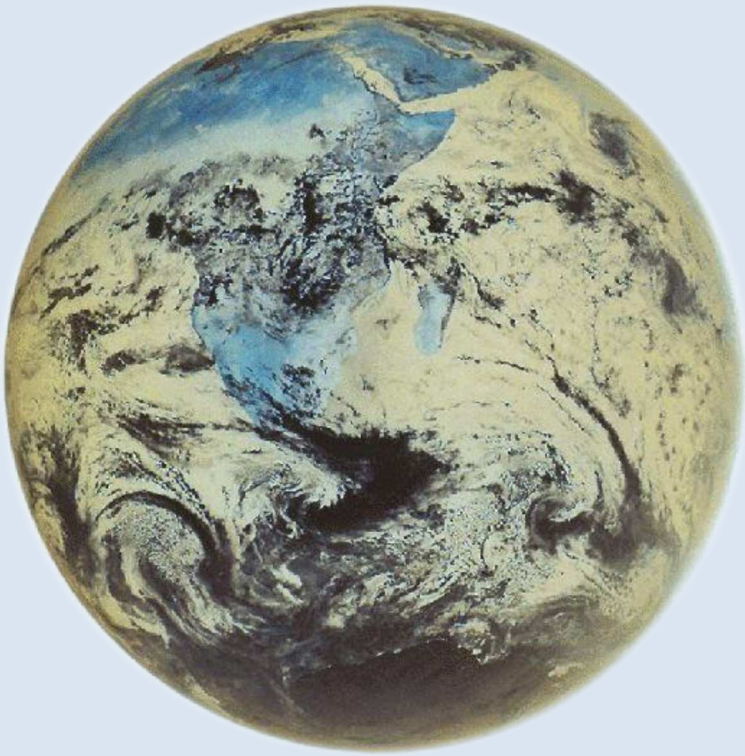
I Dewa Gede Rama

Algoritma DAN Pemrograman Java



PRAKATA

Puji syukur saya ucapkan kepada Tuhan Yang Maha Esa karena berkat bimbingan-Nya buku sederhana ini bisa diselesaikan tepat pada waktunya. Terima kasih saya ucapkan kepada orang tua yang selalu membimbing dan mendoakan saya. Terima kasih juga saya ucapkan kepada teman-teman yang selalu memberi semangat dan motivasi.

Perkembangan dunia yang semakin cepat membawa banyak pengaruh dan perubahan dalam kehidupan, salah satunya pemrograman. Pemrograman merupakan hal yang sangat menarik. Hal tersebut menyebabkan pemrograman semakin diminati oleh banyak kalangan, sehingga hampir disetiap profesi selalu ada penerapan teknologi pemrograman.

Buku ini ditujukan untuk siapa saja yang ingin mengenal pemrograman. Buku ini disusun secara ringkas, padat dan jelas, sehingga akan sangat mudah dipelajari. Buku ini memuat pengantar algoritma dan pemrograman dengan bahasa Java, sehingga hanya akan membahas dasar-dasar pemrograman saja seperti: tipe data, percabangan, perulangan, larik, metode, rekursi, pengurutan dan pencarian.

Buku ini ditulis dengan sasaran untuk memperkenalkan pemrograman Java kepada kalangan akademis, khususnya mahasiswa informatika. Untuk lebih mendalami, disarankan untuk lanjut mempelajari buku-buku berikutnya seperti: Metode Analisis Perancangan Program, Struktur Data, Pemrograman Berorientasi Objek, Basis Data, dan lain-lain.

Akhir kata, saya ucapkan terima kasih.

Bandung, Desember 2008

Dewa Rama

DAFTAR ISI

1. TIPE DATA

1.1 Definisi

Setiap data memiliki informasi yang khas. Informasi inilah yang membedakan suatu data dengan data yang lain. Informasi yang dimiliki oleh suatu data antara lain *tipe data*, *variabel* dan *nilai*.

Tipe data adalah jenis suatu data. Jenis data bisa berupa bilangan bulat, bilangan pecahan, karakter dsb. *Variabel adalah nama suatu data*. Nama inilah yang merupakan ciri pembeda utama suatu data dengan data lainnya. *Nilai adalah harga suatu data*. Nilai inilah yang nantinya dioperasikan saat program berjalan.

contoh :

1

Data : umur mahasiswa
Tipe data : bilangan bulat
Variabel : umur_mhs
Nilai : 17

2

Data : kecepatan mobil
Tipe data : bilangan pecahan
Variabel : v
Nilai : 86.77792

3

Data : alamat rumah
Tipe data : kalimat
Variabel : almt
Nilai : “Jl. Sekeloa No. 48A”

4

Data : huruf kelima
Tipe data : karakter
Variabel : fifth_char
Nilai : “e”

5

Data : nilai π
Tipe data : konstanta
Variabel : pi
Nilai : 3.14

6

Data : pengecek kebenaran
Tipe data : logika
Variabel : pengecek_kebenaran
Nilai : true

Secara umum tipe data dibagi dua, yaitu *tipe dasar* dan *tipe bentukan*. Tipe dasar adalah tipe yang sudah tersedia dalam suatu bahasa pemrograman, misalnya bilangan bulat (*integer*), bilangan pecahan (*real*), tetapan (*const*), karakter (*char*), kalimat (*string*) dan logika (*boolean*). Sedangkan tipe bentukan adalah tipe yang dibuat sendiri dengan menggabungkan tipe dasar, misalnya larik (*array*) dan rekaman (*record*).

1.2 Tipe Dasar

No	Tipe Data	Algoritma	Java	Contoh
1	Bilangan bulat	integer	long int short	-25, 0, 32
2	Bilangan pecahan	real	float double	-7.5, 0.25, 12E-2
3	Konstanta	const	final long final int final short final float final double	π (3.14)
4	Karakter	character	char	A, z, 1, *spasi*, *enter*
5	Kumpulan karakter, bisa berupa kalimat atau kata	string	String	“ayam”, “1250”
6	Logika	boolean	boolean	true, false

1.3 Penulisan Tipe Dasar

Algoritma	Java
namavariabel : *tipedata*	*tipedata namavariabel*;

contoh :

No	Tipe Data	Algoritma	Java
1	Bilangan bulat	n : integer jumlah : integer bilangan : integer	int n; int jumlah; long bilangan;
2	Bilangan pecahan	x : real hasilpembagian : real biaya : real	double x; double hasilpembagian; float biaya;
3	Konstanta	pi ←3.14 : const	final double pi = 3.14;
4	Karakter	inisial : char hurufkelima : char	char inisial; char hurufkelima;
5	Kata	masukanx : string namax : string nimx : string	String masukanx; String namax; String nimx;
6	Logika	apakahbenar : boolean kondisimengulang : boolean	boolean apakahbenar; boolean kondisimengulang;

1.4 Penulisan Tipe Bentukan

Keterangan	Algoritma	Java

Deklarasi tipe bentukan sebagai prosedur	type *namatipebentukan* : (*semua_ nilai_yang_diperbolehkan*)	enum *namatipebentukan*
		{
		semua_nilai_yang_ diperbolehkan;
		}

Deklarasi variabel	*namavariabel* : *namatipebentukan*	*namatipebentukan namavariabel*;
Pengisian nilai variabel	*namavariabel* ← *nilai_yang_diperbolehkan*	*namavariabel* = *namatipebentukan*. *nilai_yang_diperbolehkan*;

contoh 1 :

Sebagai contoh kita akan membuat tipe bentukan *umur mahasiswa* yang nilainya antara lain 17, 18, 19, 20, 21, 22, 23.

Keterangan	Algoritma	Java

Deklarasi tipe bentukan sebagai prosedur	type umurmahasiswa : (17, 18, 19, 20, 21, 22, 23)	enum umurmahasiswa
		{
		17, 18, 19, 20, 21, 22, 23;
		}

Deklarasi variabel	umursaya : umurmahasiswa	umurmahasiswa umursaya;
Pengisian nilai variabel	umursaya ← 17	umursaya = umurmahasiswa.17;

contoh 2 :

Contoh lainnya, seandainya kita akan membuat tipe bentukan *hari* yang nilainya antara lain senin, selasa, rabu, kamis, jumat, sabtu, minggu.

Keterangan	Algoritma	Java

Deklarasi tipe bentukan sebagai prosedur	type hari : (senin, selasa, rabu, kamis, jumat, sabtu, minggu)	enum hari
		{
		senin, selasa, rabu, kamis, jumat, sabtu, minggu;
		}

Deklarasi variabel	today : hari	hari today;
Pengisian nilai variabel	today ← minggu	today = hari.minggu;

1.5 Operator

Operator adalah media yang digunakan untuk memproses data sehingga memberikan hasil. Secara umum operator dibagi tiga, yaitu *operator aritmatika*, *operator relasi* dan *operator logika*. Operator aritmatika digunakan untuk mengoperasikan data secara matematika, misalnya (+), (-). Operator relasi digunakan untuk membandingkan dua buah data, misalnya (<), (>). Operator logika digunakan untuk mengaitkan dua buah kondisi menjadi sebuah kondisi, misalnya (dan), (atau).

(a) Operator aritmatika

No	Jenis Operasi	Tipe Masukan	Tipe Keluaran	Algoritma	Java
1	Penjumlahan (+)	integer real	integer real	+	+
2	Pengurangan (-)	integer real	integer real	-	-
3	Perkalian (*)	integer real	integer real	*	*
4	Pembagian (/)	integer real	integer real	/	/
5	Bilangan bulat pembagian (div)	integer	integer	div	/
6	Sisa pembagian (mod)	integer	integer	mod	%

(b) Operator relasi

No	Jenis Operasi	Algoritma	Java
1	Sama dengan (=)	=	==
2	Tidak sama dengan (≠)	<>	!=
3	Lebih dari (>)	>	>
4	Kurang dari (<)	<	<
5	Lebih dari sama dengan (≥)	≥	>=
6	Kurang dari sama dengan (≤)	≤	<=

(c) Operator logika

No	Jenis Operasi	Algoritma	Java
1	Dan (and)	AND	&&
2	Atau (or)	OR	||
3	Bukan (negasi)	NOT	!

Kondisi A	Kondisi B	A dan B	A atau B	Bukan A	Bukan B
B	B	B	B	S	S
B	S	S	B	S	B
S	B	S	B	B	S
S	S	S	S	B	B

Tabel kebenaran untuk operator logika

1.6 Komentar

Komentar adalah bagian *kode program* yang tidak dibaca saat program dijalankan. Komentar sangatlah penting untuk memperjelas *kode program* agar lebih mudah dipahami. Berikut ini adalah tata cara penulisan komentar.

Jenis Komentar	Algoritma	Java
Komentar untuk satu baris	{*komentar*}	*//komentar*

Komentar untuk beberapa baris	{
	.....
	komentar
	.....
	}
		/*
		.....
		komentar
		.....
		*/

2. PERCABANGAN

Percabangan atau *branching* merupakan sebuah blok program yang menyatakan bahwa sebuah aksi akan dijalankan jika kondisi sebuah percabangan terpenuhi. Pada umumnya konsep percabangan dibagi dua, yaitu *percabangan if* dan *percabangan case*.

2.1 Percabangan If

(a) Satu kasus

bentuk umum :

Algoritma	Java

if (*kondisi*) then
aksi
end if
	if (*kondisi*)
	{
	aksi;
	}

contoh :

Algoritma	Java

if (bilangan > 0) then
write (“bilangan positif”)
end if
	if (bilangan > 0)
	{
	System.out.println(“bilangan positif”);
	}

(b) Dua kasus

bentuk umum :

Algoritma	Java

if (*kondisi1*) then
aksi1
else {bukan kondisi1}
aksi2
end if
	if (*kondisi1*)
	{
	aksi1;
	}
	else //bukan kondisi1
	{
	aksi2;
	}

contoh :

Algoritma	Java

if (bilangan mod 2 = 0) then
write (“bilangan genap”)
else {bilangan mod 2 <> 0}
write (“bilangan ganjil”)
end if
	if (bilangan % 2 == 0)
	{
	System.out.println(“bilangan genap”);
	}
	else // bilangan mod 2 != 0
	{
	System.out.println(“bilangan ganjil”);
	}

(c) Banyak kasus

bentuk umum :

Algoritma	Java

if (*kondisi1*) then
aksi1
else
if (*kondisi2*) then
aksi2
else
if (*kondisi3*) then
aksi3
else
if (*kondisin*) then
aksin
else {bukan kondisi 1, 2, 3, n}
aksix
end if
end if
end if
end if
	if (*kondisi1*)
	{
	aksi1;
	}
	else if (*kondisi2*)
	{
	aksi2;
	}
	else if (*kondisi3*)
	{
	aksi3;
	}
	else if (*kondisin*)
	{
	aksin;
	}
	else //bukan kondisi 1, 2, 3, n
	{
	aksix;
	}

contoh :

Algoritma	Java

if (hobi = 1) then
write (“jalan-jalan”)
end if
else if (hobi = 2) then
	if (hobi==1)
	{
	System.out.println(“jalan-jalan”);
	}

write (“bermain”)
end if
else if (hobi = 3) then
write (“nonton”)
end if
else if (hobi = 4) then
write (“dengar lagu”)
end if
else {hobi <> 1, 2, 3, 4}
write (“belajar”)
end else
	else if (hobi==2)
	{
	System.out.println(“bermain”);
	}
	else if (hobi==3)
	{
	System.out.println(“nonton”);
	}
	else if (hobi==4)
	{
	System.out.println(“dengar lagu”);
	}
	else //hobi != 1, 2, 3, 4
	{
	System.out.println(“belajar”);
	}

2.2 Percabangan Case

bentuk umum :

Algoritma	Java

case (*namavariabel*)
namavariabel = 1 : *aksi1*
namavariabel = 2 : *aksi2*
namavariabel = 3 : *aksi3*
namavariabel = 4 : *aksi4*
namavariabel = 5 : *aksi5*
namavariabel = n : *aksin*
otherwise : *aksix*
end case
	switch (*namavariabel*)
	{
	case 1 : *aksi1*;
	break;
	case 2 : *aksi2*;
	break;
	case 3 : *aksi3*;
	break;
	case 4 : *aksi4*;
	break;
	case 5 : *aksi5*;
	break;
	case n : *aksin*;
	break;
	default : *aksix*;
	}

contoh :

Algoritma	Java

case (hari)
hari = 1 : write (“senin”)
hari = 2 : write (“selasa”)
hari = 3 : write (“rabu”)
hari = 4 : write (“kamis”)
hari = 5 : write (“jumat”)
hari = 6 : write (“sabtu”)
hari = 7 : write (“minggu”)
otherwise : write (“salah”)
end case
	switch (hari)
	{
	case 1 : System.out.println(“senin”);
	break;
	case 2 : System.out.println(“selasa”);
	break;
	case 3 : System.out.println(“rabu”);
	break;
	case 4 : System.out.println(“kamis”);
	break;
	case 5 : System.out.println(“jumat”);
	break;
	case 6 : System.out.println(“sabtu”);
	break;
	case 7 : System.out.println(“minggu”);
	break;
	default : System.out.println(“salah”);
	}

3. PERULANGAN

Perulangan atau *looping* adalah bagian kode program yang bertugas melakukan suatu proses terus-menerus sampai kondisi berhenti terpenuhi. Secara umum perulangan dibagi tiga, yaitu *perulangan for*, *perulangan while-do* dan *perulangan repeat-until*.

3.1 Perulangan For

Perulangan *for* digunakan jika sudah dapat dipastikan kapan pengulangan berhenti. Dengan kata lain, jumlah perulangan yang dibutuhkan sudah diketahui sebelumnya. Ada dua model perulangan dalam *for*, yaitu *perulangan for naik* dan *perulangan for turun*.

(a) Perulangan for naik

bentuk umum :

Algoritma	Java

for *namavariabel* ← *nilaiawal* to *nilaiakhir* do
  proses_perulangan
end for
	for (*namavariabel* = *nilaiawal* ; *namavariabel* *operatorrelasi nilaiakhir* ; *namavariabel*++)
	{
	  proses_perulangan;
	}

contoh :

Algoritma	Java

for i ← 1 to 10 do
  write(“angka ”, i)
end for
	for (i=1 ; i<=10 ; i++)
	{
	  System.out.println(“angka ” +i);
	}

(b) Perulangan for turun

bentuk umum :

Algoritma	Java

for *namavariabel* ← *nilaiawal* downto *nilaiakhir* do
  proses_perulangan
end for
	for (*namavariabel* = *nilaiawal* ; *namavariabel* *operatorrelasi nilaiakhir* ; *namavariabel*--)
	{
	  proses_perulangan;
	}

contoh :

Algoritma	Java

for i ← 10 downto 1 do
write(“angka ”, i)
end for
	for (i=10 ; i>=1 ; i--)
	{
	System.out.println(“angka ” +i);
	}

3.2 Perulangan While-Do

Perulangan *while-do* biasanya digunakan jika banyaknya perulangan tidak diketahui. Misalnya pada program *login password*, dimana program akan terus mengulang meminta password selama password masih salah, dan jika password benar proses perulangan berhenti.

bentuk umum :

Algoritma	Java

while *kondisimengulang* do
aksi
end while
	while (*kondisimengulang*)
	{
	aksi;
	}

contoh :

Algoritma	Java

password : string
 	
write (“masukkan password”)
read (password)
 	
while password <> “123” do
write (“password salah”)
write (“masukkan password”)
read (password)
end while
	String password;
	
	password = JoptionPane.showInput
	Dialog(null,”masukkan password”);
	
	while (!password.equals(“123”))
	{
	JoptionPane.showMessageDialog
	(null,”password salah”);
	
	password = JoptionPane.showInput
	Dialog(null,”masukkan password”);
	}

3.3 Perulangan Repeat-Until

Perulangan *repeat-until* juga dapat digunakan jika jumlah perulangan tidak diketahui, namun prinsip kerjanya berbeda dengan perulangan *while-do*. Pada perulangan *while-do* program akan menyesuaikan variabel dengan kondisi perulangan, jika tidak memenuhi maka perulangan tidak dilakukan. Berbeda dengan *repeat-until* karena minimal perulangan akan dijalankan sekali sebelum masuk ke kondisi perulangan. Hal tersebut dapat terjadi karena perulangan *repeat-until* meletakkan kondisi perulangannya dibagian akhir.

bentuk umum :

Algoritma	Java

repeat
aksi
until *kondisiberhenti*
	do
	{
	aksi;
	}
	while (*kondisiperulangan*);

contoh :

Algoritma	Java

password : string
 	
repeat
read (password)
until password = “sandi123”
	String password;
	
	do
	{
	password = JoptionPane.showInput
	Dialog(null,”masukkan password”);
	}
	while (!password.equals(“sandi123”));

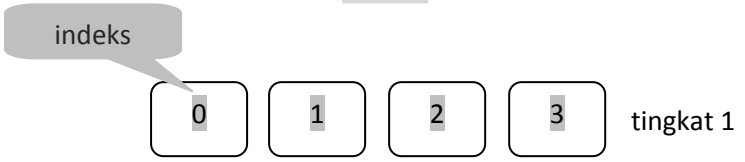
4. LARIK

4.1 Definisi

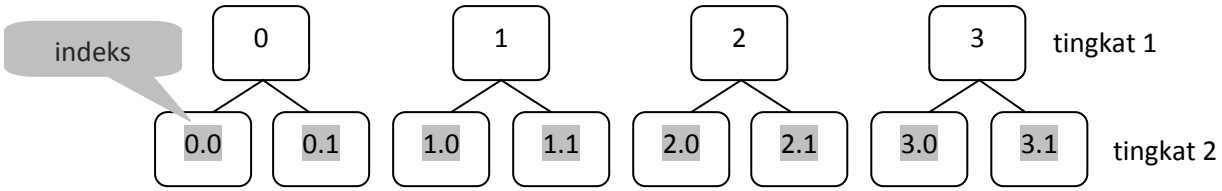
Larik atau *array* dapat didefinisikan sebagai tabel yang terstruktur. Sebuah array terdiri dari tabel-tabel yang dapat diisi dengan variabel-variabel bertipe sama. Array bertipe *integer* hanya dapat menampung integer, array bertipe *char* hanya dapat menampung karakter. Tiap tabel memiliki indeks (nomor tabel), pada java indeks dimulai dari 0 (nol). Tiap tabel dapat diisi oleh satu variabel.

Secara umum array dibedakan berdasarkan tingkatannya. Array 1 tingkat hanya memiliki tabel utama. Array 2 tingkat memiliki tabel utama dan sub-tabel. Array 3 tingkat memiliki tabel utama, sub-tabel dan sub-sub-tabel. Untuk memperjelas, perhatikan ilustrasi berikut:

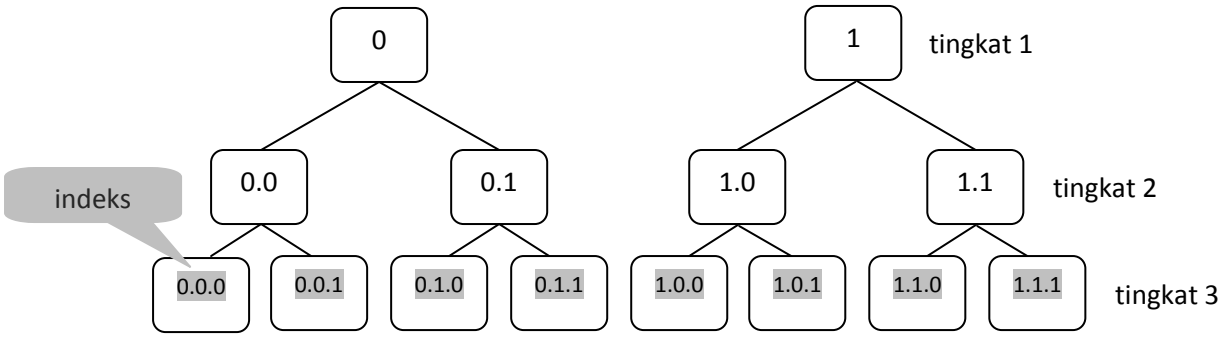
Array 1 tingkat : [4]



Array 2 tingkat : [4] [2]



Array 3 tingkat : [2] [2] [2]



4.2 Array 1 Tingkat

Array 1 tingkat hanya terdiri atas satu jenis indeks. Saat mendeklarasikan array kita harus menentukan banyaknya indeks dan tipe data untuk variabel.

(a) Deklarasi array

bentuk umum :

Algoritma	Java
namaarray : array [1...*jumlahindeks*] of *tipedata*	*tipedata namaarray* [] = new *tipedata* [*jumlahindeks*];

contoh :

Algoritma	Java
hari : array [1...7] of string	String hari [] = new String [7];
ruang : array [1...100] of integer	int ruang [] = new int [100];

(b) Mengisi indeks array

Bentuk umum :

Algoritma	Java
namaarray[*indeks*] ← *nilai*	*namaarray*[*indeks*] = *nilai*;

contoh :

Algoritma	Java
hari[1] ← “senin”	hari[0] = “senin”;
ruang[1] ← 45	ruang[0] = 45;

(c) Mengakses nilai pada indeks tertentu

Bentuk umum :

Algoritma	Java
namaarray[*indeks*]	*namaarray*[*indeks*];

contoh :

Algoritma	Java
hari[3]	hari[2];
ruang[50]	ruang[49];

4.3 Array 2 Tingkat

Pada dasarnya konsep array 2 tingkat sama seperti array 1 tingkat, hanya saja pada deklarasi array 2 tingkat ada dua macam indeks yang harus kita tentukan. Salah satu penggunaan array 2 tingkat adalah pada operasi matriks, dimana saat mendeklarasikan array kita harus menentukan *jumlah indeks kolom* dan *jumlah indeks baris*.

Berikut ini adalah penjelasan untuk array matriks yang terdiri atas 2 kolom dan 3 baris.

(a) Deklarasi array

bentuk umum :

Algoritma	Java
namaarray : array [1...*jumlah_indeks_pertama*] [1...*jumlah_indeks_kedua*] of *tipedata*	*tipedata namaarray* [] [] = new *tipedata* [*jumlah_indeks_pertama*] [*jumlah_indeks_kedua*];

contoh :

Algoritma	Java
matriks : array [1...2] [1...3] of integer	int matriks [] [] = new int [2] [3];

(b) Mengisi indeks array

Bentuk umum :

Algoritma	Java
namaarray[*indekspertama*] [*indekskedua*] ← *nilai*	*namaarray*[*indekspertama*] [*indekskedua*] = *nilai*;

contoh :

Algoritma	Java
matriks[1] [3]← 2	matriks[0][2] = 2;

(c) Mengakses nilai pada indeks tertentu

Bentuk umum :

Algoritma	Java
namaarray[*indekspertama*] [*indekskedua*]	*namaarray*[*indekspertama*] [*indekskedua*];

contoh :

Algoritma	Java
matriks[2] [2]	matriks[1] [1];

5. METODE

5.1 Definisi

Metode atau *method* adalah sebuah blok program yang berisikan instruksi singkat yang mengerjakan tugas atau proses tertentu. Metode bersifat khas, artinya metode yang satu sangat berbeda dengan metode lainnya. Metode memiliki fungsi yang unik, sehingga dengan adanya metode adalam sebuah program, program akan menjadi semakin sederhana.

Saat kita bertujuan membuat program yang berisikan beberapa instruksi yang sama, menggunakan metode merupakan pilihan yang tepat. Kita cukup membuat sebuah metode yang berisikan instruksi. Selanjutnya, jika kita membutuhkan instruksi tersebut, kita tinggal memanggil metode yang telah kita buat tadi, demikian seterusnya.

Jadi kesimpulannya, dengan metode kita tidak perlu membuat beberapa instruksi yang sama dalam sebuah program. Saat instruksi dibutuhkan, kita tinggal memanggil metode yang memuat instruksi tersebut. Metode dapat dipanggil/digunakan lebih dari sekali.

Dalam Java, ada dua macam metode, yaitu *Prosedur* dan *Fungsi*. *Prosedur* merupakan metode yang hanya berisikan proses, sedangkan *Fungsi* adalah metode yang berisikan proses dan menghasilkan nilai keluaran.

5.2 Prosedur

Berdasarkan jenis *input-output* prosedur dibagi menjadi empat macam, yaitu *Prosedur Input*, *Prosedur Output*, *Prosedur Input dan Output*, dan *Prosedur Input/Output*. Berikut ini adalah cara kerja dari masing-masing jenis prosedur tersebut.

Prosedur Input. Pada awalnya prosedur sudah mendeklarasikan variabel input. Selanjutnya variabel input dimasukkan kedalam prosedur. Setelah prosedur dijalankan nilai variabel input tersebut tidak berubah.

Prosedur Output. Pada awalnya prosedur sudah mendeklarasikan variabel output. Nilai variabel output akan didefinisikan saat prosedur dijalankan.

Prosedur Input dan Output. Pada awalnya prosedur sudah mendeklarasikan variabel input dan variabel output. Selanjutnya variabel input dimasukkan kedalam prosedur. Setelah prosedur dijalankan nilai variabel input tidak berubah. Hasil yang diperoleh dari proses akan didefinisikan sebagai nilai output.

Prosedur Input/Output. Pada awalnya prosedur sudah mendeklarasikan variabel input/output. Selanjutnya variabel input dimasukkan kedalam prosedur. Setelah prosedur

dijalankan nilai variabel input akan berubah yang kemudian akan didefinisikan menjadi variabel output.

(a) Deklarasi prosedur

Cara mendeklarasikan prosedur berbeda-beda, tergantung pada jenisnya. Pada tabel dibawah ini akan dijelaskan cara mendeklarasikan setiap jenis prosedur tersebut.

bentuk umum :

Jenis	Algoritma	Java

Prosedur Input	procedure *namaprosedur* (input : *namavariabel1* : *tipedata1, namavariabel-n* : *tipedata-n*)
	
	{*proses*}
	
	end procedure
		public static void *namaprosedur* (*tipedata1 namavariabel1*, *tipedata-n namavariabel-n*)
		{
		//*proses*
		}

Prosedur Output	procedure *namaprosedur* (output : *namavariabel1* : *tipedata1, namavariabel-n* : *tipedata-n*)
	
	{*proses*}
	
	end procedure
		public static void *namaprosedur* (*tipedata1 namavariabel1*, *tipedata-n namavariabel-n*)
		{
		//*proses*
		}

Prosedur Input dan Output	procedure *namaprosedur* (input : *namavariabelin1* : *tipedatain1, namavariabelin-n* : *tipedatain-n*, output : *namavariabelout1* : *tipedataout1, namavariabelout-n* : *tipedataout-n*)
	
	{*proses*}
	
	end procedure
		public static void *namaprosedur* (*tipedatain1 namavariabelin1*, *tipedatain-n namavariabelin-n*, *tipedataout1 namavariabelout1*, *tipedataout-n namavariabelout-n*)
		{
		//*proses*
		}

Prosedur Input/Output	procedure *namaprosedur* (input/ output : *namavariabel1* : *tipedata1, namavariabel-n* : *tipedata-n*)
	
	{*proses*}
	
	end procedure
		public static void *namaprosedur* (*tipedatainout1 namavariabelinout1*, *tipedatainout-n tipedatainout-n*)
		{
		//*proses*
		}

contoh :

Jenis	Algoritma	Java

Prosedur Input	procedure tulis (input : a : integer)
	
	write (a)
	
	end procedure
		public static void tulis (int a)
		{
		
		System.out.print(a);
		
		}

Prosedur Output	procedure baca (output : a : integer)
	
	a ← read (a)
	
	end procedure
		public static void baca (int a)
		{
		
		a = JoptionPane.showInputDialog (null,“masukkan nilai “ +a);
		
		}

Prosedur Input dan Output	procedure nilailain (input : a : integer output : b : integer)
	
	b ← a+5
	
	end procedure
		public static void nilailain (int a, int b)
		{
		
		b = a+5;
		
		}

Prosedur Input/Output	procedure nilaibaru (input/output : a : integer)
	
	a ← a*3
	
	end procedure
		public static void nilaibaru (int a)
		{
		
		a = a*3;
		
		}

(b) Memanggil prosedur

Setelah prosedur dideklarasikan, tentu saat instruksi pada prosedur tersebut dibutuhkan kita akan memanggil prosedurnya. Berikut ini adalah bentuk umum dan contoh memanggil prosedur.

bentuk umum :

Algoritma	Java
namaprosedur (*variabelinput*)	*namaprosedur* (*variabelinput*);

contoh :

Algoritma	Java
tulis (10)	tulis (10);

5.3 Fungsi

Fungsi adalah metode yang berisikan satu atau sekumpulan proses yang kemudian menghasilkan nilai keluaran.

(a) Deklarasi fungsi

bentuk umum :

Algoritma	Java

function *namafungsi*
(*namavariabelin1* : *tipedatain1*,
namavariabelin-n : *tipedatain-n*) ->
tipedataout
 	
{*proses*}
 	
return (*variabeloutput*)
 	
end function
	public static *tipedataout namafungsi*
	(*tipedatain1 namavariabelin1*,
	tipedatain-n namavariabelin-n)
	{
	
	//*proses*
	
	return(*variabeloutput*);
	
	}

contoh :

Algoritma	Java

function rumuskali (a : integer, b : integer) ->
integer
hasil : integer
hasil ← a*b
 	
return (hasil)
end function
	public static int rumuskali (int a, int b)
	{
	int hasil;
	hasil = a*b;
	
	return (hasil);
	}

(b) Memanggil fungsi

bentuk umum :

Algoritma	Java
namavariabel ← *namafungsi* (*variabelinput*)	*namavariabel* = *namafungsi* (*variabelinput*);

contoh :

Algoritma	Java
hasilperkalian ← rumuskali (2,5)	hasilperkalian = rumuskali (2,5);

6. REKURSI

6.1 Definisi

Rekursi atau *recurrence* dapat didefinisikan sebagai metode yang mengulang prosesnya sendiri. Saat sebuah rekursi dipanggil, proses akan dijalankan dan diulang sampai kondisi berhenti terpenuhi.

Dua hal yang selalu ada dalam sebuah rekursi adalah *Blok Rekursi* dan *Blok Basis*. *Blok Rekursi* merupakan kode program yang memuat kondisi mengulang. Blok inilah yang nantinya berfungsi mengulang prosesnya terus-menerus. *Blok Basis* merupakan kode program yang memuat kondisi berhenti. Blok inilah yang menyebabkan sebuah rekursi menghentikan prosesnya, sehingga diperoleh hasil akhir.

Rekursi digunakan pada metode yang menggunakan perhitungan/proses berulang, semisal perkalian, pangkat, dan faktorial. Perkalian merupakan perhitungan berulang dari proses penjumlahan. Pangkat merupakan perhitungan berulang dari proses perkalian. Begitu pula faktorial yang merupakan perhitungan berulang dari proses perkalian mundur.

6.2 Bentuk Umum Rekursi

(a) Konsep

Untuk menyusun rekursi, terlebih dahulu kita harus menentukan *konsep*. Konsep bukanlah bagian program, melainkan hanya merupakan cara kerja atau proses dasar sebuah rekursi. Konsep ini akan memudahkan kita dalam menyusun rekursi kedalam kode program.

Konsep berisikan urutan/langkah terperinci yang menjelaskan proses rekursi dari awal sampai akhir. Konsep terdiri atas *permisalan*, *langkah-langkah*, dan *hasil akhir*. *Permisalan* dapat berisikan nilai awal. *Langkah-langkah* meliputi urutan dan hasil sementara di setiap perulangan. *Hasil akhir* merupakan nilai dari proses rekursi.

(b) Deklarasi rekursi

Telah kita ketahui bersama bahwa rekursi merupakan metode. Oleh sebab itu cara mendeklarasikan rekursi mirip dengan cara mendeklarasikan metode. Ada sedikit perbedaan dalam mendeklarasikan rekursi, yaitu kita harus memisahkan antara blok perulangan dan blok penghentian.

Algoritma	Java

method* *namarekursi*
(*namavariabelin1* : *tipedatain1*,
namavariabelin-n : *tipedatain-n*) ->*tipedataout*
1 { if (*kondisiberhenti*) then
return(*nilainetral****)
2 { else {*kondisimengulang*}
return(*pemanggilanulang*)
end if
end method****
	public static *tipedataout*** *namarekursi*
	(*tipedatain1* *namavariabelin1*,
	tipedatain-n *namavariabelin-n*)
	{
	1 { if (*kondisiberhenti*)
	{
	return(*nilainetral****);
	}
	2 { else //*kondisimengulang*
	{
	return(*pemanggilanulang*);
	}
	}

ket:

* method = dapat berupa procedure atau function

** *tipedataout* = void untuk procedure dan tipe data tertentu untuk function

*** *nilainetral* = merupakan nilai yang tidak mengubah hasil sebenarnya

**** end method = dapat berupa end procedure atau end function

1 blok basis

2 blok rekursi

(c) Memanggil rekursi

Sama halnya dengan metode, rekursi juga akan dipanggil jika dibutuhkan. Saat rekursi dipanggil proses didalam rekursi akan berjalan, berulang, berhenti lalu memberikan hasil akhir. Cara memanggil rekursi juga mirip dengan cara memanggil metode.

Algoritma	Java
namavariabel ← *namarekursi* (*variabelinput*)	*namavariabel* = *namarekursi* (*variabelinput*);

6.3 Contoh Rekursi

Penerapan rekursi sangat luas, namun disini hanya akan dijelaskan beberapa contoh saja seperti *Rekursi Perkalian*, *Rekursi Pangkat*, dan *Rekursi Faktorial*.

Contoh 1 : Rekursi Perkalian dengan Proses Penjumlahan

(a) Konsep

misalnya : 5 x 3

maka : - perulangan pertama : 5 + (5x2)

- perulangan kedua : 5 + 5 + (5x1)

- perulangan ketiga : 5 + 5 + 5 + (5x0)

- perulangan keempat : 5 + 5 + 5 + 0

- hasil akhir = 15

Perkalian dengan 0 memberi hasil 0 dan merupakan kondisi berhenti

(b) Deklarasi rekursi

Algoritma	Java

function rekursikali (a : integer, b : integer) -> integer
if (b==0) then
return(0)
else {b>0}
return(a+rekursikali(a,(b-1)))
end if
end function
	public static int rekursikali (int a, int b)
	{
	if (b==0)
	{
	return(0);
	}
	else //b>0
	{
	return(a+rekursikali(a,(b-1)));
	}
	}

(c) Memanggil rekursi

Algoritma	Java
hasilkali ← rekursikali (5,3)	hasilkali = rekursikali (5,3);

Contoh 2 : Rekursi Pangkat dengan Proses Perkalian

(a) Konsep

misalnya : 2^4

maka : - perulangan pertama : 2 x (2^3)

- perulangan kedua : 2 x 2 x (2^2)
- perulangan ketiga : 2 x 2 x 2 x (2^1)
- perulangan keempat : 2 x 2 x 2 x 2 x (2^0)
- perulangan kelima : 2 x 2 x 2 x 2 x 1
- hasil akhir = 16

Pangkat 0 memberi hasil 1 dan merupakan kondisi berhenti

(b) Deklarasi rekursi

Algoritma	Java

function rekursipangkat (a: integer, b: integer) -> integer
if (b==0) then
return(1)
else {b>0}
return(a*rekursipangkat(a,(b-1)))
end if
end function
	public static int rekursipangkat (int a, int b)
	{
	if (b==0)
	{
	return(1);
	}
	else //b>0
	{
	return(a*rekursipangkat(a,(b-1)));
	}
	}

(c) Memanggil rekursi

Algoritma	Java
hasilpangkat ← rekursipangkat (2,4)	hasilpangkat = rekursipangkat (2,4);

Contoh 3 : Rekursi Faktorial dengan Proses Perkalian Mundur

(a) Konsep

misalnya : 6!

maka : - perulangan pertama : 6 x (5!)

- perulangan kedua : 6 x 5 x (4!)
- perulangan ketiga : 6 x 5 x 4 x (3!)
- perulangan keempat : 6 x 5 x 4 x 3 x (2!)
- perulangan kelima : 6 x 5 x 4 x 3 x 2 x (1!)
- perulangan keenam : 6 x 5 x 4 x 3 x 2 x 1 x (0!)
- perulangan ketujuh : 6 x 5 x 4 x 3 x 2 x 1 x 1
- hasil akhir = 720

Faktorial 0 memberi hasil 1 dan merupakan kondisi berhenti

(b) Deklarasi rekursi

Algoritma	Java

function rekursifaktorial (a : integer)
-> integer
if (a==0) then
return(1)
else {b>0}
return(a*rekursifaktorial(a -1))
end if
end function
	public static int rekursifaktorial (int a)
	{
	if (a==0)
	{
	return(1);
	}
	else //b>0
	{
	return(a*rekursifaktorial(a-1));
	}
	}

(c) Memanggil rekursi

Algoritma	Java
hasilfaktorial ← rekursifaktorial (6)	hasilfaktorial = rekursifaktorial (6);

7. PENGURUTAN

Pengurutan atau *sorting* adalah sekumpulan instruksi yang bertugas mengurutkan sekumpulan data. Ada dua jenis pengurutan yaitu pengurutan menaik (*ascending*) dan pengurutan menurun (*descending*). Berdasarkan cara kerjanya, pengurutan dibagi tiga yaitu *Pengurutan Sisipan* (*Insertion Sort*), *Pengurutan Seleksi* (*Selection Sort*), dan *Pengurutan Gelembung* (*Bubble Sort*).

Setiap metode pengurutan selalu didasarkan pada *kriteria*. Kriteria adalah suatu kondisi yang menyatakan minimum atau maksimum, tergantung pada jenis pengurutan, menaik atau menurun.

7.1 Pengurutan Sisipan

Pengurutan sisipan merupakan suatu metode pengurutan yang menjadikan sebuah data sebagai *data sisip*, yang kemudian membandingkan data sisip tersebut dengan data-data sebelumnya, lalu meletakkan data sisip tersebut pada *tempat yang sesuai*.

Pada perulangan pertama *data pada indeks kedua* dijadikan data sisip, kemudian dibandingkan dengan *data pertama*. Jika ternyata memenuhi kriteria maka data tersebut diletakkan pada tempat yang sesuai. Selanjutnya pada perulangan kedua *data pada indeks ketiga* dijadikan data sisip, kemudian dibandingkan dengan *data pertama* dan *data kedua*. Jika memenuhi kriteria maka data tersebut diletakkan pada tempat yang sesuai, demikian seterusnya.

Ket	Algoritma	Java
1		import javax.swing.JOptionPane;
2	Algorithm PengurutanSisipan	public class PengurutanSisipan { public static void main (String args []) {
3	bilangan : array [1...n] of integer	int n = Integer.parseInt (JOptionPane.showInputDialog (null,"Masukkan banyaknya bilangan"));
4	DEKLARASI : n : integer i, j : integer outputawal : string ← "" outputakhir : string ← ""	int bilangan [] = new int [n];
5	DESKRIPSI : write ("Masukkan banyaknya bilangan")	int i, j; String outputawal = "";

	read (n)	String outputakhir = "";

6	for i ← 1 to n do
	write ("Masukkan bilangan ke ", (i+1))
	bilangan[i] ← read (bilangan[i])
	
	outputawal+ ← bilangan[i], " "
	end for
		for (i=0;i<n;i++)
		{
		bilangan[i] = Integer.parseInt
		(JOptionPane.showInputDialog
		(null,"Masukkan bilangan ke " +(i+1)));
		
		outputawal += bilangan[i]+ " ";
		}

7	for i ← 2 to n do
	datasisip : integer
	datasisip ← bilangan[i]
	j ← i-1
	
	while (*kondisi kriteria** AND j>0)
	bilangan[j+1] ← bilangan [j]
	j = j-1
	end while
	
	bilangan[j+1] ← datasisip
	end for
		for (i=1;i<n;i++)
		{
		int datasisip = bilangan[i];
		j = i-1;
		
		while (*kondisi kriteria** && j>0)
		{
		bilangan[j+1] = bilangan[j];
		j = j-1;
		}
		
		bilangan[j+1] = datasisip;
		}

8	for i ← 1 to n do
	outputakhir+ ← bilangan[i], " "
	end for
		for (i=0;i<n;i++)
		{
		outputakhir += bilangan[i]+ " ";
		}

9	write ("Susunan awal: ", outputawal, "newlineSusunan akhir: ", outputakhir)	JOptionPane.showMessageDialog(null,"Susu nan awal: " +outputawal+ "\nSusunan akhir: " +outputakhir);
10		}
11	end Algorithm	}

ket:

(1) Java : menggunakan kotak dialog *JOptionPane*.

(2) Algoritma : nama algoritma,
awal algoritma.
Java : nama kelas (yang juga merupakan awal kelas),
awal program utama (main program).

(3) Algoritma : deklarasi array *bilangan* bertipe *integer* dengan jumlah indeks sebanyak *n*.
Java : meminta nilai *n* dan mendefenisikan *n* sebagai banyaknya bilangan.

(4) Algoritma : deklarasi variabel.
Java : deklarasi array *bilangan* bertipe *integer* dengan jumlah indeks sebanyak *n*.

(5) Algoritma : awal deskripsi,
meminta nilai *n* dan mendefenisikan *n* sebagai banyaknya bilangan.
Java : deklarasi variabel.

(6) Algoritma : perulangan untuk mengisi indeks array *bilangan*. Lalu merekam setiap

indeks array *bilangan* dan menyimpannya pada variabel *outputawal*.

Java : perulangan untuk mengisi indeks array *bilangan*. Lalu merekam setiap indeks array *bilangan* dan menyimpannya pada variabel *outputawal*.

(7) Algoritma : perulangan untuk mengurutkan data.

Java : perulangan untuk mengurutkan data.
ket : **kondisi kriteria*,
untuk *ascending* (datasisip<bilangan[j])
untuk *descending* (datasisip>bilangan[j])

(8) Algoritma : Merekam setiap indeks array *bilangan* yang telah diurutkan dan menyimpannya pada variabel *outputakhir*.

Java : Merekam setiap indeks array *bilangan* yang telah diurutkan dan menyimpannya pada variabel *outputakhir*.

(9) Algoritma : menampilkan hasil pengurutan.

Java : menampilkan hasil pengurutan.

(10) Java : akhir program utama (main program).

(11) Algoritma : akhir algoritma

Java : akhir kelas

7.2 Pengurutan Seleksi

Pengurutan seleksi merupakan suatu metode pengurutan yang membandingkan *beberapa data* dan memilih *satu data* yang memenuhi kriteria, kemudian data tersebut ditukartempatkan ke depan.

Pada perulangan pertama dilakukan perbandingan dari *data pertama* hingga *data terakhir*, lalu dipilih satu data yang memenuhi kriteria, kemudian *data* tersebut ditukartempatkan dengan *data pertama*. Selanjutnya pada perulangan kedua dilakukan perbandingan dari *data kedua* hingga *data terakhir*, lalu dipilih satu data yang memenuhi kriteria, kemudian *data* tersebut ditukartempatkan dengan *data kedua*, demikian seterusnya.

Ket	Algoritma	Java
1		import javax.swing.JOptionPane;
2	Algorithm PengurutanSeleksi	public class PengurutanSeleksi { public static void main (String args []) {
3	bilangan : array [1...n] of integer	int n = Integer.parseInt (JOptionPane.showInputDialog (null,"Masukkan banyaknya bilangan"));
4	DEKLARASI : n : integer i, j : integer outputawal : string ← "" outputakhir : string ← ""	int bilangan [] = new int [n];

5	DESKRIPSI : write ("Masukkan banyaknya bilangan") read (n)	int i, j; String outputawal = ""; String outputakhir = "";

6	for i ← 1 to n do
	  write ("Masukkan bilangan ke ", (i+1))
	  bilangan[i] ← read (bilangan[i])
	
	  outputawal+ ← bilangan[i], " "
	end for
		for (i=0;i<n;i++)
		{
		  bilangan[i] = Integer.parseInt
		  (JOptionPane.showInputDialog
		  (null,"Masukkan bilangan ke " +(i+1)));
		
		  outputawal += bilangan[i]+ " ";
		}

7	for i ← 1 to (n-1) do
	  for j ← (i+1) to n do
	  if (*kondisi tergantung kriteria**) then
	  temp : integer
	  temp ← bilangan[i]
	  bilangan[i] ← bilangan[j]
	  bilangan[j] ← temp
	  end if
	  end for
	end for
		for (i=0;i<n-1;i++)
		{
		  for (j=i+1;j<n;j++)
		  {
		  if (*kondisi tergantung kriteria**)
		  {
		  int temp = bilangan[i];
		  bilangan[i] = bilangan[j];
		  bilangan[j] = temp;
		  }
		  }
		}

8	for i ← 1 to n do
	  outputakhir+ ← bilangan[i], " "
	end for
		for (i=0;i<n;i++)
		{
		  outputakhir += bilangan[i]+ " ";
		}

9	write ("Susunan awal: ", outputawal, "newlineSusunan akhir: ", outputakhir)	JOptionPane.showMessageDialog(null,"Susu nan awal: " +outputawal+ "\nSusunan akhir: " +outputakhir);
10		}
11	end Algorithm	}

ket:

keterangan lainnya dapat dilihat pada (7.1 Pengurutan Sisipan)

(7) Algoritma : perulangan untuk mengurutkan data.

Java : perulangan untuk mengurutkan data.

ket : **kondisi tergantung kriteria*,

untuk *ascending* (bilangan[i]>bilangan[j])

untuk *descending* (bilangan[i]<bilangan[j])

7.3 Pengurutan Gelembung

Pengurutan gelembung merupakan suatu metode pengurutan yang membandingkan dua buah data secara terus-menerus. Proses membandingkan akan berhenti saat proses perulangan selesai.

Pada setiap perulangan hanya terjadi satu kali proses perbandingan. Pada perulangan pertama dilakukan perbandingan antara *data pertama* dan *data kedua*. Selanjutnya pada perulangan kedua dilakukan perbandingan antara *data kedua* dan *data ketiga*, demikian seterusnya.

Ket	Algoritma	Java
1		import javax.swing.JOptionPane;
2	Algorithm PengurutanGelembung	public class PengurutanGelembung { public static void main (String args []) {
3	bilangan : array [1...n] of integer	int n = Integer.parseInt (JOptionPane.showInputDialog (null,"Masukkan banyaknya bilangan"));
4	DEKLARASI : n : integer i, j : integer outputawal : string ← "" outputakhir : string ← ""	int bilangan [] = new int [n];
5	DESKRIPSI : write ("Masukkan banyaknya bilangan") read (n)	int i, j; String outputawal = ""; String outputakhir = "";

6	for i ← 1 to n do
	write ("Masukkan bilangan ke ", (i+1))
	bilangan[i] ← read (bilangan[i])
	
	outputawal+ ← bilangan[i], " "
	end for
		for (i=0;i<n;i++)
		{
		bilangan[i] = Integer.parseInt
		(JOptionPane.showInputDialog
		(null,"Masukkan bilangan ke " +(i+1)));
		
		outputawal += bilangan[i]+ " ";
		}

7	for i ← 1 to (n-1) do
	for j ← 1 to (n-1) do
	if (*kondisi tergantung kriteria**) then
	temp : integer
	temp ← bilangan[j]
	bilangan[j] ← bilangan[j+1]
	bilangan[j+1] ← temp
	end if
	end for
		for (i=0;i<n-1;i++)
		{
		for (j=0;j<n-1;j++)
		{
		if (*kondisi tergantung kriteria**)
		{
		int temp = bilangan[j];
		bilangan[j] = bilangan[j+1];
		bilangan[j+1] = temp;
		}
		}

	end for	}

8	for i ← 1 to n do
	outputakhir+ ← bilangan[i], " "
	end for
		for (i=0;i<n;i++)
		{
		outputakhir += bilangan[i]+ " ";
		}

9	write ("Susunan awal: ", outputawal, "newlineSusunan akhir: ", outputakhir)	JOptionPane.showMessageDialog(null,"Susunan awal: " +outputawal+ "\nSusunan akhir: " +outputakhir);
10		}
11	end Algorithm	}

ket:

keterangan lainnya dapat dilihat pada (7.1 Pengurutan Sisipan)

(7) Algoritma : perulangan untuk mengurutkan data.

Java : perulangan untuk mengurutkan data.

ket : **kondisi tergantung kriteria*,

untuk *ascending* (bilangan[j]>bilangan[j+1])

untuk *descending* (bilangan[j]<bilangan[j+1])

8. PENCARIAN

Pencarian atau *searching* adalah sekumpulan instruksi yang bertugas menemukan nilai suatu data. Dalam kehidupan nyata pencarian merupakan proses yang sangat penting. Berdasarkan cara kerjanya, pencarian dibagi dua yaitu *Pencarian Beruntun* (*Sequential Search*) dan *Pencarian Bagi Dua* (*Binary Search*).

8.1 Pencarian Beruntun

Pencarian beruntun dapat diimplementasikan pada data yang telah terurut maupun belum. Pencarian beruntun dilakukan dengan menelusuri data satu persatu, kemudian dicocokkan dengan data yang dicari. Jika data yang dicari sama dengan data yang dicocokkan, maka penelusuran dihentikan, sebaliknya jika data yang dicari belum sama dengan data yang dicocokkan maka penelusuran dilanjutkan, demikian seterusnya.

Ket	Algoritma	Java
1		import javax.swing.JOptionPane;
2	Algorithm PencarianBeruntun	public class PencarianBeruntun { public static void main (String args []) {
3	bilangan : array [1...n] of integer	int n = Integer.parseInt (JOptionPane.showInputDialog (null,"Masukkan banyaknya bilangan"));
4	DEKLARASI : n : integer i : integer bilygdicari : integer indeks : integer ditemukan : boolean	int bilangan [] = new int [n];
5	DESKRIPSI : write ("Masukkan banyaknya bilangan") read (n)	int i; boolean ditemukan;

6	for i ← 1 to n do
	write ("Masukkan bilangan ke ", i)
	bilangan[i] ← read (bilangan[i])
	end for
		for (i=0;i<n;i++)
		{
		bilangan[i] = Integer.parseInt
		(JOptionPane.showInputDialog
		(null,"Masukkan bilangan ke " +(i+1)));
		}

7	write ("Masukkan bilangan yang dicari :") read (bilygdicari)	int bilygdicari = Integer.parseInt (JOptionPane.showInputDialog (null,"Masukkan bilangan yang dicari :"));

8	ditemukan ← false	ditemukan = false;
9	i ← 0	i = 0;
10	indeks ← 0	int indeks = 0;

11	repeat
	  if (bilangan[i]=bilygdicari) then
	  ditemukan ← true
	  end if
	  i ← i+1
	until (ditemukan=true OR i=(n-1))
		do
		{
		  if (bilangan[i]==bilygdicari)
		  {
		  ditemukan = true;
		  }
		  i = i+1;
		}
		while (ditemukan==false || i!=(n-1))

12	indeks ← i+1	indeks = i+1;

13	if (ditemukan=true) then
	  write ("Ditemukan pada indeks ke “, indeks)
	else
	  write ("Tidak ditemukan")
	end if
		if (ditemukan==true)
		{
		  System.out.println("Ditemukan pada indeks ke " +indeks);
		}
		else
		{
		  System.out.println("Tidak ditemukan");
		}

14		}
15	end Algorithm	}

ket:

(1) Java : menggunakan kotak dialog *JOptionPane*.

(2) Algoritma : nama algoritma,
awal algoritma.
Java : nama kelas (yang juga merupakan awal kelas),
awal program utama (main program).

(3) Algoritma : deklarasi array *bilangan* bertipe *integer* dengan jumlah indeks sebanyak *n*.
Java : meminta nilai *n* dan mendefenisikan *n* sebagai banyaknya bilangan.

(4) Algoritma : deklarasi variabel.
Java : deklarasi array *bilangan* bertipe *integer* dengan jumlah indeks sebanyak *n*.

(5) Algoritma : awal deskripsi,
meminta nilai *n* dan mendefenisikan *n* sebagai banyaknya bilangan.
Java : deklarasi variabel.

(6) Algoritma : perulangan untuk mengisi indeks array *bilangan*.
Java : perulangan untuk mengisi indeks array *bilangan*.

(7) Algoritma : meminta nilai *bilygdicari* dan mendefenisikan *bilygdicari* sebagai bilangan yang ingin dicari.
Java : meminta nilai *bilygdicari* dan mendefenisikan *bilygdicari* sebagai bilangan yang ingin dicari.

(8) Algoritma : memberi nilai *ditemukan* dengan false.
Java : memberi nilai *ditemukan* dengan false.

(9) Algoritma : memberi nilai *i* dengan 0.
Java : memberi nilai *i* dengan 0.
(10) Algoritma : memberi nilai *indeks* dengan 0.
Java : deklarasi *indeks* sebagai integer, dan memberi nilai *indeks* dengan 0.
(11) Algoritma : perulangan untuk mencocokkan *bilygdicari* dengan setiap indeks array, perulangan akan berhenti jika *bilygdicari* sama dengan indeks tertentu.
Java : perulangan untuk mencocokkan *bilygdicari* dengan setiap indeks array, perulangan akan berhenti jika *bilygdicari* sama dengan indeks tertentu.
(12) Algoritma : memberi nilai *indeks* dengan i+1.
Java : memberi nilai *indeks* dengan i+1.
(13) Algoritma : menampilkan pesan ditemukan atau tidak.
Java : menampilkan pesan ditemukan atau tidak.
(14) Java : akhir program utama (main program).
(15) Algoritma : akhir algoritma
Java : akhir kelas

8.2 Pencarian Bagi Dua

Pencarian bagi dua hanya dapat diimplementasikan pada data yang telah terurut. Pencarian bagi dua dilakukan dengan membagi data menjadi dua bagian, kemudian dicocokkan dengan data yang dicari. Jika data yang dicari sama dengan data yang dicocokkan, maka pembagian dihentikan, sebaliknya jika data yang dicari belum sama dengan data yang dicocokkan maka pembagian dilanjutkan, demikian seterusnya.

Ket	Algoritma	Java
1		import javax.swing.JOptionPane;
2	Algorithm PencarianBagiDua	public class PencarianBagiDua { public static void main (String args []) {
3	bilangan : array [1...n] of integer	int n = Integer.parseInt (JOptionPane.showInputDialog (null,"Masukkan banyaknya bilangan"));
4	DEKLARASI : n : integer i : integer awal : integer akhir : integer mid : integer bilygdicari : integer ditemukan : boolean	int bilangan [] = new int [n];
5	DESKRIPSI : write ("Masukkan banyaknya bilangan") read (n)	int i; int awal; int akhir;

		int mid; boolean ditemukan;

6	for i ← 1 to n do
	  write ("Masukkan bilangan ke ", (i+1))
	  bilangan[i] ← read (bilangan[i])
	end for
		for (i=0;i<n;i++)
		{
		  bilangan[i] = Integer.parseInt
		  (JOptionPane.showInputDialog
		  (null,"Masukkan bilangan ke " +(i+1)));
		}

7	write ("Masukkan bilangan yang dicari :") read (bilygdicari)	int bilygdicari = Integer.parseInt (JOptionPane.showInputDialog (null,"Masukkan bilangan yang dicari :"));
8	ditemukan ← false	ditemukan = false;
9	awal ← 1	awal = 1;
10	akhir ← n	akhir = n;

11	repeat
	  mid ← (awal+akhir)div2
	  if (bilangan[mid]=bilygdicari) then
	  ditemukan ← true
	  else
	  if (bilangan[mid]>bilygdicari) then
	  akhir ← mid-1
	  else
	  awal ← mid+1
	  end if
	  end if
	until (ditemukan=true OR awal=akhir)
		do
		{
		  mid = (awal+akhir)/2
		  if (bilangan[mid]==bilygdicari)
		  {
		  ditemukan = true;
		  }
		  else if (bilangan[mid]>bilygdicari)
		  {
		  akhir = mid-1;
		  }
		  else
		  {
		  awal = mid+1;
		  }
		}
		while (ditemukan==false || awal!=akhir)

12	if (ditemukan=true) then
	  write ("Ditemukan pada indeks ke “,
	  mid)
	else
	  write ("Tidak ditemukan")
	end if
		if (ditemukan==true)
		{
		  System.out.println("Ditemukan pada
		  indeks ke " +mid);
		}
		else
		{
		  System.out.println("Tidak ditemukan");
		}

13		}
14	end Algorithm	}

ket:

(1) Java : menggunakan kotak dialog *JOptionPane*.

(2) Algoritma : nama algoritma,
awal algoritma.
Java : nama kelas (yang juga merupakan awal kelas),
awal program utama (main program).

(3) Algoritma : deklarasi array *bilangan* bertipe *integer* dengan jumlah indeks sebanyak *n*.
Java : meminta nilai *n* dan mendefenisikan *n* sebagai banyaknya bilangan.

(4) Algoritma : deklarasi variabel.
Java : deklarasi array *bilangan* bertipe *integer* dengan jumlah indeks sebanyak *n*.

(5) Algoritma : awal deskripsi,
meminta nilai *n* dan mendefenisikan *n* sebagai banyaknya bilangan.
Java : deklarasi variabel.

(6) Algoritma : perulangan untuk mengisi indeks array *bilangan*.
Java : perulangan untuk mengisi indeks array *bilangan*.

(7) Algoritma : meminta nilai *bilygdicari* dan mendefenisikan *bilygdicari* sebagai bilangan yang ingin dicari.
Java : meminta nilai *bilygdicari* dan mendefenisikan *bilygdicari* sebagai bilangan yang ingin dicari.

(8) Algoritma : memberi nilai *ditemukan* dengan false.
Java : memberi nilai *ditemukan* dengan false.

(9) Algoritma : memberi niali *awal* dengan 1.
Java : memberi niali *awal* dengan 1.

(10) Algoritma : memberi niali *akhir* dengan n.
Java : memberi niali *akhir* dengan n.

(11) Algoritma : perulangan untuk mencocokkan *bilygdicari* dengan setiap indeks bagian tengah array, perulangan akan berhenti jika *bilygdicari* sama dengan indeks bagian tengah tertentu.
Java : perulangan untuk mencocokkan *bilygdicari* dengan setiap indeks bagian tengah array, perulangan akan berhenti jika *bilygdicari* sama dengan indeks bagian tengah tertentu.

(12) Algoritma : menampilkan pesan ditemukan atau tidak.
Java : menampilkan pesan ditemukan atau tidak.

(13) Java : akhir program utama (main program).

(14) Algoritma : akhir algoritma
Java : akhir kelas

REFERENSI

Horstmann, Cay dan Cornell, Gary. 2000. Core Java Volume 1, Fundamentals. Sun Microsystems Inc. California.

Kadir, Abdul. 2006. Dasar Pemrograman Java 2. Penerbit ANDI. Yogyakarta.

Naughton, Patrick. 1996. The Java Handbook. McGraw-Hill Book Company. New York.

Shalahuddin, M dan Rosa, A. S. 2007. Belajar Pemrograman dengan Bahasa C++ dan Java. Penerbit Informatika. Bandung.

TENTANG

Rama, lahir di Manado tahun 1991 dengan nama lengkap I Dewa Gede Rama. Anak pertama dari tiga bersaudara ini merupakan putra dari ibu yang bernama Gusti Ayu Puspita dan ayah yang bernama Dewa Ketut Anom.

Ia telah menyelesaikan studi di SD Negeri 77 Manado, SMP Negeri 1 Manado, dan SMA Negeri 1 Manado. Seusai lulus SMA, Ia langsung mengambil studi Teknik Informatika di Institut Teknologi Harapan Bangsa. Cita-citanya menjadi seorang wirausahawan sukses yang berguna bagi bangsa dan negerinya, Indonesia. Ditengah-tengah kesibukan kuliahnya, Ia masih menyempatkan diri mengikuti kegiatan akademis lainnya.

idegeram@gmail.com