPIL: Boolean; KONSTANTA:
phi=3.14;
Begin
Input jariLingkaran, TAMPIL;
luasLingkaran = phi*
	#include <stdio.h> using namespace std; int main() {
	double jariLingkaran = 10; double luasLingkaran = 0;
	luasLingkaran = Math.PI*(Math.pow (jariLingkaran, 2));
	cout << "Luas Lingkaran : " << luasLingkaran << endl;

Pseudocode	Bahasa C++
jariLingkaran^2; If TAMPIL = TRUE then print luasLingkaran; End.	retun 0; }

Pseudocode	Bahasa Java

luas_lingkaran; deklarasi variable;
jariLingkaran, luasLingkaran: numerik;
TAMPIL: Boolean; KONSTANTA:
phi=3.14;
Begin
Input jariLingkaran, TAMPIL;
luasLingkaran = phi* jariLingkaran^2;
If TAMPIL = TRUE then print luasLingkaran;
End.
	public class Luas Lingkaran {
	public static void main(String[] args) { double jariLingkaran = 10;
	double luasLingkaran = 0;
	luasLingkaran = Math.PI*(Math.pow(jariLingkaran, 2)); System.out.println("Luas Lingkaran : " + luasLingkaran);
	}
	}

3.5 Variabel dan Tipe Data

Data merupakan sekumpulan fakta yang dapat diolah sesuai dengan kebutuhan pengguna. Data dapat diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, suara, atau kombinasinya. Pengolahan data merupakan proses manipulasi data ke dalam bentuk yang lebih berarti /bermakna sehingga dapat menghasilkan sebuah informasi.

Data yang akan diproses dalam komputer memiliki tipe yang bermacam-macam. Tipe data ini nantinya digunakan untuk menentukan tipe variabel dalam proses di komputer. Tipe data yang paling sering digunakan adalah sebagai berikut:

1. **Integer**: bilangan bulat, bukan nilai pecahan (1, 2, 500, -18, 1000)
2. **Real**: bilangan real, bilangan dengan digit desimal di belakang koma (3.5, 5.0, 20.75, - 6.25)
3. **Karakter**: data alfanumerik ('A', 'B', '*', '@', '9', '?')
4. **String**: untaian karakter ("Pemrograman", "SISTEM INFORMASI")
5. **Boolean**: hanya terdiri dari 2 nilai yaitu TRUE dan FALSE.

Variabel merupakan suatu lokasi memori komputer yang digunakan untuk menampung dan menyimpan data yang akan diproses. Ada beberapa ciri-ciri variabel, antara lain:

a. Penamaannya bersifat UNIK, artinya harus berbeda dengan variabel lainnya yang digunakan pada algoritma. Contoh: bilangan1, bilangan2, hasil, luas, panjang, lebar, nama, alamat, dan sebagainya.
b. Satu variabel hanya untuk menyimpan satu jenis data saja.
c. Tidak tergantung pada besarnya data.
d. Nilainya bisa berubah-ubah.

Penamaan variabel sebaiknya disesuaikan dengan isi data di dalamnya. Contohnya variabel yang berisi data nilai ujian sebaiknya diberi nama **Nilai**. Hal ini untuk memudahkan pemahaman terhadap alur logika suatu algoritma atau program. Ada beberapa aturan yang harus dipatuhi dalam penamaan variabel, antara lain:

a. Karakter pertama harus berupa huruf, tidak boleh berupa angka.
   Contoh :
   nilai : benar
   Nilai1 : benar
   1Nilai : salah karena diawali dengan angka
b. Tidak boleh ada spasi.
   Contoh :
   Nilai Tugas : salah, karena terdapat spasi
   NilaiTugas : benar
c. Tidak boleh ada tanda-tanda khusus (* , /, + , - , ^ , &) kecuali *underscore* (_).
   Contoh :
   nilai-ujian : salah, karena terdapat notasi pengurangan (-)
   nilai_ujian : benar

Besaran atau nilai variabel dapat diubah dengan ketentuan nilai perubahannya masih berada dalam jangkauan tipe datanya. Ada juga variabel yang nilainya tetap, yang biasa disebut sebagai **Konstanta**. Konstanta hampir sama dengan variabel. Konstanta merupakan variabel yang mempunyai besaran atau nilai yang tetap selama program dijalankan. Contoh: phi, panjangmaksimal, *username*, *password*, dan lain sebagainya.

Operasi Penjumlahan mempunyai variabel masukan BIL1, BIL2, dan TAMPIL (Gambar 3.14). BIL1 dan BIL2 berisi angka yang akan dijumlahkan. Karena kedua variabel tersebut akan dikenai operasi aritmatika, maka harus bertipe numerik. Misalkan angka yang akan dijumlahkan adalah bilangan bulat, sehingga BIL1 dan BIL2 juga bertipe numerik bilangan bulat. Salah satu tipe data bilangan bulat adalah integer (**int**).

Variabel TAMPIL dikenai operasi relasi untuk memberi keputusan apakah hasil operasi penjumlahan akan ditampilkan ke layar atau tidak. Operasi relasi ini menghasilkan dua nilai, TRUE atau FALSE. Maka TAMPIL bisa dinyatakan bertipe **BOOLEAN**.

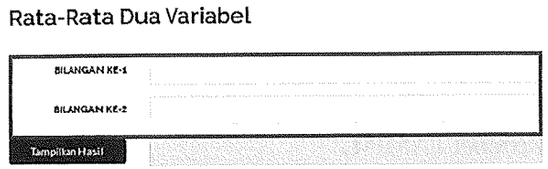
Berdasarkan pernyataan matematika **HASIL = BIL1 + BIL2** diketahui bahwa hasil operasi penjumlahan dimasukkan dalam variabel HASIL. Jika BIL1 dan BIL2 adalah bilangan bulat, maka hasil operasi penjumlahan antara bilangan bulat dengan bilangan bulat akan menghasilkan bilangan bulat juga (lihat Tabel 3.1) sehingga variabel **HASIL** bertipe **int**.

Berikutnya kita akan menentukan tipe data variabel pada Operasi Menghitung Luas Lingkaran. Operasi menghitung luas lingkaran mempunyai variabel masukan **jariLingkaran**. Pada contoh ini kita berasumsi bahwa jari-jari lingkaran dinyatakan dalam bilangan bulat atau integer. Berdasarkan alasan ini dapat ditentukan variabel **jariLingkaran** bertipe **int**. Menghitung luas lingkaran juga memerlukan nilai konstanta **phi**. Konstanta **phi** mempunyai nilai ketetapan sebesar 3.14 (bilangan pecahan). Rumus luas lingkaran **luasLingkaran = phi * jariLingkaran ^2**. Dari rumus tersebut terjadi operasi perkalian antara bilangan pecahan dengan bilangan bulat. Dalam Tabel 3.1 ditunjukkan operasi perkalian antara bilangan pecahan dengan bilangan bulat menghasilkan nilai bilangan pecahan. Hasil operasi menghitung luas lingkaran dimasukkan dalam variabel **luasLingkaran**. Kesimpulannya tipe variabel luasLingkaran adalah bilangan pecahan (***double***).

3.6 Identifikasi Variabel Berdasarkan Komponen Pengolahan Data

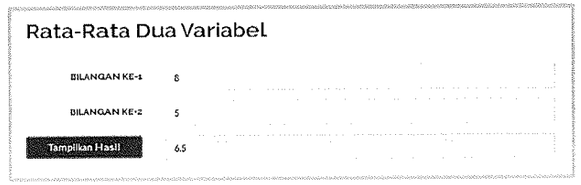
Penggunaan Variabel memang dapat diklasifikasikan berdasarkan komponen pengolah data. Masih ingat BAB 1? Komponen Pengolahan Data terdiri dari 3 komponen yaitu komponen Masukan, Proses, dan Keluaran. Pada subbab ini diharapkan kita dapat memilah dan mengklasifikasikan variabel yang digunakan pada algoritma, mana saja yang termasuk variabel Masukan, variabel Proses, dan variabel Keluaran. Untuk memudahkan proses klasifikasi variabel, kita dapat mengaktifkan penalaran.com, klik menu **Pernyataan Matematika - Aplikasi Aritmetika**, pilih dan klik **Menghitung Rata-Rata Dua Variabel**.

1. Klik *toolbar* menu MASUKAN. Operasi menghitung rata-rata dua variabel mempunyai dua masukan yang ditandai dengan kotak merah pada Gambar 3.31, Bagian masukanini adalah BILANGAN KE-1 dan BILANGAN KE-2.



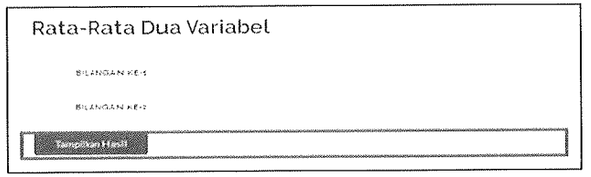
Gambar 3.31 *Masukan Operasi Menghitung Rata-rata Dua Variabel*

Ketikkan sebuah bilangan pada kotak isian BILANGAN KE-1 (misal diisi nilai 8) dan ketikkan sebuah bilangan pada kotak isian BILANGAN KE-2 (misal diisi nilai 5). Klik tombol **Tampilkan Hasil**, maka akan muncul nilai 6.5 pada kotak hasil yang artinya nilai 6.5 adalah nilai rata-rata dari penjumlahan antara 8 dan 5. (lihat Gambar 3.32).



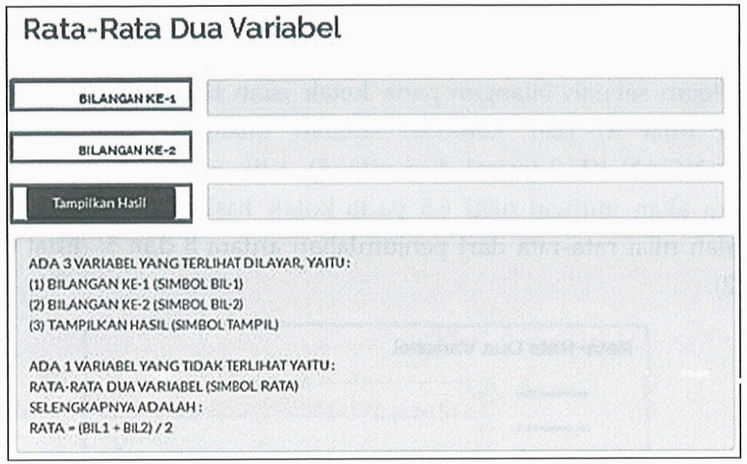
Gambar 3.32 *Contoh Operasi Menghitung Rata-rata Dua Variabel*

2. Klik *toolbar* menu KELUARAN. Pada Gambar 3.32 bagian keluaran ditandai dengan kotak merah. Pada kotak merah ini akan tertulis nilai hasil perhitungan rata-rata antara Bilangan ke-1 dan Bilangan ke-2. Pada Gambar 3.32 bagian keluaran ini berisi nilai 6.5.



Gambar 3.33 *Keluaran Operasi Menghitung Rata-rata Dua Variabel*

3. Klik *toolbar* menu VARIABEL. Ada 3 (tiga variabel yang terlihat di layar dan berfungsi sebagai variabel masukan. Variabel BILANGAN KE-1 dan BILANGAN KE-2 adalah bilangan yang akan dicari rata-ratanya. Variabel ini berturut-turut dinyatakan dengan simbol atau nama variabel BIL1 dan BIL2. Tampilkan Hasil adalah variabel keputusan untuk memutuskan apakah nilai rata-rata hasil proses ditampilkan atau tidak dan dinyatakan dengan simbol atau nama variabel TAMPIL.

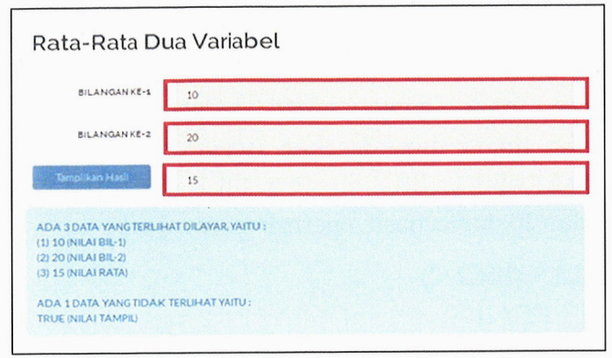


Gambar 3.34 *Variabel pada Operasi Menghitung Rata-rata Dua Variabel*

Ada satu variabel yang tidak terlihat di layar, yaitu variabel yang digunakan untuk menampung nilai hasil proses rata-rata dengan nama RATA. Variabel RATA dinyatakan dengan pernyataan matematika RATA = (BIL1 + BIL2)/2. Nilai variabel RATA inilah yang ditampilkan ke layar sebagai nilai keluaran.

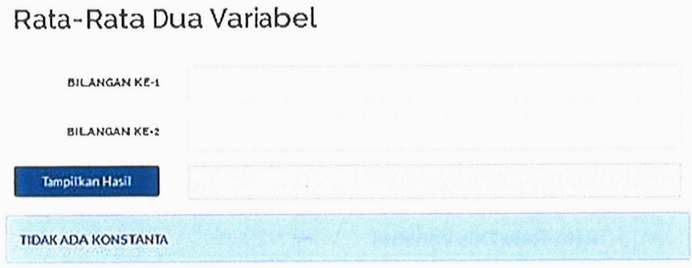
4. Klik *toolbar* menu DATA. Gambar 3.35 terdapat kotak warna merah yang menunjukkan itulah data. Data adalah nilai yang dioperasikan. Data ini masing-masing nilainya dimasukkan ke dalam variabel. Berdasarkan pokok bahasan di atas tentang variabel dan data yang terdapat pada Gambar 3.35 dapat dinyatakan bahwa data 10 akan

masuk dalam variabel BIL1, data 20 masuk dalam variabel BIL2, dan data 15 masuk dalam variabel RATA.



Gambar 3.35 *Data pada Operasi Menghitung Rata-rata Dua Variabel*

5. Klik *toolbar* menu KONSTANTA. Sesuai definisinya konstanta adalah variabel yang mempunyai besaran atau nilai tetap selama program dijalankan. Menurut rumus rata-rata dua variabel, yaitu RATA = (BIL1 + BIL2)/2 terlihat bahwa pada rumus ini tidak terdapat kontsanta.



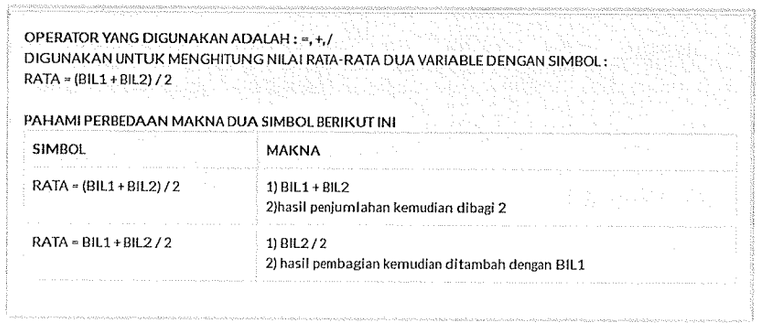
Gambar 3.36 *Konstanta pada Operasi Menghitung Rata-rata Dua Variabel*

6. Klik *toolbar* menu OPERATOR maka di layar akan terdapat tampilan seperti Gambar 3.37. Operator yang digunakan untuk menghitung rata-rata adalah operator aritmatika yang terdiri dari operator penugasan (=), operator penjumlahan (+), dan operator pembagian (/). Beserta variabel-variabel yang digunakan, operator-operator ini membentuk penyataan matematika:

RATA = (BIL1+BIL2)/2.

Hati-hati dalam menuliskan pernyataan matematika. Perhatikan prioritas dari masing-masing operator (lihat Tabel 3.1). Apabila tanda kurung pada pernyataan matematika di atas dihilangkan, maka akan menghasilkan nilai yang salah karena operator/mempunyai prioritas lebih tinggi dibandingkan operator + sehingga operasinya menjadi RATA=BIL1+(BIL2/2). Jika BIL1 dan BIL2 masing-masing bernilai 10 dan 20, maka hasil operasinya menjadi:

RATA = BIL1 + (BIL2/2)
= 10 + (20/2)
= 10 + 10
= 20 ⟶ Salah, karena rata-rata antara 10 dan 20 adalah 15

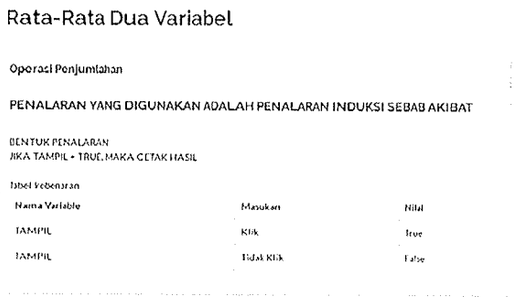

OPERATOR YANG DIGUNAKAN ADALAH : =, +, /
DIGUNAKAN UNTUK MENGHITUNG NILAI RATA-RATA DUA VARIABLE DENGAN SIMBOL :
RATA = (BIL1 + BIL2) / 2

PAHAMI PERBEDAAN MAKNA DUA SIMBOL BERIKUT INI

SIMBOL	MAKNA
RATA = (BIL1 + BIL2) / 2	1) BIL1 + BIL2 2)hasil penjumlahan kemudian dibagi 2
RATA = BIL1 + BIL2 / 2	1) BIL2 / 2 2) hasil pembagian kemudian ditambah dengan BIL1

Gambar 3.37 *Operator pada Operasi Menghitung Rata-rata Dua Variabel*

7. Klik *toolbar* menu PENALARAN


Rata-Rata Dua Variabel

Operasi Penjumlahan

PENALARAN YANG DIGUNAKAN ADALAH PENALARAN INDUKSI SEBAB AKIBAT

BENTUK PENALARAN
JIKA TAMPIL = TRUE, MAKA CETAK HASIL

Tabel Kebenaran

Nama Variable	Masukan	Nilai
TAMPIL	Klik	True
TAMPIL	Tidak Klik	False

Gambar 3.38 *Penalaran pada Operasi Menghitung Rata-rata Dua Variabel*

Penalaran yang digunakan di sini adalah penalaran induksi sebab akibat, yaitu penalaran yang mengemukakan fakta sebagai sebab untuk mendapatkan kesimpulan sebagai akibat. Perhatikan Tabel 3.5 berikut ini:

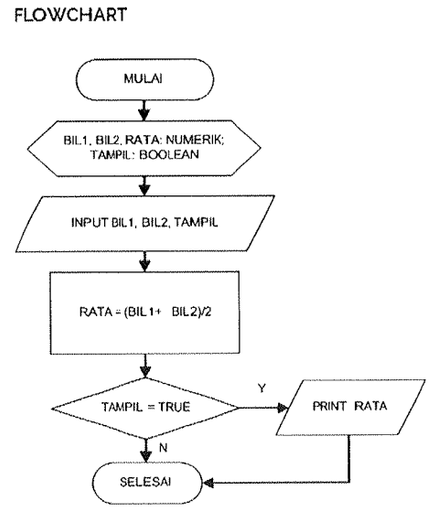
Tabel 3.5 *Tabel Kebenaran Penalaran Sebab Akibat*

Variabel	Masukan/fakta	Nilai	Kesimpulan
TAMPIL	Klik	TRUE	Cetak RATA
TAMPIL	Tidak Klik	FALSE	Tidak Cetak RATA

Berdasarkan tabel di atas diketahui bahwa jika pengguna melakukan klik pada tombol Tampilkan Hasil itu artinya variabel TAMPIL bernilai TRUE sehingga mengakibatkan nilai rata-rata dalam variabel RATA ditampilkan ke layar. Jika tombol Tampilkan Hasil tidak diklik berarti variabel TAMPIL bernilai FALSE yang mengakibatkan nilai rata-rata tidak ditampilkan ke layar. Uraian ini bisa dinyatakan dalam logika matematika:

```
Jika TAMPIL = TRUE, maka Cetak RATA
```

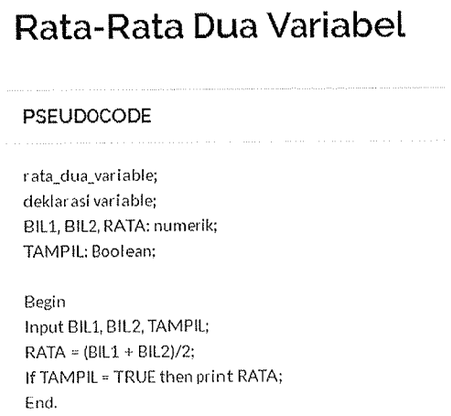
8. Klik *toolbar* menu FLOWCHART



Gambar 3.39 *Flowchart Operasi Menghitung Rata-rata Dua Variabel*

Gambar 3.39 adalah *flowchart* untuk operasi menghitung rata-rata dua variabel. *Flowchart* ini dibuat berdasarkan kaidah *Input* - Proses - *Output* dengan memperhatikan pokok bahasan di atas, yaitu Masukan, Keluaran, Variabel, Data, Operator, dan Penalaran. Kaidah-kaidah tersebut dinyatakan dalam simbol-simbol *flowchart*.

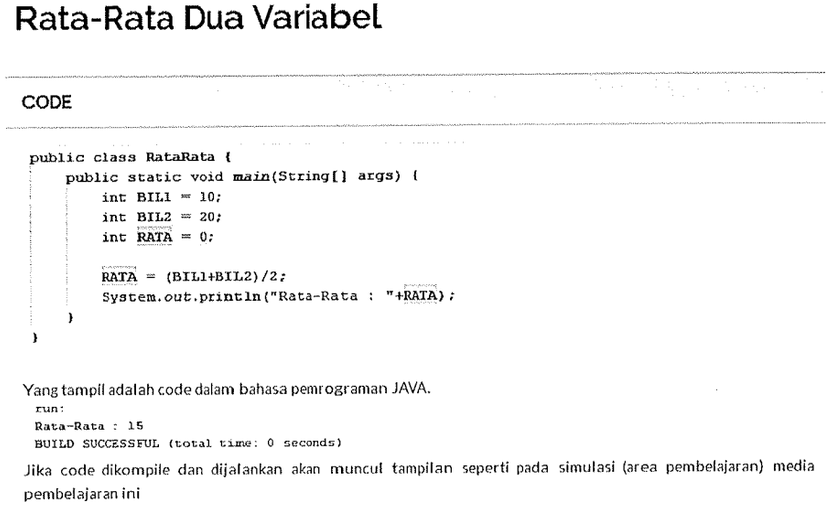
9. Klik *toolbar* menu *PSEUDOCODE*. Gambar 3.40 adalah bentuk lain dari algoritma selain *flowchart* yang terdapat pada Gambar 3.39. *Pseudocode* ini seperti arti katanya, dinyatakan dalam kalimat-kalimat yang lebih mendekati ke bahasa pemrograman sehingga bentuk *pseudocode* ini lebih mudah diterjemahkan ke dalam bahasa pemrograman dibandingkan dengan bentuk *flowchart*. Tetapi untuk pemahaman urutan proses lebih mudah menggunakan *flowchart* karena ada panah sebagai arah aliran.



Gambar 3.40 *Pseudocode Operasi Menghitung Rata-rata Dua Variabel*

10. Klik *toolbar* menu CODE maka dilayar muncul tampilan seperti Gambar 3.41. Tampilan tersebut merupakan implementasi dari algoritma *flowchart* atau *pseudocode* dalam bahasa pemrograman Java. Program tersebut tetap menggunakan kaidah *Input* - Proses - *Output*. Dalam program tersebut juga terdapat variabel dan data. Pada bahasa pemrograman, variabel harus dideklarasikan dengan tipe data tertentu sesuai dengan jenis data yang dimasukkan dalam variabel

tersebut. Di sini variabel masukan dideklarasikan dengan tipe data integer (int) karena variabel masukan tersebut berisi data bilangan bulat (lihat Tabel 3.1). Prosesnya adalah pernyataan matematika untuk menghitung nilai rata-rata di mana dalam pernyataan tersebut terdapat operasi pembagian. Operasi pembagian memungkinkan menghasilkan nilai pecahan, sehingga variabel keluarannya harus bertipe pecahan juga.



Gambar 3.41 *Kode Program Operasi Menghitung Rata-rata Dua Variabel*

Adapun bentuk *pseudocode* dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Menghitung Rata-rata Dua Variabel dapat dilihat pada tabel di bawah ini.

Menghitung Rata-rata Dua Variabel

Pseudocode	Bahasa Pascal
rata_dua_variable; deklarasi variable; BIL1, BIL2, RATA: numerik; TAMPIL: Boolean;	var BIL1 : integer; BIL2 : integer; RATA : integer;

Pseudocode	Bahasa Pascal

Begin
Input BIL1, BIL2, TAMPIL; RATA = (BIL1 + BIL2)/2;
If TAMPIL = TRUE then print RATA;
End.
	begin
	readln (BIL1); readln (BIL2);
	RATA := (BIL1 + BIL2)/2;
	writeln(RATA); end;

Pseudocode	Bahasa C++

rata_dua_variable; deklarasi variable;
BIL1, BIL2, RATA: numerik; TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; RATA = (BIL1 + BIL2)/2;
If TAMPIL = TRUE then print RATA;
End.
	#include <stdio.h> using namespace std; int main() {
	int BIL1 = 10; int BIL2 = 20; int RATA = 0;
	RATA = (BIL1 + BIL2)/2;
	cout<<"Rata-rata : " << RATA<<endl;
	return 0;
	}

Pseudocode	Bahasa Java

rata_dua_variable; deklarasi variable;
BIL1, BIL2, RATA: numerik; TAMPIL: Boolean;
 	
Begin
Input BIL1, BIL2, TAMPIL; RATA = (BIL1 + BIL2)/2;
If TAMPIL = TRUE then print RATA;
End.
	public class RataRata {
	public static void main(String[] args) { int BIL1 = 10;
	int BIL2 = 20; int RATA = 0;
	RATA = (BIL1 + BIL2)/2;
	System.out.println("Rata-rata : " + RATA);
	}
	}

3.7 Rangkuman

Berpikir komputasi dalam menangani permasalahan adalah kemampuan mengidentifikasi bagian masukan, proses, dan keluaran dalam permasalahan. Dalam masukan, proses, dan keluaran diperlukan beberapa komponen, antara lain data, variabel, operator, pernyataan matematika, dan logika matematika.

Data merupakan sekumpulan fakta yang dapat diolah sesuai dengan kebutuhan pengguna. Data dapat diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, suara, atau kombinasinya. Data yang akan diproses dalam komputer memiliki tipe yang bermacam-macam. Tipe

data ini nantinya digunakan untuk menentukan tipe variabel dalam proses di komputer.

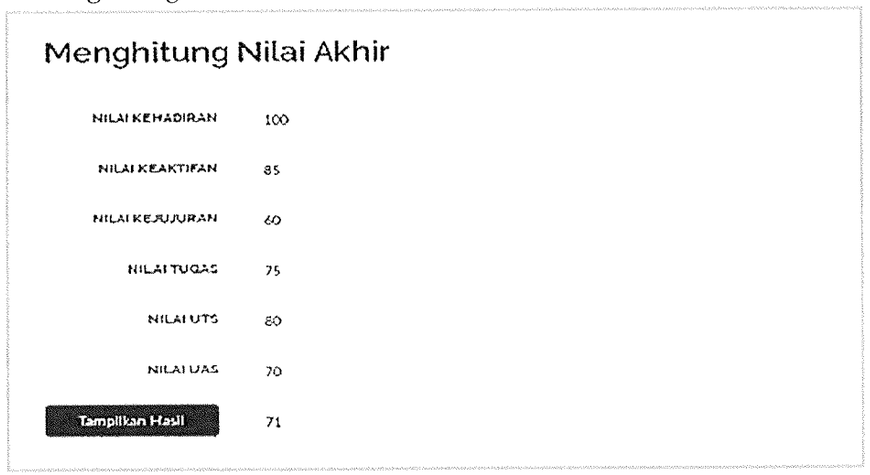
Variabel merupakan suatu lokasi memori komputer yang digunakan untuk menampung dan menyimpan data yang akan diproses. Tipe data variabel tergantung jenis data di dalamnya. Macam-macam tipe data adalah integer, real, karakter, string, boolean. Variabel dan data terdapat dalam bagian masukan, bagian proses, dan bagian keluaran.

Pernyataan matematika adalah sebuah proses yang mentransformasikan masukan menjadi keluaran yang diinginkan melalui proses perhitungan (komputasi). Pernyataan matematika terdiri dari operand dan operator aritmatika.

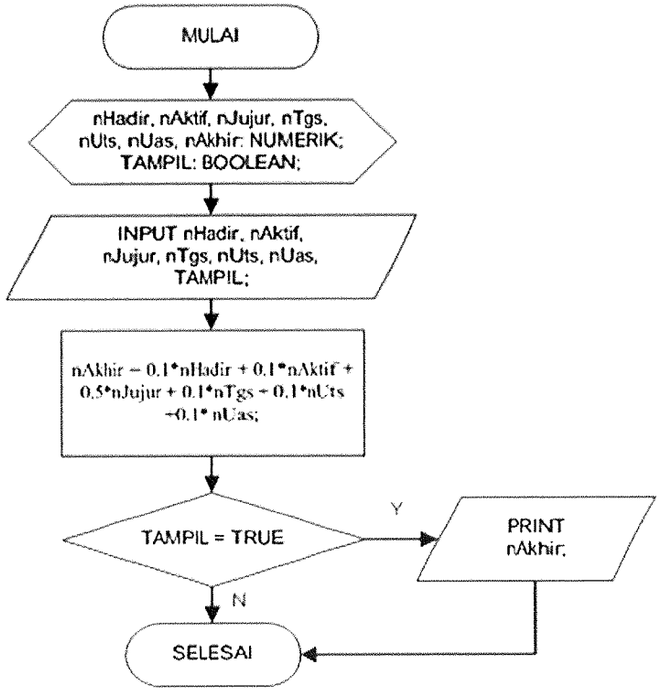
Logika Matematika merupakan pernyataan yang digunakan dalam proses yang mengandung percabangan, tepatnya pada kondisi yang berlaku sebagai syarat/aturan dari sebuah permasalahan. Logika matematika menggunakan operator relasi dan operator Logika.

3.8 Latihan Soal

1. Tentukan bagian Masukan, Keluaran, Variabel, dan Data untuk aplikasi menghitung nilai akhir di bawah ini.



2. Di bawah ini adalah *flowchart* untuk menghitung nilai akhir. Tentukan operator dan pernyataan matematika dalam *flowchart* tersebut.



3. Lakukan penalaran induksi sebab akibat untuk mengidentifikasi logika matematika pada gambar *flowchart* soal no. 2 di atas.
4. Berdasarkan jawaban soal no.1 sampai dengan jawaban soal no. 3 di atas, susunlah instruksi komputasi dalam bentuk *pseudocode* dan kode program (*Source Code*).

Berbelanja di Pasar Tradisional

Pada suatu hari, Adi disuruh kakak perempuannya ke pasar untuk membeli beberapa keperluan dapur. Ia diberi uang oleh kakaknya sebesar Rp. 150.000. Menuju pasar, Adi menggunakan transportasi umum yaitu becak dengan tarif sebesar Rp. 7.500. Sesampainya di pasar, Adi membeli cabai sebanyak 1,333 Kg, 2 Ons kunyit, 4 ikat

bayam, 2 Kg daging ayam, 1 bungkus garam, dan 2 Kg gula. Di pasar dia mengeluarkan uang sebanyak Rp. 110.000. Menuju perjalanan pulang, Adi menggunakan angkutan umum yaitu Bemo dengan tarif Rp. 5.000. Adi berhenti di depan swalayan untuk membeli sabun detergen dengan merk "R". Saat akan membayar di kasir Laki-Laki, dia mendapatkan promo Bulanan sebesar 37%. Sehingga harga sabun yang dibayarkan Adi menjadi Rp. 17.335,217. Sesampainya di Rumah, Adi mengembalikan uang kembalian sisanya ke Kakaknya.

5. Berdasarkan bacaan di atas, deklarasikan beberapa variabel yang telah digarisbawahi sesuai dengan *type* data yang benar.
6. Berdasarkan jawaban soal no. 5, buatlah kode program (*Source Code*) untuk menampilkan hasil variabel yang sudah dideklarasikan.
7. Benarkan program berikut sehingga ketika dimasukkan ke dalam program tidak terjadi error, dan tentukan *type* data apa yang sesuai dengan program tersebut.

```
a. float A = 188.55565;
b. doubleB =2288880;
c. int C = 222909990890038;
d. char D = "Rizal Arif Firmansyah";
e. string E = "Muhammad Arif" , "Niko Hasda Prayogo";
f. double F = 55.?2938f;
g. string G = 'L';
```

8. Tampilkan program hasil jawaban soal no. 7 sesuai dengan urutan abjad.
9. Perbaikilah program di bawah ini yang masih terdapat error, lalu tampilkan data yang memanggil variabel H.

```
int H = 400; double I = H;
```

10. Perhatikan program di bawah ini: Perbaiki program di bawah ini sesuai dengan *value* yang tersedia, dan ketika menampilkan yang bukan sama dengan **nilai2** maka yang tampil yaitu *value* TRUE, dan begitu juga sebaliknya.

```
string nilail = true;
int nilai2 = false;
```

Soal cerita kapal selam: "Sebuah kapal selam berada pada kedalaman 64 m di bawah permukaan air laut. Satu jam kemudian, letak kapal tersebut berada pada kedalaman 72 m di bawah

permukaan air laut. Sekarang kapal tersebut berada pada kedalaman 120 m di bawah permukaan air laut. Tentukan jarak yang ditempuh kapal tersebut dari posisi semula". (Suparmin, 2017: 11).

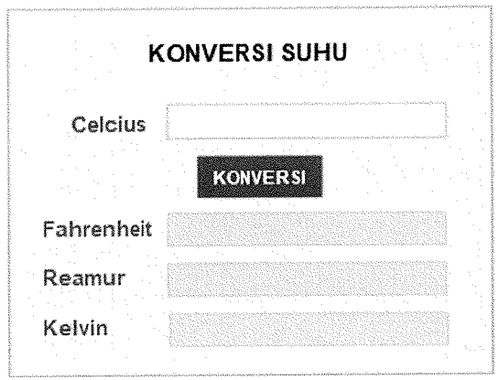
Berdasarkan soal cerita kapal selam di atas, selesaikan pertanyaan soal no. 11 sampai dengan soal no. 14.

11. Tentukan masukan, keluaran, variabel, dan tipe data untuk soal cerita kapal selam.
12. Tentukan operator dan pernyataan matematika untuk proses penyelesaian soal cerita kapal selam.
13. Buatlah *flowchart* dan *pseudocode* untuk menyelesaikan soal cerita kapal selam.
14. Ubahlah algoritma dari jawaban soal no.13 menjadi kode program (*Source Code*).

Soal cerita Siput: "Seekor siput berada di dasar sumur kering sedalam 6,3 meter. Setiap hari siput tersebut bergerak naik sejauh 120 centimeter dan tergelincir turun sejauh 50 centimeter. Setelah berapa hari siput dapat naik ke permukaan sumur?" (Suparmin, 2017: 13).

Berdasarkan soal cerita siput di atas, selesaikan pertanyaan soal no. 15 s/d soal no. 17.

15. Tentukan masukan, keluaran, variabel, dan tipe data untuk soal cerita siput.
16. Tentukan operator dan pernyataan matematika untuk proses penyelesaian soal cerita siput.
17. Buatlah *pseudocode* dan kode program untuk menyelesaikan soal cerita siput.
18. Tentukan masukan, keluaran, variabel, dan tipe data untuk aplikasi konversi suhu di bawah ini.



19. Tentukan operator, pernyataan matematika, dan logika matematika untuk proses konversi suhu dari derajat Celcius ke derajat Fahrenheit, Reamur, dan Kelvin jika diketahui rumus konversi sebagai berikut:

$$\frac{99}{55}$$

$$FF = \frac{44}{CC} + 3333$$

$$RR = \frac{CC}{CC}$$

$$KK = CC + 332233$$

20. Buatlah *flowchart* dan kode program (*Source Code*) untuk konversi duhu dari derajat Celcius ke derajat Fahrenheit, Reamur, dan Kelvin.

-oo0o-

BAB 4

PERCABANGAN

Deskripsi Singkat

Bab 4 menjelaskan pentingnya kemampuan berpikir komputasi yaitu masukan, proses, dan keluaran bagi semua orang dalam penggunaan konsep percabangan. Percabangan adalah sebuah perintah/instruksi yang dapat dijalankan dengan syarat/kondisi tertentu harus dipenuhi terlebih dahulu. Apabila syarat/kondisi belum terpenuhi, maka perintah/instruksi tersebut tidak akan pernah dijalankan.

Pokok bahasan dalam bab ini meliputi pendahuluan, percabangan dan pernyataan matematika, percabangan dan logika matematika, percabangan dan instruksi komputasi, percabangan dan variabel, klasifikasi percabangan pada pengolahan data, rangkuman, dan ditutup dengan latihan soal.

Bab ini membahas tentang konsep dasar percabangan dan simbol percabangan di flowchart dan pseudocode. Proses percabangan berkaitan erat dengan logika matematika serta penggunaan operator relasi dan operator logika. Konsep penalaran induksi sebab akibat dibahas juga dalam bab ini.

Tujuan Pembelajaran

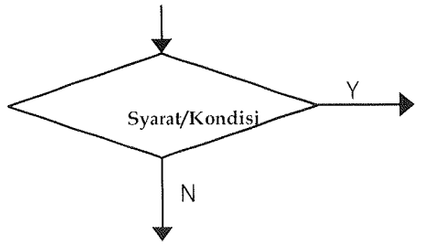
Akhir pembelajaran atau diskusi, diharapkan pembaca mempunyai beberapa ketrampilan berikut:

1. Jika diberikan soal pembaca dapat mengidentifikasi masukan, keluaran, Variabel, dan Data pada operasi percabangan.
2. Jika diberikan soal pembaca dapat mengidentifikasi bagian proses, operator dan pernyataan matematika pada bagian proses percabangan.
3. Jika diberikan soal pembaca dapat melakukan penalaran induksi sebab akibat untuk mengidentifikasi logika matematika dalam proses percabangan..
4. Berdasarkan ketrampilan 1 sampai dengan 3 di atas diharapkan pembaca dapat menyusun instruksi komputasi operasi percabangan dalam bentuk flowchart, pseudocode, dan code programnya.

4.1 Pendahuluan

BYPada proses percabangan, sebuah perintah/instruksi dapat dijalankan dengan syarat/kondisi tertentu harus dipenuhi terlebih dahulu. Apabila syarat/kondisi belum terpenuhi, maka perintah/instruksi tersebut tidak akan pernah dijalankan. Syarat atau kondisi yang digunakan boleh lebih dari satu kondisi (tidak terbatas). Proses percabangan juga dikenal sebagai proses pemilihan atau proses pengambilan keputusan.

Pada algoritma berbentuk flowchart, percabangan dinotasikan dengan simbol berbentuk belah ketupat yang mempunyai dua panah keluar. Satu panah mewakili syarat/kondisi bernilai TRUE (T) atau YES (Y) atau YA (Y). Satu panah lagi mewakili syarat/kondisi FALSE (F) atau NO (N) atau TIDAK (T).



Gambar 4.1 *Simbol Flowchart Percabangan*

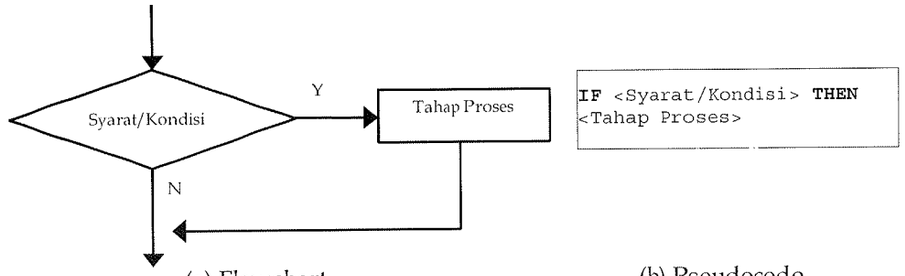
Apabila algoritma percabangan dinyatakan dengan pseudocode ditandai dengan kata kunci IF-THEN.


```
IF <Syarat/Kondisi> THEN
```

Gambar 4.2 *Simbol Pseudocode Percabangan*

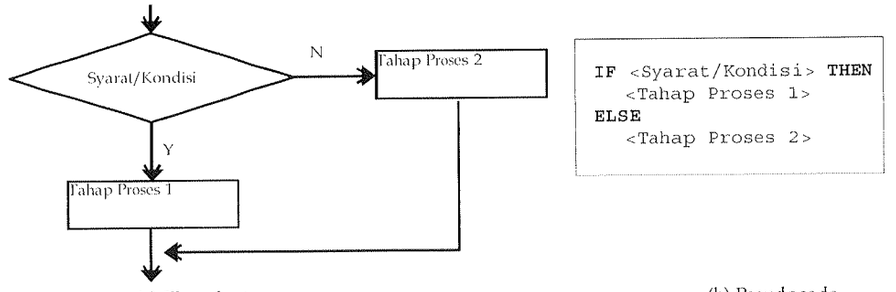
Ada dua macam bentuk dasar proses percabangan. Bentuk tunggal dan bentuk ganda. Percabangan bentuk tunggal mempunyai prinsip kerja jika syarat/kondisi bernilai benar (Yes/Y) maka Tahap Proses dikerjakan. Jika syarat/kondisi bernilai salah (No/N) maka Tahap Proses tidak dikerjakan (lihat Gambar 4.3).

Percabangan bentuk ganda mempunyai prinsip kerja jika syarat/kondisi bernilai benar, maka Tahap Proses 1 dikerjakan dan jika syarat/kondisi bernilai salah, maka Tahap Proses 2 dikerjakan. Pada pseudocode percabangan bentuk ganda ditandai dengan kata ELSE. Hal ini dapat dilihat dalam Gambar 4.4.



(a) Flowchart (b) Pseudocode

Gambar 4.3 *Percabangan Bentuk Tunggal*



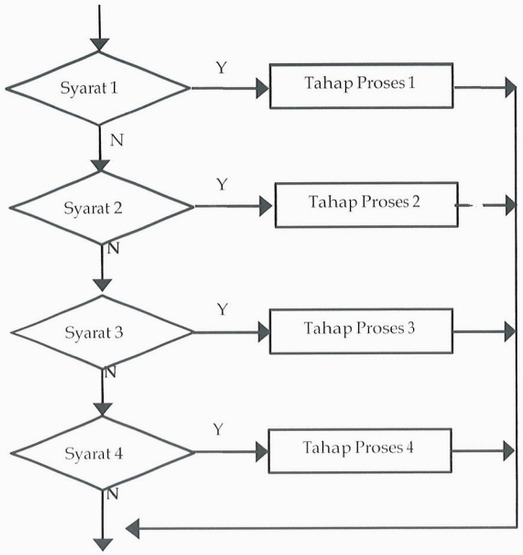
(a) Flowchart (b) Pseudocode

Gambar 4.4 *Percabangan Bentuk Ganda*

Gabungan dari bentuk tunggal dan bentuk ganda bisa menghasilkan percabangan bentuk jamak seperti terlihat pada Gambar 4.5.

Syarat/Kondisi adalah logika matematika yang mengunakan operator relasi (Tabel 3.2) dengan hasil operasi bernilai TRUE atau FALSE. Logika matematika yang menyatakan syarat/kondisi boleh juga menggunakan operator aritmatika (Tabel 3.1) maupun operator logika (Tabel 3.3) apabila diperlukan.

Tahap Proses adalah segala instruksi komputasi yang harus dikerjakan dan berkaitan dengan benar (Y) atau salah (N) hasil operasi relasi pada syarat/kondisi.




```
IF <Syarat 1 > THEN
   <Tahap Proses 1>
ELSE IF <Syarat 2 > THEN
   <Tahap Proses 2>
ELSE IF <Syarat 3 > THEN
   <Tahap Proses 3>
ELSE IF <Syarat 4 > THEN
   <Tahap Proses 4>
```

(a) Flowchart (b) Pseudocode

Gambar 4.5 *Percabangan Bentuk Jamak*

Berikutnya kita coba contoh dari **penalaran.com.** Klik menu **Pernyataan Matematika - Operasi Aritmatika**. Pilih dan klik **Operasi Pembagian** (lihat Gambar 4.6).

Gambar 4.6 *Menu Operasi Aritmetika Pembagian*

Setelah menu Operasi Pembagian diklik, muncul tampilan seperti Gambar 4.7. Kotak merah di gambar tersebut menunjukkan bagian masukan dari operasi pembagian. Operasi pembagian mempunyai dua masukan, yaitu BILANGAN KE-1 dan BILANGAN KE-2. Bagian ini tempat memasukkan dua bilangan yang akan dioperasikan.

Gambar 4.7 *Masukan Operasi Aritmetika Pembagian*

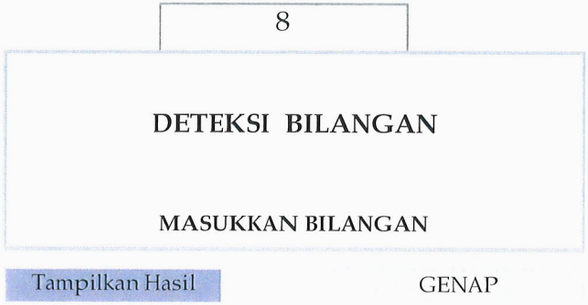
Selain masukan, operasi pembagian juga mempunyai keluaran seperti yang ditunjukkan pada Gambar 4.8.

Gambar 4.8 *Keluaran Operasi Aritmetika Pembagian*

Keluaran ini sebagai tempat untuk menampilkan nilai hasil operasi pembagian. Bagaimana dua nilai masukan ini bisa menghasilkan nilai keluaran yang benar ? Akan dijelaskan pada subbab berikutnya.

4.2 Percabangan dan Pernyataan Matematika

Salah satu contoh kasus yang menerapkan percabangan adalah mendeteksi sebuah bilangan yang diinputkan apakah sebagai bilangan genap atau bilangan ganjil.



Gambar 4.9 *Operasi Deteksi Bilangan*

Gambar 4.9 adalah operasi deteksi bilangan. Jika diberi masukan nilai genap (misal : 8) maka bagian keluaran akan menampilkan keterangan "GENAP". Jika diberi masukan bilangan ganjil (misal : 5) maka keluaran menampilkan keterangan "GANJIL".

Kasus ini memerlukan penyelesaian percabangan karena setiap bilangan yang dimasukkan mempunyai 2(dua) kemungkinan nilai, yaitu nilai Genap atau nilai Ganjil. Masalah utama yang harus diketahui pada kasus ini adalah apa syarat suatu bilangan dikatakan genap dan apa syarat dikatakan bilangan ganjil ?

Pernyataan 1: "Bilangan genap adalah bilangan yang habis dibagi 2" (bilangan dibagi 2 mempunyai sisa 0)

Pernyataan 2: "Bilangan ganjil adalah bilangan yang tidak habis dibagi 2" (bilangan dibagi 2 mempunyai sisa 1)

Ada 2 (dua) variabel yang tidak terlihat, yaitu:

1. **BIL** , untuk menampung nilai masukan, yaitu bilangan yang akan dideteksi
2. **TAMPIL**, untuk menampung nilai TRUE atau FALSE dari tombol **Tampilkan Hasil** sebagai keputusan untuk menampilkan hasil deteksi atau tidak.

Ada 2(dua) variabel yang tidak terlihat di layar, yaitu :

1. **SISA**, untuk menampung nilai sisa pembagian antara bilangan dengan 2.
2. **HASIL**, untuk menampung nilai keluaran hasil deteksi, yaitu "GENAP" atau "GANJIL".

Agar bisa menyelesaikan kasus ini berarti harus melakukan operasi aritmatika untuk mencari nilai sisa pembagian. Operator aritmatika yang digunakan adalah penugasan (=) dan operator aritmatika untuk mencari nilai sisa pembagian (**MOD**). Pernyataan matematika untuk mencari sisa pembagian ini adalah:

```
SISA = BIL MOD 2
```

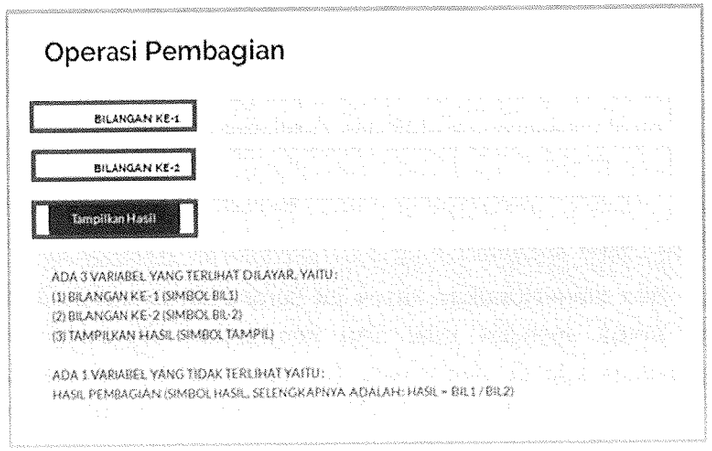
Nilai SISA ini nantinya akan dikenai operasi relasi untuk mendapatkan keputusan bilangan yang dideteksi sebagai bilangan genap atau bilangan ganjil. Jika SISA bernilai 0 berarti bilangan yang dideteksi adalah bilangan GENAP, dan jika SISA bernilai 1 maka bilangan yang dideteksi adalah bilangan GANJIL.

Contoh lain adalah operasi pembagian yang sudah dibahas di subbab Pendahuluan. Coba masuk ke penalaran.com. Aktifkan menu **Pernyataan Matematika - Operasi Aritmetika - Operasi Pembagian**. Klik toolbar menu VARIABEL yang ada di bagian kiri layar. Pada Gambar 4.10 ditunjukkan ada 3(tiga) variabel yang terlihat di layar, yaitu :

1. BILANGAN KE-1, dinyatakan dengan nama variabel BIL1
2. BILANGAN KE-2, dinyatakan dengan nama variabel BIL2
3. Tampilkan Hasil, dinyatakan dengan nama variabel TAMPIL

Selain itu ada 1(satu) variabel yang tidak terlihat di layar, yaitu variabel HASIL yang nilainya didapatkan dari operasi pembagian. Operasi pembagian di contoh ini adalah proses pembagian antara variabel BIL1 dan variabel BIL2 menggunakan pernyataan matematika:

HASIL = BIL1 / BIL2



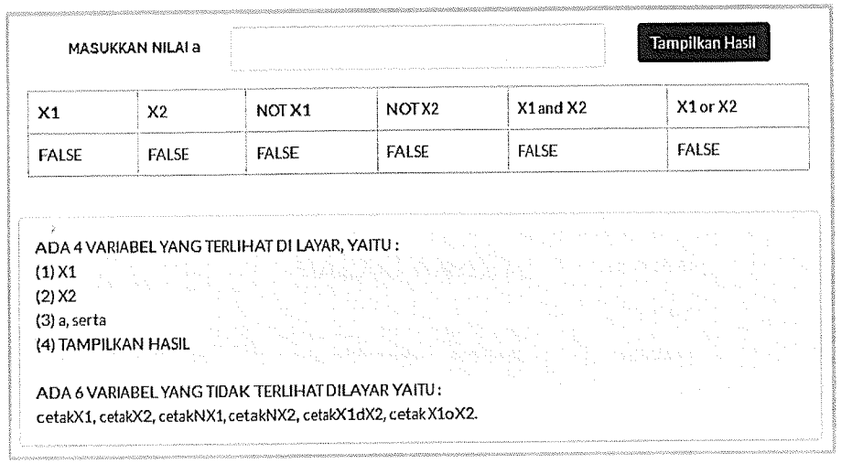
Gambar 4.10 *Variabel Operasi Aritmetika Pembagian*

4.3 Percabangan dan Logika Matematika

Implementasi konsep percabangan dapat dilihat dalam **penalaran.com.** Klik menu **Logika Matematika**, pilih dan klik **Logika Dasar** (Gambar 3.19). Klik toolbar menu VARIABEL maka akan tampil Gambar 4.11.

Ada 4(empat) variabel yang terlihat di layar sebagai masukan, yaitu

1. **MASUKKAN NILAI a**, dinyatakan dengan nama variabel **a**
2. **Tampilkan Hasil**, dinyatakan dengan nama variabel **TAMPIL**
3. **X1**, dinyatakan dengan nama variabel **X1**
4. **X2**, dinyatakan dengan nama variabel **X2**



Gambar 4.11 *Variabel dalam Operasi Logika Dasar*

Ada 6(enam) variabel yang tidak terlihat di layar berlaku sebagai keluaran, yaitu:

1. X1, nama variabel **cetakX1**
2. X2, nama variabel **cetakX2**
3. NOT X1, nama variabel **cetakNX1**
4. NOT X2, nama variabel **cetakNX2**
5. X1 AND X2, nama variabel **cetakX1dX2**
6. X1 OR X2, nama variabel **cetakX1oX2**

Logika matematika pada operasi logika dasar akan lebih mudah dipahami dengan mempelajari PENALARAN di penalaran.com. Klik toolbar menu PENALARAN dan perhatikan Gambar 4.12 dan aturan dalam Tabel 3.3.

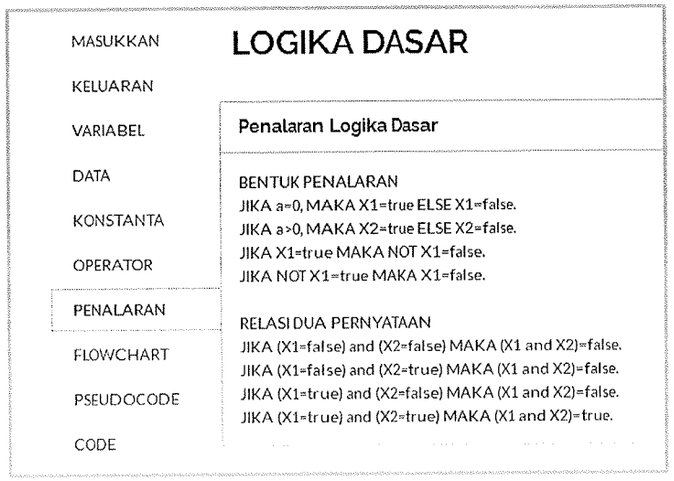
Pada Operasi Logika Dasar di penalaran.com terdapat syarat seperti di bawah ini :

X1 ADALAH SIMBOL PERNYATAAN 1: a = 0
X2 ADALAH SIMBOL PERNYATAAN 2: a > 0

Syarat tersebut bisa dinyatakan dalam Tabel 4.1.

Tabel 4.1 *Syarat dalam Operasi Logika Dasar*

Syarat	X1	X2
a=0	TRUE	FALSE
a>0	FALSE	TRUE



Gambar 4.12 *Penalaran Operasi Logika Dasar*

Dalam keadaan awal (inisialisasi) semua keluaran bernilai FALSE.

X1	X2	NOT X1	NOT X2	X1 and X2	X1 or X2
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

BENTUK PENALARAN dan RELASI DUA PERNYATAAN pada Gambar 4.12 dapat dituliskan dalam bentuk tabel berikut jika **Masukkan Nilai a** diisi dengan nilai **0**.

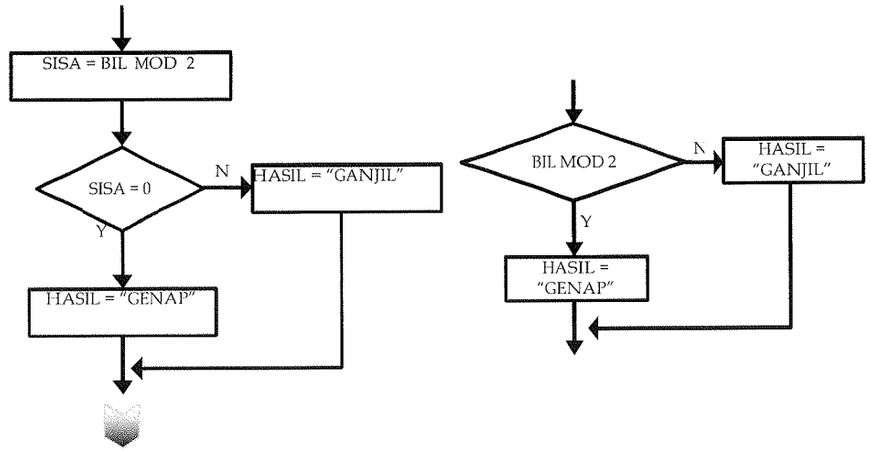
Nilai a	Syarat	X1	X2	NOT X1	NOT X2	X1 and X2	X1 or X2
0	a=0	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE

Sedangkan apabila Masukan Nilai a diisi dengan 5, yaitu (a>0), maka penalaran dan pernyataan dalam Gambar 4.12 dapat dinyatakan dengan tabel di bawah ini.

Nilai a	Syarat	X1	X2	NOT X1	NOT X2	X1 and X2	X1 or X2
5	a>0	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE

Pada subbab PERCABANGAN DAN PERNYATAAN MATEMATIKA terdapat kasus deteksi bilangan genap dan ganjil. Pada kasus tersebut terdapat pernyataan matematika untuk mencari nilai sisa pembagian bilangan dengan 2. Nilai sisa inilah yang nantinya dideteksi menggunakan logika matematika untuk mendapatkan keputusan bilangan genap atau ganjil.

Pernyataan matematika dan logika matematika ini bisa dijadikan satu menjadi sebuah logika matematika seperti Gambar 4.13.



Gambar 4.13 *Logika Matematika untuk Deteksi Bilangan Genap/Ganjil Tabel 4. 2 Contoh Operasi Deteksi Bilangan Genap/Ganjil*

BIL	BIL MOD 2	BIL MOD 2 = 0	KEPUTUSAN
5	1	FALSE	GANJIL
8	0	TRUE	GENAP

Tabel 4.2 adalah contoh operasi deteksi bilangan. Bilangan 5 adalah bilangan ganjil karena 5 dibagi 2 menghasilkan sisa 1. Nilai sisa 1 ini tidak memenuhi syarat BIL MOD 2 = 0 sehingga didapatkan keputusan 5 adalah bilangan ganjil.

Bilangan 8 adalah bilangan genap karena 8 dibagi 2 menghasilkan sisa 0. Nilai sisa 0 ini memenuhi syarat BIL MOD 2 = 0 sehingga didapatkan keputusan 8 adalah bilangan genap.

4.4 Percabangan dan Instruksi Komputasi

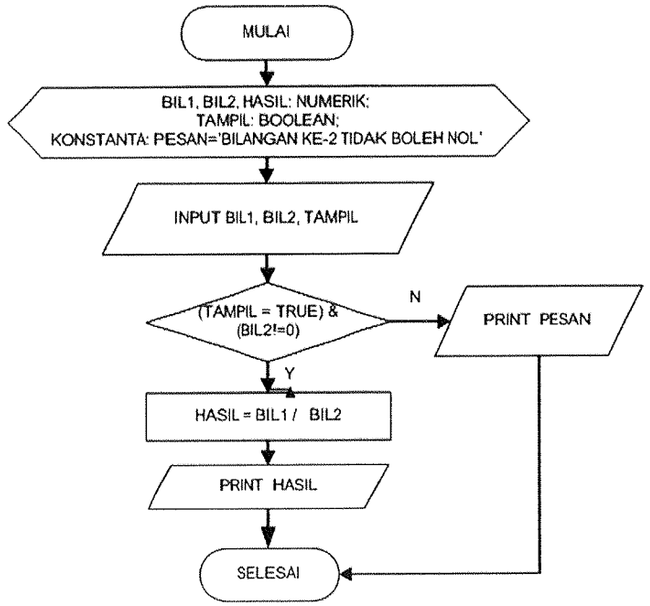
Operasi Aritmetika Pembagian di **penalaran.com** mempunyai variabel masukan BIL1 dan BIL2 yang berisi bilangan yang akan dioperasikan. Selain itu ada variabel masukan lain yang bernama TAMPIL yang bernilai TRUE atau FALSE untuk menentukan apakah hasil operasi pembagian ditampilkan atau tidak.

Operasi pembagian mempunyai syarat yang tidak boleh dilanggar, yaitu bilangan penyebut (pembagi) tidak boleh bernilai nol. Pada contoh ini bilangan pembagi berada pada variabel BIL2. Sebelum dioperasikan, variabel BIL2 harus diuji dulu apakah bernilai nol atau tidak. Uji ini dinyatakan dalam logika matematika. Apabila syarat ini terpenuhi (BIL2 ≠ 0) maka operasi perhitungan pembagian dilakukan. Operasi ini memerlukan operator pemuatan (=) dan operator pembagian (/).

Hasil operasi pembagian dimasukkan dalam variabel HASIL. Ada dua syarat yang harus dipenuhi untuk bisa menampilkan hasil pembagian. Syarat pertama, variabel TAMPIL harus bernilai TRUE. Syarat kedua nilai variabel BIL2 tidak boleh nol. Karena kedua syarat ini harus terpenuhi semua, maka kedua syarat ini dihubungkan dengan operator logika AND.

Uraian tentang operasi pembagian ini kemudian dinyatakan dalam instruksi komputasi berupa algoritma (*Flowchart* dan *Pseudocode*) dan kode program mengunakan bahasa pemrograman Java (lihat Gambar 4.14, Gambar 4.15, dan Gambar 4.16).

FLOWCHART



Gambar 4.14 *Flowchart Operasi Aritmetika Pembagian*

PSEUDOCODE

```
pembagian;
deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
KONSTANTA: PESAN='BILANGAN KE-2 TIDAK BOLEH NOL'

Begin
Input BIL1, BIL2, TAMPIL;
If (TAMPIL = TRUE) and (BIL2!= 0) then
Begin HASIL = BIL1/BIL2; print HASIL; end
Else print PESAN;
End.
```

Gambar 4.15 *Pseudocode Operasi Aritmetika Pembagian*

Operasi Pembagian

CODE

```
public class Pembagian {
    public static void main(String[] args) {
        int BIL1 = 400;
        int BIL2 = 20;
        int HASIL = 0;
        String PESAN = "BILANGAN KE-2 TIDAK BOLEH NOL";

        if(BIL2 == 0) System.out.println(PESAN);
        else {
            HASIL = BIL1 / BIL2;
            System.out.println("HASIL : "+HASIL);
        }
    }
}
```

Yang tampil adalah code dalam bahasa pemrograman JAVA.

```
run:
HASIL : 20
BUILD SUCCESSFUL (total time: 0 seconds)
```

Jika code dikompile dan dijalankan akan muncul tampilan seperti pada simulasi (area pembelajaran) media pembelajaran ini

Gambar 4.16 *Kode Program Operasi Aritmetika Pembagian*

Adapun bentuk *pseudocode* dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Menghitung Pembagian dapat dilihat pada tabel di bawah ini.

Operasi Pembagian

Pseudocode	Bahasa Pascal
pembagian; deklarasi variable; BIL1, BIL2, HASIL: numerik;	var BIL1 : integer; BIL2 : integer;

Pseudocode	Bahasa Pascal

TAMPIL: Boolean;
KONSTANTA: PESAN='BILANGAN KE-2
TIDAK BOLEH NOL'
 	
Begin
Input BIL1, BIL2, TAMPIL;
If (TAMPIL = TRUE) and (BIL2!= 0)
then
Begin HASIL = BIL1 / BIL2; print
HASIL; end
Else print PESAN; End.
	HASIL : integer;
	
	begin
	readln (BIL1); readln (BIL2);
	PESAN : String; begin
	PESAN := 'BILANGAN KE-2
	TIDAK BOLEH NOL';
	if(BIL2 = 0)
	begin
	writeln (PESAN); end
	Else begin
	HASIL := BIL1 / BIL2;
	writeln('HASIL : ' , HASIL);
	end; end;

Pseudocode	Bahasa C++

pembagian; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
KONSTANTA: PESAN='BILANGAN KE-2
TIDAK BOLEH NOL'
 	
Begin
Input BIL1, BIL2, TAMPIL;
If (TAMPIL = TRUE) and (BIL2!= 0)
then
Begin HASIL = BIL1 / BIL2; print
HASIL; end
Else print PESAN; End.
	#include <stdio.h> using namespace
	std; int main() {
	int BIL1 = 400; int BIL2 = 20; int
	HASIL = 0;
	String PESAN = "BILANGAN KE-2 TIDAK
	BOLEH NOL";
	If(BIL2 == 0) cout<<PESAN<<endl;
	Else {
	HASIL = BIL1 / BIL2; Cout<<"HASIL :
	" << HASIL<<endl;
	}
	return 0;
	}

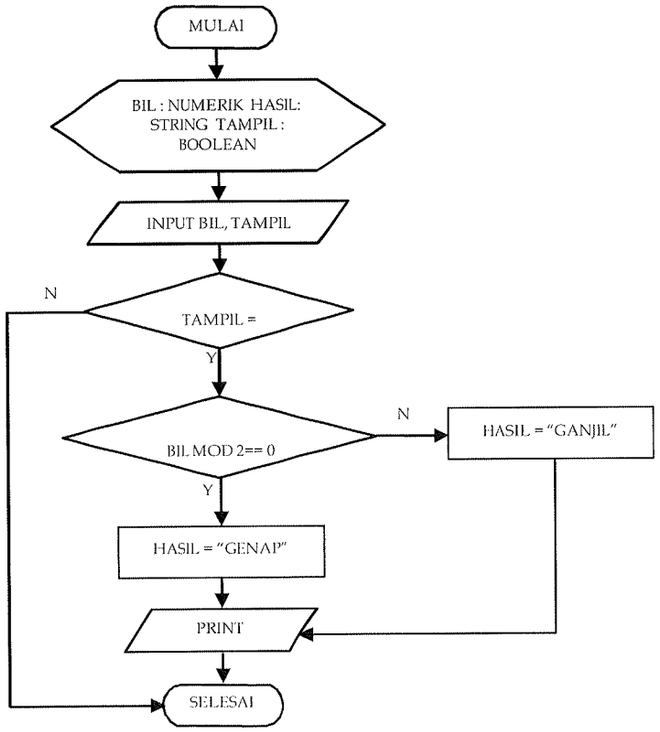
Pseudocode	Bahasa Java

pembagian; deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
KONSTANTA: PESAN='BILANGAN KE-2
TIDAK BOLEH NOL'
Begin
Input BIL1, BIL2, TAMPIL;
If (TAMPIL = TRUE) and (BIL2!= 0)
then
Begin HASIL = BIL1 / BIL2; print
HASIL; end
Else print PESAN; End.
	public class Pembagian {
	public static void main(String[]
	args) { int BIL1 = 400;
	int BIL2 = 20; int HASIL = 0;
	String PESAN = "BILANGAN KE-2
	TIDAK BOLEH NOL";
	If(BIL2 == 0)
	System.out.println(PESAN); Else {
	HASIL = BIL1 / BIL2;
	System.out.println("HASIL : " +
	HASIL);
	}
	} }

Selanjutnya kita bahas contoh tentang deteksi bilangan genap atau ganjil. Program ini memerlukan variabel BIL sebagai variabel masukan yang berisi angka yang akan dideteksi. Karena BIL berisi angka, maka dideklarasikan sebagai NUMERIK. Ada variabel keluaran HASIL yang berisi hasil keputusan deteksi bilangan. Jika BIL berisi bilangan genap maka HASIL bernilai teks "GENAP". Jika BIL berisi bilangan ganjil maka HASIL bernilai teks "GANJIL" sehingga variabel HASIL dideklarasikan bertipe STRING. Berikutnya ada variabel TAMPIL sebagai variabel masukan untuk menentukan apakah nilai hasil operasi akan ditampilkan ke layar atau tidak. Jika variabel TAMPIL bernilai TRUE maka nilai HASIL ditampilkan, sedangkan jika bernilai FALSE maka nilai Hasil tidak ditampilkan.

Proses yang terjadi di contoh ini proses deteksi apakah nilai variabel masukan BIL sebagai bilangan genap atau ganjil. Proses ini memerlukan

pernyataan matematika sekaligus logika matematika. Pernyataan matematika berupa operasi aritmatika dengan operator MOD untuk mendapatkan nilai sisa pembagian antara BIL dengan 2. Jika sisa yang dihasilkan adalah 0 maka BIL dinyatakan sebagai bilangan genap dan jika sisa bernilai 1 maka BIL dinyatakan sebagai bilangan ganjil. Proses ini memerlukan logika matematika dengan operator relasi sama dengan (==). Hasil keputusan operasi ini berupa teks "GENAP" atau "GANJIL" dimasukkan dalam variabel HASIL.



Gambar 4.17 *Flowchart Deteksi Bilangan Genap/Ganjil*

Hasil operasi yang sudah dimasukkan dalam variabel HASIL selanjutnya ditampilkan ke layar agar diketahui oleh *user*. Seluruh operasi Masukan – Proses – Keluaran ini dinyatakan dalam instruksi komputasi berturut-turut seperti Gambar 4.17, Gambar 4.18, dan

Gambar 4.19 berupa algoritma *Flowchart*, algoritma *Pseudocode*, dan Kode program dalam bahasa Java.

```
bilGanjilGenap; deklarasi variabel; BIL : NUMERIK; HASIL :
STRING TAMPIL : BOOLEAN;

Begin
Input BIL, TAMPIL
If TAMPIL = TRUE Then
Begin
IF Bil MOD 2 == 0 THEN HASIL = "GENAP" ELSE HASIL = "GANJIL"
Print HASIL End;
End.
```

Gambar 4.18 *Pseudocode Deteksi Bilangan Genap/Ganjil*

```
CODE

public class bilGanjilGenap {
public static void main(String[] args) { String HASIL;
int BIL=6;

if(BIL % 2 == 0) HASIL = "GENAP";
else
HASIL = "GANJIL";
System.out.print(BIL + " adalah " + HASIL);
}
}

Run:
6 adalah GENAP
```

Gambar 4.19 *Kode Program Deteksi Bilangan Ganjil/Genap*

4.5 Percabangan dan Variabel

Pada BAB 3 dijelaskan bahwa variabel merupakan suatu lokasi memori komputer yang digunakan untuk menampung dan menyimpan data yang akan diproses. Dalam pemrograman variabel harus mempunyai tipe data. Tipe data ini menunjukkan data jenis apa yang boleh dimasukkan dalam variabel. Ada beberapa jenis tipe data, antara lain: integer, real, karakter, string, dan boolean. Tipe data integer adalah tipe data numerik bilangan bulat. Tipe data real adalah tipe data numerik bilangan pecahan. Data alfanumerik dinyatakan dengan tipe data karakter. Kumpulan dari data karakter merupakan tipe data

string. Ada tipe data yang hanya mempunyai dua nilai, TRUE dan FALSE, yaitu tipe data boolean.

Perhatikan Gambar 4.15 tentang *Pseudocode* Operasi Pembagian. Pada *pseudocode* tersebut terdapat bagian deklarasi variabel.

```
deklarasi variable;
BIL1, BIL2, HASIL: numerik;
TAMPIL: Boolean;
KONSTANTA: PESAN='BILANGAN KE-2 TIDAK BOLEH NOL'
```

Variabel BIL1 dan BIL2 adalah variabel masukan bertipe numerik. Jika data yang dimasukkan dalam variabel tersebut adalah bilangan bulat, maka variabel BIL1 dan BIL2 bisa dinyatakan bertipe integer. Variabel TAMPIL juga sebagai variabel masukan. Variabel ini berisi keputusan apakah hasil operasi pembagian ditampilkan ke layar atau tidak. Karena ada dua nilai keputusan YA atau TIDAK, maka variabel TAMPIL dideklarasikan bertipe boolean.

Program operasi pembagian mempunyai satu variabel keluaran yang bernama HASIL. Variabel HASIL berisi nilai hasil operasi pembagian antara BIL1 dan BIL2. Operasi pembagian memungkinkan menghasilkan nilai pecahan. Lihat Tabel 4.2. Tabel ini berdasarkan Tabel 3.1 Operator Aritmatika. Operasi Pembagain dianggap selalu menghasilkan bilangan pecahan, meskipun kedua operandnya berjenis bilangan bulat (integer). Berdasarkan aturan ini maka variabel HASIL bisa dideklarasikan bertipe numerik pecahan (real).

Tabel 4.2 *Operator Pembagian*

Operator	Operasi	Tipe Operand	Tipe Hasil
/	Pembagian Real	Real, Real Integer, Integer Real, Integer	Real. Real Real

4.6 Klasifikasi Percabangan pada Pengolahan Data

Pada pokok bahasan ini kita gunakan contoh operasi Kombinasi Dua Buah Relasi. Operasi ini digunakan untuk mencari nilai terkecil dari tiga buah nilai yang dimasukkan. Pada Gambar 4.20, contoh ada tiga bilangan yang dimasukkan, yaitu **9**, **4**, dan **7**. Tekan tombol Tampilkan Nilai

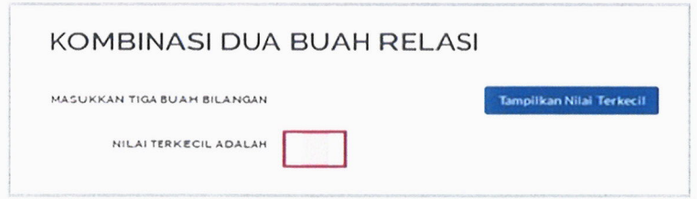
Terkecil. Maka di bagian keluaran muncul nilai **4** yang dinyatakan sebagai nilai terkecil.

Gambar 4.20 *Operasi Kombinasi Dua Buah Relasi*

1. Klik *toolbar* menu MASUKAN. Lihat Gambar 4.21. Kombinasi Dua Buah Relasi mempunyai 4 (empat) masukan yang ditandai kotak merah. Tiga kotak masukan digunakan untuk menampung bilangan yang diseleksi dan satu masukan berupa tombol **Tampilkan Nilai Terkecil** untuk memutuskan nilai terkecil hasil seleksi ditampilkan atau tidak.

Gambar 4.21 *Masukan Kombinasi Dua Buah Relasi*

2. Klik *toolbar* menu KELUARAN. Pada Gambar 4.22 bagian keluaran ditunjukkan dengan tanda kotak merah. Nilai terkecil dari tiga nilai masukan akan tampil di bagian kotak merah (keluaran).



Gambar 4. 22 *Keluaran Kombinasi Dua Variabel*

3. Klik *toolbar* menu VARIABEL

Gambar 4.23 *Variabel Kombinasi Dua Buah Relasi*

Gambar 4.23 menjelaskan ada satu variabel array bernama **bil** yang terlihat di layar. Jika diilustrasikan menggunakan data Gambar 4.20, variabel array ini sebagai berikut:

9	4	7
bil [1]	bil [2]	bil [3]

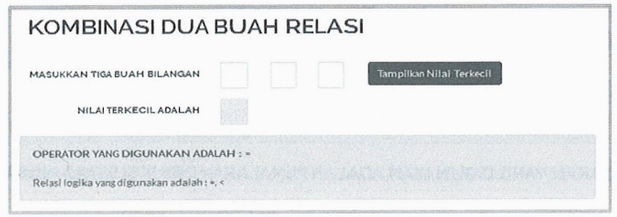
Selain variabel array **bil**, ada variabel lain bernama **nTerkecil**. Variabel **nTerkecil** berisi angka terkecil dari ketiga bilangan 9, 4, dan 7, yaitu 4. Ada variabel lain yang terlihat di layar, yaitu **tampilNilTerkecil**. Variabel ini bertipe BOOLEAN karena mempunyai nilai TRUE atau FALSE.

4. Klik *toolbar* menu DATA. Data ini sebagai data masukan. Sesuai dengan Gambar 4.24 ada tiga variabel masukan bil[1], bil[2], dan bil[3]. Variabel bil[1] berisi data masukan 12, variabel bi[2] berisi data masukan 4, dan bil[3] berisi data masukan 45. Data masukan ini selanjutnya dikenai proses operasi relasi. Proses ini menghasilkan data keluaran 4, artinya data terkecil dari tiga data masukan adalah 4. Data keluaran ini selanjutnya

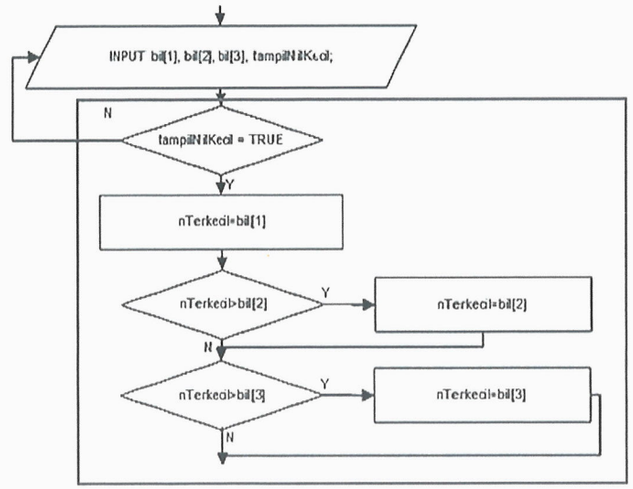
Gambar 4.24 *Data Kombinasi Dua Buah Relasi*

Ada satu data masukan yang tidak terlihat di layar. Data masukan ini bertempat di variabel masukan bernama **tampilNilKecil**. Variabel tampilNilKecil bernilai TRUE, yang artinya keputusan bahwa data terkecil sebagai data keluaran ditampilkan ke layar.

5. Klik *toolbar* menu OPERATOR. Proses yang terjadi pada kombinasi dua buah relasi adalah operasi relasi. Operasi ini memerlukan operator perbandingan/relasi sama dengan (=) dan lebih besar (>). Sesuai dengan nama, kombinasi dua buah relasi memerlukan dua kali operasi relasi seperti Gambar 4.26.



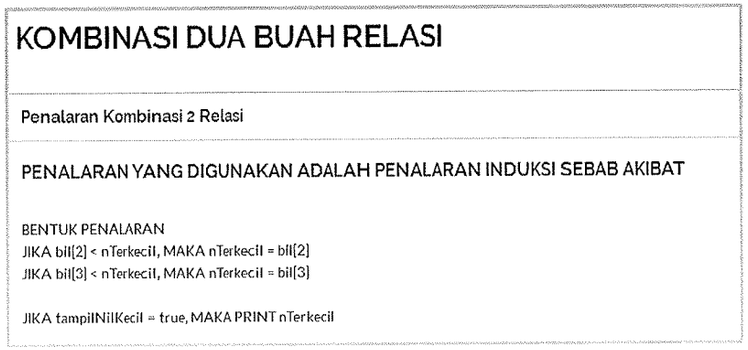
Gambar 4.25 *Operator Kombinasi Dua Buah Relasi*



Gambar 4.26 *Operasi Relasi pada Kombinasi Dua Buah Relasi*

Operasi relasi yang pertama **nTerkecil > bil[2]?**. Hasil operasi relasi pertama dikenai operasi relasi kedua **nTerkecil > bil[3]?**.

6. Klik *toolbar* menu PENALARAN. Kombinasi dua buah relasi menggunakan penalaran induksi sebab akibat. Variabel **nTerkecil** selalu diisi dengan nilai yang terkecil. nTerkecil diberi nilai awal sama dengan nilai bil[1]. Selanjutnya nTerkecil dibandingkan dengan bil[2]. Jika bil[2] lebih kecil dari nTerkecil, maka nilai nTerkecil diganti dengan nilai bil[2]. Proses berikutnya nTerkecil dibandingkan dengan bil[3]. Jika nilai bil[3] lebih kecil dari nTerkecil, maka nilai nTerkecil diganti dengan bil[3]. Hasil akhir operasi relasi akan mendapatkan nilai terkecil dalam variabel nTerkecil (lihat Gambar 4.24).

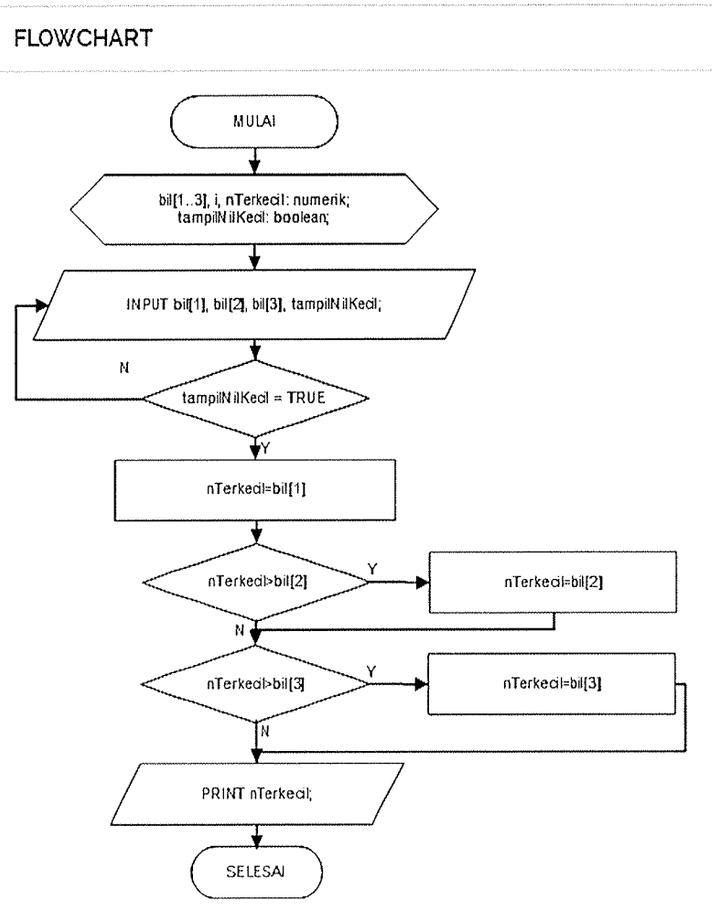


Gambar 4.27 *Penalaran Kombinasi Dua Buah Relasi*

7. Klik *toolbar* menu *FLOWCHART*. Gambar 4.28 adalah *flowchart* Kombinasi Dua Buah Relasi. *Flowchart* ini mempunyai bagian masukan, bagian proses, dan bagian keluaran. Bagian masukan adalah memasukkan/*input* nilai dalam variabel **bil[1]**, **bil[2]**, **bil[3]** sebagai bilangan yang diseleksi dan variabel **tampilNilKecil** sebagai keputusan bilangan terkecil akan ditampilkan ke layar atau tidak. Bagian masukan ditandai dengan simbol jajaran genjang.

   Bagian proses berupa operasi relasi dengan simbol belah ketupat pada variabel tampilNilKecil. Jika tampilNilKecil bernilai TRUE maka seleksi bilangan dilakukan dan ditampilkan.Proses seleksi dilakukan dengan dua buah operasi relasi. Hasil seleksi operasi relasi pertama dikenai proses seleksi pada operasi relasi kedua.

Hasil seleksi berupa nilai terkecil menjadi data keluaran yang diletakkan dalam variabel nTerkecil. Data dalam nTerkecil inilah yang ditampilkan ke layar sebagai keluaran dengan simbol jajaran genjang.



Gambar 4.28 *Flowchart Kombinasi Dua Buah Relasi*

8. Klik *toolbar* menu *PSEUDOCODE*. Bentuk lain dari algoritma adalah *pseudocode*. Seperti *flowchart*, *pseudocode* juga mempunyai bagian Masukan, Proses, dan Keluaran. Pada Gambar 4.29, bagian masukan ditandai dengan kotak merah, bagian proses ditandai dengan kotak hijau, dan bagian keluaran ditandai dengan kotak biru.

```
PSEUDOCODE

logikaDasar:
deklarasi variable:
    bil[1..3], nTerkecil: numerik;
    tampilNilKecil boolean;
Begin
    INPUT bil[1], bil[2], bil[3];
    REPEAT
        INPUT tampilNilKecil;
    UNTIL tampilNilKecil = true;

    nTerkecil = bil[1];
    IF nTerkecil >bil[1] THEN nTerkecil=bil[2];
    IF nTerkecil >bil[3] THEN nTerkecil=bil[3];

    PRINT nTerkecil;
END.
```

MASUKAN

PROSES

KELUARAN

Gambar 4.29 *Pseudocode Kombinasi Dua Buah Relasi*

9. Klik *toolbar* menu CODE. Pada Gambar 4.30 ditunjukkan bagian keluaran, proses, dan masukan dalam kode program menggunakan bahasa Java.

```
public class KombinasiRelasi {
    public static void main(String[] args) {
//        bil[1..3], nTerkecil: numerik;
        int[] bil = new int[3];
        int nTerkecil;

        Scanner sc = new Scanner(System.in);
        for(int i=0;i<bil.length;i++){
            System.out.print("Masukkan bilangan "+(i+1)+" : ");
            bil[i] = sc.nextInt();
        }

        nTerkecil = bil[0];
        if(nTerkecil > bil[1]) nTerkecil = bil[1];
        if(nTerkecil > bil[2]) nTerkecil = bil[2];

        System.out.println("Bilangan Terkecil adalah "+nTerkecil);
    }
}
```

MASUKAN

PROSES

KELUARAN

```
Yang tampil adalah code dalam bahasa pemrograman JAVA.
run:
Masukkan bilangan 1 : 12
Masukkan bilangan 2 : 4
Masukkan bilangan 3 : 45
Bilangan Terkecil adalah 4
BUILD SUCCESSFUL (total time: 4 seconds)
```

Jika code dikompile dan dijalankan akan muncul tampilan seperti pada simulasi (area pembelajaran) media pembelajaran ini

Gambar 4.30 *Kode Program dan Keluaran Kombinasi Dua Buah Relasi*

Adapun bentuk *pseudocode* dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Kombinasi Dua Buah Relasi dapat dilihat pada tabel di bawah ini.

Kombinasi Dua Relasi

Pseudocode	Bahasa Pascal

logikaDasar; deklarasi variable;
bil[1..3], nTerkecil: numerik;
tampilNilKecil boolean;
Begin
INPUT bil[1], bil[2], bil[3];
REPEAT
INPUT tampilNilKecil; UNTIL
tampilNilKecil = true;
 	
nTerkecil = bil[1];
	var
	bil : array [1..3] of integer;
	nTerkecil : integer;
	i : integer; begin
	i = 0;
	for i :=0 to bil.length begin
	write ('Masukkan bilangan', (i+1)
	, ' : ');
	readln(bil[i]);

Pseudocode	Bahasa Pascal

IF nTerkecil>bil[1] THEN
nTerkecil=bil[2];
IF nTerkecil>bil[3] THEN
nTerkecil=bil[3];
 	
PRINT nTerkecil; END.
	end;
	nTerkenal := bil[0]; if(nTerkecil
	> bil[1]) begin
	nTerkecil := bil[1]; end;
	if(nTerkecil > bil[2]) begin
	nTerkecil := bil[2]; end;
	writeln('Bilangan Terkecil
	adalah ', nTerkecil);
	end;

Pseudocode	Bahasa C++

logikaDasar; deklarasi variable;
bil[1..3], nTerkecil: numerik;
tampilNilKecil boolean;
Begin
INPUT bil[1], bil[2], bil[3];
REPEAT
INPUT tampilNilKecil; UNTIL
tampilNilKecil = true;
 	
nTerkecil = bil[1];
IF nTerkecil>bil[1] THEN
nTerkecil=bil[2];
IF nTerkecil>bil[3] THEN
nTerkecil=bil[3];
 	
PRINT nTerkecil; END.
	#include <stdio.h> using namespace
	std; int main() {
	int bil[3];
	int nTerkecil;
	For (int i=0;i<bil.length;i++) {
	cout <<"Masukkan bilangan " <<
	(i+1)
	<< " : " ;
	cin>>bil[i];
	}
	nTerkenal = bil[0];
	if(nTerkecil > bil[1]) nTerkecil =
	bil[1]; if(nTerkecil > bil[2])
	nTerkecil = bil[2]; cout <<
	"Bilangan Terkecil adalah " +
	nTerkecil << endl;
	return 0;
	}

Pseudocode	Bahasa Java

logikaDasar; deklarasi variable;
bil[1..3], nTerkecil: numerik;
tampilNilKecil boolean;
Begin
INPUT bil[1], bil[2], bil[3]; REPEAT
INPUT tampilNilKecil; UNTIL
tampilNilKecil = true;
 	
nTerkecil = bil[1];
	public class KombinasiRelasi {
	public static void main(String[]
	args) { int[] bil = new int[3];
	int nTerkecil;
	Scanner sc = new
	Scanner(System.in);
	For (int i=0;i<bil.length;i++) {
	System.out.print("Masukkan bilangan
	" + (i+1) + " : ");
	bil[i] = sc.nextInt();
	}

Pseudocode	Bahasa Java

IF nTerkecil>bil[1] THEN
nTerkecil=bil[2];
IF nTerkecil>bil[3] THEN
nTerkecil=bil[3];
 	
PRINT nTerkecil; END.
	nTerkenal = bil[0];
	if(nTerkecil > bil[1]) nTerkecil =
	bil[1]; if(nTerkecil > bil[2])
	nTerkecil = bil[2];
	System.out.println("Bilangan
	Terkecil adalah " + nTerkecil);
	}
	}

4.7 Rangkuman

Percabangan adalah sebuah perintah/instruksi yang dapat dijalankan dengan syarat/kondisi tertentu harus dipenuhi terlebih dahulu. Apabila syarat/kondisi belum terpenuhi, maka perintah/instruksi tersebut tidak akan pernah dijalankan. Syarat atau kondisi yang digunakan boleh lebih dari satu kondisi (tidak terbatas).

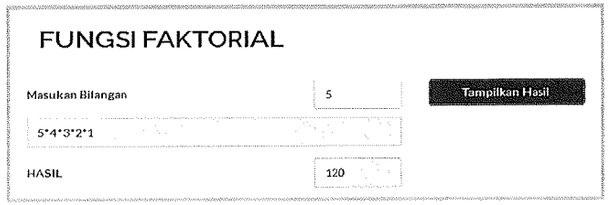
Percabangan ditandai dengan kata kunci IF-THEN untuk percabangan bentuk tunggal dan kata kunci IF-THEN-ELSE untuk percabangan bentuk ganda. Percabangan bentuk tunggal mempunyai prinsip kerja jika syarat/kondisi bernilai benar (*Yes*/*Y*) maka Tahap Proses dikerjakan. Jika syarat/kondisi bernilai salah (*No*/*N*) maka Tahap Proses tidak dikerjakan. Sedangkan percabangan bentuk ganda mempunyai prinsip kerja jika syarat/kondisi bernilai benar, maka Tahap Proses 1 dikerjakan dan jika syarat/kondisi bernilai salah, maka Tahap Proses 2 dikerjakan.

Syarat/Kondisi adalah logika matematika yang menggunakan operator relasi dengan hasil operasi bernilai *TRUE* atau *FALSE*. Logika matematika yang menyatakan syarat/kondisi boleh juga menggunakan operator aritmatika maupun operator logika apabila diperlukan.

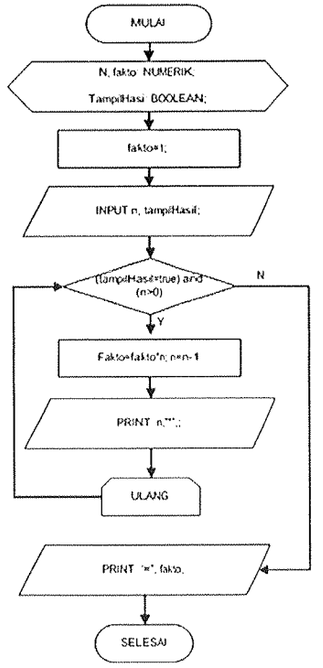
Tahap Proses adalah segala instruksi komputasi yang harus dikerjakan dan berkaitan dengan benar (*Y*) atau salah (*N*) hasil operasi relasi pada syarat/kondisi.

4.8 Latihan Soal

1. Tentukan bagian masukan, keluaran, variabel, dan data pada aplikasi berikut ini:



2. Di bawah ini adalah *flowchart* untuk menghitung nilai faktorial. Tentukan operator dan pernyataan matematika dalam *flowchart* tersebut



3. Lakukan penalaran pada percabangan yang terdapat pada gambar *flowchart* soal no. 2 di atas.
4. Buatlah *pseudocode* dan kode program (*Source Code*) untuk gambar *flowchart* soal no. 2.
5. Buat program untuk menentukan status kelulusan Mahasiswa dengan ketentuan sebagai berikut:

   - Jika total nilai >=75, maka dinyatakan "Lulus"
   - Jika total nilai <75, maka dinyatakan "Tidak Lulus"

   Soal Cerita Dieng:

   "Dieng merupakan daerah dataran tinggi Indonesia. Dieng pernah mengalami suhu dingin ekstrim. Pada tahun 2012, suhu ekstrim Dieng mencapai -6℃. Pada tahun 2016, suhu dingin ekstrim Dieng mencapai -2℃. Buatlah program untuk menampilkan suhu dingin terekstrim Dieng". (Suparmin, 2017: 9).

   Berdasarkan soal cerita Dieng di atas, selesaikan pertanyaan soal no. 6 sampai dengan soal no. 9.
6. Tentukan masukan, keluaran, variabel, dan tipe data untuk soal cerita Dieng di atas.
7. Tentukan operator dan pernyataan matematika untuk proses penyelesaian soal cerita Dieng.
8. Tentukan logika matematika dan lakukan penalaran induksi sebab akibat pada soal cerita Dieng.
9. Buatlah *flowchart* dan kode program (*Source Code*) untuk menyelesaikan soal cerita Dieng.
   Kerjakan soal no.10 sampai dengan soal no.13 untuk membuat program menentukan status kelulusan mahasiswa dengan ketentuan sebagai berikut:

   - Jika total nilai >=75, maka dinyatakan "Lulus"
   - Jika total nilai <75, maka dinyatakan "Tidak Lulus"

10. Tentukan masukan, keluaran, variabel, dan tipe data untuk membuat program kelulusan.
11. Tentukan logika matematika dan lakukan penalaran induksi sebab akibat untuk program kelulusan.
12. Buatlah *flowchart* dan *pseudocode* untuk membuat program kelulusan.
13. Ubahlah jawaban soal no.12 menjadi kode program (*Source Code*).
    Kerjakan soal no.14 sampai dengan soal no. 20 untuk membuat sebuah program Sistem Informasi Hotel dengan ketentuan seperti uraian berikut:

    Jenis Kamar:

    - *Deluxe Room* harga Rp 275.000/hari
    - *Suite Room* harga Rp 300.000/hari
    - *Precident Room* harga Rp 375.000/hari

    Diskon:

    - Jika Pelanggan Biasa
      - Jika menginap 4 - 7 hari, maka mendapatkan diskon 10%
      - Jika menginap > 7 hari, maka akan mendapatkan diskon 15%
    - Jika *Member*
      - Jika menginap 4 - 7 hari, maka mendapatkan diskon 20%
      - Jika menginap > 7 hari, maka mendapatkan diskon 35%
14. Buatlah tampilan *input*an data dari pelanggan yang berisi informasi: nama pelanggan, alamat pelanggan, pilihan jenis kamar, lama menginap, dan status *member* dari pelanggan.
15. Buatlah tampilan keluaran berupa Nota Pembayaran Hotel untuk Sistem Informasi Hotel di atas.
16. Tentukan variabel dan tipe data untuk menyelesaikan Sistem Informasi Hotel.
17. Tentukan operator dan pernyataan matematika untuk proses penyelesaian Sistem Informasi Hotel.

18. Tentukan logika matematika dan lakukan penalaran induksi sebab akibat untuk menyelesaikan Sistem Informasi Hotel di atas.
19. Buatlah *flowchart* dan *pseudocode* untuk Sistem Informasi Hotel di atas.
20. Ubahlah algoritma jawaban soal no. 19 menjadi kode program (*Source Code*).

-oo0o-

PERULANGAN 1

Deskripsi Singkat

Pada bab 5 ini akan dibahas mengenai Alur Algoritma Perulangan. Diawali dengan Pendahuluan yang membahas tentang contoh sederhana penerapan proses perulangan dalam kehiduan sehari-hari. Proses Perulangan pada dasarnya bertugas untuk melakukan kegiatan mengulang sesuai yang diinginkan dengan batasan tertentu. Pembahasan Alur Algoritma Perulangan yang akan disampaikan pada buku ini mencakup penerapan perulangan pada Pernyataan Matematika, penerapan perulangan pada Logika Matematika, penerapan perulangan untuk Instruksi Komputasi, penerapan perulangan pada penggunaan variabel.

Penerapan Perulangan pada Pernyataan Matematika mencakup deskripsi tentang penggunaan kounter, akumulator dan multiplikator. Pembaca akan dibawa berpikir kritis mengenai operasi aritmatika yang diolah pada proses perulangan. Sedangkan sub bab penerapan perulangan pada logika matematika akan membahas tentang hasil boolean *true/false* dari kondisi yang ditentukan pada perulangan. Pembaca akan diberikan gambaran dalam bentuk tabel *tracing* untuk menganalisa hasil kondisi atau batasan perulangan.

Hampir sama seperti sub bab sebelumnya, bahwa pada sub bab penerapan perulangan untuk instruksi komputasi akan membahas

perintah-perintah yang berjalan pada sistem selama proses perulangan dilakukan. Hal ini juga berlaku pada perulangan dan penggunaan variabel. Terdapat 2 jenis variabel yang digunakan pada perulangan ini yaitu variabel yang terlihat secara eksplisit, maupun variabel yang tidak terlihat alias implisit.

Pada sub bab terakhir terdapat pembahasan mengenai perbedaan Alur Algoritma Perulangan dibandingkan Alur Algoritma Percabangan. Sebelum akhir bab, terdapat Rangkuman dan Latihan Soal. Ditutup dengan daftar pustaka yang digunakan untuk menyusun bab 5.

Tujuan Pembelajaran

Tujuan akhir pembelajaran pada Bab 5 ini dibagi menjadi 3 bagian, antara lain:

1. Jika diberi soal atau studi kasus, mahasiswa dapat **menganalisis** hasil penerapan perulangan pada Pernyataan Matematika, penerapan perulangan pada Logika Matematika, penerapan perulangan untuk Instruksi Komputasi, serta penerapan perulangan pada penggunaan variabel.
2. Jika diberi soal atau studi kasus, mahasiswa dapat **membuktikan** hasil penerapan perulangan pada Pernyataan Matematika, penerapan perulangan pada Logika Matematika, penerapan perulangan untuk Instruksi Komputasi, serta penerapan perulangan pada penggunaan variabel.
3. Tujuan terakhir adalah mahasiswa dapat **menyimpulkan** hasil akhir penerapan perulangan pada Pernyataan Matematika, penerapan perulangan pada Logika Matematika, penerapan perulangan untuk Instruksi Komputasi, serta penerapan perulangan pada penggunaan variabel.

5.1 Pendahuluan

Alur Algoritma Perulangan bertugas untuk melakukan kegiatan mengulang dengan batasan tertentu. Banyak aplikasi komputer yang melakukan operasi atau pekerjaan berulang-ulang sampai sebuah

kondisi yang diinginkan terpenuji. Dengan memanfaatkan alur algoritma perulangan, seorang programmer tidak perlu lagi menuliskan baris kode program yang sama berulang kali secara sekuensial (satu per satu). Oleh karena itu, alur algoritma perulangan menjadi hal yang penting dalam aktivitas pemrograman.

Beberapa contoh proses perulangan dalam kehidupan sehari-hari diantaranya adalah sebagai berikut:

1. Melakukan kegiatan pencatatan pendaftaran mahasiswa baru yang biasa dilakukan oleh bagian administrasi/penerimaan mahasiswa baru. Pada umumnya terdapat seorang petugas yang bertugas untuk melakukan entry data calon mahasiswa baru berdasarkan data yang tertulis pada formulir pendaftaran. Data yang di*input*kan pada komputer terdiri dari data pribadi, data keluarga, data riwayat sekolah, dsb. Apabila dalam sekali proses penerimaan, kampus tersebut menerima 500 orang calon mahasiswa baru, maka petugas tersebut akan melakukan pencataan pendaftaran sebanyak 500 kali proses entry dengan komposisi data yang sama ke dalam sistem komputer. Proses seperti ini dapat diterapkan dengan menggunakan alur algoritma perulangan.
2. Seorang murid sekolah dasar yang biasanya mendapatkan hukuman keterlambatan masuk kelas oleh Guru, dihukum dengan menuliskan kalimat yang sama "Saya akan selalu datang tepat waktu untuk hari esok" pada bukunya sebanyak 100 kali. Menulis kalimat yang sama inilah, proses penerapan alur perulangan diterapkan dalam kehidupan nyata sehari-hari.
3. Seorang pelanggan di supermaket membeli kebutuhan pokok bulanan dengan 10 jenis barang yang berbeda. Kasir akan meng*input*kan data pembelian barang tersebut sesuai data barang yang dibeli. Proses menginputkan komposisi data barang yang sama sebanyak 10 kali juga merupakan bagian dari proses perulangan.

Proses perulangan mempunyai 4 tahapan yang harus dilakukan, antara lain sebagai berikut:

- Inisialisasi
  Tahap ini merupakan tahap persiapan membuat kondisi awal atau batas awal perulangan. Batas awal dapat dibentuk melalui pengisian sebuah variabel dengan nilai awal.

  Contoh: membuat inisialisasi variabel **batas = 1.**
- Proses
  Tahap ini merupakan tahap utama dalam sebuah perulangan, dimana berisi proses/kegiatan yang akan dilakukan secara berulang-ulang.

  Contoh: proses yang akan diulang adalah menerima *input*an pendaftaran 10 nama mahasiswa, sehingga dapat dituliskan **input(nama_mhs)** sekali saja. Biarkan alur proses perulangan yang bekerja sebanyak 10 kali inputan.
- Kounter
  Tahap ini merupakan tahap yang paling penting, dikarenakan kounter merupakan pemicu terjadinya proses perulangan. Kounter dapat berfungsi sebagai penambah satu nilai atau pengurang satu nilai. Kounter akan menjaga dan menjamin bahwa proses perulangan tetap berjalan sebagaimana mestinya. Detail pem-bahasan tentang Kounter akan dibahas pada Bab 6.

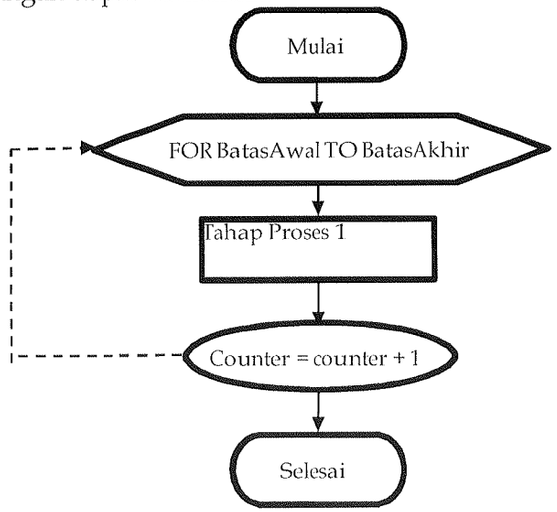
  Contoh: Kounter penambah satu untuk variabel **batas = batas + 1.**
- Terminasi/Kondisi Perulangan
  Tahap ini merupakan tahap akhir dari sebuah proses perulangan. Hal ini dapat dilakukan dengan pengisian sebuah variabel dengan nilai akhir. Dengan catatan bahwa variabel yang digunakan pada tahap terminasi harus sama dengan variabel yang digunakan pada tahap inisialisasi. Hanya mengganti nilai awal (batas awal) dengan nilai akhir (batas akhir).

  Contoh: pada tahap inisialisasi dibuat variabel batas = 1, kemudian tahap terminasi mengubah isi **variabel batas = 10.**

5.2 Perulangan dan Pernyataan Matematika

Proses perulangan adalah sebuah proses algoritma/program dimana setiap perintah/instruksi dapat dijalankan secara berulang dengan kondisi tertentu. Kondisi tersebut digunakan sebagai batas awal, batas akhir perulangan, dan kounter. Kounter dapat berfungsi sebagai penambah satu nilai atau pengurang satu nilai pada proses perulangan. Salah satu contoh kounter adalah membuat deret angka, misal 1, 2, 3, dst. Pada operasi aritmatika, kounter digunakan untuk membuat indeks.

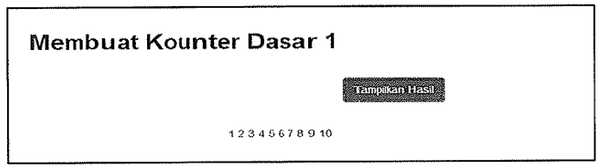
Untuk *Flowchart* dasar Perulangan dapat dilihat pada Gambar 5.1. Pada gambar tersebut terdapat batasan awal dan batasan akhir yang harus diketahui terlebih dahulu (simbol warna biru, setelah simbol Mulai). Batasan awal dan batas akhir juga dapat diinputkan dari pengguna. Kemudian tentukan proses yang akan diulang (simbol warna hijau). Perintah terakhir adalah melengkapi proses perulangan dengan menambahkan kounter (simbol warna biru, sebelum simbol Selesai). Oleh karena itu, pembaca diharapkan dapat menganalisis kelengkapan 3 faktor perulangan tersebut. Jika 3 faktor tersebut tersedia, maka perulangan dapat dilakukan.



Gambar 5.1 *Flowchart Dasar Perulangan For*

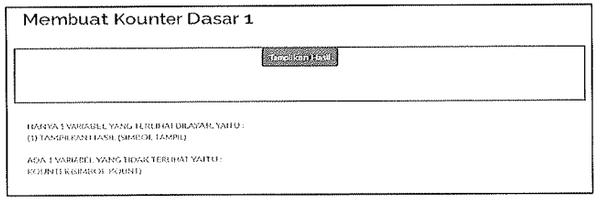
Cobalah untuk mengakses Aplikasi Penalaran.com. Pada Aplikasi Penalaran.com tersebut terdapat **Menu Pernyataan Matematika - Operasi Aritmatika - Membuat Kounter (Kounter Dasar 1). Pada menu tersebut** ditampilkan proses membuat bilangan secara urut mulai dari 1 sampai dengan 10. Bilangan tersebut akan muncul setelah ditekan tombol Tampilkan Hasil. Bentuk deret yang dihasilkan dapat dilihat pada Gambar 5.2.

Pada gambar 5.2 ditampilkan hasil 1, 2, 3, 4, 5, 6, 7, 8, 9 ,10. Proses ini merupakan penerapan kounter dengan penambah satu angka. Dengan nilai awal = 1, jika ditambah 1, maka hasilnya 2. Angka 2 akan ditambah lagi dengan 1, maka hasilnya 3. Proses ini berlangsung sama hingga batas terakhir yaitu 10.



Gambar 5.2 *Hasil Membuat Kounter Dasar di Penalaran.com*

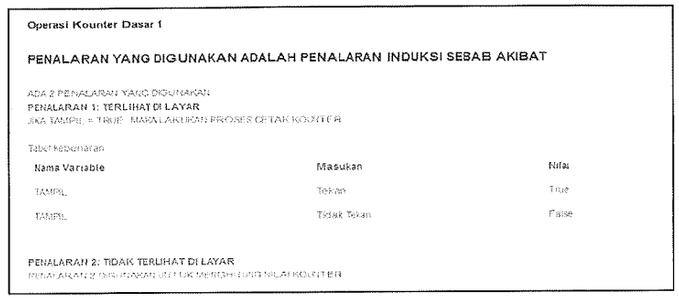
Sebenarnya terdapat 2 buah variabel yang digunakan pada proses membuat Kounter Dasar 1 tersebut, yaitu: **Variabel Tampil** yang terlihat berupa tombol 'Tampilkan Hasil' dan **Variabel Kount** yang tidak terlihat, namun berdampak pada hasil deret yang ditampilkan di layar aplikasi.



Gambar 5.3 *Penggunaan Variabel pada Menu Kounter Dasar 1*

Sedangkan Tabel Kebenarannya dapat dilihat pada Gambar 5.4. Tabel Kebenaran dapat ditampilkan dengan menekan menu yang

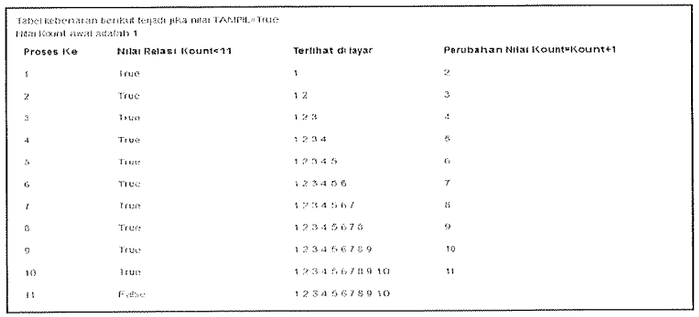
disediakan pada sisi kiri. Pada tabel kebenaran tersebut berisi kondisi dari Variabel Tampil. Apabila variabel Tampil mendapat masukan 'Tekan', maka bernilai True. Sebaliknya, apabila variabel Tampil mendapat masukan 'Tidak ditekan', maka bernilai False.



Gambar 5.4 *Model Penalaran Pembuatan Kounter Dasar 1.*

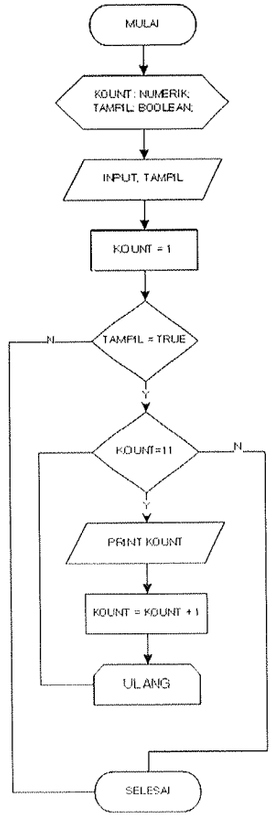
Pernyataan matematika yang digunakan pada proses perulangan tersebut terdapat pada variabel Kount dengan Rumus Kount = Kount + 1.

Nilai Kount akan ditambah dengan angka 1 hingga kondisi Kount < 11. Jika kount bernilai 11. maka perulangan berhenti, karena syaratmya harus lebih kecil dari 11, bukan lebih kecil sama dengan 11. **Sedangkan operator yang digunakan adalah: =, +** yang digunakan untuk menghitung kount, dengan simbol → kount = kount + 1. Jika dilakukan proses perulangan, maka setiap tahapan akan menghasilkan deret angka seperti pada Gambar 5.5.



Gambar 5.5 *Tabel Kebenaran Pembuatan Kounter Dasar 1*

Apabila dibuat *flowchart* dari proses pembuatan Kounter Dasar 1, dapat dilihat pada Gambar 5.6. *Flowchart* dimulai dengan mendeklarasikan 2 buah variabel, yaitu Variabel Kount dengan tipe data Numerik, dan Variabel Tampil dengan tipe data Boolean. Proses selanjutnya adalah menekan tombol Tampil. Berikan nilai variabel Kount sama dengan 1. Apabila tampil menerima inputan/ditekan, maka bernilai True, maka proses perulangan dengan batas kount = 11 dimulai. Selama Kount lebih kecil dari 11, maka cetak nilai Kount dilayar, kemudian Kount akan ditambahkan dengan angka 1, agar proses *looping* dapat terus berjalan. Jika variabel Kount sudah tidak memenuhi kondisi akhir lebih kecil 11, maka proses selesai.



Gambar 5.6 *Flowchart Pembuatan Kounter Dasar 1*

Gambar 5.6. tersebut jika diidentifikasi ke dalam komponen pengolahan data menjadi seperti berikut ini:

a. Komponen masukan
   Pada komponen masukan ada 1 instruksi yang dituliskan menjadi satu, yaitu: *Input* TAMPIL.

   Maknanya, pengguna diminta memasukkan atau memberi nilai pada 1 buah variabel, yaitu: TAMPIL melalui peralatan masukan.
b. Komponen proses
   Pada komponen proses ada satu pernyataan matematika di setiap *pseudocode* berikut ini.

   1) Deklarasi Awal: Kount = 1.
   2) Penjumlahan: KOUNT = KOUNT + 1

   Terdapat pula logika matematika yang digunakan yaitu pada proses percabangan:

   - Tampil = *true*
   - Kount = 11
c. Komponen keluaran.
   Pada komponen keluaran ada satu instruksi, yaitu: Print KOUNT.

   Maknanya adalah, mesin komputer diminta menampilkan nilai yang disimpan atau ditampung pada variabel KOUNT.

Setelah *Flowchart* selesai dibuat, langkah selanjutnya adalah membuat *pseudocode*. Adapun bentuk *pseudocode* dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk pembuatan Kounter Dasar 1 dapat dilihat pada tabel di bawah ini.

Pseudocode	Bahasa Pascal

kounterSederhana; deklarasi variable;
KOUNT: numerik; TAMPIL: Boolean;
 	
Begin
Input TAMPIL; KOUNT = 1;
if TAMPIL = TRUE then While KOUNT<11
do
print KOUNT;
KOUNT = KOUNT + 1;
End;
	var
	kount : integer;
	
	begin kount:=1;
	while (kount < 11) begin
	writeln(kount, " "); kount :=
	kount + 1;
	end; writeln (' '); end;

Pseudocode	Bahasa C++

kounterSederhana; deklarasi variable;
KOUNT: numerik; TAMPIL: Boolean;
 	
Begin
Input TAMPIL; KOUNT = 1;
if TAMPIL = TRUE then While KOUNT<11 do
print KOUNT;
KOUNT = KOUNT + 1;
End;
End.
	#include <stdio.h> using namespace std; int main() {
	int kount = 1;
	while (kount < 11) {
	cout << kount<<" "<<endl; kount = kount + 1;
	}
	cout << "" << endl; return 0;
	}

Pseudocode	Bahasa Java

kounterSederhana; deklarasi variable;
KOUNT: numerik; TAMPIL: Boolean;
 	
Begin
Input TAMPIL; KOUNT = 1;
if TAMPIL = TRUE then While KOUNT<11 do
print KOUNT;
KOUNT = KOUNT + 1;
End;
End.
	public class Kounter_1 {
	public static void main(String[] args) { int kount = 1;
	while (kount < 11) {
	System.out.println(kount+" "); Kount = kount + 1;
	}
	System.out.println("");
	}
	}

5.3 Perulangan dan Logika Matematika

Logika matematika adalah cabang logika dan matematika yang mengandung kajian matematis logika dan aplikasi kajian ini pada bidang-bidang lain di luar matematika. Logika matematika berhubungan erat dengan ilmu komputer dan logika filosofis. Pada umumnya, logika matematika hanya akan menghasilkan 2 nilai boolean, yaitu nilai *true* dan nilai *false*. Hal ini bergantung pada kondisi yang diberikan pada algoritma. Contoh Logika Matematika dapat dilihat pada Gambar 5.7.

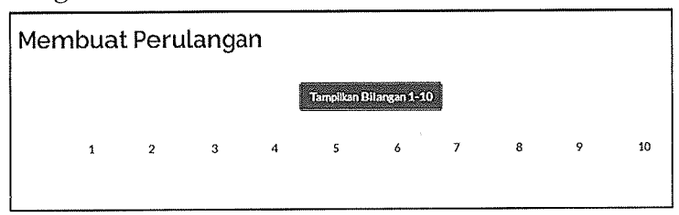
Contoh logika matematika:

$$p(x) : 3x + 1 > 6, x \in \mathbb{R}$$

Saat $x = 1$, maka $p(1) : 3(1) + 1 > 6$ bernilai salah
Saat $x = 2$, maka $p(2) : 3(2) + 1 > 6$ bernilai benar

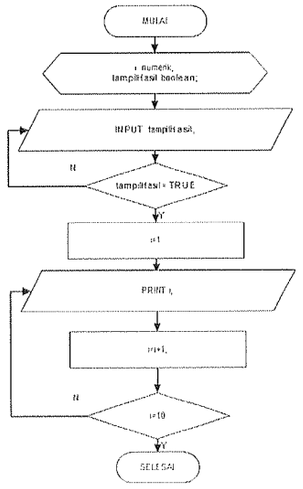
Gambar 5.7 *Contoh Logika Matematika*

Pada Aplikasi Penalaran.com, Menu Logika Matematika dibagi menjadi beberapa menu seperti Logika Dasar, Kombinasi Dua Relasi, Konsep Perulangan, dan Konsep Perulangan Array. Untuk Menu Konsep Perulangan, menampilkan deret angka yang sama seperti yang telah dibahas pada Gambar 5.2 di sub bab sebelumnya yaitu menampilkan bilangan 1 hinga 10.



Gambar 5.8 *Menu Konsep Perulangan.*

Berdasarkan Gambar 5.8, ditampilkan deret angka dengan kounter penambah 1. Selain itu, terdapat 1 variabel sederhana yang terlihat di layar yaitu *i* dan 1 variabel boolean yaitu tampilBil. Variabel tampilBil ini juga bernilai *true*, namun tidak terlihat pada layar komputer dikarenakan diproses otomatis oleh sistem. Sedangkan bentuk penalaran yang digunakan adalah jika i < 11, maka print i, i=i+1.



Gambar 5.9 *Flowchart Konsep Perulangan pada Menu Logika Matematika*

Gambar 5.9. tersebut jika diidentifikasi ke dalam komponen pengolahan data menjadi seperti berikut ini:

a. Komponen masukan.
   Pada komponen masukan ada 1 instruksi yang dituliskan menjadi satu, yaitu: *Input* tampilHasil.

   Maknanya, pengguna diminta memasukkan atau memberi nilai pada 1 buah variable, yaitu tampilHasil melalui peralatan masukan.
b. Komponen proses.
   Pada komponen proses ada satu pernyataan matematika di setiap *pseudocode* berikut ini.

   1) Deklarasi Awal: $i = 1$.
   2) Penjumlahan: $i = i + 1$

   Terdapat pula logika matematika yang digunakan yaitu pada proses percabangan:

   - TampilHasil = true
   - $i = 10$
c. Komponen keluaran.
   Pada komponen keluaran ada satu instruksi, yaitu: Print *i*.

   Maknanya adalah, mesin komputer diminta menampilkan nilai yang disimpan atau ditampung pada variabel *i*.

Pada Gambar 5.9, *flowchart* tersebut terdapat 2 kondisi yang harus diperhatikan adalah:

a. tampilHasil = True
   Jika *input*an tampilHasil (berupa tombol tampilkan hasil) ditekan oleh pengguna, maka secara otomatis, variabel ini akan bernilai *True*. Sebaliknya, apabila tombol tampilkan hasil tidak ditekan, maka akan bernilai *false* dan berulang pada proses sebelumnya yaitu *input* tampilHasil. Hasil kondisi ini berdampak pada proses selanjutnya. Lihat tabel *tracing* berikut ini. Tabel *tracing* dapat membantu pembaca untuk melakukan analisis proses logika matematika.

Input tampilHasil	Kondisi	Hasil Logika Matematika	Proses selanjutnya yang akan dijalankan
Tombol tampilkan ditekan (true)	tampilHasil == true ?	Yes → true	Berlanjut ke proses i = 1
Tombol tampilkan tidak ditekan (false)	tampilHasil == true ?	No → false	Berulang ke posisi atas untuk input ulang tampilHasil.

b. i = 10.
   Kondisi i=10 merupakan pengecekan batas akhir. Hal ini dikarenakan batas awal telah diinisialisasi bahwa i=1. Pengecekan kondisi tersebut harus melewati 2 proses sebelumnya, yaitu print(i) dan i=i+1. Apabila hasil penambahan kounter i=i+1 belum sama dengan 10, maka proses perulangan masih terus berlanjut. Sebaliknya apabila kounter i=i+1 telah menyamai batas i=10, maka perulangan akan berhenti. Perhatikan hasil analisis tabel *tracing* di bawah ini.

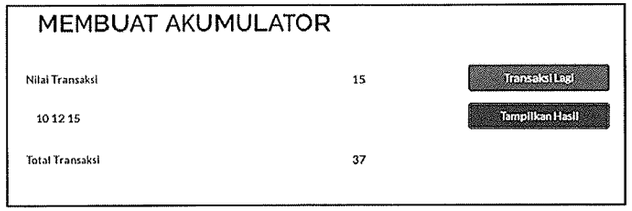
Variabel awal i=1	Print(i)	i=i+1	Kondisi i=10	Hasil Logika Matematika	Proses selanjutnya yang akan dijalankan
I=1	Cetak 1	I=i+1=1+1=2	I=10 ? 2 = 10 ?	False	Print(i)
	Cetak 2	I=i+1=2+1=3	I=10 ? 3 = 10 ?	False	Print(i)
	Cetak 3	I=i+1=3+1=4	I=10 ? 4 = 10 ?	False	Print(i)
	Cetak	I=i+1=...+1=....	I=10 ? ... = 10 ?	False	Print(i)
	Cetak 9	I=i+1=9+1=10	I=10 ? 10 = 10 ?	True	End

Berdasarkan tabel tersebut, hasilnya tercetak angka 1 s.d 9. Angka 10 tidak tercetak, karena kondisi pada saat i=10 langsung diarahkan pada simbol terminal selesai. Hal ini mengakibatkan angka 10 tidak sempat tercetak pada layar komputer.

5.4 Perulangan dan Instruksi Komputasi

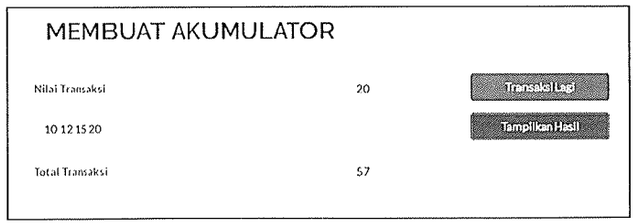
Instruksi Komputasi merupakan bidang ilmu penyusunan model matematika dengan teknik penyelesaian numerik serta penggunaan komputer yang membahas apakah dan bagaimanakah suatu masalah dapat dipecahkan pada model komputasi, menggunakan algoritma yang sesuai dengan permasalahan yang ditentukan. Beberapa komputasi sering kali dikaitkan dengan Operasi Aritmatika. Operasi matematika

menekankan pada pemikiran tentang model aljabar untuk symbol-simbol matematika yang tidak ada pada keybord, seperti sigma (Σ) dan phi (Π). Simbol sigma (Σ) mewakili penjumlahan berulang atau akumulasi penjumlahan (Akumulator). Sedangan simbol phi (Π) mewakili akumulasi perkalian.



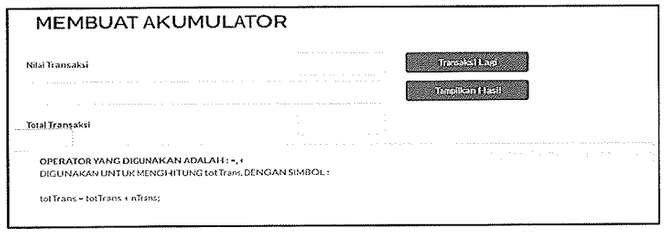
Gambar 5.10 *Menu Membuat Akumulator pada Penalaran.com (1)*

Pada gambar 5.10 terdapat satu *input*an Nilai Transaksi. Nilai transaksi merupakan data masukan yang harus diisi dengan angka. Setelah mengisi sebuah angka (misalnya *input* 10), pengguna dapat menekan tombol Transaksi Lagi, apabila ingin menginputkan angka yang kedua pada Nilai Transaksi (misalnya *input* angka 12 dan 15). Angka yang di*input*kan, akan muncul pada *field* bagian bawahnya. Angka-angka nilai transaksi tersebut akan membentuk sebuah deret angka yaitu 10, 12, 15 dan dapat dijumlahkan. Untuk menjumlahkan angka-angka tersebut, pengguna dapat menekan tombol Tampilkan Hasil saat selesai meng*input*kan nilai transaksi. Hal ini akan memunculkan nilai hasil penjumlahan pada Total Transaksi yaitu 37. Hasil Penjumlahan tersebut didapatkan dari menjumlahkan nilai-nilai yang di*input*kan secara berulang pada kolom Nilai Transaksi (10+12+15). Proses inilah yang dapat dikategorikan sebagai instruksi komputasi berupa Akumulator (Hasil Penjumlahan Berulang).



Gambar 5.11 *Menu Membuat Akumulator pada Penalaran.com (2)*

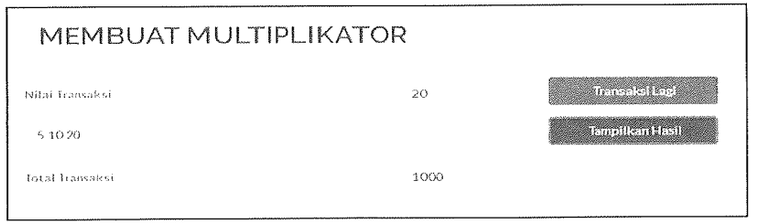
Di sisi lain, apabila pengguna ingin meng*input*kan kembali satu angka lagi yaitu 20, dapat dilakukan dengan memasukkan angka tersebut pada Nilai Transaksi. Kemudian tekan tombol Transaksi Lagi. Hasilnya, angka 20 akan bertambah pada posisi akhir deret. Setelah itu, pengguna dapat menekan tombol Tampilkan Hasil untuk dapat mengetahui hasil Total Transaksi yang baru yaitu 57.



Gambar 5.12 *Instruksi Komputasi pada Akumulator*

Instruksi Komputasi yang digunakan pada pembuatan Akumulator ini dapat dilihat pada Gambar 5.12. Bagaimana sebuah deret angka dapat dijumlahkan adalah dengan menghitung totTrans dengan operasi aritmatika totTrans = totTrans+nTrans. Operator yang digunakan adalah tanda sama dengan (=) dan tanda plus (+).

Variabel nTrans	Variabel tot Trans	Instruksi Komputasi totTrans = totTrans + nTrans	Hasil totTrans	Mengisi angka lagi ? (Y/N)
Input 10	totTrans = 0	totTrans = 0 + 10	10	Yes
Input 12	totTrans = 10	totTrans = 10 + 12	22	Yes
Input 15	totTrans = 22	totTrans = 22 + 15	37	Yes
Input 20	totTrans = 37	totTrans = 37 + 20	57	No
Proses berhenti saat tidak ada angka yang diisi lagi pada kolom Nilai Transaksi.				



Gambar 5.13 *Menu Multiplikator pada Penalaran.com (1)*

Gambar 5.13. tersebut jika diidentifikasi ke dalam komponen pengolahan data menjadi seperti berikut ini:

a. Komponen masukan
   Pada komponen masukan ada 1 instruksi yang dituliskan menjadi satu, yaitu: *Input* nTrans.

   Maknanya, pengguna diminta memasukkan atau memberi nilai pada 1 buah variabel, yaitu: nTrans melalui peralatan masukan.
b. Komponen proses
   Pada komponen proses ada satu pernyataan matematika di setiap *pseudocode* berikut ini.

   1) Deklarasi Awal: totTrans = 1.
   2) Perkalian: totTrans = totTrans * nTrans

   Terdapat pula logika matematika yang digunakan yaitu pada proses percabangan:

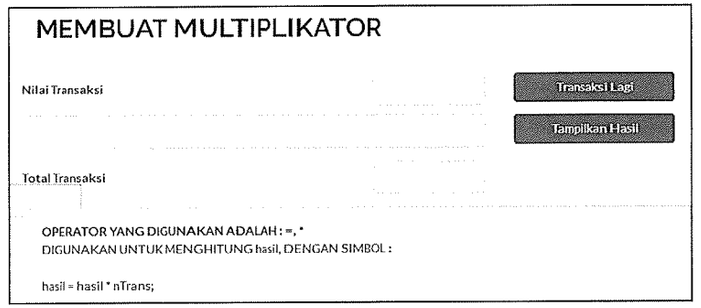
   - Apakah mengisi angka lagi? (jawaban *Y/N*)
c. Komponen keluaran
   Pada komponen keluaran ada satu instruksi, yaitu: Print totTrans.

   Maknanya adalah, mesin komputer diminta menampilkan nilai yang disimpan atau ditampung pada variabel totTrans.

Hal yang sama juga terjadi pada Pembuatan Multiplikator. Perbedaan mendasar antara Akumulator dengan Multiplikator adalah proses perhitungan aritmatika yang digunakan. Jika pada Akumulator, proses perhitungan yang digunakan adalah proses penjumlahan secara berulang. Sedangkan pada Multiplikator, proses perhitungan yang digunakan adalah proses perkalian secara berulang.

Perhatikan gambar 5.13. Bentuk *input*an data masukannya sama dengan tampilan Akumulator. Pengguna diharuskan memasukkan data nilai pada kolom Nilai Transaksi (misal *input* angka 5), kemudian dilanjutkan dengan menekan tombol Transaksi Lagi, apabila diinginkan untuk meng*input*kan kembali. Angka 5 tersebut akan muncul di kolom bagian bawahnya. Pengguna dapat meng*input*kan data nilai lebih dari

satu (misalnya *input* angka 10 dan 20), kemudian tekan tombol Transaksi Lagi. Jika telah muncul pada kolom bagian bawahnya yaitu 5, 10, dan 20, pengguna dapat mengakhiri proses dengan menekan tombol Tampilkan Hasil. Maka pada kolom Total Transaksi akan muncul nilai 1000. Nilai ini didapatkan dengan proses perkalian secara berulang yaitu 5×10×20 = 1000.



Gambar 5.14 *Menu Multiplikator pada Penalaran.com (2)*

Instruksi Komputasi yang digunakan pada pembuatan Multiplikator ini dapat dilihat pada Gambar 5.14. Bagaimana sebuah deret angka dapat dikalikan adalah dengan menghitung nTrans dengan operasi aritmatika hasil = hasil*nTrans. Operator yang digunakan adalah tanda sama dengan (=) dan tanda perkalian (*).

Variabel nTrans	Variabel hasil	Instruksi Komputasi hasil = hasil * nTrans	Nilai Hasil	Mengisi angka lagi ? (Y/N)
Input 5	hasil = 1	hasil = 1 * 5	5	Yes
Input 10	hasil = 5	hasil = 5 * 10	50	Yes
Input 20	hasil = 50	hasil = 50 * 20	1000	No
Proses berhenti saat tidak ada angka yang diisi lagi pada kolom Nilai Transaksi.				

Gambar 5.14. tersebut jika diidentifikasi ke dalam komponen pengolahan data menjadi seperti berikut ini:

a. Komponen masukan
   Pada komponen masukan ada 1 instruksi yang dituliskan menjadi satu, yaitu: *Input* nTrans.

   Maknanya, pengguna diminta memasukan atau memberi nilai pada 1 buah variabel, yaitu: nTrans melalui peralatan masukan.

b. Komponen proses
Pada komponen proses ada satu pernyataan matematika di setiap *pseudocode* berikut ini.

1) Deklarasi Awal: Hasil = 1.
2) Perkalian: Hasil = Hasil * nTrans

Terdapat pula logika matematika yang digunakan yaitu pada proses percabangan:

- Apakah mengisi angka lagi? (jawaban *Y/N*)

c. Komponen keluaran
Pada komponen keluaran ada satu instruksi, yaitu: Print Hasil.

Maknanya adalah, mesin komputer diminta menampilkan nilai yang disimpan atau ditampung pada variabel Hasil.

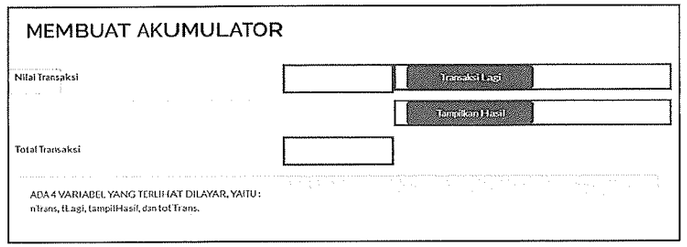
5.5 Perulangan dan Variabel

Pada alur algoritma apapun, variabel selalu merupakan kebutuhan utama bagi seorang programmer. Hal pertama yang harus dikuasai untuk membuat sebuah algoritma adalah mengidentifikasi kebutuhan variabel. Jika variabel yang dibutuhkan kurang, maka algoritma tersebut tidak dapat berjalan sebagaimana mestinya.

Penentuan variabel untuk kebutuhan algoritma perulangan pada aplikasi penalaran.com dibagi menjadi 2 jenis, yaitu variabel yang terlihat dilayar dan variabel yang tidak terlihat di layar.. Seperti pada Gambar 5.15. Tampilan menu tersebut dapat diakses pada **Menu Pernyataan Matematika - Operasi Aritmatika, kemudian pilih Akumulasi Penjumlahan.**

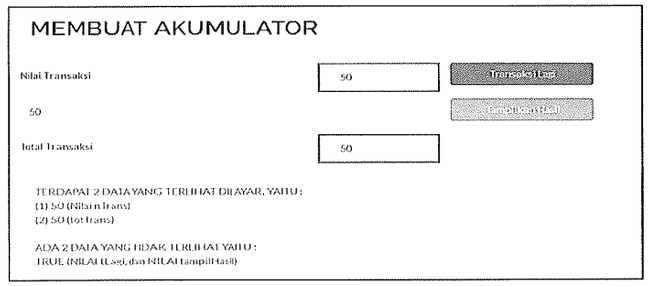
a. Variabel yang terlihat di layar dapat diidentifikasi melalui kolom data masukan dan kolom data keluaran pada Menu Pembuatan Akumulator yaitu kolom data masukan Nilai Transaksi dan kolom data keluaran Total Transaksi. (1) Kolom data masukan Nilai Transaksi disimpan pada variabel **nTrans.** (2) Kolom data keluaran Total Transaksi disimpan pada variabel **totTrans.**

b. Sedangkan variabel yang tidak terlihat di layar dapat diidentifikasi melalui tombol yang ditekan atau tidak ditekan oleh pengguna. Pada form Akumulator tersebut terdapat 2 tombol, yaitu (3) tombol Transaksi Lagi disimpan melalui variabel **tLagi** dan (4) tombol Tampilkan Hasil disimpan melalui variabel **tampilHasil.**



Gambar 5.15 *Kebutuhan Variabel pada Menu Pembuatan Akumulator pada Penalaran.com*

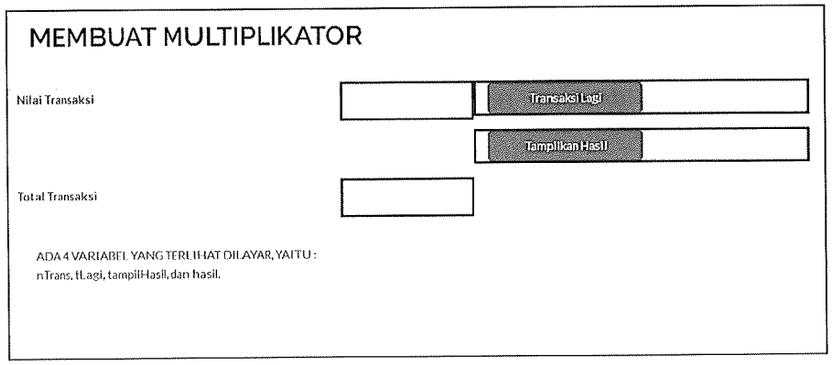
Oleh karena itu, jumlah total variabel yang digunakan pada Form Akumulator tersebut terdiri dari 4 variabel yaitu nTrans, totTrans, tLagi, dan tampilHasil. Setelah variabel sudah teridentifikasi, selanjutnya pengguna harus mengetahui jenis data yang diinputkan pada kolom data masukan dan data keluaran. Lihat pada Gambar 5.16. Terdapat 2 data yang terlihat di layar yaitu data nilai 50 untuk variabel nTrans dan data nilai 50 untuk variabel totTrans. Sedangkan data yang tidak terlihat di layar yaitu data menekan tombol Transaksi Lagi dan tombol Tampilkan Hasil dengan nilai *True*. Apabila tidak ditekan, maka bernilai *false*.



Gambar 5.16 *Jenis Data yang digunakan pada Akumulator menu Penalaran.com.*

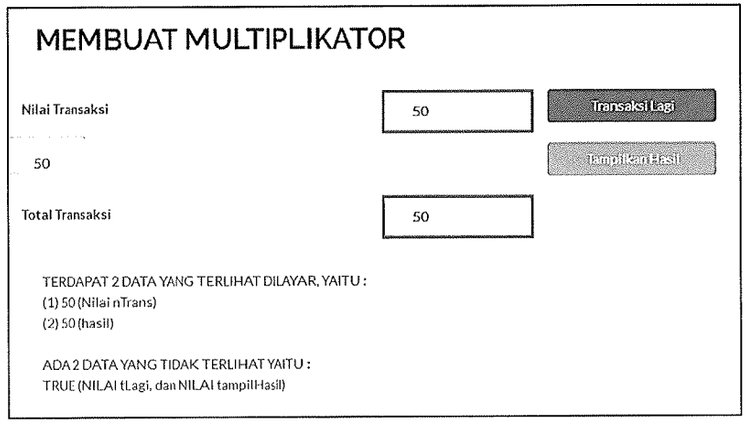
Setelah memahami kebutuhan variabel untuk Akumulator, maka pada Gambar 5.17 digambarkan bagaimana cara untuk mengidentifikasi kebutuhan variabel untuk Multiplikator. Sebenarnya tidak jauh berbeda dengan Akumulator. Penentuan variabel untuk kebutuhan algoritma perulangan pada Multiplikator juga dibagi menjadi 2 jenis, yaitu variabel yang terlihat dilayar dan variabel yang tidak terlihat di layar. Tampilan menu tersebut dapat diakses pada **Menu Pernyataan Matematika - Operasi Aritmatika, kemudia pilih Akumulasi Perkalian.**

a. Variabel yang terlihat di layar dapat diidentifikasi melalui kolom data masukan dan kolom data keluaran pada Menu Pembuatan Multiplikator yaitu kolom data masukan Nilai Transaksi dan kolom data keluaran Total Transaksi. (1) Kolom data masukan Nilai Transaksi disimpan pada variabel **nTrans.** (2) Kolom data keluaran Total Transaksi disimpan pada variabel **hasil.**
b. Sedangkan variabel yang tidak terlihat di layar dapat diidentifikasi melalui tombol yang ditekan atau tidak ditekan oleh pengguna. Pada form Multiplikator tersebut terdapat 2 tombol, yaitu (3) tombol Transaksi Lagi disimpan melalui variabel **tLagi** dan (4) tombol Tampilkan Hasil disimpan melalui variabel **tampilHasil.**



Gambar 5.17 *Kebutuhan Variabel pada Menu Pembuatan Multiplikator pada Penalaran.com*

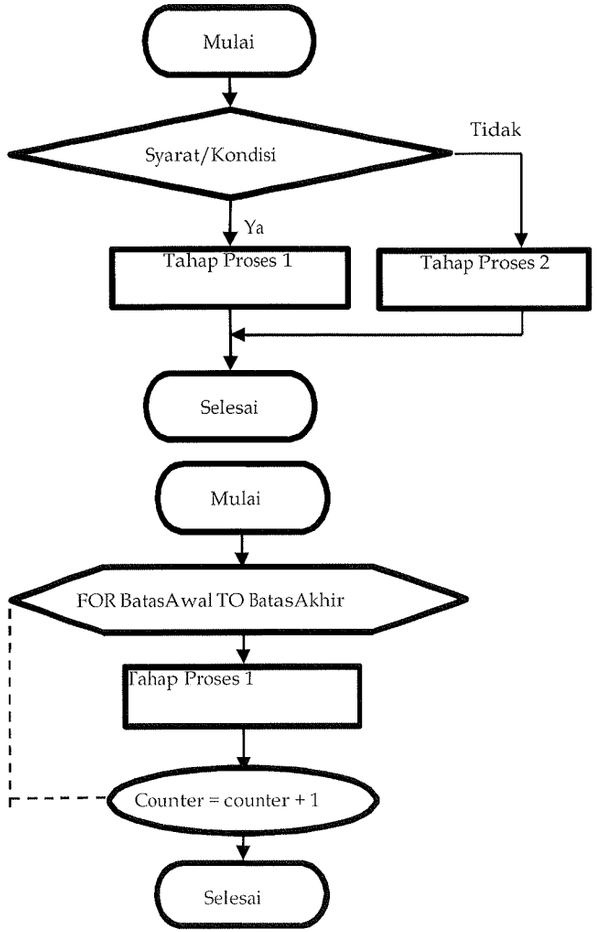
Oleh karena itu, jumlah total variabel yang digunakan pada Form Multiplikator tersebut terdiri dari 4 variabel yaitu nTrans, hasil, tLagi, dan tampilHasil. Setelah variabel sudah teridentifikasi, selanjutnya pengguna harus mengetahui jenis data yang diinputkan pada kolom data masukan dan data keluaran. Lihat pada Gambar 5.18. Terdapat 2 data yang terlihat di layar yaitu data nilai 50 untuk variabel nTrans dan data nilai 50 untuk variabel hasil. Sedangkan data yang tidak terlihat di layar yaitu data menekan tombol Transaksi Lagi dan tombol Tampilkan Hasil dengan nilai *True*. Apabila tidak ditekan, maka bernilai *false*.



Gambar 5.18 *Jenis Data yang digunakan pada Akumulator menu Penalaran.com.*

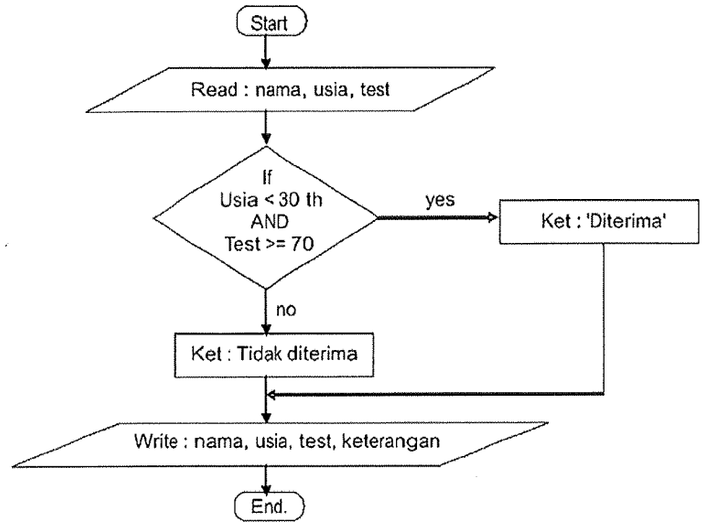
5.6 Perulangan dan Percabangan

Alur algoritma perulangan dan percabangan sangat berbeda jauh. Perulangan merupakan proses yang dilakukan secara berulangan dengan kondisi atau batas tertentu. Sedangkan Percabangan merupakan proses yang terjadi karena terdapat kondisi atau syarat yang harus dipenuhi, sehingga menyebabkan pengguna harus menentukan pilihan dan mengambil keputusan.



Gambar 5.19 *(A) Flowchart Alur Percabangan (B) Flowchart Alur Perulangan.*

Selain perbedaan yang mendasar di atas, kedua alur algoritma ini juga memiliki persamaan yaitu sama-sama memiliki kondisi yang harus dipenuhi. Jika pada alur perulangan, kondisi digunakan sebagai batas awal dan batas akhir perulangan. Sedangkan pada alur algoritma percabangan, kondisi digunakan sebagai syarat yang harus dipenuhi atau dipilih, agar proses selanjutnya dapat dijalankan. Adapun contoh *flowchart* dasar yang membedakan keduanya, dapat dilihat pada Gambar 5.19.



Gambar 5.20 *Contoh Flowchart Percabangan*

Perhatikan Gambar 5.20. Pada *Flowchart* tersebut terdapat 3 buah variabel yang di*input*kan, yaitu nama, usia, dan (nilai) *test*. Kondisi yang harus dipenuhi adalah Jika usia lebih kecil dari 30 DAN (nilai) tes lebih besar sama dengan 70. Apabila variabel yang di*input*kan memenuhi kondisi maka akan diproses "Diterima". Namun, sebaliknya apabila tidak terpenuhi, maka akan diproses "Tidak Diterima".

Gambar 5.20. tersebut jika diidentifikasi ke dalam komponen pengolahan data menjadi seperti berikut ini:

a. Komponen masukan
   Pada komponen masukan ada 3 instruksi yang dituliskan menjadi satu, yaitu: *Input* nama, usia, *test*

   Maknanya, pengguna diminta memasukan atau memberi nilai pada 3 buah variabel, yaitu nama, usia, *test* melalui peralatan masukan.
b. Komponen proses
   Pada komponen proses ada pernyataan matematika di setiap *pseudocode* berikut ini.

1) Pengisian nilai jika kondisi benar: Ket =Diterima
2) Pengisian nilai jika kondisi salah: Ket = Tidak Diterima

Terdapat pula logika matematika yang digunakan yaitu pada proses percabangan:

- If Usia < 30 AND test >= 70

c. Komponen keluaran.
Pada komponen keluaran ada satu instruksi, yaitu: Print nama, usia, *test*, dan keterangan.

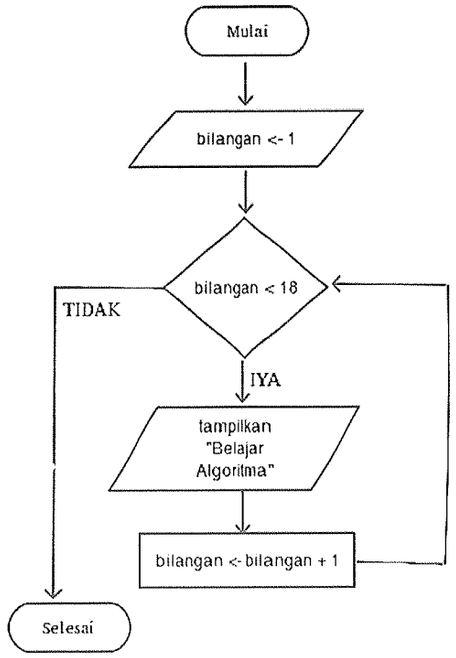
Maknanya adalah, mesin komputer diminta menampilkan nilai yang disimpan atau ditampung pada variable nama, usia, *test*. Keterangan.

Untuk memudahkan proses menalar Anda, gunakan bantuan menggunakan Tabel *Tracing*. Tabel *Tracing* ini digunakan untuk menguji apakah kondisi tersebut dapat terpenuhi atau tidak. Sekaligus menentukan hasil akhir dari proses percabangan tersebut. Adapun contoh tabel *tracing* yang dapat digunakan untuk menganalisa bentuk penalarannya adalah sebagai berikut:

Input variabel Nama	Input variabel Usia	Input variabel test	kondisi	Hasil boolean (tidak terlihat)	Hasil keterangan
Dian	25	50	Usia < 30 DAN test >= 70	True DAN false = False	Tidak Diterima
Ayu	28	85	Usia < 30 DAN test >= 70	True DAN true = true	Diterima
Joko	30	70	Usia < 30 DAN test >= 70	False DAN true	Tidak Diterima

Cobalah untuk mengingat kembali mengenai penggunaan Operator penghubung AND dan OR. Terdapat aturan yang mengikat keduanya. Operator penghubung AND bernilai TRUE apabila kedua kondisinya bernilai *true*. Apabila terdapat salah satu saja yang bernilai *false*, maka nilai akhir akan tetap FALSE. Hasil akhir dari *flowchart* tersebut akan menampilkan variabel nama, usia, *test*, dan keterangan.

Perhatikan contoh *flowchart* perulangan pada Gambar 5.21. Di awal telah diinisialisasi bahwa variabel bilangan bernilai 1. Selama bilangan bernilai lebih kecil dari 18, maka akan tampil kalimat "Belajar Algoritma". Kounter perulangan akan berjalan dengan penambah 1. Namun apabila variabel bilangan telah mencapai lebih dari 18, maka proses perulangan akan berhenti. Lihat hasil tabel tracing berikut ini.



Gambar 5.21 *Contoh Flowchart Perulangan*

Variabel bilangan	Kondisi	Output Layar	Kounter Bilangan = bilangan + 1
Bilangan = 1	Bilangan < 18 = true	"Belajar Algoritma"	Bilangan = 1+1=2
Bilangan = 2	Bilangan < 18 = true	"Belajar Algoritma"	Bilangan = 2+1=3
Bilangan = 3	Bilangan < 18 = true	"Belajar Algoritma"	Bilangan = 3+1=4
....	....	.....	....
Bilangan = 17	Bilangan < 18 = true	"Belajar Algoritma"	Bilangan = 17+1=18
Bilangan = 18	Bilangan < 18 = false	(kosong–proses selesai)	(kounter tidak berjalan lagi)

Gambar 5.21. tersebut jika diidentifikasi ke dalam komponen pengolahan data menjadi seperti berikut ini:

a. Komponen masukan
   Pada komponen masukan, tidak ada perintah *input* nilai. Hal ini kemungkinan terdapat variabel yang langsung dideklarasikan.
b. Komponen proses
   Pada komponen proses ada pernyataan matematika di setiap *pseudocode* berikut ini.

   - Penjumlahan kounter: bilangan = bilangan + 1

   Terdapat pula logika matematika yang digunakan yaitu pada proses percabangan:

   - If bilangan < 18
c. Komponen keluaran
   Pada komponen keluaran ada satu instruksi, yaitu: Print/menampilkan kalimat "Belajar Algoritma"

   Maknanya adalah, mesin komputer diminta menampilkan kalimat yang langsung ditampilkan pada layar komputer "Belajar Algoritma"

5.7 Rangkuman

- Proses perulangan adalah sebuah proses program di mana setiap perintah/instruksi dapat dijalankan secara berulang dengan kondisi tertentu.
- Kounter adalah sebuah proses otomatis yang biasa digunakan dalam aktivitas sehari-hari. Contoh *counter* adalah membuat nomer urut, misal 1, 2, 3, dst.
- Operasi matematika menekankan pada pemikiran tentang model aljabar untuk simbol-simbol matematika yang tidak ada pada keyboard, seperti sigma (Σ) dan phi (Π).

5.8 Latihan Soal

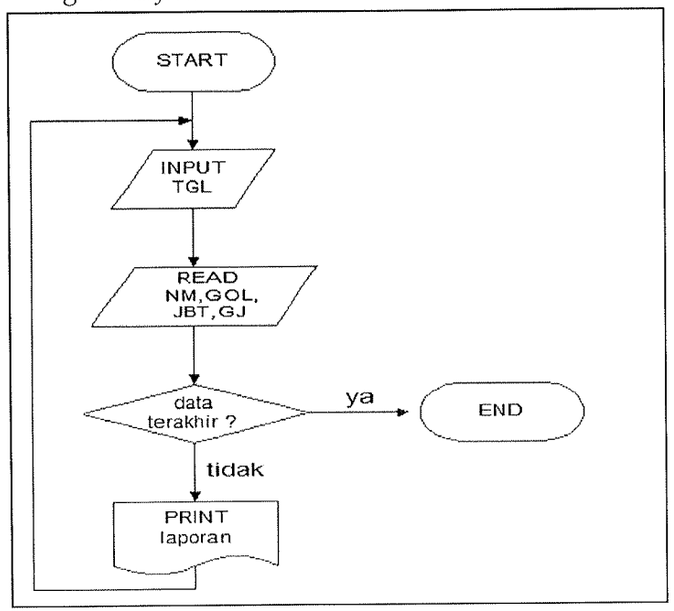
Buatlah algoritma berupa *flowchart* untuk kasus di bawah ini:

1. Buatlah algoritma untuk memecahkan masalah dalam program untuk struktur perulangan. Misalkan anda diminta untuk membuat algoritma dan program menampilkan bilangan dari 10 sampai 1 secara menurun, yaitu: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
2. Buatlah *flowchart* dengan inputan berupa tombol, yaitu jika tombol ditekan sebanyak 10 kali maka akan mengirim data serial berupa kalimat tercetak di layar ke komputer "TOMBOL TELAH DITEKAN 10 KALI".
3. Buatlah *pseudocde* untuk menghitung *n*? = 1+2+3+.....+*n* dan menampilkan hasilnya.
   Ketentuan:

   *n di-*input*kan oleh *user*
   Output, jika *n* = 10 maka akan tampil sebagai berikut:
   n = 10
   1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 +
   total = 55
4. Buatlah algoritma untuk mencetak deret bilangan genap yang terletak antara -4 sampai dengan 50.
5. Buatlah autentifikasi login aplikasi. Jika *user* dan *password* tidak cocok, maka harus mengulang hingga *user* dan *password* yang di*input*kan benar.
6. Buatlah algoritma untuk menampilkan status bilangan ganjil dan genap dari deret angka inputan nilai awal dan nilai akhir di*input*kan *user* dengan kelipatan 1.
7. Gambarlah *flowchart* untuk memasukkan nama mahasiswa, nilai UTS, nilai UAS, dan nilai Tugas dari *N* mahasiswa melalui keyboard. Nilai *N* sendiri diisi sebelumnya melalui keyboard juga dengan validitas *harus bilangan positif*. *Flowchart* akan menghitung dan menampilkan rata-rata nilai akhir, nilai akhir terbesar dan nama mahasiswa yang memperolehnya, nilai akhir terkecil dan nama

mahasiswa yang memperolehnya. Perhatikan bahwa: Nilai akhir untuk seorang mahasiswa diperoleh dari 10% Nilai Tugas 45% Nilai UTS + 45% Nilai UAS.

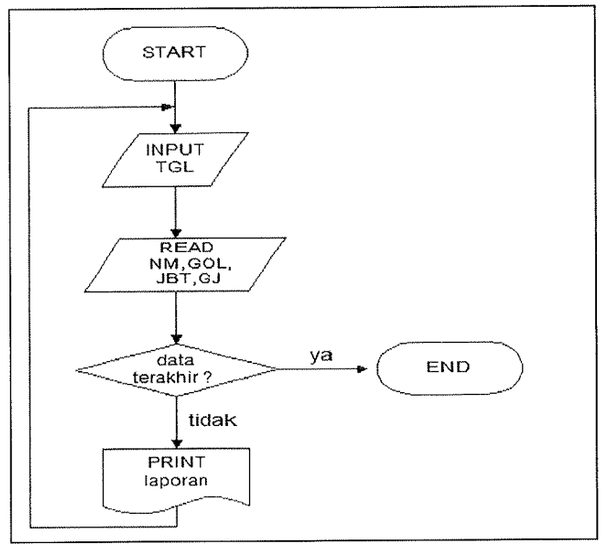
8. Perhatikan gambar *flowchart* berikut ini:



Berdasarkan *flowchart* di atas, tuliskan jawaban soal di bawah ini dengan benar.

a. Sebutkan semua variabel yang berperan dalam keseluruhan proses pada *flowchart* tersebut ?
b. Sebutkan semua instruksi/perintah komputer yang berjalan pada *flowchart* tersebut?
c. Sampai kapan *flowchart* tersebut berhenti berulang?

9. Perhatikan gambar *flowchart* berikut ini:



Berdasarkan *flowchart* di atas, Berikan 3 contoh data *input*, data proses, dan data *output* yang didapatkan dari flowchart tersebut.

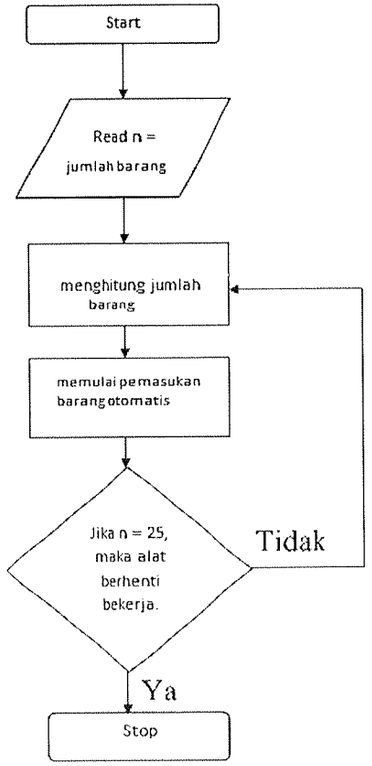
10. Perhatikan potongan code program berikut ini:

```
1   #include <iostream>
2
3   using namespace std;
4   int main(){
5       int umur;
6       cout << "Umur Kamu Berapa ? ";
7       cin >> umur;
8
9       if (umur >= 17){
10          cout << "Yeah , Kamu Boleh Main Tiktok ! :)" << endl;
11      }
12      else{
13          cout << "Aduh, Masih Kecil Jangan Main Tiktok ! " << endl;
14      }
15  }
16
```

Berdasarkan *code* program tersebut, lakukan analisis hasil *output*, jika di*input*kan sejumlah nilai berikut ini:

a. Umur 5 tahun
b. Umur 16 tahun
c. Umur 25 tahun
d. Umur 50 tahun

11. Buatlah *pseudocode* yang dapat mencetak deret angka dengan kounter atau jangkauan deret adalah penambah 3. *Input*kan terlebih dahulu batas awal dan batas akhir perulangan.
    Contoh Hasil *Output*: *Input* Batas Awal = 1 Input Batas Akhir = 15
    Hasil Deret = 3, 6, 9, 12, 15
12. Buatlah *pseudocode* yang dapat mencetak deret angka dengan kounter atau jangkauan deret adalah pengurang 3. Inputkan terlebih dahulu batas awal dan batas akhir perulangan.
    Contoh Hasil *Output*: *Input* Batas Awal = 15 *Input* Batas Akhir = 1
    Hasil Deret = 15, 12, 9, 6, 3
13. Perhatikan *flowchart* berikut ini:



Berdasarkan *flowchart* tersebut, jawablah pertanyaan yang benar berikut ini:

a. Berapa kali perulangan tersebut berulang selama *flowchart* dijalankan?
b. Sebutkan semua variabel yang dibutuhkan/digunakan pada *flowchart* tersebut.
c. Gantilah Perintah "memulai pemasukan barang otomatis" menggunakan intruksi *pseudocode* yang tepat dan benar.

14. Berdasarkan gambar *flowchart* nomor 13, gantilah semua intruksi yang masih tertulis pada *flowchart* tersebut menggunakan pola *pseudocode* sesuai aturan penulisan algoritmik yang benar.
15. Berdasarkan gambar *flowchart* nomor 13, berikan 3 contoh data *input*, data proses, data *output* apabila *flowchart* tersebut dijalankan dari awal hingga akhir.

-oo0o-

BAB 6

PERULANGAN 2

DESKRIPSI SINGKAT

Pada bab 6 ini akan dibahas mengenai bentuk-bentuk perulangan serta beberapa contoh solusi permasalahan dalam bentuk perulangan sederhana. Penerapan sederhana tentang alur algoritma Perulangan ini diterapkan dan dibahas lebih detail pada konsep Akumulator, Multiplikator, dan perhitungan Faktorial.

Bentuk-bentuk perulangan yang disajikan pada buku ini dibagi menjadi 4 jenis, yaitu bentuk For, bentuk Do-While, bentuk While, bentuk Nested For. Setiap bentuk perulangan tersebut akan dibahas bentuk *flowchart* dan *pseudocode* yang dapat digunakan sebagai acuan pembelajaran di kelas.

Setelah memahami bentuk setiap alur algoritma perulangan, sub bab selanjutnya akan dibahas mengenai Akumulator, Multiplikator, dan Faktorial secara utuh, mulai dari kebutuhan data masukan hingga kode program yang dapat digunakan.

Di akhir sub bab telah disiapkan rangkuman sederhana dari pembahasan bab 6 ini beserta latihan soal yang dapat dikerjakan sebagai tugas mandiri oleh mahasiswa di kelas berdasarkan studi kasus yang telah ditentukan.

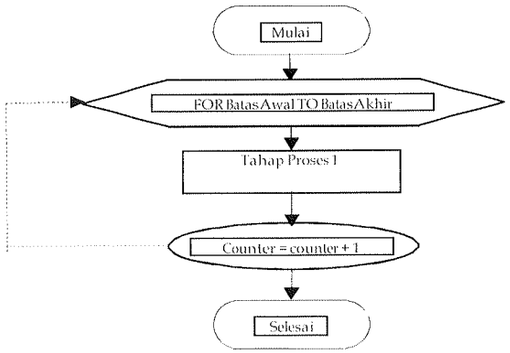
Tujuan Pembelajaran

Tujuan akhir pembelajaran pada Bab 5 ini dibagi menjadi 3 bagian, antara lain:

1. Jika diberi soal atau studi kasus, mahasiswa dapat **menganalisis** jenis-jenis alur algoritma Perulangan yang tepat untuk kasus tersebut.
2. Jika diberi soal atau studi kasus, mahasiswa dapat **membuktikan** hasil penerapan algoritma perulangan pada permasalahan proses bisnis.
3. Tujuan terakhir adalah mahasiswa dapat **menyimpulkan** perbedaan jenis alur perulangan pada Akumulator, Multiplikator, dan Faktorial.

6.1 Pendahuluan

Dalam sebuah algoritma, tahapan penyelesaian sebuah permasalahan dapat berupa struktur berurutan (*sequence*), struktur pemilihan atau pengambilan keputusan (*selection*), dan struktur pengulangan (*iteration*). Ketiga jenis langkah tersebut membentuk struktur konstruktif suatu algoritma. Proses perulangan adalah sebuah proses program di mana setiap perintah/instruksi dapat dijalankan secara berulang dengan kondisi tertentu. Kondisi pada proses perulangan ini biasanya digunakan sebagai batas awal dan batas akhir perulangan. Apabila tidak ada kondisi sebagai batas perulangan, maka proses pada algoritma tersebut akan berjalan terus-menerus, sehingga mengakibatkan adanya *out of space memory*.



Gambar 6.1 *Flowchart Proses Perulangan*

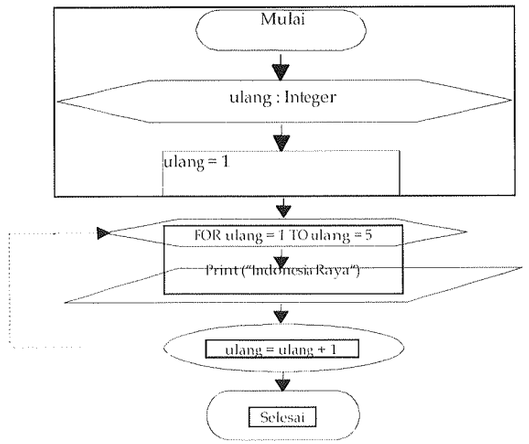
Berdasarkan Gambar 6.1, *flowchart* berjalan pada awal tahapan mulai, kemudian berjalan pada proses perulangan dengan batas awal dan batas akhir yang harus dipenuhi terlebih dahulu. Jika batas awal dan batas akhir telah diberi nilai awal, maka Tahap Proses 1 siap untuk dijalankan secara berulang. Proses perulangan tahap proses 1 akan terjadi melalui bantuan *counter*. *Counter* ini berfungsi sebagai penambah satu angka menuju tingkat perulangan berikutnya. Penambahan satu pada *counter* akan terjadi sampai batas akhir yang telah ditentukan sebelumnya. Jika penambahan *counter* telah lebih kecil atau sama dengan batas akhir, maka proses perulangan berhenti.

Proses perulangan ini dapat terjadi pada 3 bagian *flowchart*, yaitu pada bagian awal, tengah, maupun akhir *flowchart*. Sesuaikan dengan kondisi permasalahan yang ingin diselesaikan. Apabila proses percabangan diletakkan pada bagian awal, pada tahap bagian tengah dan akhir dapat diisi dengan proses sekuensial/percabangan lainnya yang merupakan proses kelanjutan dari proses perulangan.

Contoh: Flowchart mencetak Kata "Indonesia Raya" sebanyak 5 kali.

```
Input : -
Batas Awal = 1 Batas Akhir = 5
```

Proses yang diulang: Mencetak kata Indonesia Raya *Output*: Kata "Indoensia Raya" dicetak sebanyak 5 kali



Gambar 6.2 *Flowchart Mencetak Kata "Indonesia Raya" Sebanyak 5 Kali*

6.2 Jenis Perulangan

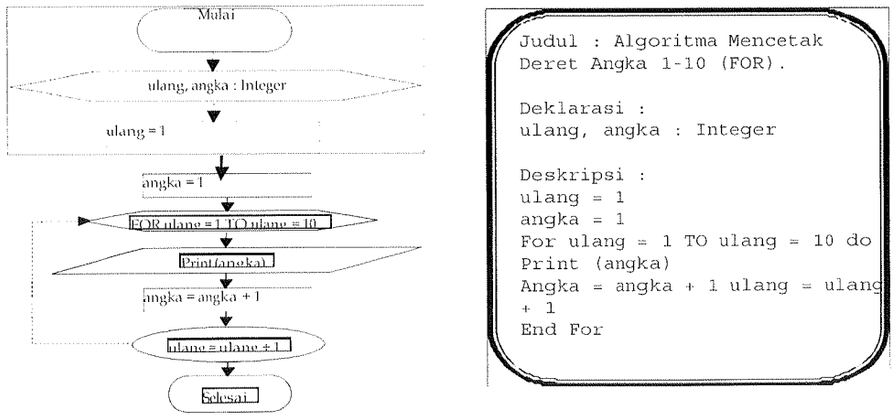
Terdapat beberapa jenis bentuk proses perulangan yang penting untuk dipelajari sebagai konsep dasar perulangan. Bentuk *Flowchart* dan *Pseuducode* yang telah dipelajari pada sub bab sebelumnya merupakan bentuk umum dari proses perulangan. Adapun bentuk proses perulangan dibagi menjadi 4 bentuk, yaitu sebagai berikut:

1. Bentuk For

   Bentuk For ini merupakan bentuk dasar yang paling sederhana dibandingkan bentuk for yang lainnya. Bentuk ini dikatakan paling sederhana karena paling mudah dalam prosesnya, masukkan batas awal dan akhir, dan gunakan *counter* sebagai variabel penambah satu. Dalam Bahasa Indonesia, bentuk ini dikenal dengan sebutan Perulangan untuk batas awal sama dengan sekian sampai batas akhir sama dengan sekian. Bentuk For juga dapat digunakan sebagai dasar pemrosesan untuk Array 1D.

   Contoh 1 : *Flowchart* mencetak deret angka 1-10 secara Ascending (FOR).

   Input: -
   Batas Awal = 1 Batas Akhir = 10
   Proses yang diulang: Mencetak angka
   Output: Mencetak deret angka 1, 2, 3, 4, 5, 6, 7, 8, 9, 10



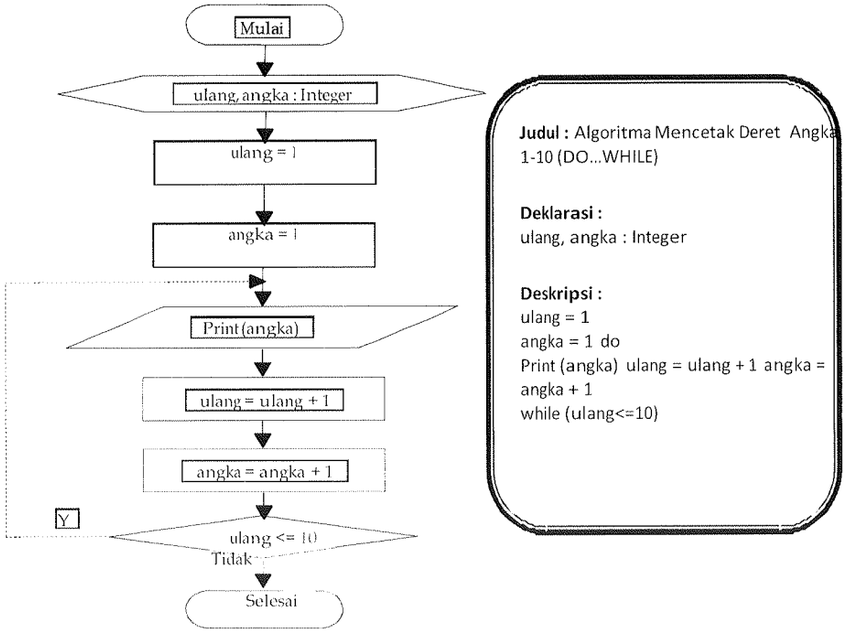
Gambar 6.3 *Flowchart Mencetak Deret Angka 1-10 secara Ascending (FOR)*

2. Bentuk Do-While

   Bentuk Do-While merupakan bentuk perulangan selain for yang menggunakan kondisi, seperti pada IF-THEN. Kondisi tersebut merupakan batas perulangan yang dapat ditentukan pada program. Proses perulangan dan counter dapat dijalankan apabila kondisi batasnya masih terpenuhi. Dalam Bahasa Indonesia, bentuk ini dikenal dengan kata LAKUKAN proses perulangan SELAMA masih memenuhi kondisi batas yang ditentukan. Hanya saja, pada penulisan pseudocode, kondisi berada di posisi paling bawah.

   Contoh: Flowchart mencetak deret angka 1-10 secara Ascending (DO...WHILE).

   Input : -
   Batas Awal = 1 Batas Akhir = 10
   Proses yang diulang : Mencetak angka
   Output : Mencetak deret angka 1, 2, 3, 4, 5, 6, 7, 8, 9, 10



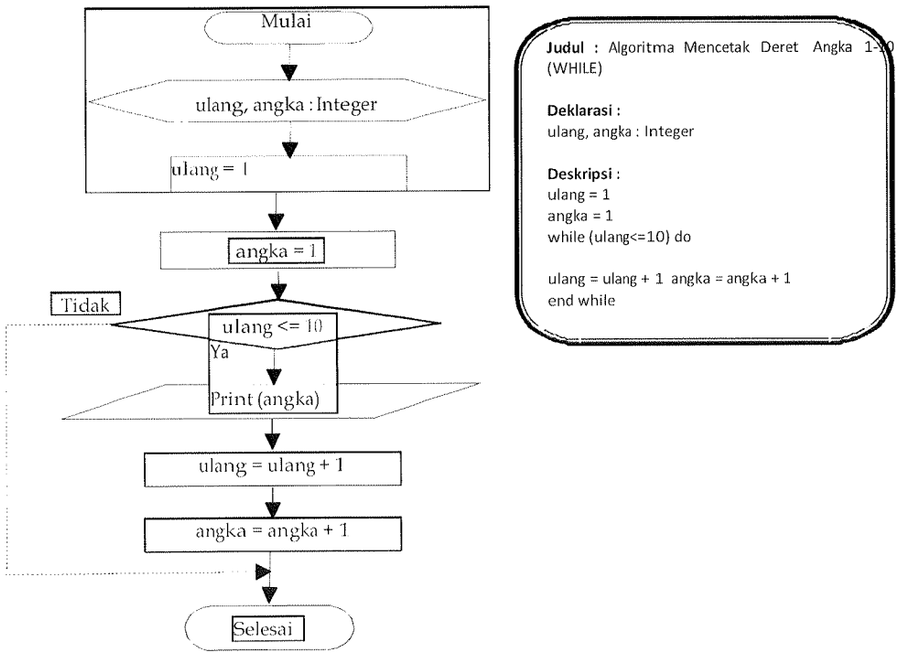
Gambar 6.4 *Flowchart Mencetak Deret Angka 1-10 secara Ascending (DO WHILE)*

3. Bentuk While

Bentuk While merupakan bentuk perulangan kebalikan dari Do-While. Jika pada perulangan yang menggunakan Do-While, kondisi batas berada di posisi paling bawah, sedangkan pada perulangan yang menggunakan While, kondisi batas berada di posisi paling atas. Dalam Bahasa Indonesia, bentuk ini dikenal dengan kata SELAMA kondisi masih memenuhi batas yang ditentukan, LAKUKAN proses perulangan.

Contoh: *Flowchart* mencetak deret angka 1-10 secara Ascending (WHILE).

Input : -
Batas Awal = 1 Batas Akhir = 10
Proses yang diulang : Mencetak angka
Output : Mencetak deret angka 1, 2, 3, 4, 5, 6, 7, 8, 9, 10



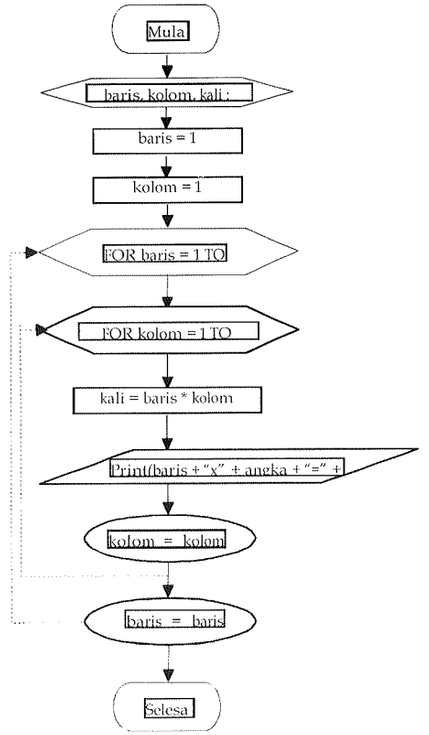
Gambar 6.5 *Flowchart Mencetak Deret Angka 1-10 secara Ascending (WHILE).*

4. Bentuk Nested For

Bentuk Nested For memiliki konsep yang sama dengan bentuk Nested IF. Bentuk Nested For merupakan bentuk for di dalam for, yang biasa disebut For Bersarang atau For Bertingkat. Bentuk Nested For biasanya sering diaplikasikan untuk mencetak angka atau karakter yang mempunyai struktur baris dan kolom. Oleh karena itu, nested for merupakan dasar pemrosesan untuk Array 2D.

Contoh: *Flowchart* mencetak deret perkalian angka 1-5 secara urut.
Input: -
Batas Baris Awal = 1 Batas Baris Akhir = 10 Batas Kolom Awal = 1 Batas Kolom Akhir = 5

Proses yang diulang: Mengalikan angka 1-5 *Output*: Mencetak deret perkalian angka 1-5.



Gambar 6.6 *Flowchart Mencetak Deret Perkalian Angka 1-5Secara Urut*

6.3 Masalah Perulangan Sederhana

Permasalahan perulangan sederhana sering kali disajikan dalam bentuk deret angka atau inputan data yang dilakukan secara berulang. Contoh: terdapat sejumlah nama mahasiswa yang harus dimasukkan ke dalam sebuah sistem. Lihat tabel di bawah ini. (1) Tentukan terlebih dahulu variabel yang akan digunakan sebagai batas awal dan batas akhir serta variabel untuk proses yang akan diulang. (2) Lakukan proses perulangan dengan menggunakan salah satu bentuk perulangan, misalnya for.

No.	Kalimat Deskriptif	Pseudocode
1	Menentukan variabel mendeklarasikan tipe data berdasarkan data yang telah disediakan.	nama_mhs : String i: integer

2	Melakukan proses perulangan.
	• Tentukan jumlah perulangan atau batas akhir, misalnya 5 kali
	• Tentukan nilai awal counter yaitu 1
	• Counter akan dihitung menggunakan increment (i++).
	• Tracing nilai I : i = 0
	i++ ≈ i = i + 1
	i = 0 + 1 = 1
	i = 1 + 1 = 2
	i = 2 + 1 = 3
	i = 3 + 1 = 4
	i = 4 + 1 = 5
	(i = 5 + 1 = 6) → x
	Tidak digunakan, karena batas akhir adalah 5
		for i = 1 to 5 do
		
		input (nama_mhs) print (nama_mhs) i++
		end for

6.4 Akumulator

Akumulator adalah proses penjumlahan berulang dengan menambahkan secara berurutan hasil penjumlahan dengan bilangan berikutnya. Simbol secara matematisnya adalah sebagai berikut.

$$S = \frac{n}{x_i}$$
$$i = 1$$

Nilai variabel S diperoleh dengan cara menjumlahkan nilai x dari 1 sampai ke n. Contoh 1, untuk n=5 dan nilai x adalah sebagai berikut.

n	x
1	10
2	35
3	25
4	20
5	40

Maka

$$S = \frac{n}{x_i} = 10 + 35 + 25 + 20 + 40 = 130$$
$$i = 1$$

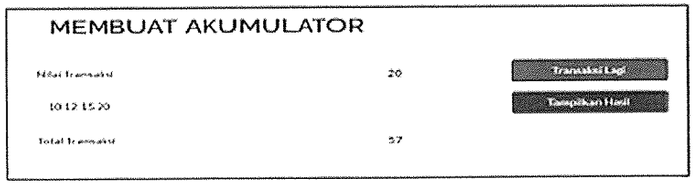
Contoh 2, untuk n=7 dan nilai x adalah sebagai berikut.

n	x
1	10
2	35
3	25
4	20
5	40
6	10
7	10

Maka :

$$S = \sum_{i=1}^{n} x_i = 10 + 35 + 25 + 20 + 40 + 10 + 10 = 150$$

Contoh implementasi Akumulasi Penjumlahan pada Aplikasi Belajar Dasar-dasar Algoritma Komputasi : Menu Pernyataan Matematika - Operasi Aritmatika (Akumulasi Penjumlahan), dapat dilihat pada Gambar 6.7 di bawah ini.



Gambar 6.7 *Menu Membuat Akumulator pada Penalaran.com*

Pada gambar 6.7 terdapat satu *input*an Nilai Transaksi. Nilai transaksi merupakan data masukan yang harus diisi dengan angka. Setelah mengisi sebuah angka (misalnya *input* 10), pengguna dapat menekan tombol Transaksi Lagi, apabila ingin meng*input*kan angka yang kedua pada Nilai Transaksi (misalnya input angka 12 dan 15). Angka yang diinputkan, akan muncul pada *field* bagian bawahnya. Angka-angka nilai transaksi tersebut akan membentuk sebuah deret angka yaitu 10, 12, 15 dan dapat dijumlahkan. Untuk menjumlahkan angka-angka tersebut, pengguna dapat menekan tombol Tampilkan Hasil saat selesai menginputkan nilai transaksi. Hal ini akan memunculkan nilai hasil penjumlahan pada Total Transaksi yaitu 37. Hasil Penjumlahan tersebut didapatkan dari menjumlahkan nilai-nilai yang diinputkan secara berulang pada kolom Nilai Transaksi (10+12+15) = 37.

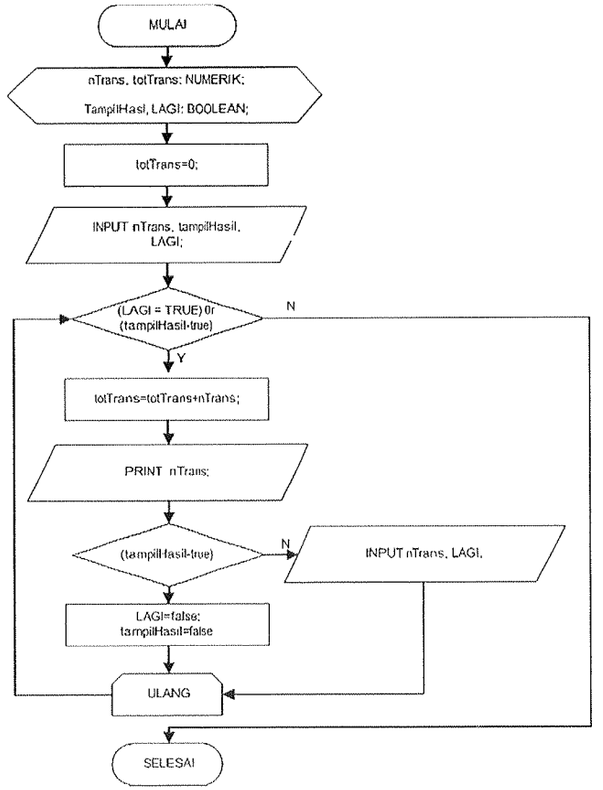
Pada gambar *Flowchart* 6.8, dapat dilihat bahwa terdapat 4 buah variavel yang harus dideklarasikan yaitu variabel (1) nTrans dan (2) totTrans bertipe Numerik (karena inputan data berupa angka) serta variabel (3) TampilHasil dan (4) Lagi bertipe Boolean (karena penekanan tombol bersifat *true/false*). Variabel totTrans awalnya diisi dengan angka 0 (nol). Kemudian sistem aplikasi akan menerima inputan dari nTrans, tampilHasil, dan LAGI. Jika LAGI bernilai true atau tampilHasil bernilai *true*, maka penjumlahan variabel totTrans = totTrans + nTrans dilakukan. Setelah itu proses cetak/*print* nTrans dilakukan.

Namun, di bagian tengah perulangan, terdapat pengecekan kembali variabel tampilHasil apakah bernilai *true*. Jika ya, maka variabel LAGI disetting false dan tampilHasil disetting *false*. Sebaliknya, apabila

tampilHasil tidak bernilai *true* (*false*), maka lakukan *input* data kembali untuk variabel nTrans dan LAGI. Proses ini akan berulang terus dengan batasan bahwa variabel LAGI bernilai *false* atau variabel tampilHasil bernilai *false*.

Untuk memudahkan proses logikanya, pengguna dapat menganalisis melalui tabel penalaran/*tracing* berikut ini:

Variabel nTrans	Variabel totTrans	totTrans = totTrans + nTrans	Hasil totTrans	Mengisi angka lagi ? (Y/N)
Input 10	totTrans = 0	totTrans = 0 + 10	10	Yes
Input 12	totTrans = 10	totTrans = 10 + 12	22	Yes
Input 15	totTrans = 22	totTrans = 22 + 15	37	No
Proses berhenti saat tidak ada angka yang diisi lagi pada kolom Nilai Transaksi.				



Gambar 6.8 *Flowchart Menu Akumulasi Penjumlahan (Akumulator)*

Gambar 6.8. tersebut jika diidentifikasi ke dalam komponen pengolahan data menjadi seperti berikut ini:

a. Komponen masukan

Pada komponen masukan ada 3 instruksi yang dituliskan menjadi satu, yaitu: *Input* nTrans, tampilHasil, LAGI.

Maknanya, pengguna diminta memasukkan atau memberi nilai pada 3 buah variabel, yaitu: nTrans, tampilHasil, LAGI melalui peralatan masukan.

b. Komponen proses

Pada komponen proses ada satu pernyataan matematika di setiap *pseudocode* berikut ini.

1) Deklarasi Awal: totTrans = 0.
2) Penjumlahan: totTrans = totTrans + nTrans
3) Pemberian nilai LAGI= *false*
4) Pemberian nilai tampilHasil = *false*

Terdapat pula logika matematika yang digunakan yaitu pada proses percabangan:

- Apakah mengisi angka lagi? (jawaban *Y/N*) → LAGI = *true OR* tampilHasil = *true*
- Jika tampilHasil = *true*

c. Komponen keluaran

Pada komponen keluaran ada satu instruksi, yaitu: Print nTrans.

Maknanya adalah, mesin komputer diminta menampilkan nilai yang disimpan atau ditampung pada variable nTrans.

Setelah membuat *flowchart* selesai Langkah selanjutnya adalah membuat *pseudocode*. Adapun bentuk *pseudocode* dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Akumulasi Penjumlahan dapat dilihat pada tabel di bawah ini.

Pseudocode	Bahasa Pascal

akumulator; deklarasi variable;
nTrans, totTrans: numerik; LAGI,
tampilHasil: Boolean;
 	
Begin
totTrans =0;
input nTrans, LAGI, tampilHasil;
While (LAGI = TRUE) or
(tampilHasil=true) do
totTrans= totTrans +nTrans; Print
nTrans.
If tampilHasil = true then
	var
	nTrans: integer; totTrans: integer;
	LAGI: bool; lagi_str: string;
	
	begin nTrans := 0;
	totTrans := 0; LAGI = true;
	while (LAGI = true) begin
	write (' Masukkan nTrans = ');

Pseudocode	Bahasa Pascal

Begin
Print hasil; tampilHasil=false;
LAGI=false;
End;
Else begin
input nTrans; LAGI,
tampilHasil;
End;
 	
End;
End
	readln(nTrans);
	totTrans := totTrans + nTrans;
	writeln('totTrans = ' , totTrans);
	write(' Masukkan nTrans lagi (y/n)
	? ');
	readln(Lagi_str);
	If (lagi_str.compareTo('n') = 0)
	begin
	LAGI := false; end;
	end; end;

Pseudocode	Bahasa C++

akumulator; deklarasi variable;
nTrans, totTrans: numerik; LAGI,
tampilHasil: Boolean;
 	
Begin
totTrans =0;
input nTrans, LAGI, tampilHasil;
While (LAGI = TRUE) or
(tampilHasil=true) do
totTrans= totTrans +nTrans; Print
nTrans.
If tampilHasil = true then Begin
Print hasil; tampilHasil=false;
LAGI=false;
End;
Else begin
input nTrans; LAGI,
tampilHasil;
End;
 	
End;
End
	#include <stdio.h> using namespace
	std; int main() {
	int nTrans = 0; int totTrans = 0;
	boolean LAGI = true; String
	lagi_str;
	
	while (LAGI == true) {
	cout << " Masukkan nTrans = "<<
	endl;
	cin >> nTrans;
	totTrans = totTrans + nTrans;
	cout << "totTrans =
	" << totTrans<<endl;
	cout << " Masukkan nTrans lagi (y/n)
	? "<< endl;
	cin >> Lagi_str;
	If(lagi_str.compareTo("n") == 0) {
	LAGI = false;
	}
	}
	return 0;
	}

Pseudocode	Bahasa Java

akumulator; deklarasi variable;
nTrans, totTrans: numerik; LAGI,
tampilHasil: Boolean;
 	
Begin
totTrans =0;
input nTrans, LAGI, tampilHasil;
While (LAGI = TRUE) or
(tampilHasil=true) do
totTrans= totTrans +nTrans; Print
nTrans.
	Package Akumulator; Import
	java.util.Scanner;
	
	public class AkumulatorJumlah {
	public static void main(String[]
	args)
	{
	int nTrans = 0; int totTrans = 0;
	boolean LAGI = true; String
	lagi_str;
	
	while (LAGI == true) {

Pseudocode	Bahasa Java

If tampilHasil = true then Begin
Print hasil; tampilHasil=false;
LAGI=false;
End;
Else begin
input nTrans; LAGI,
tampilHasil;
End;
 	
End;
End
	Scanner sc =
	new Scanner(System.in);
	System.out.println("
	Masukkan nTrans = ");
	nTrans = sc.nextLine(); totTrans =
	totTrans + nTrans;
	System.out.println("totTrans = "
	+ totTrans);
	Scanner again = new
	Scanner(System.in);
	System.out.println(" Masukkan
	nTrans lagi (y/n) ? ");
	Lagi_str = sc.next();
	If(lagi_str.compareTo("n") == 0) {
	LAGI = false;
	}
	}
	}
	}

6.5 Multiplikator

Multiplikator adalah proses perkalian berulang secara berurutan, hasil perkalian dikalikan dengan dengan berikutnya. Simbol secara matematisnya adalah sebagai berikut.

$$S = \prod_{i=1}^{n} x_i$$

Nilai variabel S diperoleh dengan cara melakukan perkalian nilai x dari 1 sampai ke-...

Contoh 1, untuk n=5 dan nilai x adalah sebagai berikut.

n	x
1	10
2	35
3	25
4	20
5	40

Maka:

$$S = \prod_{i=1}^{n} = 10 * 35 * 25 * 20 * 40 = 7.000.000$$

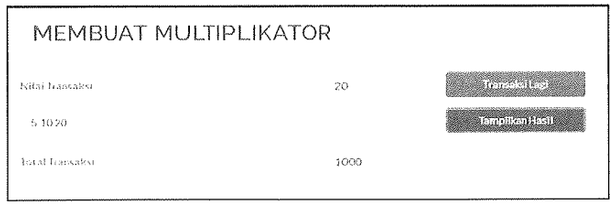
Contoh 2, untuk n=7 dan nilai x adalah sebagai berikut:

n	x
1	2
2	-3
3	2
4	0
5	40
6	10
7	10

Maka

$$S = \prod^{n} = 2 * (-3) * 2 * 0 * 40 * 10 * 10 = 0$$

Contoh implementasi Akumulasi Perkalian pada Aplikasi Belajar Dasar-dasar Algoritma Komputasi: Menu Pernyataan Matematika - Operasi Aritmatika (Akumulasi Perkalian), dapat dilihat pada Gambar 6.9.



Gambar 6.9 *Menu Multiplikator pada Penalaran.com*

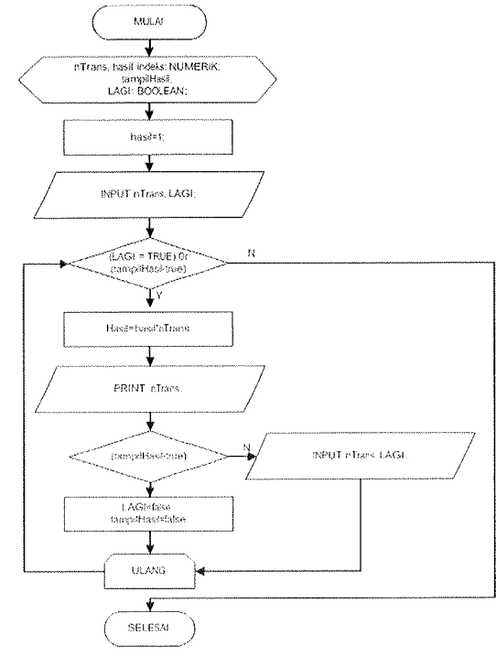
Perhatikan gambar 6.9. Bentuk inputan data masukannya sama dengan tampilan Akumulator. Pengguna diharuskan memasukkan data nilai pada kolom Nilai Transaksi (misal input angka 5), kemudian dilanjutkan dengan menekan tombol Transaksi Lagi, apabila diinginkan untuk menginputkan kembali. Angka 5 tersebut akan muncul di kolom bagian bawahnya. Pengguna dapat menginputkan data nilai lebih dari satu (misalnya *input* angka 10 dan 20), kemudian tekan tombol Transaksi Lagi. Jika telah muncul pada kolom bagian bawahnya yaitu 5, 10, dan 20, pengguna dapat mengakhiri proses dengan menekan tombol Tampilkan Hasil. Maka pada kolom Total Transaksi akan muncul nilai 1000. Nilai ini didapatkan dengan proses perkalian secara berulang yaitu 5×10×20 = 1000.

Pada gambar *Flowchart* 6.10, dapat dilihat bahwa terdapat 4 buah variavel yang harus dideklarasikan yaitu variabel (1) nTrans dan (2) hasil bertipe Numerik (karena inputan data berupa angka) serta variabel (3) TampilHasil dan (4) Lagi bertipe Boolean (karena penekanan tombol bersifat *true/false*). Variabel totTrans awalnya diisi dengan angka 1 (satu). Kemudian sistem aplikasi akan menerima inputan dari nTrans, dan LAGI. Jika LAGI bernilai true atau tampilHasil bernilai *true*, maka penjumlahan variabel hasil = hasil * nTrans dilakukan. Setelah itu proses cetak/*print* nTrans dilakukan.

Namun, di bagian tengah perulangan, terdapat pengecekan kembali variabel tampilHasil apakah bernilai true. Jika ya, maka variabel LAGI disetting *false* dan tampilHasil disetting *false*. Sebaliknya, apabila tampilHasil tidak bernilai true (false), maka lakukan input data kembali untuk variabel nTrans dan LAGI. Proses ini akan berulang terus dengan batassan bahwa variabel LAGI bernilai *false* atau variabel tampilHasil bernilai *false*.

Untuk memudahkan proses logikanya, pengguna dapat menganalisis melalui tabel penalaran/ *tracing* berikut ini:

Variabel nTrans	Variabel hasil	Instruksi Komputasi hasil = hasil * nTrans	Nilai Hasil	Mengisi angka lagi ? (Y/N)
Input 5	hasil = 1	hasil = 1 * 5	5	Yes
Input 10	hasil = 5	hasil = 5 * 10	50	Yes
Input 20	hasil = 50	hasil = 50 * 20	1000	No
Proses berhenti saat tidak ada angka yang diisi lagi pada kolom Nilai Transaksi.				



Gambar 6.10 *Flowchart Menu Akumulasi Perkalian (Multiplikator)*

Gambar 6.10. tersebut jika diidentifikasi ke dalam komponen pengolahan data menjadi seperti berikut ini:

a. Komponen masukan.
   Pada komponen masukan ada 3 instruksi yang dituliskan menjadi satu, yaitu: Input nTrans, tampilHasil, LAGI.

   Maknanya, pengguna diminta memasukan atau memberi nilai pada 3 buah variabel, yaitu: nTrans, tampilHasil, LAGI melalui peralatan masukan.

b. Komponen proses

Pada komponen proses ada satu pernyataan matematika di setiap *pseudocode* berikut ini.

1) Deklarasi Awal: hasil = 1.
2) Perkalian: hasil = hasil * nTrans
3) Pemberian nilai LAGI= *false*
4) Pemberian nilai tampilHasil = *false*

Terdapat pula logika matematika yang digunakan yaitu pada proses percabangan:

- Apakah mengisi angka lagi? (jawaban *Y/N*) → LAGI = *true OR* tampilHasil = *true*
- Jika tampilHasil = *true*

c. Komponen keluaran

Pada komponen keluaran ada satu instruksi, yaitu: Print nTrans.

Maknanya adalah, mesin komputer diminta menampilkan nilai yang disimpan atau ditampung pada variable nTrans.

Setelah membuat *flowchart* selesai Langkah selanjutnya adalah membuat *pseudocode*. Adapun bentuk *pseudocode* dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Akumulasi Perkalian dapat dilihat pada tabel di bawah ini.

Pseudocode	Bahasa Pascal

multiplikator; deklarasi variable;
nTrans, hasil: numerik; LAGI,
tampilHasil: Boolean;
 	
Begin
hasil=1;
input nTrans, LAGI, tampilHasil;
While (LAGI = TRUE) or
(tampilHasil=true) do
hasil=hasil*nTrans; Print nTrans.
If tampilHasil = true then Begin
	var
	nTrans: integer; totTrans: integer;
	LAGI: bool; lagi_str: string;
	
	begin nTrans := 0;
	totTrans := 0; LAGI := true;
	while (LAGI = true) do begin
	write (' Masukkan nTrans = ');
	readln(nTrans);

Pseudocode	Bahasa Pascal

Print hasil; tampilHasil=false;
LAGI=false;
End;
Else begin
input nTrans; LAGI,
tampilHasil;
End;
 	
End;
End
	totTrans := totTrans * nTrans;
	writeln('totTrans = ' , totTrans);
	write(' Masukkan nTrans lagi
	(y/n) ? ');
	readln(Lagi_str);
	If (lagi_str.compareTo('n') = 0)
	begin
	LAGI := false; end;
	end; end;

Pseudocode	Bahasa C++

multiplikator; deklarasi variable;
nTrans, hasil: numerik; LAGI,
tampilHasil: Boolean;
 	
Begin
hasil=1;
input nTrans, LAGI, tampilHasil;
While (LAGI = TRUE) or
(tampilHasil=true) do
hasil=hasil*nTrans; Print nTrans.
If tampilHasil = true then Begin
Print hasil; tampilHasil=false;
LAGI=false;
End;
Else begin
input nTrans; LAGI,
tampilHasil;
End;
 	
End;
End
	#include <stdio.h> using
	namespace std; int main() {
	int nTrans = 0; int totTrans = 0;
	boolean LAGI = true; String
	lagi_str;
	
	while (LAGI == true) {
	cout << " Masukkan nTrans =
	"<< endl;
	cin >> nTrans;
	totTrans = totTrans * nTrans;
	cout << "totTrans =
	" << totTrans<<endl;
	cout << " Masukkan nTrans lagi
	(y/n)
	? "<< end';
	cin >> Lagi_str;
	If(lagi_str.compareTo("n") == 0) {
	LAGI = false;
	}
	}
	return 0;
	}

Pseudocode	Bahasa Java

multiplikator; deklarasi variable;
nTrans, hasil: numerik; LAGI,
tampilHasil: Boolean;
 	
Begin
hasil=1;
input nTrans, LAGI, tampilHasil;
While (LAGI = TRUE) or
(tampilHasil=true) do
hasil=hasil*nTrans; Print nTrans.
	Import java.util.Scanner;
	
	public class AkumulatorJumlah {
	public static void main(String[]
	args)
	{
	int nTrans = 0; int totTrans = 0;
	boolean LAGI = true; String
	lagi_str;

Pseudocode	Bahasa Java

Begin
Print hasil; tampilHasil=false;
LAGI=false;
End;
Else begin
input nTrans; LAGI,
tampilHasil;
End;
 	
End;
End
	System.out.println(" Masukkan nTrans = ");
	nTrans = sc.nextLine(); totTrans = totTrans * nTrans;
	System.out.println("totTrans = " + totTrans);
	Scanner again = new Scanner(System.in);
	System.out.println(" Masukkan nTrans lagi (y/n) ? ");
	Lagi_str = sc.next();
	If(lagi_str.compareTo("n") == 0) {
	LAGI = false;
	}
	}
	}}

6.6 Faktorial

Faktorial adalah proses perkalian berulang secara berurutan, hasil perkalian dikalikan dengan bilangan berikutnya, dengan nilai bilangan otomatis, dimulai dari nol sampai dengan n (sesuai kebutuhan). Simbol secara matematisnya adalah sebagai berikut.

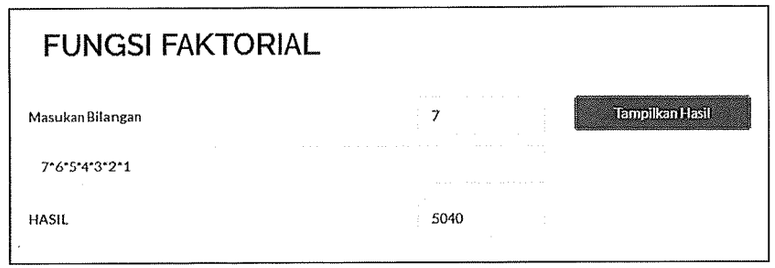
$$faktor(n) = \prod_{i=1}^{n} i$$

Nilai fungsi *fakto(n)* diperoleh dengan cara melakukan perkalian bilangan 1 sampai dengan n.

Contoh fungsi *fakto(n):*

n	*fakto(n)*
1	1
2	1*2=2
3	1*2*3=6
4	1*2*3*4=24
...	.....
n	1*2*3*.. .*n

Contoh implementasi Perhitungan Faktorial pada Aplikasi Belajar Dasar-dasar Algoritma Komputasi: **Menu Pernyataan Matematika - Operasi Aritmatika (Perhitungan Faktorial).**



Gambar 6.11 *Menu Faktorial pada Penalaran.com*

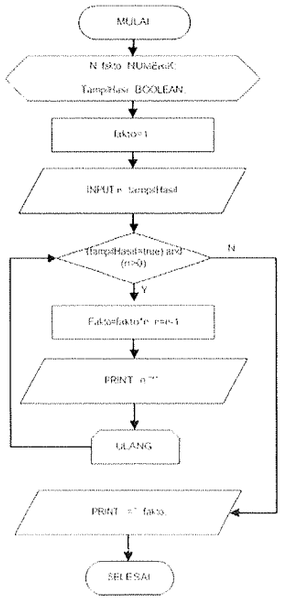
Pada gambar 6.11 terdapat satu inputan Masukkan Bilangan. Masukkan Bilangan merupakan data *input*an/masukan yang harus diisi dengan angka. Setelah mengisi sebuah angka (misalnya *input* 7), tekan tombol Tampilkan Hasil yang ada di sebelah kanannya. Hasilnya akan muncul pada kolom di bawahnya uraian perhitungan faktorial, bahwa faktorial 7 bisa didapatkan dari 7*6*5*4*3*2*1. Hasil perkalian tersebut ditampilkan pada kolom bagian bawahnya lagi yaitu 5040.

Bagaimana hanya *input* angka 7, namun dapat menghasilkan 7*6*5*4*3*2*1. Hal ini menggunakan konsep deret perulangan dengan kounter pengurang 1. Angka 7 dikurangi 1 sama dengan 6. Angka 6 dikurangi 1 sama dengan 5. Lakukan perulangan ini secara terus menerus hingga batas akhir sama dengan 1. Maka akan menghasilkan deret 7, 6, 5, 4, 3, 2, 1 Untuk dapat menghasilkan nilai perkalian deret, sistem aplikasi menggunakan variabel fakto dengan rumus fakto = fakto * n. Variabel *n* berisi angka yang diinputkan atau yang akan dicari faktorialnya (input angka 7).

Untuk memudahkan proses logika berpikirnya, pengguna dapat menganalisis melalui tabel penalaran/*tracing* berikut ini:

Variabel awal fakto	Input Variabel *n*	Input tampil Hasil (tekan tombol/tidak)	Kondisi tampilHasil = true AND N > 0	fakto = fakto * n	Print *n*	*n* = *n*-1	Print Hasil fakto
fakto=1	Input 7	true	tampilHasil=true AND 7>0 → Yes, maka berulang	fakto=1*7 =7	Cetak 7	N = 7-1=6	-

Variabel awal fakto	Input Variabel *n*	Input tampil Hasil (tekan tombol/tidak)	Kondisi tampilHasil = true AND N > 0	fakto = fakto * *n*	Print *n*	*n* = *n*-1	Print Hasil fakto
-	-	-	tampilHasil=true AND 6>0 → Yes, maka berulang	fakto=7*6 =42	Cetak 6	N = 6-1=5	-
-	-	-	tampilHasil=true AND 5>0 → Yes, maka berulang	fakto=42*5 =210	Cetak 5	N = 5-1=4	-
-	-	-	tampilHasil=true AND 4>0 →Yes,	fakto=210 *4	Cetak 4	-	-
-	-	-	tampilHasil=true AND 3>0→Yes, maka berulang	fakto=840 *3 =2520	Cetak 3	-	-
-	-	-	tampilHasil=true AND 3>0→Yes, maka berulang	fakto=252 0*2 =5040	Cetak2	-	-
-	-	-	tampilHasil=true AND 3>0 →Yes, maka berulang	fakto=504 0*1 =5040	Cetak 1	-	-
-	-	-	tampilHasil=true AND 0>0→NO, maka berulang berhenti	-	-	-	-



Gambar 6.12 *Flowchart Faktorial pada Penalaran.com*

Pada gambar *Flowchart* 6.12 dapat dilihat bahwa *flowchart* faktorial menggunakan konsep perulangan. Dengan dilengkapi 3 buah variabel yaitu (1) variabel *n* untuk menampung inputan pengguna, angka berapa yang ingin dicari nilai faktorialnya. (2) Variabel fakto untuk menampung hasil perkaliannya. (3) Variabel tampilHasil untuk menampung nilai boolean dari tombol Tampilkan Hasil ditekan, sehingga secara default bernilai true. Variabel fakto awalnya dideklarasikan berisi angka 1 (satu). Kemudian input n dan tampilHasil. Pada saat perulangan, terjadi pengecekan kondisi, apakah tampilHasil=true AND n > 0. Jika kondisi terpenuhi, maka dilakukan perhitungan variabel fakto = fakto * *n*. Untuk menampilkan deret angka faktorial di layar 7*6*5*4*3*2*1, variabel *n* harus dikurangi 1 dengan menggunakan operasi aritmatika *n* = *n*-1.

Selain itu, Gambar *Flowchart* tersebut jika diidentifikasi ke dalam komponen pengolahan data menjadi seperti berikut ini:

a. Komponen masukan
   Pada komponen masukan ada 2 instruksi yang dituliskan menjadi satu, yaitu: *Input n*, tampilHasil.
   Maknanya, pengguna diminta memasukan atau memberi nilai pada 2 buah variabel, yaitu: n, tampilHasil melalui peralatan masukan.
b. Komponen proses
   Pada komponen proses ada satu pernyataan matematika di setiap *pseudocode* berikut ini.

   1) Deklarasi Awal: fakto = 1.
   2) Perkalian: fakto = fakto* *n*
   3) Pengurangan *n* = *n*-1

   Terdapat pula logika matematika yang digunakan yaitu pada proses percabangan:

   - Jika tampilHasil = *true* AND n>0

c. Komponen keluaran
   Pada komponen keluaran ada satu instruksi, yaitu: Print fakto. Maknanya adalah, mesin komputer diminta menampilkan nilai yang disimpan atau ditampung pada variabel fakto.

Setelah membuat *flowchart* selesai Langkah selanjutnya adalah membuat *pseudocode*. Adapun bentuk *pseudocode* dan konversi ke 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Faktorial dapat dilihat pada tabel di bawah ini.

Pseudocode	Bahasa Pascal

faktorial; deklarasi variable;
n, fakto: numerik;
tampilHasil: Boolean;
 	
Begin
fakto=1;
input n, tampilHasil;
While (tampilHasil=true) and (n>0)
do
fakto=fakto*n; n=n-1;
Print n,"*";
End;
Print "=", fakto;
End.
	var
	fakto: integer; n : integer;
	begin
	fakto := 1;
	n := 3;
	while (n > 0) begin
	fakto := fakto*n; if (n>1)
	begin
	writeln(n , '*'); end;
	else begin
	writeln(n);
	end;
	n := n-1;
	end;
	writeln('=' , fakto); end;

Pseudocode	Bahasa C++

faktorial; deklarasi variable;
n, fakto: numerik; tampilHasil:
Boolean;
 	
Begin
fakto=1;
input n, tampilHasil;
While (tampilHasil=true) and (n>0)
do
fakto=fakto*n; n=n-1; Print n,"*";
End;
Print "=", fakto;
End.
	#include <stdio.h> using namespace
	std; int main() {
	int fakto = 1; int n = 3;
	
	while (n > 0) { fakto = fakto*n;
	if (n>1) cout << n << "*"<<endl;
	else cout << n << endl;
	n--;
	}
	cout << "=" << fakto<<endl;
	}
	}

Pseudocode	Bahasa Java

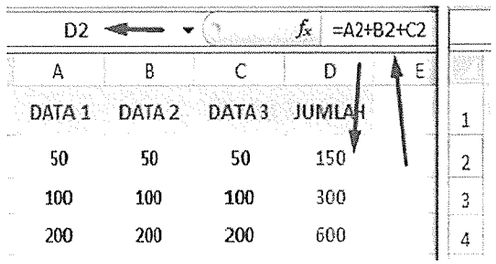
faktorial; deklarasi variable;
n, fakto: numerik; tampilHasil:
Boolean;
 	
Begin
fakto=1;
input n, tampilHasil;
While (tampilHasil=true) and (n>0)
do
fakto=fakto*n; n=n-1; Print n,"*";
End;
Print "=", fakto;
End.
	public class Faktorial {
	public static void main(String[]
	args)
	{
	int fakto = 1; int n = 3;
	
	while (n > 0) { fakto = fakto*n;
	if (n>1) System.out.println(n +
	"*"); else System.out.println(n);
	n--;
	}
	System.out.println("=" + fakto);
	}
	}

6.7 Rangkuman

- Akumulator adalah proses penjumlahan berulang dengan menambahkan secara berurutan hasil penjumlahan dengan bilangan berikutnya.
- Multiplikator adalah proses perkalian berulang secara berurutan, hasil perkalian dikalikan dengan berikutnya.
- Faktorial adalah proses perkalian berulang secara berurutan, hasil perkalian dikalikan dengan bilangan berikutnya, dengan nilai bilangan otomatis, dimulai dari nol sampai dengan *n* (sesuai kebutuhan).

6.8 Latihan Soal

1. Perhatikan ilustrasi gambar berikut ini:



   Apabila diberikan Data1, Data2, dan Data3, masing-masing sebanyak 3 baris. Buatlah algoritma yang tepat untuk menjumlahkan hasil akhir dari ketiga data tersebut. Kemudian cetak data *input* dan *output* yang dihasilkan pada akhir program.

2. Buatlah *pseudocode* untuk menghitung penjumlahan deret angka yang ganjil dengan batas awal = 1, batas bawah = *n* dan menampilkan hasil akhir penjumlahannya.

   Ketentuan:

   - Deret merupakan bilangan ganjil
   - *Output*, **jika n = 10** maka akan tampil sebagai berikut:

n = 10

Deret : 1 + 3 + 5 + 7+ 9 + 10
total deret = ???

3. Buatlah *pseudocode* untuk menghitung perkalian deret angka yang ganjil dengan batas awal = 1, batas bawah = *n* dan menampilkan hasil akhir perkaliannya.

   Ketentuan:

   - Deret merupakan bilangan ganjil
   - *Output*, **jika n = 10** maka akan tampil sebagai berikut :

   n = 10
   Deret : 1 × 3 × 5 × 7 × 9 × 10 total deret = ???

4. Buatlah *pseudocode* untuk menghitung penjumlahan deret angka yang genap dengan batas awal = 1, batas bawah = *n* dan menampilkan hasil akhir penjumlahannya.

   Ketentuan:

   - Deret merupakan bilangan genap
   - *Output*, **jika n = 10** maka akan tampil sebagai berikut:

   n = 10
   Deret : 2 + 4 + 6 + 8 + 10
   total deret = ???

5. Buatlah *pseudocode* untuk menghitung perkalian deret angka yang genap dengan batas awal = 1, batas bawah = *n* dan menampilkan hasil akhir perkaliannya.

   Ketentuan:

   - Deret merupakan bilangan genap
   - *Output*, **jika n = 10** maka akan tampil sebagai berikut:

   n = 10
   Deret: 2 × 4 × 6 × 8 × 10 , total deret = ???

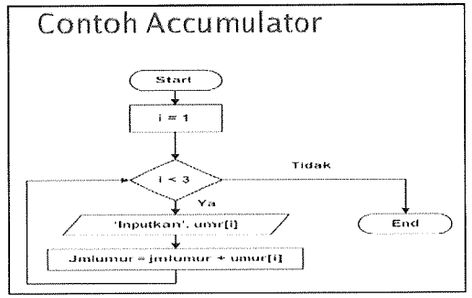
6. Perhatikan ilustrasi gambar berikut ini:

	BUAH	JUMLAH
4	PISANG	10
5	MANGGA	25
6	JERUK	13
7	RAMBUTAN	15
8	KELAPA	25
9	TOTAL	88

   Jika seorang *customer* melakukan transaksi pembelian buah di supermarket, program layanan penjualan pada kasir akan meng-inputkan sejumlah buah yang dibeli. Apabila *customer* tersebut ingin mengetahui total jumlah buah yang dibeli (*qty*), maka program tersebut harus dapat melakukan akumulasi penjumlahan buah.

   Untuk mendukung hal tersebut, Anda sebagai programmer diminta untuk membuat algoritma yang tepat untuk dapat mencetak hasil akhir jumlah buah yang dibeli.

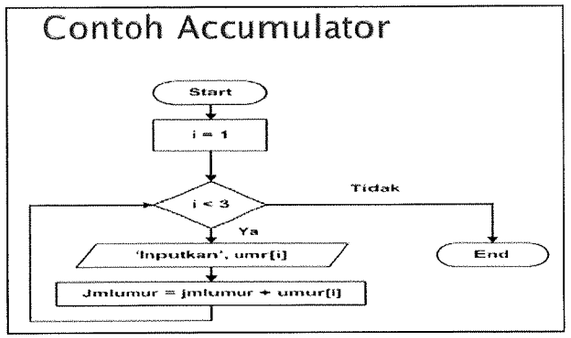
7. Perhatikan *flowchart* berikut ini:



   Berdasarkan *flowchart* di atas, tuliskan jawaban soal di bawah ini dengan benar.

   a. Berapa Nilai awal variabel I?
   b. Berapa Batas akhir perulangan variabel I?
   c. Variabel umr[*i*] merupakan jenis variabel apa?
   d. Tuliskan kembali perintah/instruksi untuk meng*input*kan sebuah nilai berdasarkan *flowchart* tersebut?
   e. Adakah pernyataan matematika yang digunakan pada *flowchart* tersebut? Jelaskan dengan contoh dan alasan.

8. Perhatikan *flowchart* berikut ini:



Berdasarkan *flowchart* di atas, jawablah pertanyaan berikut ini dengan benar.

a. Lakukan proses *input* sejumlah nilai umr[*i*] sesuai dengan keinginan Anda.
b. Cetak hasil akhir variabel Jmlumur untuk hasil penjumlahan seluruh nilai yang telah Anda inputkan pada soal poin a.

9. Perhatikan *code* program berikut ini:

```
package loops;


public class ForLoops {

    public static void main(String[] args) {

        int loopVal;
        int end_value = 11;

        for (loopVal = 0; loopVal < end_value; loopVal++) {

            System.out.println("Loop Value = " + loopVal);
        }
    }

}
```

a. Sebutkan semua variabel yang digunakan pada *code* program tersebut.
b. Cetaklah data hasil *output* apabila program tersebut dijalankan.

10. Perhatikan code program berikut ini:

```
int main()
{
  int n = 0;
  while(n < 100)
  {
    n = n + 5;
    print("n = %d\n", n);
    pause(200);
    if(n == 50) break;                    // <- add
  }
  print("All done!");
}
```

a. Sebutkan semua variabel yang digunakan pada *code* program tersebut.
b. Pernyataan matematika yang mana yang berperan sebagai kounter?
c. Sebutkan logika matematika yang digunakan pada *code* tersebut?

11. Perhatikan *code* program berikut ini:

```
int main()
{
  int n = 0;
  while(n < 100)
  {
    n = n + 5;
    print("n = %d\n", n);
    pause(200);
    if(n == 50) break;                    // <- add
  }
  print("All done!");
}
```

Cetaklah hasil akhir dari *code* program tersebut apabila dijalankan hingga selesai.

12. Perhatikan potongan code berikut ini:

```
File  Edit  Format  Run  Options  Windows  Help
def factorial (x):
    if x==1:
        return 1
    else:
        return x * factorial(x-1)
print (factorial(5))
```

a. Tulis semua variabel yang digunakan pada *code* tersebut.
b. Sebutkan logika matematika yang digunakan pada *code* tersebut.
c. Tuliskan *code* mana yang berperan sebagai pernyataan matematika.

13. Perhatikan potongan *code* berikut ini:

```
File  Edit  Format  Run  Options  Windows  Help
def factorial (x):
    if x==1:
        return 1
    else:
        return x * factorial(x-1)
print (factorial(5))
```

Berikan hasil akhir dari *code* program tersebut apabila dijalankan.

14. Pada sebuah toko swalayan, Ani menemukan promo baru yang ditawarkan oleh pihak swalayan. Promo tersebut diberlakukan untuk pembelian shampoo dengan jumlah genap, minimal pembelian 2 botol shampoo akan diberikan gratis 1 botol shampoo yang sama. Promo ini berlaku untuk kelipatan. Buatlah daftar untuk mencatat berapa jumlah shampoo gratis yang harus diberikan kepada *customer* apabila dibatasi pembelian shampoo maksimal adalah 10 botol.

2 botol	4 botol	6 botol	8 botol	10 botol	12 botol	14 botol	16 botol	18 botol	20 botol
1 botol	2 botol	3 botol	4 botol	5 botol	6 botol	7 botol	8 botol	9 botol	10 botol

Buatlah *Flowchart* dan *Pseudocode* untuk permasalahan di atas menggunakan konsep Proses Perulangan.

15. Buatlah program kasir minimart untuk mendata barang yang dibeli konsumen dengan menginputkan nama petugas kasir, banyak jenis barang, nama barang, harga dan banyaknya.

Ketentuan:

- Saat akhir pendataan beri peringatan apakah *user* ingin menginputkan data lainnya apa tidak
- Jika ya maka pendataan dilanjutkan hingga tidak ada lagi yang di data.
- Tampilkan total semua barang pembeli yang didata pada proses di atas.

Bentuk *Output*:

```
---------APRILMART---------
Petugas Kasir: James  Banyak Jenis Barang: 2

-------pesanan ke-1-------- Nama Barang  : Odol
Harga       : Rp.3500
Banyak      : 2
-------pesanan ke-2-------- Nama Barang  : Indomie  Harga : Rp.2700
Banyak      : 4

apakah ada pesanan lainnya? (0=tidak/1=Ya)  jawaban: 1
Banyak jenis barang: 1
-------pesanan ke-3-------- Nama Barang  : Pocky
Harga       : Rp.5000
Banyak      : 2

apakah ada pesanan lainnya? (0=tidak/1=Ya)  jawaban: 0

total semuanya: Rp.27800
```

-oo0o-

BAB 7

MEMBUAT *PSEUDOCODE* DARI *FLOWCHART* DAN VARIABEL ARRAY 1 DIMENSI

Deskripsi Singkat

Pada bab 7 ini akan dibahas mengenai variabel array 1 dimensi di mana variabel array sendiri berfungsi untuk menyimpan sejumlah data dengan tipe data yang sama dan memiliki index untuk mengakses setiap elemen pada variabel tersebut. Pada bagian pendahuluan akan dijelaskan perbedaan dan urgensi dari variabel array dibandingkan dengan variabel lainnya. Pembahasan Algoritma variabel array yang akan dijelaskan pada buku ajar ini meliputi pengertian array/larik, deklarasi dan inisialisasi elemen array 1D, menginputkaan dan memproses nilai elemen array 1D, menampilkan Nilai Elemen Array, penerapan variabel array untuk instruksi komputasi pada penjumlahan dan pengurangan matriks, serta penerapan perulangan pada penggunaan variabel array.

Penerapan variabel array pada komputasi Matematika meliputi proses penjumlahan dan pengurangan pada matriks di mana jumlah kolom pada matriks yaitu 2,3 atau 4 kolom. Pembaca akan diberikan gambaran bagaimana menyusun *flowchart*, *pseudocode* hingga *code* dari studi kasus yang diberikan.

Hampir sama seperti sub bab sebelumnya, bahwa pada sub bab penerapan perulangan menggunakan variabel array 1Dimensi untuk

instruksi komputasi akan membahas perintah-perintah yang berjalan pada sistem selama proses perulangan dilakukan. Hal ini juga berlaku pada perulangan dan penggunaan variabel array 1 D.

Pada sub bab terakhir terdapat pembahasan mengenai studi kasus penerapan variabel array. Sebelum akhir bab, terdapat Rangkuman dan Latihan Soal. Ditutup dengan daftar pustaka yang digunakan untuk menyusun bab 7.

Tujuan Pembelajaran

Tujuan akhir pembelajaran pada Bab 7 ini dibagi menjadi 4 bagian, antara lain:

1. Jika diberikan pernyataan menggunakan variabel array 1 Dimensi, pembaca dapat **menilai kebenaran** pernyataan matematika tersebut.
2. Jika diberi soal atau studi kasus, pembaca dapat **meng-implementasikan** pengolahan menggunakan variabel array 1 Dimensi dengan berbagai tipe data.
3. Jika diberi soal atau studi kasus, pembaca dapat **membuktikan** hasil penerapan algoritma perulangan mengguankan variabel array 1 Dimensi pada permasalahan proses bisnis.
4. Tujuan terakhir adalah pembaca dapat **menyimpulkan** perbedaan antara variabel biasa dan variabel array.

7.1 Pendahuluan

Sebuah variabel yang memiliki tipe data literal hanya bisa digunakan untuk menyimpan satu data saja. Apabila dalam sebuah studi kasus membutuhkan beberapa data dengan tipe data yan sama maka variable literal tersebut kurang efisien untuk digunakan. Sebagai contoh ketika seorang guru ingin menyimpan data nilai seluruh siswa nya yang terdiri dari 40 siswa. Dengan menerapkan variable literal berarti guru tersebut membutuhkan 40 variabel untuk menyimpan nilai dari masing-masing siswanya. Namun hal ini dapat di sederhanakan menjadi satu variabel saja bila guru tersebut memilih untuk menggunakan variabel array dalam menyimpan seluruh data nilai siswa.

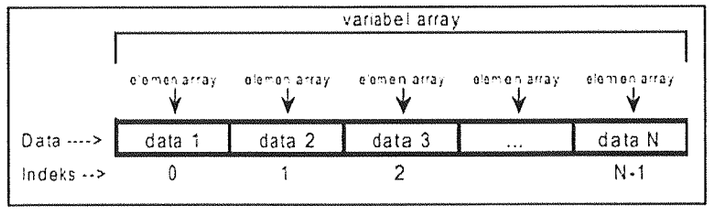
7.1.1 Pengertian Array/Larik

Array atau Larik merupakan kumpulan dari beberapa nilai yang mempunyai tipe data yang sama. Array dapat disebut sebagai sebuah variabel yang mempunyai lebih dari satu nilai. Array merupakan sejumlah variabel berbeda tetapi memiliki nama variabel yang sama. Perbedaan antara variabel yang sama pada array adalah indeksnya. Array disebut juga sebagai Larik/Tabel/Vektor/Variabel Majemuk.

Ada beberapa hal penting yang menjadi ciri-ciri dari sebuah array, antara lain sebagai berikut:

a. Array dibagi menjadi 2 bentuk dimensi, yaitu array satu dimensi dan array dua dimensi.
b. Variabel array dapat terdiri dari satu atau beberapa elemen.
c. Nilai yang mengisi setiap elemen tersebut harus spesifik, yaitu mempunyai tipe data yang sama.
d. Setiap elemen pada sebuah array akan ditandai dengan sebuah indeks. Indeks tersebut berfungsi sebagai nomor urut elemen.
e. Semua proses yang terjadi pada sebuah array, mulai dari proses pengisian nilai elemen array, proses perhitungan, maupun proses penampilan data array dapat menggunakan 2 cara, yaitu melalui proses inisialisasi langsung atau menggunakan proses *looping for*.

Indeks adalah sebuah angka yang menyatakan urutan elemen pada suatu variabel array 1 dimensi. Data merupakan nama variabel array. Elemen array merupakan isi dari array. Dan Indeks selalui dimulai dari indeks ke-0 hingga (N-1) untuk elemen terakhir, di mana N adalah jumlah total elemen array. Berikut konsep dasar dari array:



Gambar 7.1 *Indeks pada Array*

7.1.2 Deklarasi Array 1D

Variabel array dideklarasikan dengan menerapkan operator dimension []. Berikut contoh pendeklarasian array:

```
Var Nama_variabel_array : Array [jangkauan] Of Tipe_Data
```

Contoh :

```
Var nilaiSiswa : Array [0..9] Of Integer
```

Gambar 7.2 *Deklarasi Array*

7.1.3 Cara Inisialisasi Elemen Array 1D

Sebuah array dapat diinisialisasi ketika array tersebut dideklarasikan. Setelah melakukan proses deklarasi array, langkah selanjutnya yang harus dilakukan adalah mengisi nilai elemen array. Berikut contoh cara inisialisasi elemen array:

```
Var Nama_variabel_array : Array [jangkauan] Of Tipe_Data =
(nilai_elemen_array)
```

Contoh :

```
Var nilaiSiswa : Array [0..4] Of Integer = (30, 95, 75, 63, 100)
```

Gambar 7.3 *Inisialisasi Aray*

7.1.4 Cara Menginputan Nilai Elemen Array 1D

Konsep perulangan (*looping*) yang paling mudah digunakan adalah menggunakan konsep perulangan bentuk pertama, yaitu FOR. Jika pengisian nilai elemen array dilakukan menggunakan looping FOR, maka hanya membutuhkan 3 baris notasi algoritma. Perhatikan penulisan notasi algoritma untuk pembacaan inputan *user array* satu dimensi pada Gambar berikut:


```
FOR nilai_awal TO nilai_akhir DO
    INPUT (Nama_Var_Array[indeks_ke]
END FOR
```

Contoh :

```
FOR i = 0 TO 9 DO
    INPUT (nilaiSiswa[i])
END FOR
```

Gambar 7.4 *Input Elemen Array*

7.1.5 Cara Pemrosesan Nilai Elemen Array 1D

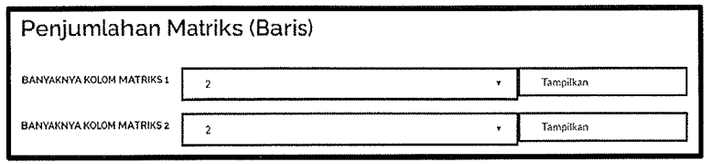
Cara memproses nilai elemen array merupakan tahap lanjutan setelah proses penginputan nilai elemen array. Pada dasarnya, tahapan paling mudah pada array adalah pengisian nilai elemen array dilanjutkan dengan proses menampilkan nilai elemen array, tanpa adanya pemrosesan lebih lanjut terhadap nilai elemen array. Namun, pemrosesan nilai elemen array mempunyai banyak manfaat yang dapat diambil, terutama dapat diaplikasikan terhadap semua kebutuhan pengguna. Satu hal yang penting dan harus diingat pada pemrosesan array satu dimensi adalah, selalu gunakan konsep perulangan (*looping*) bentuk FOR.

7.2 Penyajian Instruksi dalam Komputasi

Sesuai dengan pengertian array yaitu tipe data yang terstruktur yang berfungsi untuk menampung beberapa data yang bertipe sama. Di mana array terdiri dari beberapa elemen dan masing-masing elemen dapat diakses menggunakan indeks array. Array biasa digunakan untuk mengimplementasikan tabel yang terdiri dari baris dan kolom. Sehingga dalam implementasinya array dapat digunakan untuk melakukan komputasi matriks.

7.2.1 Operasi Penjumlahan Matriks

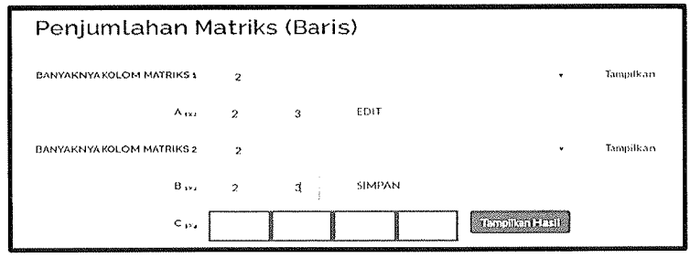
Pada gambar 7.5 menunjukkan masukan atau inputan untuk menentukan jumlah kolom matriks 1 dan jumlah kolom pada matriks 2. Pada gambar ditampilkan masing-masing kolom diinputkan dengan nilai 2 sehingga dari 2 kolom matriks yang tersedia, masing-masing memiliki 2 nilai.



Gambar 7.5 *Inputan Matriks Penjumlahan*

Pada gambar 7.6 menunjukkan proses menginputkan nilai untuk masing-masing kolom pada matriks. Dapat dilihat nilai yang di*input*kan adalah 2 dan 3 untuk kolom 1 dan kolom dua sehingga matriks yang terbentuk adalah 2×2.

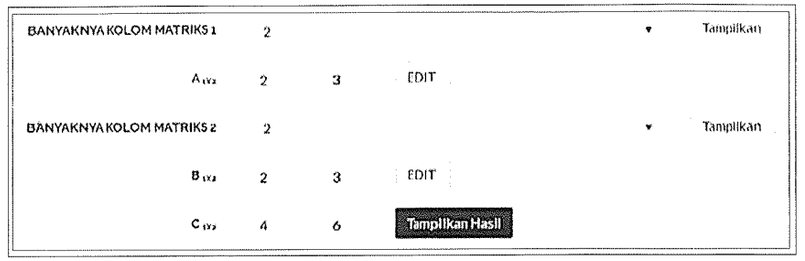
Matriks :	2	2
	3	3



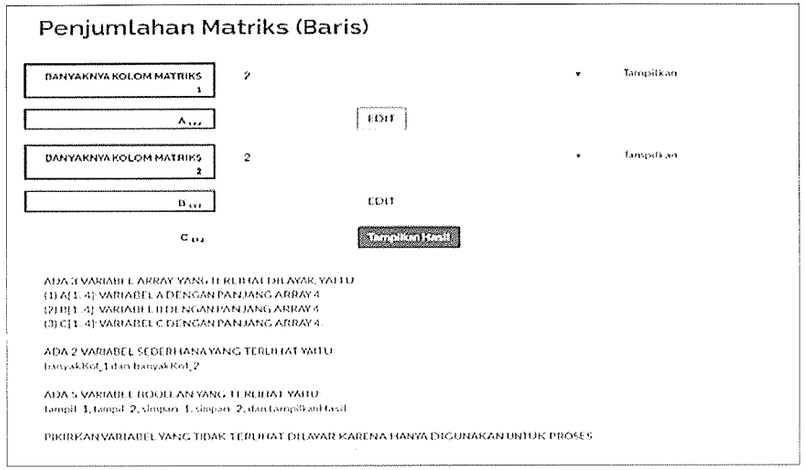
Gambar 7.6 *Output Matriks Penjumlahan*

Pada gambar 7.7 menunjukkan hasil komputasi penjumlahan untuk matriks dua kolom yang sudah di*input*kan pada tahap sebelumnya. Dari proses penjumlahan diperoleh hasil yaitu 4 dan 6 untuk penjumlahan 2 kolom matriks dengan nilai masing-masing kolom adalah 2 dan 3.

Matriks :	kol.1	kol.2	hasil
	2	2	4
	3	3	6



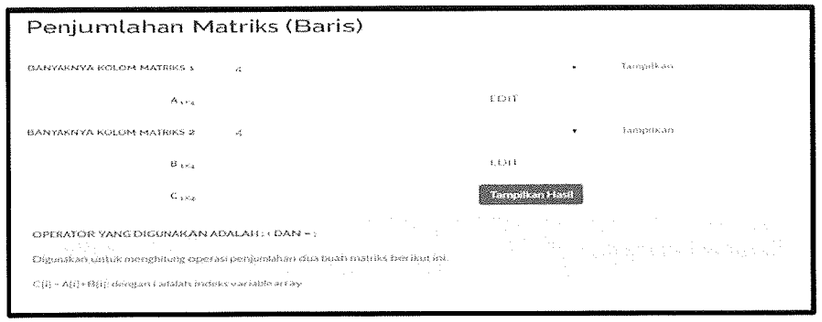
Gambar 7.7 *Hasil Matriks Penjumlahan*



Gambar 7.8 *Data pada Matriks Pnjumlahan*

Gambar 7.8 menunjukkan tampilan dari penjumlahan matriks. Banyaknya variabel yang digunakan untuk menjumlahkan matriks terdiri dari beberapa variabel yaitu:

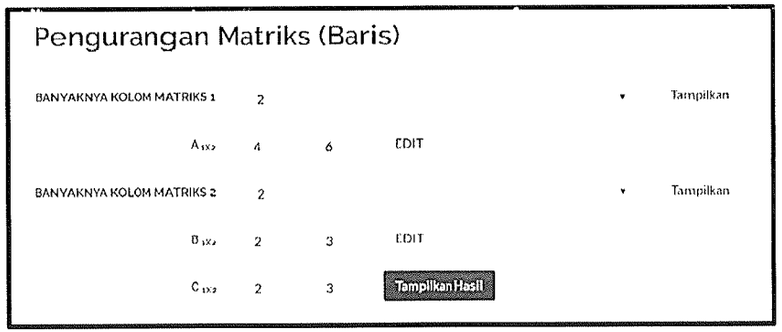
1. Terdapat 3 variabel array yaitu A,B dan C di mana masing-masing memiliki 4 elemen (panjang array adalah 4)
2. Terdapat 2 variabel sederhana untuk menampung kolom1 dan kolom 2



Gambar 7.9 *Operator Pada Matriks Penjumlahan*

Terdapat 5 variabel boolean untuk menampilkan *input*an dan hasil serta menyimpan proses. Ada 2 operator yang digunakan untuk melakukan komputasi penjumlahan matriks yaitu menggunakan operator '+' dan '='. Sebagai contoh yaitu C[i] =A[i] + B[i] di mana i adalah indeks variabel array untuk mengakses setiap elemen array.

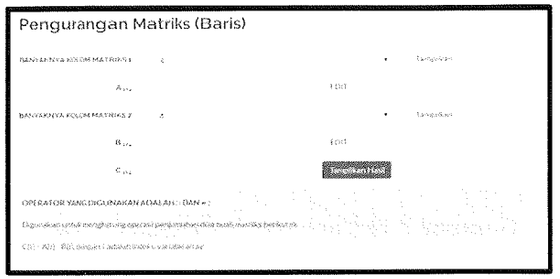
7.2.2 Operasi Pengurangan Matriks



Gambar 7.10 *Hasil Pengurangan Matriks*

Pada gambar 7.10 menunjukkan hasil komputasi pengurangan untuk matriks dua kolom yang sudah diinputkan pada tahap sebelumnya. Dari proses penjumlahan diperoleh hasil yaitu 2 dan 3 untuk penjumlahan 2 kolom matriks dengan nilai masing-masing kolom adalah 4 dan 6 untuk kolom 1 serta 2 dan 3 untuk kolom 2.

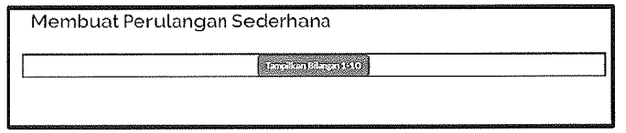
Matriks :	kol.1	kol.2	hasil
	4	2	2
	6	3	3



Gambar 7.11 *Operator pada Matriks Pengurangan.*

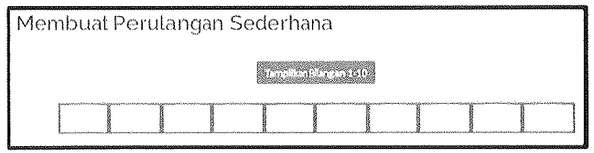
Ada 2 operator yang digunakan untuk melakukan komputasi pengurangan matriks yaitu menggunakan operator '-' dan '='. Sebagai contoh yaitu C[i] =A[i] + B[i] di mana i adalah indeks variabel array untuk mengakses setiap elemen array. Hal ini dapat dilihat pada gambar 7.11.

7.2.3 Konsep Perulangan Array



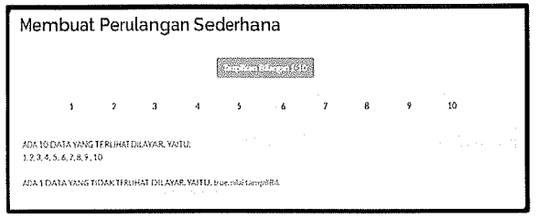
Gambar 7.12 *Masukan Perulangan Array*

Gambar 7.12 menunjukkan tampilan button Tampilkan bilangan 10 di mana hal ini merupakan variabel boolean yaitu tampilBil. Variabel tampilBil ini juga bernilai *true*, namun tidak terlihat pada layar komputer dikarenakan diproses otomatis oleh sistem.



Gambar 7.13 *Keluaran Perulangan Array*

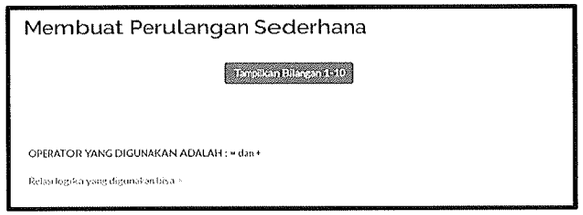
Gambar 7.13 menunjukkan hasil keluaran dari proses perulangan dimulai dari bilangan 1 hingga bilangan 10.



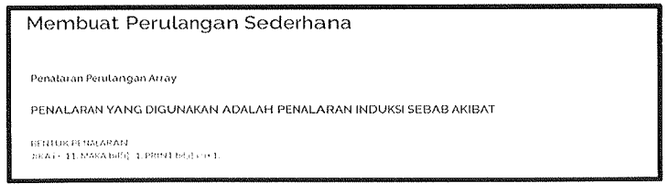
Gambar 7.14 *Data pada Perulangan Array*

Berdasarkan Gambar 7.14, ditampilkan deret angka dengan kounter penambah 1. Selain itu, terdapat 1 variabel sederhana yang terlihat di layar yaitu *i* dan 1 variabel boolean yaitu tampilBil. Variabel tampilBil ini juga bernilai *true,* namun tidak terlihat pada layar komputer dikarenakan diproses otomatis oleh sistem. Sedangkan bentuk penalaran yang digunakan adalah jika $i < 11$, maka print *i*, *i*=*i*+1.

Ada 2 operator yang digunakan untuk melakukan perulangan array 1 Dimensi yaitu menggunakan operator '+' dan '='. Hal ini dapat dilihat pada gambar 7.16.



Gambar 7.15 *Operator pada Perulangan Array*



Gambar 7. 16 *Penalaran pada Perulangan Array*

7.3 Pengenalan *Pseudocode*

7.3.1 Penjumlahan Matriks

Pseudocode

```
penjumlahanMatriks; deklarasi variable;
banyakKol_1, banyakKol_2, I: numerik;
    tampil_1, tampil_2, simpan_1, simpan_2, tampilkanHasil: boolean;
    A[1..4], B[1..4], C[1..4]: numerik;

Begin
    INPUT banyakKol_1; REPEAT
    INPUT tampil_1; UNTIL tampil_1 = TRUE;

    IF (banyakKol_1 = 2) THEN (PRINT matriks_Arr1_2)
    ELSE IF(banyakKol_1 = 3) THEN (PRINT matriks_Arr1_3) ELSE (PRINT
    matriks_Arr1_4);
    FOR i=1 TO banyakKol_1 DO BEGIN
    IF(banyakKol_1 = 2)THEN INPUT matriks_Arr1_2[i]
    ELSE IF(banyakKol_1 = 3) THEN INPUT matriks_Arr1_2[i] ELSE INPUT
    matriks_Arr1_4[i];
    END FOR

    IF simpan_1 = TRUE
    FOR i=1 TO banyakKol_1 DO
    IF(banyakKol_1 = 2) THEN A[i]=matriks_Arr1_2[i]
    ELSE IF(banyakKol_1=3) THEN A[i]=matriks_Arr1_2[i] ELSE
    A[i]=matriks_Arr1_2[i];
```

Pseudocode

```
    INPUT banyakKol_2; REPEAT
    INPUT tampil_2; UNTIL tampil_2 = TRUE;

    IF(banyakKol_2 = 2) THEN (PRINT matriks_Arr1_2)
    ELSE IF (banyakKol_2=3) THEN (PRINT matriks_Arr1_3) ELSE (PRINT
    matriks_Arr1_4);
    FOR i=1 TO banyakKol_2 DO BEGIN
    IF (banyakKol_2 = 2) THEN INPUT matriks_Arr1_2[i]
    ELSE IF(banyakKol_2 = 3) THEN INPUT matriks_Arr1_3[i] ELSE INPUT
    matriks_Arr1_4[i];
    END FOR

    IF simpan_2 = TRUE
    FOR i=1 TO banyakKol_2 DO
    IF(banyakKol_1 = 2) THEN B[i]=matriks_Arr1_2[i]
    ELSE IF(banyakKol_1 = 3)THEN B[i]=matriks_Arr1_2[i] ELSE B[i] =
    matriks_Arr1_2[i];

    IF banyakKol_1 = banyakKol_2 THEN BEGIN
    FOR i=1 TO banyakKol_1 DO C[i] = A[i] + B[i]; INPUT tampilkanHasil;
    REPEAT
    FOR i=1 TO banyakKol_1 DO PRINT C[i] UNTIL tampilkanHasil=TRUE;
    END IF
    ELSE PRINT "Matriks A tidak bisa dijumlahkan dengan Matriks B";
END
```

Adapun bentuk konversi *pseudocode* ke dalam 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Penjumlahan Matriks dapat dilihat pada tabel di bawah ini.

Operasi Penjumlahan Matriks

Pseudocode	Bahasa Pascal

penjumlahanMatriks; deklarasi variable;
banyakKol_1, banyakKol_2, I: numerik;
tampil_1, tampil_2, simpan_1, simpan_2, tampilkanHasil: boolean;
A[1..4], B[1..4], C[1..4]: numerik;
 	
Begin
INPUT banyakKol_1; REPEAT
INPUT tampil_1; UNTIL tampil_1 = TRUE;
 	
IF (banyakKol_1 = 2) THEN (PRINT
	var
	banyakKol_1: integer; banyakKol_2 : integer; i : integer;
	A : array [1..4] of integer; B : array [1..4] of integer; C : array [1..4] of integer; Ctemp : string;
	
	begin banyakKol_1 := 4;
	banyakKol_2 := 4;
	If (banyakKol_1 != banyakKol_2)
	begin
	writeln ('Matriks A tidak bisa

Pseudocode	Bahasa Pascal

matriks_Arr1_2)
ELSE IF(banyakKol_1 = 3) THEN (PRINT matriks_Arr1_3)
ELSE (PRINT matriks_Arr1_4);
FOR i=1 TO banyakKol_1 DO BEGIN
IF(banyakKol_1 = 2)THEN INPUT matriks_Arr1_2[i]
ELSE IF(banyakKol_1 = 3) THEN INPUT matriks_Arr1_2[i]
ELSE INPUT matriks_Arr1_4[i]; END FOR
 	
IF simpan_1 = TRUE
FOR i=1 TO banyakKol_1 DO
IF(banyakKol_1 = 2)
THEN A[i]=matriks_Arr1_2[i]
ELSE
IF(banyakKol_1=3) THEN
A[i]=matriks_Arr1_2[i]
ELSE
A[i]=matriks_Arr1_2[i];
INPUT banyakKol_2; REPEAT
 	
INPUT tampil_2; UNTIL tampil_2 = TRUE;
IF(banyakKol_2 = 2) THEN (PRINT matriks_Arr1_2)
ELSE IF (banyakKol_2=3) THEN (PRINT matriks_Arr1_3)
	dijumlahkan dengan matriks B'); end;
	Else begin
	for i := 0 to banyakKol_1 do begin
	write ('Masukkan Matriks A [', I,']:'); readln(A[i]);
	end; writeln (' ');
	For i :=0 to banyakKol_2 begin
	write ('Masukkan Matriks B [', I,']:'); readln(B[i]);
	end; writeln (' ');
	for i := 0 to banyakKol_1 do begin
	C[i] := A[i] + B[i];
	Ctemp := Ctemp+ '['+C[i]+']'; end;
	writeln (' Matriks C:', Ctemp); end;
	end;

Pseudocode	Bahasa Pascal

ELSE (PRINT matriks_Arr1_4);
FOR i=1 TO banyakKol_2 DO BEGIN
IF (banyakKol_2 = 2) THEN INPUT matriks_Arr1_2[i]
ELSE IF(banyakKol_2 = 3) THEN INPUT matriks_Arr1_3[i]
ELSE INPUT matriks_Arr1_4[i]; END FOR
IF simpan_2 = TRUE
FOR i=1 TO banyakKol_2 DO
IF(banyakKol_1 = 2)
THEN B[i]=matriks_Arr1_2[i]
ELSE
IF(banyakKol_1 = 3)THEN
B[i]=matriks_Arr1_2[i]
ELSE B[i] = matriks_Arr1_2[i];
 	
IF banyakKol_1 = banyakKol_2 THEN
BEGIN
FOR i=1 TO banyakKol_1 DO C[i] = A[i] + B[i];
INPUT tampilkanHasil; REPEAT
FOR i=1 TO banyakKol_1 DO PRINT C[i]
UNTIL tampilkanHasil=TRUE; END IF
ELSE PRINT "Matriks A tidak bisa dijumlahkan dengan Matriks B";
 	
END	

Pseudocode	Bahasa C++

penjumlahanMatriks; deklarasi variable;
banyakKol_1, banyakKol_2, I: numerik;
tampil_1, tampil_2, simpan_1, simpan_2, tampilkanHasil: boolean;
A[1..4], B[1..4], C[1..4]: numerik;
 	
Begin
INPUT banyakKol_1; REPEAT
INPUT tampil_1; UNTIL tampil_1 = TRUE;
 	
IF (banyakKol_1 = 2) THEN (PRINT matriks_Arr1_2)
ELSE IF(banyakKol_1 = 3) THEN (PRINT matriks_Arr1_3)
ELSE (PRINT matriks_Arr1_4);
	#include <stdio.h> using namespace std; int main() {
	int banyakKol_1 = 4; int banyakKol_2 = 4; int i;
	int A[4];
	int B[4];
	int C[4];
	String Ctemp="";
	If (banyakKol_1 != banyakKol_2) cout << "Matriks A tidak bisa ditambahkan dengan matriks B"<<endl;
	Else {
	For(i=0;i< banyakKol_1; i++) {
	cout<<"Masukkan Matriks A ["<< i<< "] :";
	cin >> A[i] ;
	}

Pseudocode	Bahasa C++

FOR i=1 TO banyakKol_1 DO BEGIN
IF(banyakKol_1 = 2)THEN INPUT
matriks_Arr1_2[i]
ELSE IF(banyakKol_1 =
3) THEN INPUT matriks_Arr1_2[i]
ELSE INPUT
matriks_Arr1_4[i]; END FOR
 	
IF simpan_1 = TRUE
FOR i=1 TO banyakKol_1 DO
IF(banyakKol_1 = 2)
THEN A[i]=matriks_Arr1_2[i]
ELSE
IF(banyakKol_1=3) THEN
A[i]=matriks_Arr1_2[i]
ELSE
A[i]=matriks_Arr1_2[i];
 	
INPUT banyakKol_2; REPEAT
INPUT tampil_2; UNTIL tampil_2 = TRUE;
 	
IF(banyakKol_2 = 2) THEN (PRINT
matriks_Arr1_2)
ELSE IF (banyakKol_2=3) THEN (PRINT
matriks_Arr1_3)
ELSE (PRINT
matriks_Arr1_4);
FOR i=1 TO banyakKol_2 DO BEGIN
IF (banyakKol_2 = 2) THEN INPUT
matriks_Arr1_2[i]
ELSE IF(banyakKol_2 =
3) THEN INPUT matriks_Arr1_3[i]
ELSE INPUT
matriks_Arr1_4[i]; END FOR
 	
IF simpan_2 = TRUE
FOR i=1 TO banyakKol_2 DO
IF(banyakKol_1 = 2)
THEN B[i]=matriks_Arr1_2[i]
ELSE
 	
IF(banyakKol_1 = 3)THEN
B[i]=matriks_Arr1_2[i]
ELSE B[i] =
matriks_Arr1_2[i];
 	
IF banyakKol_1 = banyakKol_2 THEN BEGIN
FOR i=1 TO banyakKol_1 DO C[i] = A[i] +
B[i];
INPUT tampilkanHasil; REPEAT
FOR i=1 TO banyakKol_1 DO PRINT C[i]
UNTIL tampilkanHasil=TRUE; END IF
ELSE PRINT "Matriks A tidak bisa
dijumlahkan dengan Matriks B";
 	
END
	cout << ""<<endl;
	For(i=0;i< banyakKol_2; i++) {
	cout << "Masukkan Matriks B ["<<
	I
	<< "]:";
	cin>> B[i];
	}
	cout << ""<endl;
	For(i=0;i< banyakKol_1; i++) {
	C[i] = A[i] + B[i];
	Ctemp = Ctemp+ "["+C[i]+"]";
	}
	cout << "Matriks C:"<<
	Ctemp<<endl;
	}
	Return 0;
	}

Pseudocode	Bahasa Java

penjumlahanMatriks; deklarasi
variable;
banyakKol_1, banyakKol_2, I: numerik;
tampil_1, tampil_2, simpan_1,
simpan_2, tampilkanHasil: boolean;
A[1..4], B[1..4], C[1..4]: numerik;
 	
Begin
INPUT banyakKol_1; REPEAT
INPUT tampil_1; UNTIL tampil_1 =
TRUE;
 	
IF (banyakKol_1 = 2) THEN (PRINT
matriks_Arr1_2)
ELSE IF(banyakKol_1 = 3) THEN (PRINT
matriks_Arr1_3)
ELSE (PRINT
matriks_Arr1_4);
FOR i=1 TO banyakKol_1 DO BEGIN
IF(banyakKol_1 = 2)THEN INPUT
matriks_Arr1_2[i]
ELSE IF(banyakKol_1 =
 	
3) THEN INPUT matriks_Arr1_2[i]
ELSE INPUT
matriks_Arr1_4[i]; END FOR
 	
IF simpan_1 = TRUE
FOR i=1 TO banyakKol_1 DO
IF(banyakKol_1 = 2)
THEN A[i]=matriks_Arr1_2[i]
ELSE
IF(banyakKol_1=3) THEN
A[i]=matriks_Arr1_2[i]
ELSE
A[i]=matriks_Arr1_2[i];
 	
INPUT banyakKol_2; REPEAT
INPUT tampil_2; UNTIL tampil_2 =
TRUE;
 	
IF(banyakKol_2 = 2) THEN (PRINT
matriks_Arr1_2)
ELSE IF (banyakKol_2=3) THEN (PRINT
matriks_Arr1_3)
ELSE (PRINT
matriks_Arr1_4);
FOR i=1 TO banyakKol_2 DO BEGIN
IF (banyakKol_2 = 2) THEN INPUT
matriks_Arr1_2[i]
ELSE IF(banyakKol_2 =
3) THEN INPUT matriks_Arr1_3[i]
ELSE INPUT
matriks_Arr1_4[i]; END FOR
	public class MatriksPenjumlahan {
	public static void main(String[]
	args)
	{
	int banyakKol_1 = 4; int
	banyakKol_2 = 4; int i;
	int[] A = new int[4]; int[] B = new
	int[4]; int[] C = new int[4];
	String Ctemp="";
	If (banyakKol_1 != banyakKol_2)
	System.out.println ("Matriks A
	tidak bisa dijumlahkan dengan
	matriks B");
	Else {
	Scanner sc = new
	Scanner(System.in);
	For(i=0;i< banyakKol_1; i++) {
	System.out.print ("Masukkan Matriks
	A ["+ i+"]:");
	A[i] = sc.nextInt();
	}
	System.out.println (""); For(i=0;i<
	banyakKol_2; i++) {
	System.out.print ("Masukkan Matriks

Pseudocode	Bahasa Java

IF simpan_2 = TRUE
FOR i=1 TO banyakKol_2 DO
IF(banyakKol_1 = 2)
THEN B[i]=matriks_Arr1_2[i]
ELSE
IF(banyakKol_1 = 3)THEN
B[i]=matriks_Arr1_2[i]
ELSE B[i] =
matriks_Arr1_2[i];
 	
IF banyakKol_1 = banyakKol_2 THEN BEGIN
FOR i=1 TO banyakKol_1 DO C[i] = A[i] +
B[i];
INPUT tampilkanHasil; REPEAT
FOR i=1 TO banyakKol_1 DO PRINT C[i]
UNTIL tampilkanHasil=TRUE; END IF
ELSE PRINT "Matriks A tidak bisa
dijumlahkan dengan Matriks B";
 	
END
	B ["+ i+"]:");
	B[i] = sc.nextInt();
	}
	System.out.println ("");
	For(i=0;i< banyakKol_1; i++) {
	C[i] = A[i] + B[i];
	Ctemp = Ctemp+ "["+C[i]+"]";
	}
	System.out.println ("Matriks
	C:"+ Ctemp);
	}
	}
	}

7.3.2 Pengurangan Matriks

Pseudocode

```
pennguranganMatriks; deklarasi variable;
banyakKol_1, banyakKol_2, I: numerik;
    tampil_1, tampil_2, simpan_1, simpan_2, tampilkanHasil: boolean;
    A[1..4], B[1..4], C[1..4]: numerik;

Begin
    INPUT banyakKol_1; REPEAT
    INPUT tampil_1; UNTIL tampil_1 = TRUE;

    IF (banyakKol_1 = 2) THEN (PRINT matriks_Arr1_2)
    ELSE IF(banyakKol_1 = 3) THEN (PRINT matriks_Arr1_3) ELSE (PRINT
    matriks_Arr1_4);
    FOR i=1 TO banyakKol_1 DO BEGIN
    IF(banyakKol_1 = 2)THEN INPUT matriks_Arr1_2[i]
    ELSE IF(banyakKol_1 = 3) THEN INPUT matriks_Arr1_2[i] ELSE INPUT
    matriks_Arr1_4[i];
    END FOR

    IF simpan_1 = TRUE
    FOR i=1 TO banyakKol_1 DO
    IF(banyakKol_1 = 2) THEN A[i]=matriks_Arr1_2[i]
    ELSE IF(banyakKol_1=3) THEN A[i]=matriks_Arr1_2[i] ELSE
    A[i]=matriks_Arr1_2[i];
```


Pseudocode

```
    INPUT banyakKol_2; REPEAT
    INPUT tampil_2; UNTIL tampil_2 = TRUE;

    IF(banyakKol_2 = 2) THEN (PRINT matriks_Arr1_2)
    ELSE IF (banyakKol_2=3) THEN (PRINT matriks_Arr1_3) ELSE (PRINT
    matriks_Arr1_4);
    FOR i=1 TO banyakKol_2 DO BEGIN
    IF (banyakKol_2 = 2) THEN INPUT matriks_Arr1_2[i]
    ELSE IF(banyakKol_2 = 3) THEN INPUT matriks_Arr1_3[i] ELSE INPUT
    matriks_Arr1_4[i];
    END FOR

    IF simpan_2 = TRUE
    FOR i=1 TO banyakKol_2 DO
    IF(banyakKol_1 = 2) THEN B[i]=matriks_Arr1_2[i]
    ELSE IF(banyakKol_1 = 3)THEN B[i]=matriks_Arr1_2[i] ELSE B[i] =
    matriks_Arr1_2[i];
    IF banyakKol_1 = banyakKol_2 THEN BEGIN FOR i=1 TO banyakKol_1 DO
    C[i] = A[i] - B[i]; INPUT tampilkanHasil;
    REPEAT
    FOR i=1 TO banyakKol_1 DO PRINT C[i] UNTIL tampilkanHasil=TRUE;
    END IF
    ELSE PRINT "Matriks A tidak bisa dikurangkan dengan Matriks B";

    END
```

Adapun bentuk konversi *pseudocode* ke dalam 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Pengurangan Matriks dapat dilihat pada tabel di bawah ini.

Operasi Pengurangan Matriks

Pseudocode	Bahasa Pascal

penngurangan Matriks; deklarasi variable;
banyakKol_1, banyakKol_2, I: numerik;
tampil_1, tampil_2, simpan_1, simpan_2, tampilkanHasil: boolean;
A[1..4], B[1..4], C[1..4]: numerik;
 	
Begin
INPUT banyakKol_1; REPEAT
INPUT tampil_1; UNTIL tampil_1 = TRUE;
 	
IF (banyakKol_1 = 2) THEN (PRINT
	var
	banyakKol_1: integer;
	banyakKol_2 : integer; i : integer;
	A : array [1..4] of integer; B : array [1..4] of integer; C : array [1..4] of integer; Ctemp : string;
	
	begin banyakKol_1 := 4;
	banyakKol_2 := 4;
	If (banyakKol_1 != banyakKol_2)
	begin
	writeln ('Matriks A tidak bisa

Pseudocode	Bahasa Pascal

matriks_Arr1_2)
ELSE IF(banyakKol_1 = 3) THEN
(PRINT matriks_Arr1_3)
ELSE (PRINT
matriks_Arr1_4);
FOR i=1 TO banyakKol_1 DO BEGIN
IF(banyakKol_1 = 2)THEN INPUT
matriks_Arr1_2[i]
ELSE IF(banyakKol_1 =
3) THEN INPUT matriks_Arr1_2[i]
ELSE INPUT
matriks_Arr1_4[i]; END FOR
 	
IF simpan_1 = TRUE
FOR i=1 TO banyakKol_1 DO
IF(banyakKol_1 = 2)
THEN A[i]=matriks_Arr1_2[i]
ELSE
IF(banyakKol_1=3) THEN
A[i]=matriks_Arr1_2[i]
ELSE
A[i]=matriks_Arr1_2[i];
 	
INPUT banyakKol_2; REPEAT
INPUT tampil_2; UNTIL tampil_2 =
TRUE;
 	
IF(banyakKol_2 = 2) THEN (PRINT
matriks_Arr1_2)
ELSE IF (banyakKol_2=3) THEN
(PRINT matriks_Arr1_3)
ELSE (PRINT
matriks_Arr1_4);
FOR i=1 TO banyakKol_2 DO BEGIN
IF (banyakKol_2 = 2) THEN INPUT
matriks_Arr1_2[i]
ELSE IF(banyakKol_2 =
3) THEN INPUT matriks_Arr1_3[i]
ELSE INPUT
matriks_Arr1_4[i]; END FOR
 	
IF simpan_2 = TRUE
FOR i=1 TO banyakKol_2 DO
IF(banyakKol_1 = 2)
THEN B[i]=matriks_Arr1_2[i]
ELSE
IF(banyakKol_1 = 3)THEN
B[i]=matriks_Arr1_2[i]
ELSE B[i] =
matriks_Arr1_2[i];
 	
IF banyakKol_1 = banyakKol_2 THEN
BEGIN
FOR i=1 TO banyakKol_1 DO C[i] =
A[i] - B[i];
	dikurangkan dengan matriks B');
	end;
	Else begin
	for i := 0 to banyakKol_1 do
	begin
	write ('Masukkan Matriks A [',
	I,']:'); readln(A[i]);
	end; writeln (' ');
	For i :=0 to banyakKol_2 begin
	write ('Masukkan Matriks B [',
	I,']:'); readln(B[i]);
	end; writeln (' ');
	for i := 0 to banyakKol_1 do
	begin
	C[i] := A[i] - B[i];
	Ctemp := Ctemp+ '['+C[i]+']'; end;
	writeln (' Matriks C:', Ctemp);
	end;
	end;

Pseudocode	Bahasa Pascal
INPUT tampilkanHasil; REPEAT FOR i=1 TO banyakKol_1 DO PRINT C[i] UNTIL tampilkanHasil=TRUE; END IF ELSE PRINT "Matriks A tidak bisa dikurangkan dengan Matriks B"; END	

Pseudocode	Bahasa C++

pennguranganMatriks; deklarasi
variable;
banyakKol_1, banyakKol_2, I:
numerik;
tampil_1, tampil_2, simpan_1,
simpan_2, tampilkanHasil: boolean;
A[1..4], B[1..4], C[1..4]: numerik;
 	
Begin
INPUT banyakKol_1; REPEAT
INPUT tampil_1; UNTIL tampil_1 =
TRUE;
 	
IF (banyakKol_1 = 2) THEN (PRINT
matriks_Arr1_2)
ELSE IF(banyakKol_1 = 3) THEN (PRINT
matriks_Arr1_3)
ELSE (PRINT
matriks_Arr1_4);
FOR i=1 TO banyakKol_1 DO BEGIN
IF(banyakKol_1 = 2)THEN INPUT
matriks_Arr1_2[i]
ELSE IF(banyakKol_1 =
3) THEN INPUT matriks_Arr1_2[i]
ELSE INPUT
matriks_Arr1_4[i]; END FOR
 	
IF simpan_1 = TRUE
FOR i=1 TO banyakKol_1 DO
IF(banyakKol_1 = 2)
THEN A[i]=matriks_Arr1_2[i]
ELSE
IF(banyakKol_1=3) THEN
A[i]=matriks_Arr1_2[i]
ELSE
A[i]=matriks_Arr1_2[i];
INPUT banyakKol_2; REPEAT
INPUT tampil_2; UNTIL tampil_2 =
TRUE;
 	
IF(banyakKol_2 = 2) THEN (PRINT
matriks_Arr1_2)
	#include <stdio.h> using namespace
	std; int main() {
	int banyakKol_1 = 4; int banyakKol_2
	= 4; int i;
	int A[4];
	int B[4];
	int C[4];
	String Ctemp="";
	If (banyakKol_1 != banyakKol_2) cout
	<< "Matriks A
	tidak bisa
	dikurangkan dengan matriks
	B"<<endl;
	Else {
	For(i=0;i< banyakKol_1; i++) {
	cout<<"Masukkan Matriks A ["<< i<<
	"]:";
	cin >> A[i] ;
	}
	cout << ""<<endl;
	For(i=0;i< banyakKol_2; i++) {
	cout << "Masukkan Matriks B ["<< I
	<< "]:";
	cin>> B[i];
	}
	cout << ""<endl;
	For(i=0;i< banyakKol_1; i++) { C[i]
	= A[i] - B[i];
	Ctemp = Ctemp+ "["+C[i]+"]";
	}
	cout << "Matriks C:"<<
	Ctemp<<endl;
	}
	Return 0;
	}

Pseudocode	Bahasa C++

ELSE IF (banyakKol_2=3) THEN (PRINT matriks_Arr1_3)
ELSE (PRINT matriks_Arr1_4);
FOR i=1 TO banyakKol_2 DO BEGIN
IF (banyakKol_2 = 2) THEN INPUT matriks_Arr1_2[i]
ELSE IF(banyakKol_2 = 3) THEN INPUT matriks_Arr1_3[i]
ELSE INPUT matriks_Arr1_4[i]; END FOR
 	
IF simpan_2 = TRUE
FOR i=1 TO banyakKol_2 DO
IF(banyakKol_1 = 2)
THEN B[i]=matriks_Arr1_2[i]
ELSE
IF(banyakKol_1 = 3)THEN
B[i]=matriks_Arr1_2[i]
ELSE B[i] = matriks_Arr1_2[i];
 	
IF banyakKol_1 = banyakKol_2 THEN
BEGIN
FOR i=1 TO banyakKol_1 DO C[i] = A[i] - B[i];
INPUT tampilkanHasil; REPEAT
FOR i=1 TO banyakKol_1 DO PRINT C[i]
UNTIL tampilkanHasil=TRUE; END IF
ELSE PRINT "Matriks A tidak bisa dikurangkan dengan Matriks B";	

Pseudocode	Bahasa Java

penguranganMatriks; deklarasi variable;
banyakKol_1, banyakKol_2, I: numerik;
tampil_1, tampil_2, simpan_1,
simpan_2, tampilkanHasil: boolean;
A[1..4], B[1..4], C[1..4]: numerik;
 	
Begin
INPUT banyakKol_1; REPEAT
INPUT tampil_1; UNTIL tampil_1 = TRUE;
 	
IF (banyakKol_1 = 2) THEN (PRINT matriks_Arr1_2)
ELSE IF(banyakKol_1 = 3) THEN (PRINT matriks_Arr1_3)
ELSE (PRINT matriks_Arr1_4);
	public class MatriksPengurangan {
	public static void main(String[] args)
	{
	int banyakKol_1 = 4; int banyakKol_2 = 4; int i;
	int[] A = new int[4]; int[] B = new int[4]; int[] C = new int[4];
	String Ctemp="";
	If (banyakKol_1 != banyakKol_2)
	System.out.println ("Matriks A tidak bisa dikurangkan dengan matriks B");
	Else {
	Scanner sc = new Scanner(System.in);
	For(i=0;i< banyakKol_1; i++) {

Pseudocode	Bahasa Java

FOR i=1 TO banyakKol_1 DO BEGIN
IF(banyakKol_1 = 2)THEN INPUT matriks_Arr1_2[i]
ELSE IF(banyakKol_1 = 3) THEN INPUT matriks_Arr1_2[i]
ELSE INPUT matriks_Arr1_4[i]; END FOR
 	
IF simpan_1 = TRUE
FOR i=1 TO banyakKol_1 DO
IF(banyakKol_1 = 2)
THEN A[i]=matriks_Arr1_2[i]
ELSE
IF(banyakKol_1=3) THEN
A[i]=matriks_Arr1_2[i]
ELSE
A[i]=matriks_Arr1_2[i];
INPUT banyakKol_2; REPEAT
INPUT tampil_2; UNTIL tampil_2 = TRUE;
IF(banyakKol_2 = 2) THEN (PRINT matriks_Arr1_2)
ELSE IF (banyakKol_2=3) THEN (PRINT matriks_Arr1_3)
ELSE (PRINT matriks_Arr1_4);
FOR i=1 TO banyakKol_2 DO BEGIN
IF (banyakKol_2 = 2) THEN INPUT matriks_Arr1_2[i]
ELSE IF(banyakKol_2 = 3) THEN INPUT matriks_Arr1_3[i]
ELSE INPUT matriks_Arr1_4[i]; END FOR
IF simpan_2 = TRUE
FOR i=1 TO banyakKol_2 DO
IF(banyakKol_1 = 2)
THEN B[i]=matriks_Arr1_2[i]
	System.out.print ("Masukkan Matriks A ["+ i+"]:");
	A[i] = sc.nextInt();
	}
	System.out.println (""); For(i=0;i< banyakKol_2; i++) {
	System.out.print ("Masukkan Matriks B ["+ i+"]:");
	B[i] = sc.nextInt();
	}
	System.out.println (""); For(i=0;i< banyakKol_1; i++) { C[i] = A[i] - B[i];
	Ctemp = Ctemp+ "["+C[i]+"]";
	}
	System.out.println ("Matriks C:"+ Ctemp);
	}
	}

7.3.3 Perulangan Array 1 Dimensi

Pseudocode

perulanganArraySederhana;
deklarasi variable;
bil[i]; i: numerik;
tampilHasil: boolean;
Begin
REPEAT
INPUT tampilHasil; UNTIL tampilHasil = true;
 	
i=1; bil[i]=i; REPEAT
PRINT i;
i=i+1; bil[i]=i;
UNTIL i = 10;
END.

Adapun bentuk konversi *pseudocode* ke dalam 3 bentuk bahasa pemrograman (Pascal, C++, dan Java) untuk Pengurangan Matriks dapat dilihat pada tabel di bawah ini.

Konsep Perulangan Array

Pseudocode	Bahasa Pascal

perulanganSederhana; deklarasi
variable;
bil[i]; i: numerik; tampilHasil:
boolean;
Begin
REPEAT
INPUT tampilHasil; UNTIL tampilHasil
= true;
i=1; bil[i]=i; REPEAT
PRINT i;
i=i+1; bil[i]=i;
UNTIL i = 10; END.
	var
	bil : array [1..11] of integer;
	i : integer;
	
	begin i :=1;
	while (i<=10) do begin
	write (i , ' '); bil[i] :=i;
	i :=i+1;
	end; writeln (' '); end;

Pseudocode	Bahasa C++

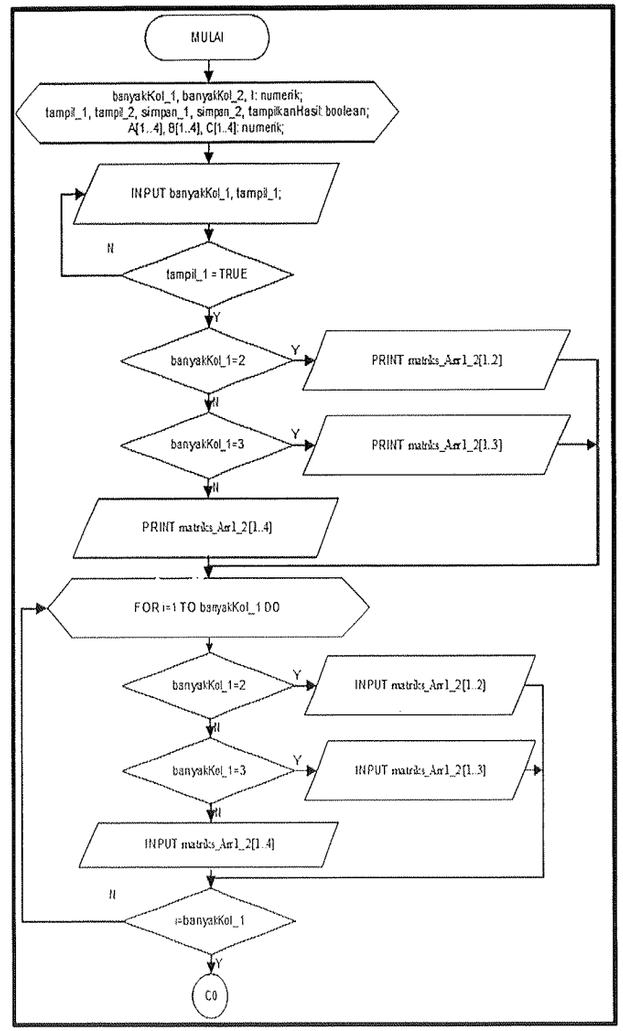
perulanganSederhana; deklarasi
variable;
bil[i]; i: numerik; tampilHasil:
boolean;
 	
Begin
REPEAT
INPUT tampilHasil; UNTIL tampilHasil
= true;
 	
i=1; bil[i]=i; REPEAT
PRINT i;
i=i+1; bil[i]=i;
UNTIL i = 10; END.
	#include <stdio.h> using namespace
	std; int main() {
	int i=1;
	int bil [11]; while (i<=10) {
	cout<< i << " "; bil[i]=i;
	i=i+1·
	}
	cout<< " " << endl; return 0;
	}

Pseudocode	Bahasa Java

perulanganSederhana; deklarasi
variable;
bil[i]; i: numerik; tampilHasil:
boolean;
 	
Begin
REPEAT
INPUT tampilHasil; UNTIL tampilHasil
= true;
i=1; bil[i]=i; REPEAT
PRINT i;
i=i+1; bil[i]=i;
UNTIL i = 10; END.
	public class PerulanganArr {
	public static void main(String[]
	args)
	{
	int i=1;
	int[] bil = new int[11]; while
	(i<=10) { System.out.print(i + "
	"); bil[i]=i;
	i=i+1;
	}
	System.out.println(" ");
	}
	}

7.4 Pengenalan *Flowchart*

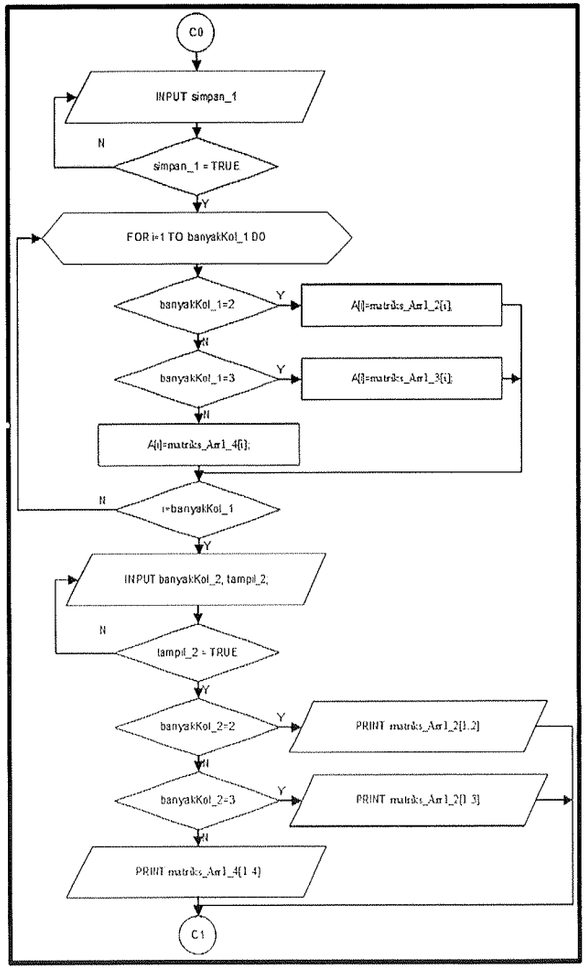
7.4.1 *Flowchart* Penjumalahan Matriks



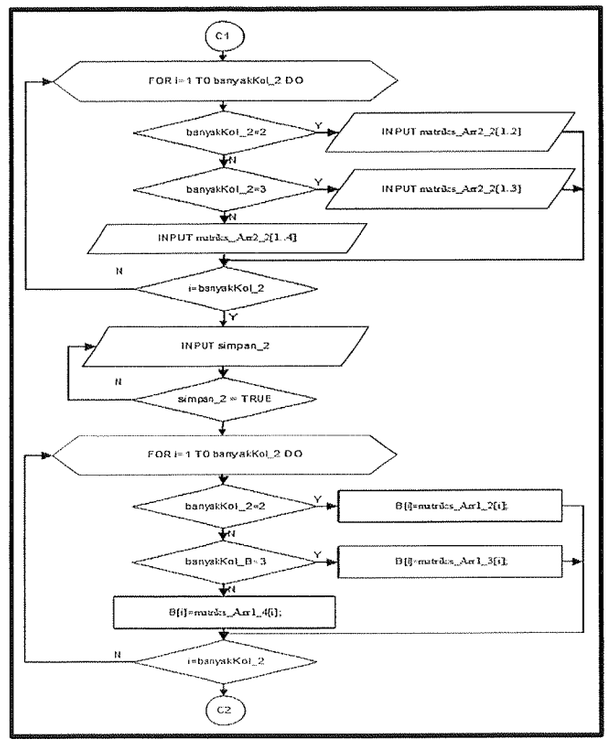
Gambar 7.17 *Flowchart Penjumlahan Matriks Bagian 1*

Pada Gambar 7.17. *Flowchart* dimulai dengan mendeklarasikan variabel banyakKol_1 dan banyakKol2 dengan tipe numerik; deklarasi

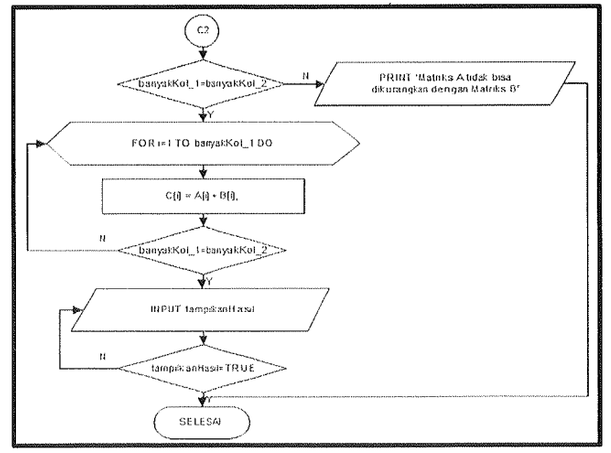
variasi tampil_1, tampil_2, simpan_1, simpan_2 dan tampilkanHasil dengan tipe boolean; serta variabel array A[1..4], B[1..4] dan C[1..4] dengan tipe numerik. Proses selanjutnya adalah meng*input*kan banyaknya kolom serta menekan tombol tampil, maka akan keluar sesuai dengan nilai yang diinputkan. Banyaknya kolom mulai dari 1 hingga 4. Kemudian dilakukan *looping* atau perulangan untuk meng*input*kan nilai dari masing-masing elemen.



Gambar 7.18 *Flowchart Penjumlahan Matriks Bagian 2.*

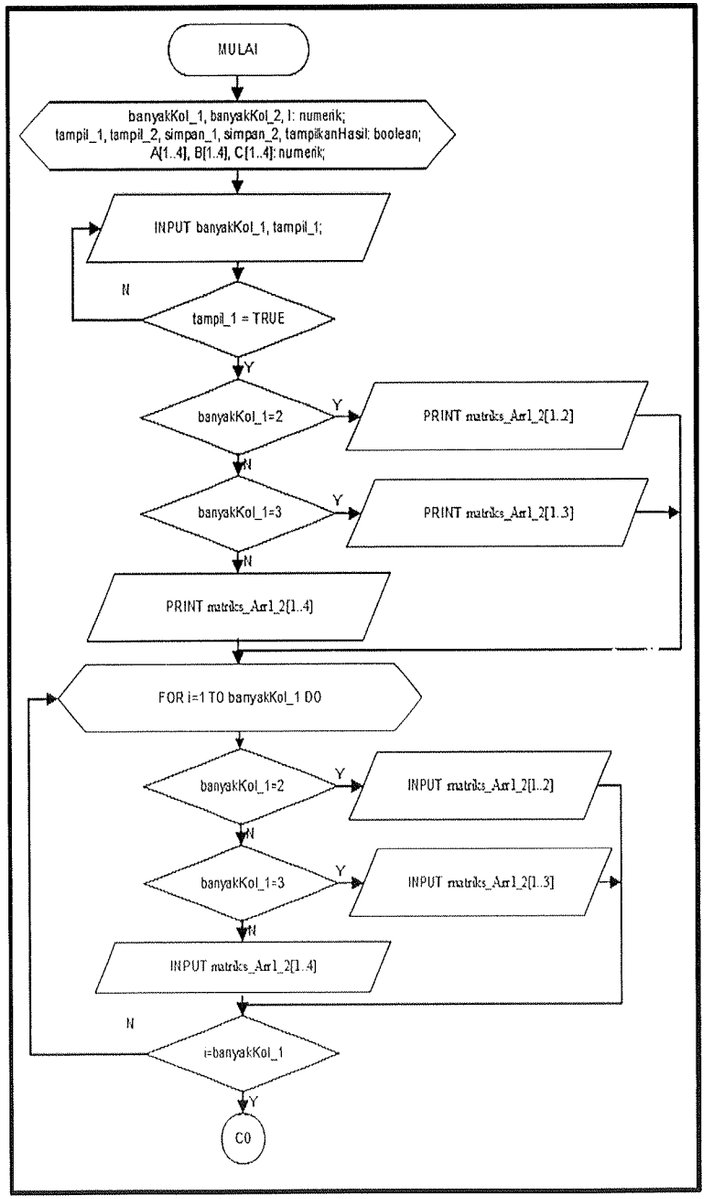


Gambar 7.19 *Flowchart Penjumlahan Matriks Bagian 3*

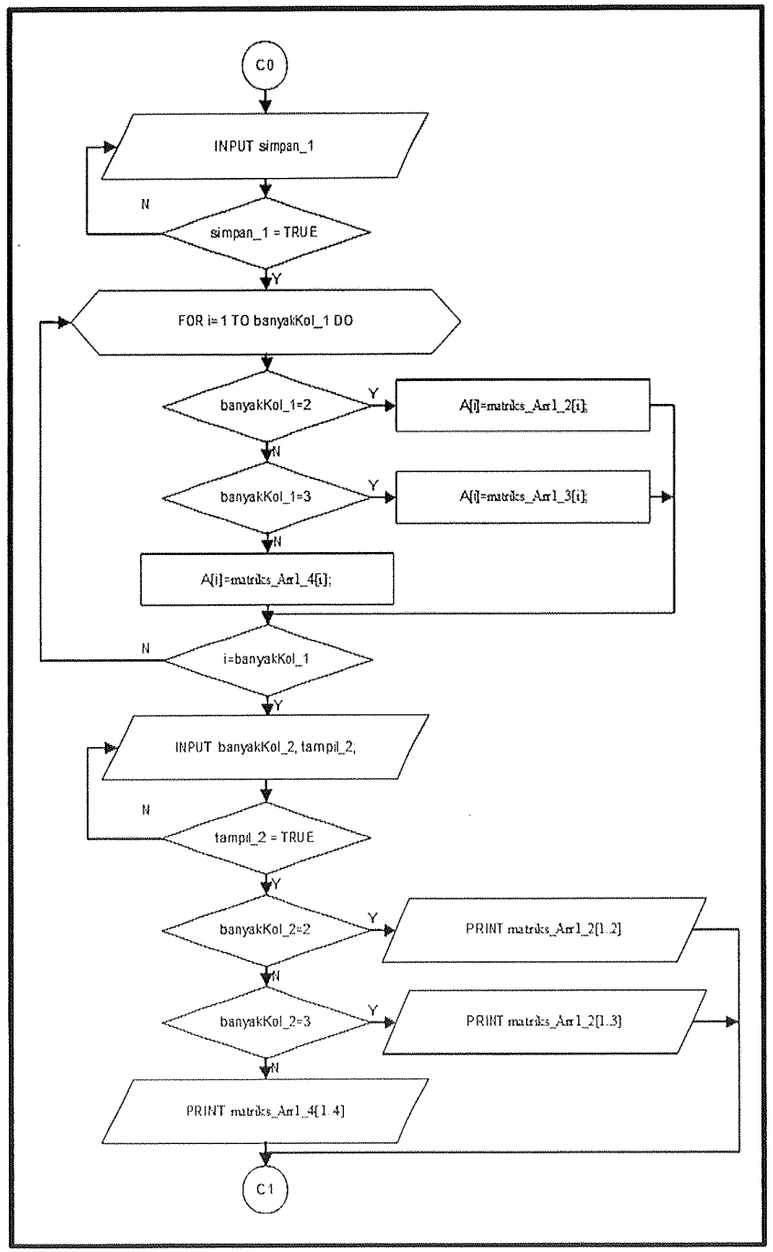


Gambar 7.20 *Flowchart Penjumlahan Matriks Bagian 4*

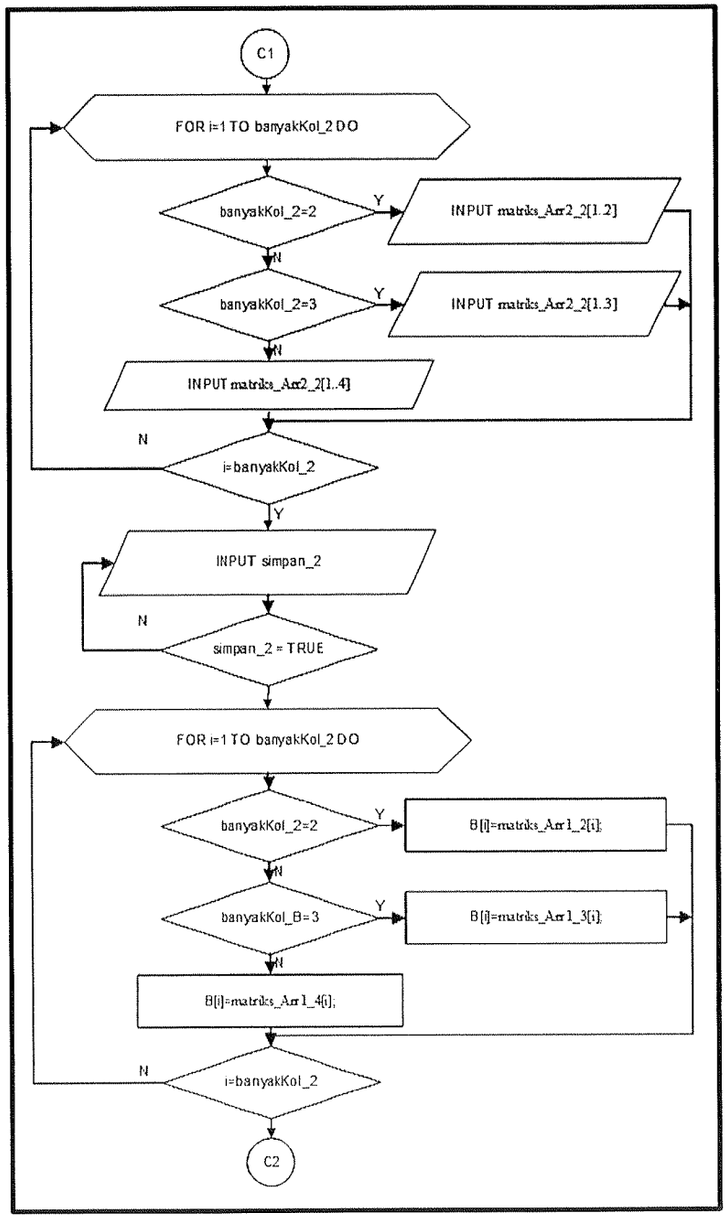
7.4.2 *Flowchart* Pengurangan Matriks



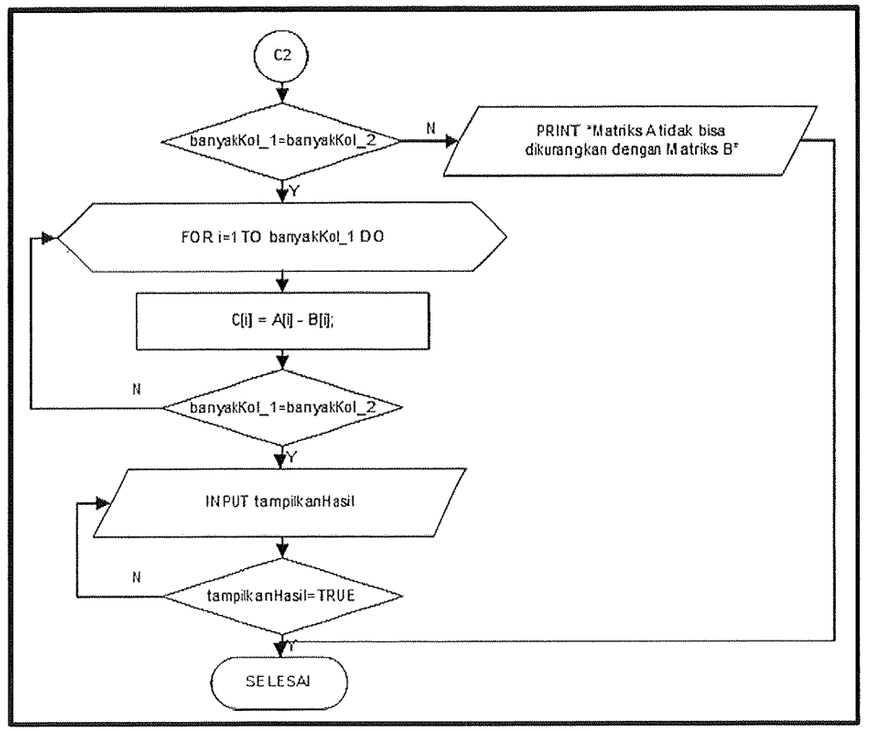
Gambar 7.21 *Flowchart Pengurangan Matriks Bagian 1*



Gambar 7.22 *Flowchart Pengurangan Matriks Bagian 2*



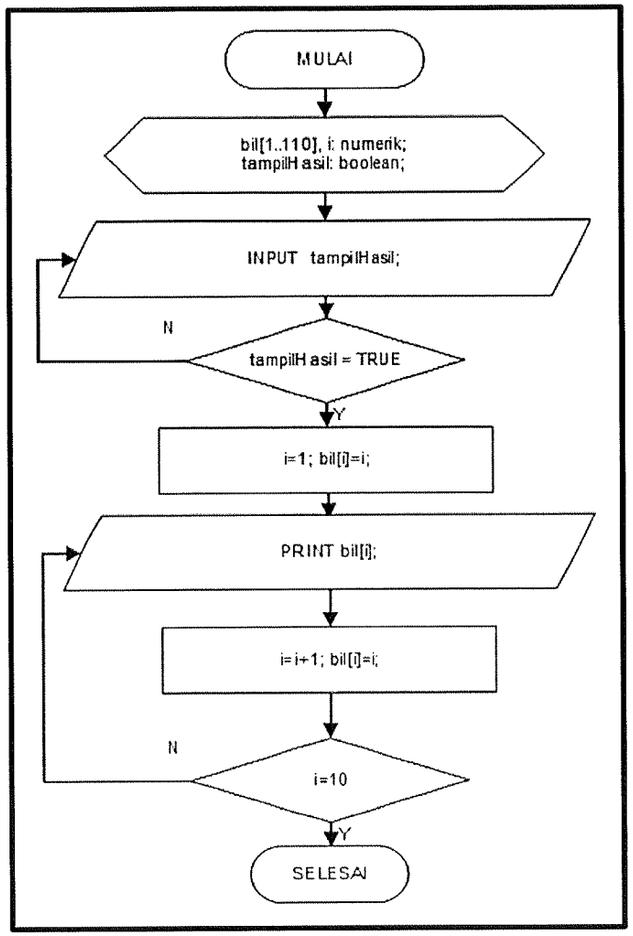
Gambar 7.23 *Flowchart Pengurangan Matriks Bagian 3*



Gambar 7.24 *Flowchart Pengurangan Matriks Bagian 4*

7.4.3 *Flowchart* Perulangan Array 1 Dimensi

Pada Gambar 7.25 *Flowchart* dimulai dengan mendeklarasikan variabel array yaitu bil[1...10] dan i dengan tipe numerik, serta deklasi variabel tampilHasil dengan tipe boolean. Proses selanjutnya adalah menekan tombol tampilHasil. Kemudian dilanjutkan pada proses inisialisasi variabel i=1 dan inisialisasi variabel array bil[i] = 1. Kemudian dilakukan perulangan untuk mencetak hasil bilangan 1 hingga sepuluh. Untuk mencetak bilangan digunakan proses perulangan dengan kondisi i=10. Jika nilai i tidak sama dengan sepuluh maka perulangan akan terus berlanjut, namun bila i sama dengan sepuluh maka akan ke akhir proses.



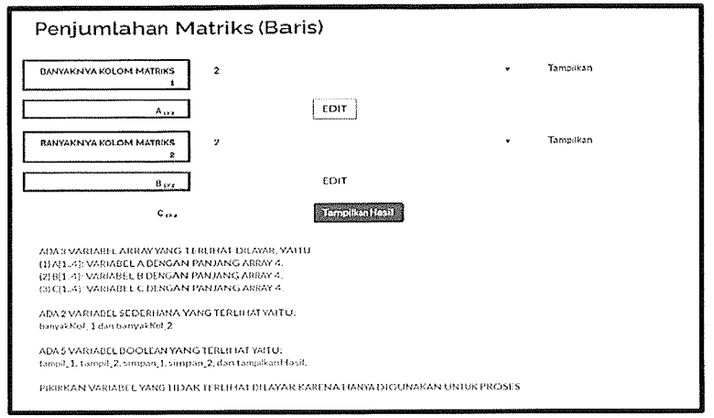
Gambar 7.25 *Flowchart Perulangan Array 1 Dimensi*

7.5 Variabel Array 1 Dimensi

7.5.1 Variabel Penjumlahan Matriks

Gambar 7.26 menunjukkan tampilan dari penjumlahan matriks. Banyaknya variabel yang digunakan untuk menjumlahkan matriks terdiri dari beberapa variabel yaitu:

1. Terdapat 3 variabel array yaitu A,B dan C di mana masing-masing memiliki 4 elemen (panjang array adalah 4)
2. Terdapat 2 variabel sederhana untuk menampung kolom1 dan kolom 2
3. Terdapat 5 variabel boolean untuk menampilkan *input*an dan hasil serta menyimpan proses.

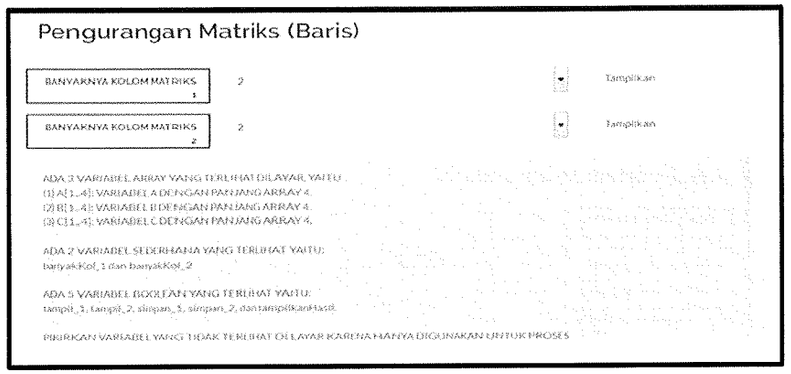


Gambar 7.26 *Variabel Penjumlahan Matriks*

Pada Gambar 7.26 digambarkan bagaimana cara untuk meng-identifikasi kebutuhan variabel untuk proses pengurangan matriks. yang tidak jauh berbeda dengan proses penjumlahan matriks. Penentuan variabel untuk kebutuhan pengurangan matriks juga dibagi menjadi 3 jenis, yaitu variabel yang terlihat dilayar dan variabel yang tidak terlihat di layar. Tampilan menu tersebut dapat diakses pada **Menu Pernyataan Matematika - Operasi Matriks, kemudia pilih Operasi Pengurangan.**

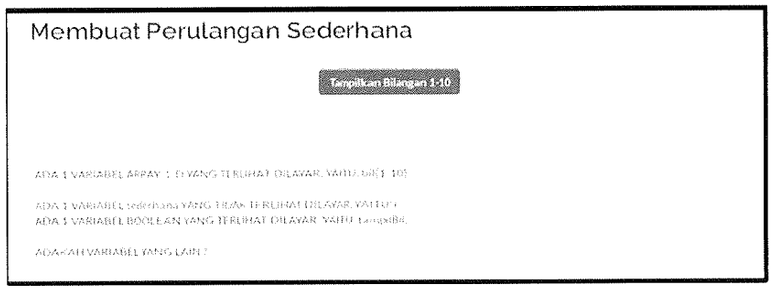
a. Variabel yang terlihat di layar dapat diidentifikasi melalui kolom data masukan dan kolom data keluaran pada Menu pengurangan yaitu **banyak kolom matriks 1** dan **banyak kolom matriks 2** serta keluaran untuk hasil pengurangannya.
b. Sedangkan variabel yang tidak terlihat di layar dapat diidentifikasi melalui tombol yang ditekan atau tidak ditekan oleh pengguna. Pada form Multiplikator tersebut terdapat 5 tombol, yaitu 2 tombol tampil, 2 tombol simpang, dan tombol tampilkan hasil.

7.5.2 Variabel Pengurangan Matriks



Gambar 7.27 *Variabel Pengurangan Matriks*

7.5.3 Variabel Perulangan Array 1 Dimensi



Gambar 7.28 *Variabel Perulangan Array 1 Dimensi*

Gambar 7.28 menunjukkan tampilan dari penjumlahan matriks. Banyaknya variabel yang digunakan untuk menjumlahkan matriks terdiri dari beberapa variabel yaitu:

1. Terdapat 1 variabel array 1 Dimensi yaitu yang terlihat pada layar yaitu bil [1...10]
2. Terdapat 1 variabel sederhana yang tidak terlihat yaitu *i*, untuk indeks setiap elemen array
3. Terdapat 1 variabel boolean yang tidak terlihat untuk menampilkan bilangan yaitu tombol 'tampilkan bilangan 1 sampai 10'

7.6 Rangkuman

a. Sebuah Array atau Larik merupakan kumpulan dari beberapa nilai yang mempunyai tipe data yang sama. Array disebut juga sebagai Larik/Tabel/Vektor/Variabel Majemuk dan serangkaian elemen array dibedakan pada nilai indeksnya.
b. Semua proses yang berkaitan dengan array 1D, paling mudah dilakukan dengan menggunakan konsep *looping* FOR.
c. Array satu dimensi merupakan array yang terdiri dari satu baris banyak kolom atau satu kolom banyak baris.

7.7 Latihan Soal

1. Deklarasikan sebuah variabel array (misalkan nama variabelnya = **letters)** yang mengalokasikan untuk 10 elemen bertipe char. Kemudian masukkan karakter 'Z' pada elemen yang keempat dari array **letters.**
2. Buatlah algoritma untuk menginputkan data ke dalam variabel array dengan tipe integer sebanyak *n* kali. Berdasarkan data *input*an tersebut dapat dihitung total dari seluruh data masukan.
3. Berdasarakan soal no. 1 dari *n* data *input*an, tampilkan nilai terbesar dan nilai terkecil dari variabel array yang diinputkan.
4. Berdasarakan soal no. 1 dari *n* data inputan, Kemudian tampilkan isi array tersebut dengan indeks ganjil.
5. Buatlah *flowchart* dengan menerapkan variabel array 1 Dimensi untuk menampung dan menampilkan nama dan IPK mahasiswa. Di mana nilai IPK mahasiswa secara simultan bertambah sebesar 0.1. berikut *layout* tampilannya:
   Tabel Penilaian

NAMA	IPK
Budi	4.00
Ani	4.10
Sudrajat	4.20
Fajar	4.30

6. Buat *flowchart* untuk mencari nilai rata-rata seorang mahasiswa, berikut contoh studi kasus yang diberikan.

   Input :
   Nama : Agus
   Jumlah nilai : 3
   Nilai-1 : 60
   Nilai-2 : 70
   Nilai-3 : 80

   Output :
   Agus, nilai rata-rata adalah 70.

7. Dari soal no. 5 buatlah agar jumlah mahasiswa yang nantinya akan di*input*kan nilai menjadi lebih dinamis (ditentukan oleh *user* sendiri berapa jumlah mahasiswanya).
8. Buatlah *flowchart* untuk studi kasus berikut ini Setiap akhir semester, kelas XII IPA 5 selalu mengadakan pemilihan untuk siswa paling baik. Pak Saiful selaku wali kelas hendak memberikan *award* untuk *siswa of the year*. Metode yang digunakan untuk memilih siswa of the year adalah dengan cara menyebutkan nilai akhir hasil belajar para siswa. Nilai akhir yang terbesar akan dinobatkan sebagai *siswa of the year*.

   Contoh Tampilan:
   Jumlah siswa : 5 [inputan]
   Nilai akhir siswa 1 : 5 [inputan]
   Nilai akhir siswa 2 : 4 [inputan]
   Nilai akhir siswa 3 : 10 [inputan]
   Nilai akhir siswa 4 : 9 [inputan]
   Nilai akhir siswa 5 : 2 [inputan]
   siswa of the Year adalah siswa ke-3 dengan nilai akhir 10

9. Ibu menyuruh Titi untuk berbelanja sejumlah barang di supermarket. Setibanya di supermarket, ternyata sedang diadakan diskon untuk semua jenis barang. Diskon yang diberikan setiap barang berbeda dengan barang lainnya. Berdasarkan catatan yang

dititipkan ibu kepada Titi, Titi membeli sejumlah barang dengan daftar diskon barang sebagai berikut:

a. Sabun : diskon 5% (harga per item 5.500 sebelum diskon dan *Qty* adalah 10)
b. Beras : diskon 7% (harga per item 14.000 sebelum diskon dan *Qty* adalah 5 kg)
c. Gula dan Minyak Goreng : diskon 10% (harga per item 20.000 sebelum diskon dan *Qty* adalah 3 kg)

Anda diminta untuk membuat *flowchart* dan *pseudocode* dengan menampilkan hasil akhir yaitu Daftar barang yang dibeli Titi dan Diskon per barang

10. Dari soal nomer 7, buatkan *flowchart* untuk menampilkan total harga per jenis barang dan total harga belanja Titi secara keseluruhan.
11. Buatlah sebuah *pseudocode* untuk menyimpan data nilai akhir mahasiswa menggunakan variabel array 1 Dimensi (minimal 2 orang mahasiswa). Data yang diinputkan berupa NIM, Nama, Mata Kuliah, nilai tugas, UTS dan UAS untuk memperoleh NA (nilai akhir) setiap mahasiswa. *Layout* Menu Utama Program:

```
========================================

APLIKASI PENILAIAN

========================================
```

1) Input Nilai Mahasiswa
2) Nilai Akhir Mahasiswa
3) Keluar

Masukkan no. Pilihan menu anda (1-3):
Ketika pengguna memilih menu 1, tampilan program berikut

1) NIM : 1900001
   Nama : Kurniawan
   Mata Kuliah : Algoritma
   Tugas : 95
   UTS : 70
   UAS : 80

Ketika pengguna memilih menu 2, tampilan program berikut:

1) NIM : 1900001
   Nama : Kurniawan Mata Kuliah: Algoritma
   Nilai Akhir : 90 (NA =((tugas × 40%)+(UTS × 30%)+(UAS × 30%))/3)

   Ketika pengguna memilih menu 3, maka akan keluar dari aplikasi

12. Modifikasilah *flowchart* yang anda buat pada soal no. 10 sehingga jumlah mahasiswa yang di*input*kan menjadi lebih fleksibel (tergantung *input*an pengguna)
13. Modifikasilah *flowchart* yang anda buat pada soal no. 10 menu ke dua sehingga tampilan seperti berikut ini:
    NIM : 1900001
    Nama : Kurniawan
    Mata Kuliah : Algoritma
    Nilai Akhir : 90
    Nilai huruf : A

Nilai Akhir	Nilai Huruf
00 – 44	E
44 – 54	D
55 – 59	C
60 – 64	C+
65 – 74	B
75 – 79	B+
80 - 100	A

14. Buatlah *pseudocode* untuk menampilkan transaski penjualan seperti pada tampilan berikut ini:

    Aplikasi Penjualan Kantin

    1) Daftar Makanan
    2) Transaksi
    3) Omset
    4) *Exit*

Menu 1 : daftar Makanan

Makanan		Harga
Nasi Pecel		15.000
Bakso		12.000
Sate	20.000	
Gado-Gado		10.000

Menu 2: Transaksi
Masukkan nama makanan= Bakso
Harga Bakso = 12.000
Jumlah bakso yang dibeli = 2
Total = 24.000

Menu 3: Omset

Makanan	Jumlah Penjualan	Total
Nasi Pecel	2	30.000
Bakso	2	24.000
Sate	3	60.000
Gado-gado	0	0
TOTAL OMSET		**114.000**

15. Pada soal No. 14 tambahkan menu dengan jenis minuman yang akan dibeli oleh *customer.*

-oo0o-

DAFTAR PUSTAKA

[1] Ariani S, R. (2018). *Logika Algoritma dan Pemrograman Dasar*. Bandung: Modula.

[2] Attridge N, Inglis M. (2013). *Advanced Mathematical Study and the Development of Conditional Reasoning Skills*. PLoS ONE. 8 (7). USA: University of Minnesota.

[3] COMBÉFIS, S., *et al.* (2017). Learning and Teaching Algorithm Design and Optimisation Using Contests Tasks. *Olympiads in Informatics*. 11. 19–28. Litunia: Vilnius University.

[4] Dix, A.J., Finlay, J.E., Abowd, G.D. and Beale, R. (2004). *Human-Computer Interaction*, Third Edition. USA: Prentice Hall.

[5] E. Rahmawati and S. J. Hartati. (2013). *The Application of Computer Aided Learning to Learn Basic Conceptsof Branching and Looping on Logic.* vol. 5. no. 6. pp. 15-24.

[6] Eglen, SJ (2009) A Quick Guide to Teaching R Programming to Computational Biology Students. PLoS Comput Biol 5(8): e1000482. doi:10.1371/*journal.pcbi*.1000482

[7] Farrell, J. (2011). *Programming Logic and Design Introductory*, sixth edition. Canada: Course Technology.

[8] Hartati, S.J. (2014). Design of Learning Model of Logic and Algorithms Based on APOS Theory. *International Journal of Evaluation and Research in Education* (IJERE). 3 (2), 109 - 118. Yogyakarta: UAD.

[9] Hartati, S.J. (2017). *Kerangka Teori untuk Peranan Penalaran Matematika pada Pembelajaran Algoritma Komputasi.* Prosiding: Seminar Nasional Matematika IndoMS Wilayah Sulawesi. Volume 2. 58-68. Makasar: Universitas Hasanuddin.

[10] Hartati, S.J., Rahmawati, E, Vitianingsih, A. V., Kurniati, N., Sayidah, N., Muhajir. (2018). *Buku Ajar Penalaran Matematika pada Pembuatan Algoritma Komputasi Dasar.* Surbaya: Unitomo Press.

[11] Haseski, H.I., İlic, U., Tuğtekin, U. (2018) *Defining a New 21st Century Skill- Computational Thinking: Concepts and Trends. International Education Studies*. 11 (4). Canada: Canadian Center of Science and Education

[12] Hsu, T.C. , Chang, S.C., Hung, Y.T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*. 126. 296–310. Taiwan: Elsevier.

[13] Inglis, M., Attridge, N. (2016). *Does Mathematical Study Develop Logical Thinking?: Testing the Theory of Formal Discipline.* Beaverton. 3 (46). Beaverton: Ringgold Inc.

[14] Johar, R., Yusniarti, S., Saminan. (2018). The Analysis of Proportional Reasoning Problem in The Indonesian Mathematics Textbook for The Junior High School. *Journal on Mathematics Education* (IndoMS-JME)). 9 (1). 55-68. Palembang: IndoMS.

[15] Kozen, D.C. (1991). *The Design and Analysis of Algorithms.* New York: Springer Verlag Inc

[16] McConnell, J.J. (2001). *Analysis of Algorithms: An Active Learning Approach.* Massachusetts: Jones and Bartlett Publishers, Inc.

[17] Mendikbud. (2018). *Peraturan Menteri Pendidikan dan Kebudayaan Republik Indonesia Nomor 37 Tahun 2018 tentang Perubahan Atas Peraturan Menteri Pendidikan dan Kebudayaan Nomor 24 Tahun 2016 Tentang Kompetensi Inti dan Kompetensi Dasar Pelajaran Pada Kurikulum 2013 Pada Pendidikan Dasar dan Pendidikan Menengah.* Jakarta: Kementrian Pendidikan dan Kebudayaan.

[18] Mohaghegh, M., McCauley, M. (2016). Computational Thinking: The Skill Set of the 21st Century. *International Journal of Computer Science and Information Technologies* (IJCSIT). 7 (3). 1524-1530.

[19] Ou, X. (2005). *A logic-programming approach to network security analysis. Unpublished: Dissertation.* New Jersey: Princeton University. Retrieved from ftp://ftp.cs.princeton.edu /techreports /2005/735.pdf.

[20] Papert, S. (1990). *A critique of technocentrism in thinking about the school of the future.* Epistemology and learning memo #2. September 1990. Cambridge MA: MIT. Retrieved 29 August 2015 from www.papert.org/articles/ACritiqueofTechnocentrism.html.

[21] Post, T., Behr, M., & Lesh, R. (1988). *Proportionality and the development of pre- algebra understanding*. In A. Coxford (Ed.), Algebraic concepts in the curriculum K-12.1988 Yearbook (pp. 78-90). Reston, VA: NCTM.

[22] S.J. Hartati, A. V. Vitianingsih. (2018). *Belajar Dasar-dasar Algoritma Komputasi.* penalaran.com.

[23] S.J. Hartati. (2014). Design of Learning Model of Logic and Algorithms Based on APOS Theory. *Int. J. Eval. Res. Educ.* vol. 3. no. 2. pp. 109-118.

[24] S.J. Hartati. (2018). Framework Design Learning of Introduction to Computational Algorithms by Using the Theory Learning by Doing. *J. Phys. Conf. Stat. Math. Teaching,* Res. vol.1028, no.1. p. 1.

[25] Septia, T., Prahmana, R.C.I., Pebrianto, Wahyu, R.. (2018). Improving Students Spatial Reasoning With Course LAB. *Journal on*

Mathematics Education (IndoMS-JME). 9 (2). 327-336. Palembang: IndoMS.

[26] Shafaei, A., Nejati, M., & Abd Razak, N. (2016). *Out of sight, out of mind: psychological consequences of attachment and adjustment attitude.* Studies in Higher Education.accessed on August 10, 2019 at url https://tandfonline.com/doi/full/10.1080/ 03075079. 2016.1162 780?scroll=top&needAcces s=true.

[27] Suparmin, Kurniawati, S. Nuraini. (2017). *Buku Siswa Matematika untuk SD/MI Kelas VI.* Surakarta: Mediatama.

[28] Tourniaire. F. & Pulos, S. (1985). Proportional reasoning: A review of the literature. *Educational Studies in Mathematics, 16*(2), 181-204.

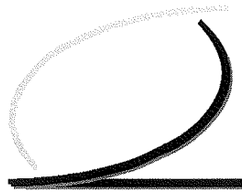
[29] Weintrop, D. et al. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *J Sci Educ Technol.* 25. 127–147. New York: Springer.

[30] Wing, J.M. (2006). *Computational Thinking. Communications of the ACM.* 49 (3). 33-35.

[31] Wing, J.M. (2008). *Computational Thinking and Thinking about Computing.* Philosophical Transactions of the Royal Society of London a: Mathematical, Physical and Engineering Sciences. 366 (1881) 3717 - 3725.

[32]. Wing, J.M. (2010). *Computational Thinking: What and Why?* Retrieved from http://www.cs.cmu.edu/~CompThik/resources /TheLinkWing.pdf.

-oo0o-



GLOSARIUM

A

Algoritma

adalah sebuah prosedur langkah demi langkah untuk memecahkan masalah atau menyelesaikan suatu tugas.

Array atau Larik

adalah variabel yang terdiri dari kumpulan beberapa nilai yang mempunyai tipe data yang sama.

Aplikasi

adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna.

Aritmatika

adalah ilmu hitung merupakan cabang (atau pendahulu) matematika yang mempelajari operasi dasar bilangan.

Akumulasi

adalah tambahan secara berkala atas suatu jumlah pokok.

B

Boolean

adalah sebuah tipe data yang digunakan untuk menyimpan data yang hanya mempunyai 2 nilai, yaitu *true* dan *false*.

Bilangan

adalah suatu konsep matematika yang digunakan untuk pencacahan dan pengukuran atau suatu sebutan untuk menyatakan jumlah/banyaknya sesuatu.

C

CPU

adalah singkatan dari *Central Processing Unit*, yaitu perangkat keras komputer (*hardware*) yang bertugas melaksanakan perintah dan mengolah data dari perangkat lunak.

Code

adalah suatu rangkaian pernyataan atau deklarasi yang ditulis dalam bahasa pemrograman komputer yang terbaca manusia.

Coding

adalah sebuah sebutan dari pada programing/menulis program. Dalam hal ini yang dimagsud dengan **koding** adalah menulis sebuah *script* menggunakan bahasa perograman tertentu dengan aturan tertentu sehingga menghasilkan sebuah karya.

D

Data

adalah sekumpulan keterangan atau fakta mentah berupa simbol, angka, kata-kata, atau citra, yang didapatkan melalui proses pengamatan atau pencarian ke sumber-sumber tertentu.

Digital

adalah penggambaran dari suatu keadaan bilangan yang terdiri dari angka 0 dan 1 atau *off* dan *on* (bilangan biner).

Dekomposisi

adalah salah satu perubahan secara kimia yang membuat objek, biasanya makhluk hidup yang mati dapat mengalami perusakan susunan/struktur yang dilakukan oleh dekomposer (termasuk semut, belatung, bakteri dan jamur).

F

Flowchart

adalah suatu bagan terurut untuk menggambarkan alur yang terjadi pada suat proses, dengan menggunakan simbol-simbol tertentu.

Flowchart Program

adalah bagan yang menggambarkan arus logika dari data yang akan diproses dalam suatu program dari awal sampai akhir.

Faktorial

adalah hasil perkalian antara bilangan bulat positif yang kurang dari atau sama dengan *n*.

I

Indeks

adalah bentuk penomoran atau tanda yang digunakan pada array/larik.

Input

adalah data masukan pada algoritma, baik berupa deklarasi variabel yang akan digunakan selama proses berjalan maupun variabel yang nilainya berasal dari inputan user.

Integer

adalah sebuah tipe data yang digunakan untuk menyimpan bilangan bulat.

Instruksi

adalah perintah atau arahan (untuk melakukan suatu pekerjaan atau melaksanakan suatu tugas).

K

Kounter

adalah variabel yang berfungsi sebagai penambah satu angka menuju tingkat perulangan berikutnya.

Konstanta

adalah variabel yang nilainya tetap.

Komputasi

adalah algoritma yang digunakan untuk menemukan suatu cara dalam memecahkan masalah dari sebuah data *input*.

Komponen

adalah keseluruhan makna yang terdiri dari sejumlah elemen, di mana antara elemen yang satu dengan elemen lainnya memiliki ciri khusus yang berbeda-beda.

Konsep

adalah abstrak, entitas mental yang universal yang menunjuk pada kategori atau kelas dari suatu entitas, kejadian atau hubungan.

Komputer

adalah perangkat elektronik yang memanipulasi informasi atau data.

Komputer

mampu menyimpan, mengambil, dan mengolah data.

Keluaran

adalah suatu nilai yang dikeluarkan (dihasilkan, diterbitkan, dan sebagainya).

Kombinasi

adalah menggabungkan beberapa objek dari suatu grup tanpa memperhatikan urutan.

L

Logika

dapat didefinisikan sebagai ilmu yang harus menggunakan prinsip-prinsip tertentu agar dapat berfikir secara logis (tepat dan benar) menurut kaidah/aturan tertentu.

Logika Algoritma

adalah suatu cabang khusus dalam ilmu komputer yang mempelajari bagaimana para peserta didik dapat berpikir secara logis dalam pembuatan sebuah bentuk algoritma, baik menggunakan *flowchart* maupun *pseudocode*.

Looping Process

adalah sebuah proses algoritma di mana sebuah perintah/instruksi dapat dijalankan secara berulang sampai kondisi/batas tertentu.

Login

adalah proses masuk ke jaringan komputer dengan memasukkan identitas akun minimal terdiri dari *username*/akun pengguna dan *password* untuk mendapatkan hak akses.

M

Masukan

adalah hasil memasukkan.

Matematika

adalah ilmu yang mempelajari hal-hal seperti besaran, struktur, ruang, dan perubahan.

Matriks

adalah sekumpulan bilangan yang disusun secara baris dan kolom dan ditempatkan pada kurung biasa atau kurung siku.

Materi

adalah setiap objek atau bahan yang membutuhkan ruang, yang jumlahnya diukur oleh suatu sifat yang disebut massa.

N

Navigasi

adalah penentuan kedudukan (*position*) dan arah perjalanan baik di medan sebenarnya.

Numerik

adalah sebuah simbol atau kumpulan dari simbol yang merepresentasikan sebuah bilangan.

O

Operator

adalah penghubung antar operand.

Output

adalah data keluaran yang dihasilkan berdasarkan inputan dan proses yang terjadi pada sebuah algoritma.

Online

adalah menunjukkan keadaan terhubung di mana suatu keadaan komputer yang dapat saling bertukar informasi karena sudah terhubung ke sebuah internet.

Operasional

adalah konsep yang bersifat abstrak untuk memudahkan pengukuran suatu variabel.

Operasi Dasar

adalah penjumlahan, pengurangan, perkalian dan pembagian.

P

Proses

adalah tahap pengolahan data, baik menggunakan logika proses maupun model matematika yang sesuai.

Pseudo

adalah imitasi/mirip/menyerupai.

Pseudocode

adalah bentuk penyajian algoritma dengan menggunakan struktur bahasa tertentu.

Paradigma

adalah model utama, pola atau metode (untuk meraih beberapa jenis tujuan). Seringkali **paradigma** merupakan sifat yang paling khas atau dasar dari sebuah teori atau cabang ilmu.

Penalaran

adalah proses berpikir yang bertolak dari pengamatan indera (pengamatan empirik) yang menghasilkan sejumlah konsep dan pengertian.

Pakar

adalah ahli ialah seseorang yang banyak dianggap sebagai sumber tepercaya atas teknik maupun keahlian tertentu yang bakatnya untuk menilai dan memutuskan sesuatu dengan benar dan baik.

Pengolahan

adalah kumpulan metode dan teknik yang digunakan untuk mengubah bahan mentah menjadi bahan jadi.

Perangkat Lunak

adalah istilah khusus untuk data yang diformat, dan disimpan secara digital, termasuk program komputer, dokumentasinya, dan berbagai informasi yang bisa dibaca, dan ditulis oleh komputer.

Proposisi

adalah kata yang merangkaikan kata-kata atau bagian kalimat dan biasanya diikuti oleh nomina atau pronomina.

Pernyataan Matematika

adalah tunggal maupun majemuk operasi **matematika** terhadap suatu **pernyataan**, baik tunggal maupun majemuk.

Persamaan

adalah suatu pernyataan matematika dalam bentuk simbol yang menyatakan bahwa dua hal adalah persis sama.

Pertidaksamaan

adalah kalimat terbuka memuat tanda ketidaksamaan.

R

Register

adalah sebagian kecil memori komputer yang dipakai untuk tempat penampungan data dengan ketentuan bahwa data yang terdapat dalam **register** dapat diproses dalam berbagai operasi dengan melihat berapa besar kemampuan menampung **register** tersebut.

Relasi

adalah hubungan; perhubungan; pertalian.

S

Sistem adalah sekelompok komponen dan elemen yang digabungkan menjadi satu untuk mencapai tujuan tertentu.

INDEKS

A

akar persamaan kuadrat, *55*
Akumulator, *164, 184*
algoritma, *22*
algoritma komputasi, *10*
Aplikasi Aritmetika, *59*
Aritmetika Dasar, *59*
array 1 dimensi, *209*
Ascending, *180*

B

berpikir komputasi, *1, 2, 3, 10, 12, 22, 25, 27, 28*
Boolean, *87, 91, 98*

C

Central Processing Unit (CPU), *7*

D

Data, *84*
Do-While, *177, 181, 182*

E

elemen array, *209*

F

Faktorial, *196*
flowchat, *23*
For, *177*

I

ilmuwan komputer, *4, 15, 18*
Inisialisasi, *124, 148*
INPUT, *41*
Instruksi, *41*
Integer, *98*

K

Kalimat Deskriptif, *184*
Karakter, *98*
Kombinasi, *132*
Komponen keluaran, *50, 54, 153*
Komponen masukan, *12, 50*
komponen pengolahan data, *1, 4*

Komponen proses, *4, 44*
Konstanta, *93*
Kounter, *135, 137*

L

Larik, *190*
Logika Dasar, *84*
logika matematika, *31, 34, 53*

M

Matriks, *193*
mesin computer, *53*
Multiplikator, *146, 173*

N

navigasi, *11*
Nested For, *165*

O

operand, *77*
Operasi Aritmetika, *39, 73*
operasional, *10*
operator, *54*
Operator Aritmatika, *77*
Operator Logika, *83*
Operator Relasi, *83*
OUTPUT, 36

P

Pembagian, *110*
Penalaran matematika, *8*
penalaran matematika kreatif, *32*
penalaran.com, *23, 24, 54, 72*
pengguna, *29*
pengolahan data, *1, 29*
Penjumlahan, *23, 78*
Percabangan, *86, 115*
Percabangan Bentuk Tunggal, *117*
pernyataan matematika, *35*
Pernyataan matematika, *35*
persamaan kuadrat, *55*
Perulangan, *145, 146*
pseudo code, *10, 36, 40, 41*

R

Real, *98*

S

sigma, *158, 170*
String, *98*

T

Tabel Kebenaran, *60*
Tabel Tracing, *145, 156, 168*
Terminasi, *148*
Tipe data, *75, 98*
tools, *12*

V

variabel, *39, 41, 42*
Variabel, *83, 86*
Variabel Majemuk, *211*
Vektor, *211, 241*

W

While, *181, 182*

-oo0o-

String

adalah tipe data untuk teks yang merupakan gabungan huruf, angka, *whitespace* (spasi), dan berbagai karakter.

Tipe Data adalah pola penyajian data dalam memori komputer, seperti alamat yang dapat menunjukkan di mana objek data disimpan.

Tombol

adalah sebuah homonim karena arti-artinya memiliki ejaan dan pelafalan yang sama tetapi maknanya berbeda.

Tool

adalah alat bantu.

Transformasi

adalah sebuah proses perubahan secara berangsur-angsur sehingga sampai pada tahap *ultimate,* perubahan yang dilakukan dengan cara memberi respon terhadap pengaruh unsur eksternal dan internal yang akan mengarahkan perubahan dari bentuk yang sudah dikenal sebelumnya melalui proses menggandakan secara berulang-ulang.

V

Variabel

adalah suatu lokasi memori komputer yang digunakan untuk menampung dan menyimpan data yang akan diolah.

-oo0o-

TENTANG PENULIS

Dr. Dra. Sulis Janu Hartati, M.T., terlahir di Kediri, pada tanggal 22 Januari 1964. Sekolah Dasar sampai Menengah ditempuhnya di kota kelahirannya. Sementara S1 sampai dengan S3 ditempuh di kota Pahlawan, Surabaya. S1 diselesaikan pada bulan Agustus 1987, MIPA Matematika UNAIR Surabaya. Berkarir sebagai dosen sejak Maret 1988 di STIKOM Surabaya. Tiga belas tahun kemudian menempuh S2 di Teknik Informatika ITS Surabaya, dan lulus pada tahun 1998. Sembilan tahun kemudian baru melanjutkan S3 Pendidikan Matematika di UNESA Surabaya. S3 diselesaikan pada Pebruari 2012. Sejak bulan Maret 2015 pindah *home base* ke Program Studi Pendidikan Matemtaika, FKIP, Universitas Dr. Soetomo Surabaya.

Aktif mengikuti hibah DRPM sejak tahun 2010. Hasil karyanya selama ini banyak dipublikasikan melalui prosiding nasional dan internasional, serta jurnal nasional dan internasional. Karya-karyanya focus pada topik algoritma, pembelajaran matematika dan keterkaitannya dengan pembuatan algoritma komputasi. Penulis juga rajin mengurus Hak Cipta dari karyanya sejak 2016. Sampai dengan saat ini, sudah ada 5 karya yang memiliki HKI, dua diantaranya berupa program computer. Karya-karya tersebut dapat diakses melalui https://scholar.google.co.id/citations

?user=tPLOvk0AAAAJ&hl=id. Sementara menerbitkan buku baru ditekuninya pada tahun 2017. Buku pertamanya yang ber-ISBN berjudul *"Pembelajaran Penalaran Matematika pada Algoritma Komputasi"*.

Neny Kurniati, S.Kom., M.T., lahir di Sidoarjo tanggal 29 Juni 1971. Setelah menyelesaikan SD, SLTP, dan SLTA di Sidoarjo, melanjutkan pendidikan S1 dan S2 di Surabaya. Pendidikan S1 dituntaskan pada tahun 1996 untuk program studi Manajemen Informatika di STIKOM Surabaya. Selepas lulus S1, di tahun yang sama menjadi dosen luar biasa di STIKOM Surabaya. Tahun 1999 sampai dengan 2016 menjadi dosen tetap di Politeknik SURABAYA sehingga memutuskan menempuh Pendidikan Magister Terapan di Politeknik Elektronika Negeri Surabaya untuk program studi Teknik Informatika dan Komputer, lulus tahun 2016. Sejak September 2018 sampai sekarang memutuskan menjadi dosen tetap di Universitas Nahdlatul Ulama Sidoarjo (UNUSIDA).

Sejak awal menjadi dosen sampai sekarang dipercaya mengampu mata kuliah Algoritma dan Pemrograman, selain mata kuliah Struktur Data, Pemrograman Berorientasi Objek, Pengolahan Citra Digital, dan Grafika Komputer. Berdasarkan pengalaman mengajar Algoritma dan Pemrograman dengan berbagai macam kemampuan pemahaman dan penalaran mahasiswa, sehingga memutuskan konsentrasi penelitian dan penulisan buku bidang Algoritma Komputasi'

Anik Vega Vitianingsih, S.Kom., MT, terlahir di Bojonegoro, pada tanggal 12 Juni 1981. Beliau sampai saat ini fokus pada bidang penelitian Sistem Informasi Geografis untuk Analisa Spasial pada bidang topik tertentu yang dimulai dari riset pertamanya tahun 2004 saat menyelesaikan Tugas Akhir. S1 diselesaikan pada bulan Maret Tahun 2004 di Universitas Dr. Soetomo. Berkarir sebagai dosen sejak lulus kuliah S1 Tahun 2004 di Prodi Teknik Informatika Universitas Dr. Soetomo, kemudian melanjutkan studi lanjut S2 di Jaringan Cerdas Multimedia ITS Surabaya

dengan bidang topik Teknologi Game, dan lulus pada tahun 2011. Tahun 2017 sampai sekarang semester VI Beliau melanjutkan Studi Lanjut S3 di Universiti Teknikal Malaysia Melaka-Malaysia dengan topik Penelitian Spatial Analysis.

Aktif mengikuti hibah DRPM sejak tahun 2013. Hasil karyanya selama ini banyak dipublikasikan melalui prosiding nasional dan internasional, serta jurnal nasional dan internasional. Karya-karyanya banyak yang fokus pada bidang Sistem Informasi Geografis. Penulis juga rajin mengurus Hak Cipta dari karyanya. Sampai dengan saat ini, sudah ada 8 karya yang memiliki HKI, dua diantaranya berupa program komputer. Karya-karya tersebut dapat diakses melalui https://scholar.google.co.id/citations ?user=acUhDoIAAAAJ&hl=id.

Endra Rahmawati, S.Kom., M.Kom. lahir di Surabaya pada tanggal 12 Oktober 1987. Ia memperoleh Gelar Sarjana dari Jurusan S1 Teknik Informatika Universitas 17 Agustus 1945 Surabaya pada tahun 2009. Karir pertamanya di bidang IT Support selama 1 tahun memberikan semangat baginya untuk menempuh S2 di bidang Teknologi Informasi di STTS Surabaya hingga tahun 2012. Sebagai Dosen Tetap di Universitas Dinamika Surabaya, penulis mengampu mata kuliah Pemrograman Dasar yang lebih fokus ke *Java Programming*.

Adapun beberapa karya ilmiah hasil penelitiannya telah diterbitkan di beberapa jurnal nasional dengan Topik Multimedia, Pengolahan Citra Digital, Teknik Informatika, dan Sistem Informasi.

Norma Ningsih, S.ST., M.T. lahir di Sampang Madura pada tanggal 29 September 1990. Pendidikan S1 ditempuh di Politeknik Elektronika Negeri Surabaya pada jurusan Teknik Telekomunikasi. sedangankan program Magister diselesaikan pada Institut Teknologi Sepuluh Nopember pada jurusan Teknik Elektro spesialisasi Teknik Telekomunikasi. Sejak 2015 hingga sekarang telah bekerja di Institut Bisnis dan Infromatika

Stikom Surabaya, sekarang menjadi Universitas Dinamika dan mengampu beberapa mata kuliah terkait *Software Development* seperti Pemrograman, Analysis dan Desain hingga Pengujian Sistem. Penelitian yang sudah di terbitkan berada dalam topik *software engineering* dan jaringan serta sistem sinyal radar

-oo0o-