**DAFTAR ISI**

**BAB I : PENGANTAR**

**A. Pengertian Algoritma & Pemrograman**

0. Sejarah Algoritma

1. Algoritma

2. Pemrograman

3. Penyajian Algoritma

4. Pemrograman Prosedural

**B. Tipe-tipe Algoritma Berdasarkan Format Penulisan**

1. Deskriptif

2. Flow Chart (Diagram Alir)

3. Pseudocode

**C. Flow Chart (Diagram Alir)**

1. Pengertian

2. Simbol-simbol Flow Chart

3. Bentuk-bentuk Dasar Struktur Logika Flow Chart

**D. Tipe Data**

1. Tipe Data Sederhana

2. Tipe Data terstruktur

**BAB I**

**PENGANTAR**

1. **Pengertian Algoritma & Pemrograman**

**0 Sejarah Singkat Algoritma**

Ditinjau dari asal-usul katanya, kata Algoritma sendiri mempunyai sejarah yang aneh. Orang hanya menemukan kata *algorism* yang berarti proses menghitung dengan angka arab. Anda dikatakan *algorist* jika Anda menghitung menggunakan angka arab. Para ahli bahasa berusaha

menemukan asal kata ini namun hasilnya kurang memuaskan. Akhirnya para ahli sejarah matematika menemukan asal kata tersebut yang berasal dari nama penulis buku arab yang terkenal yaitu Abu Ja’far Muhammad Ibnu Musa Al-Khuwarizmi. Al-Khuwarizmi dibaca orang barat menjadi *Algorism.* Al-Khuwarizmi menulis buku yang berjudul *Kitab Al Jabar Wal-Muqabala* yang artinya “Buku pemugaran dan pengurangan” (*The book ofrestoration and reduction*). Dari judul buku itu kita juga memperoleh akar kata “Aljabar” (*Algebra*). Perubahan kata dari *algorism* menjadi *algorithm* muncul karena kata *algorism* sering dikelirukan dengan *arithmetic*, sehingga akhiran *–sm* berubah menjadi *–thm.* Karena perhitungan dengan angka Arab sudah menjadi hal yang biasa, maka lambat laun kata *algorithm*berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna kata aslinya. Dalam bahasa Indonesia,kata *algorithm* diserap menjadi *algoritma*.

**1. Algoritma**

Asal usul kata algoritma dapat Anda baca dalam buku “The Art of Computer Programming Second Edition Volume I” yang ditulis oleh Donald E. Knuth (1973, p1-2). Menurut Donald E. Knuth (1973,p4), algoritma dalam pengertian modern mempunyai kemiripan dengan istilah **resep**, **proses, metode, teknik, prosedur, rutin**. Algoritma adalah sekumpulan aturan-aturan berhingga yang memberikan sederetan operasi-operasi untuk menyelesaikan suatu jenis masalah yang khusus. Menurut Rinaldi Munir, algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis. Berdasarkan dua pengertian algoritma di atas, dapat disimpulkan bahwa algoritma merupakan suatu istilah yang luas, yang tidak hanya berkaitan dengan dunia komputer.

***“Algoritma*** *adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis dan logis”.* Kata *logis* merupakan kata kunci dalam algoritma. Langkah-langkah dalam algoritma harus logis dan harus dapat ditentukan bernilai salah atau benar.

**Penyajian Algoritma**

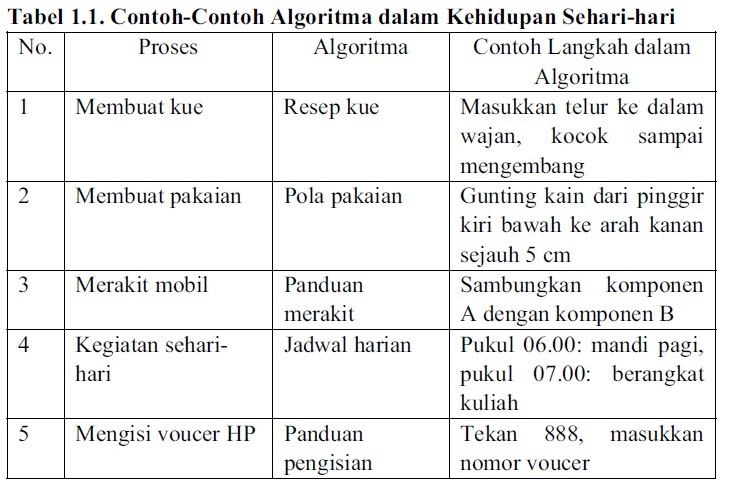
1. Menggunakan Flow Chart (Diagram Alir)
2. Menggunakan Pseudo-Code

**Pembuatan algoritma mempunyai banyak keuntungan di antaranya:**

1. Pembuatan atau penulisan algoritma tidak tergantung pada bahasa pemrograman manapun, artinya penulisan algoritma independen dari bahasa pemrograman dan komputer yang melaksanakannya.

2. Notasi algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman.

3. Apapun bahasa pemrogramannya, *output* yang akan dikeluarkan sama karena algoritmanya sama.



**2. Pemrograman**

Pemrograman berasal dari kata **program** yang diberi awalan pe– dan akhiran –an. Dalam buku ini, program berarti program komputer. Pengertian **program komputer** menurut John M. Zelle, Ph.D. dalam bukunya yang berjudul “Python Programming: An Introduction to Computer Science” (2002, p1) adalah sekumpulan instruksi langkah per langkah yang memberitahukan mengenai yang harus dilakukan komputer secara tepat. **Pemrograman** adalah segala kegiatan pembuatan program komputer. Kemudian terdapat pula istilah **bahasa pemrograman** yang berarti bahasa yang digunakan dalam pembuatan program komputer. Berdasarkan pengertian algoritma dan pemrograman, maka dapat dikatakan bahwa program merupakan hasil penerapan dari algoritma-algoritma. Akan tetapi, dalam buku ini tidak dibahas materi mengenai pembuatan program komputer. Buku ini memfokuskan teknik-teknik pembuatan algoritma itu sendiri. Nama mata kuliah Algoritma dan Pemrograman dalam hal ini berarti mempelajari pembuatan algoritma-algoritma yang dapat diterapkan dalam pemrograman.

**Untuk Menyelesaikan persoalan Programmer haruslah**

1. Dapat mendesain algoritma yang menjelaskan bagaimana persoalan tersebut diselesaikan
2. Menulis/merubah algoritma menjadi suatu program dengan menggunakan suatu Bahasa pemrograman yang sesuai.
3. Menggunakan komputer untuk menjalankan program.

**Beda Algoritma dan Program**

Program adalah kumpulan pernyataan komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma. Program ditulis dengan menggunakan bahasa pemrograman. Jadi bisa disebut bahwa program adalah suatu implementasi dari bahasa pemrograman. Beberapa pakar memberi formula bahwa:

**Program = Algoritma + Bahasa (Struktur Data)**

Bagaimanapun juga struktur data dan algoritma berhubungan sangat erat pada sebuah program. Algoritma yang baik tanpa pemilihan struktur data yang tepat akan membuat program menjadi kurang baik, demikian juga sebaliknya.

**Ada beberapa hal yang harus diperhatikanpada translasi tersebut, yaitu:**

a. Pendeklarasian variabel

Untuk mengetahui dibutuhkannya pendeklarasian variabel dalam penggunaan bahasa pemrograman apabila tidak semua bahasa pemrograman membutuhkannya.

b. Pemilihan tipe data

Apabila bahasa pemrograman yang akan digunakan membutuhkan pendeklarasian variabel maka perlu hal ini dipertimbangkan pada saat pemilihan tipe data.

c. Pemakaian instruksi-instruksi

Beberapa instruksi mempunyai kegunaan yang sama tetapi masing masing memiliki kelebihan dan kekurangan yang berbeda.

d. Aturan sintaksis

Pada saat menuliskan program kita terikat dengan aturan sintaksis dalam bahasa pemrograman yang akan digunakan.

e. Tampilan hasil

Pada saat membuat algoritma kita tidak memikirkan tampilan hasil yang akan disajikan. Hal-hal teknis ini diperhatikan ketika mengkonversikannya menjadi program.

f. Cara pengoperasian *compiler* atau *interpreter*.

Bahasa pemrograman yang digunakan termasuk dalam kelompok *compiler* atau *interpreter*.

Contoh :

1. Judul program

2. Blok program

a. Bagian deklarasi

- deklarasi label

- deklarasi konstanta

- deklarasi tipe

- deklarasi variable

- deklarasi prosedur

- deklarasi fungsi

b. Bagian pernyataan

1. **Penyajian Algoritma**

Beberapa persyaratan untuk menjadi algoritma yang baik adalah:

• **Tingkat kepercayaannya tinggi (*realibility*).** Hasil yang diperoleh dari proses harus berakurasi tinggi dan benar.

• **Pemrosesan yang efisien (*cost* rendah).** Proses harus diselesaikan secepat mungkin dan frekuensi kalkulasi yang sependek mungkin.

• **Sifatnya general.** Bukan sesuatu yang hanya untuk menyelesaikan satu kasus saja, tapi juga untuk kasus lain yang lebih general.

• **Bisa dikembangkan (*expandable*).** Haruslah sesuatu yang dapat kita kembangkan lebih jauh berdasarkan perubahan *requirement* yang ada.

• **Mudah dimengerti.** Siapapun yang melihat, dia akan bisa memahami algoritma Anda. Susah dimengertinya suatu program akan membuat susah di-*maintenance* (kelola).

• **Portabilitas yang tinggi (*portability*).** Bisa dengan mudah diimplementasikan di berbagai *platform* komputer.

• ***Precise* (tepat, betul, teliti).** Setiap instruksi harus ditulis dengan seksama dan tidak ada keragu-raguan, dengan demikian setiap instruksi harus dinyatakan secara eksplisit dan tidak ada bagian yang dihilangkan karena pemroses dianggap sudah mengerti. Setiap langkah harus jelas dan pasti.

**Contoh: Tambahkan 1 atau 2 pada x.**

Instruksi di atas terdapat keraguan.

• Jumlah langkah atau instruksi berhingga dan tertentu. Artinya, untuk kasus yang sama banyaknya, langkah harus tetap dan tertentu meskipun datanya berbeda.

• Efektif. Tidak boleh ada instruksi yang tidak mungkin dikerjakan oleh pemroses yang akan menjalankannya.

Contoh: Hitung akar 2 dengan presisi sempurna.

Instruksi di atas tidak efektif, agar efektif instruksi tersebut diubah.

Misal: Hitung akar 2 sampai lima digit di belakang koma.

• Harus *terminate*. Jalannya algoritma harus ada kriteria berhenti. Pertanyaannya adalah apakah bila jumlah instruksinya berhingga maka pasti *terminate*?

• *Output* yang dihasilkan tepat. Jika langkah-langkah algoritmanya logis dan diikuti dengan seksama maka dihasilkan *output* yang diinginkan.

**Mencari Kesalahan Algoritma atau Pemrograman**

a. Kesalahan sintaks (penulisan program).

b. Kesalahan pelaksanaan: semantik, logika, dan ketelitian.

**Standar penyusunan program**

- Kebenaran logika dan penulisan.

- Waktu minimum untuk penulisan program.

- Kecepatan maksimum eksekusi program.

- Ekspresi penggunaan memori.

- Kemudahan merawat dan mengembangkan program.

- *User Friendly*.

- *Portability.*

- Pemrograman modular.

**Belajar Memprogram secara sistematik**

**a. Belajar Memprogram**

• Belajar memprogram: belajar bahasa pemrograman.

• Belajar memprogram: belajar tentang strategi pemecahan masalah, metodologi dan sistematika pemecahan masalah kemudian menuliskannya dalam notasi yang disepakati bersama.

• Belajar memprogram: bersifat pemahaman persoalan, analisis dan sintesis.

• Belajar memprogram, titik berat: *designer* program.

**b. Belajar Bahasa Pemrograman**

• Belajar bahasa pemrograman: belajar memakai suatu bahasa pemrograman, aturan sintaks, tatacara untuk memanfaatkan pernyataan yang spesifik untuk setiap bahasa.

• Belajar bahasa pemrograman, titik berat: *coder.*

**c. Produk yang Dihasilkan Pemrogram**

• Program dengan rancangan yang baik (metodologis, sistematis).

• Dapat dieksekusi oleh mesin.

• Berfungsi dengan benar.

• Sanggup melayani segala kemungkinan masukan.

• Disertai dokumentasi.

• Belajar memprogram, titik berat: *designer* program.

**Urutan dasar pemecahan suatu masalah:**

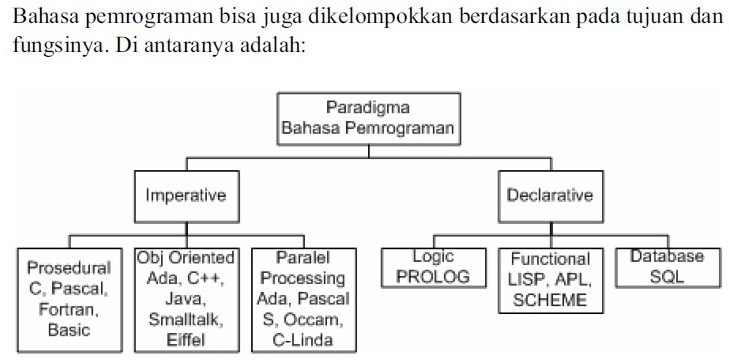
􀂾 START, berisi pernyataan untuk persiapan peralatan yang diperlukan sebelum menangani pemecahan persoalan.

􀂾 READ, berisi pernyataan kegiatan untuk membaca data dari suatu peralatan *input.*

􀂾 PROSES, berisi kegiatan yang berkaitan dengan pemecahan persoalan sesuai dengan data yang dibaca.

􀂾 WRITE, berisi pernyataan untuk merekam hasil kegiatan ke peralatan *output.*

􀂾 END, mengakhiri kegiatan pengolahan.



**Beda Algoritma dan Program**

Program adalah kumpulan pernyataan komputer, sedangkan metode dan tahapan sistematis dalam program adalah algoritma. Program ditulis dengan menggunakan bahasa pemrograman. Jadi bisa disebut bahwa program adalah suatu implementasi dari bahasa pemrograman.

Beberapa pakar memberi formula bahwa:

**Program = Algoritma + Bahasa (Struktur Data)**

Bagaimanapun juga struktur data dan algoritma berhubungan sangat erat pada sebuah program. Algoritma yang baik tanpa pemilihan struktur data yang tepat akan membuat program menjadi kurang baik, demikian juga sebaliknya.

**Produk yang Dihasilkan Pemrogram**

• Program dengan rancangan yang baik (metodologis, sistematis).

• Dapat dieksekusi oleh mesin.

• Berfungsi dengan benar.

• Sanggup melayani segala kemungkinan masukan.

• Disertai dokumentasi.

• Belajar memprogram, titik berat: *designer* program.

1. **Pemrograman Prosedural**

Algoritma berisi urutan langkah-langkah penyelesaian masalah. Ini berarti algoritma adalah proses yang prosedural. Pada program prosedural, program dibedakan antara bagian data dengan bagian instruksi. Bagian instruksi terdiri dari atas runtunan (*sequence)* instruksi yang dilaksanakan satu per satu secara berurutan oleh sebuah pemroses. Alur pelaksanaan instruksi

dapat berubah karena adanya pencabangan kondisional. Data yang disimpan di dalam memori dimanipulasi oleh instruksi secara beruntun. Kita katakan bahwa tahapan pelaksanaan program mengikuti pola beruntun atau prosedural.

Paradigma pemrograman seperti ini dinamakan pemrograman prosedural. Bahasa-bahasa tingkat tinggi seperti *Cobol, Basic, Pascal, Fortran,* dan C/C++ mendukung kegiatan pemrograman prosedural, karena itu mereka dinamakan juga bahasa prosedural. Selain paradigma pemrograman prosedural, ada lagi paradigma yang lain yaitu pemrograman berorientasi objek (*Object Oriented Programming* atau OOP). Paradigma pemrograman ini merupakan *trend* baru dan sangat popular akhir-akhir ini. Pada paradigma OOP, data dan instruksi dibungkus (*encapsulation)* menjadi satu. Kesatuan ini disebut kelas (*class*) dan instansiasi kelas pada saat *run-time* disebut objek (*object).* Data di dalam objek hanya dapat diakses oleh instruksi yang ada di dalam objek itu saja.

**B. Tipe-tipe Algoritma Berdasarkan Format Penulisan**

Algoritma adalah independen terhadap bahasa pemrograman tertentu, artinya algoritma yang telah dibuat tidak boleh hanya dapat diterapkan pada bahasa pemrograman tertentu. Penulisan algoritma tidak terikat pada suatu aturan tertentu, tetapi harus jelas maksudnya untuk tiap langkah algoritmanya. Namun pada dasarnya algoritma dibagi menjadi beberapa macam berdasarkan format penulisannya, yaitu:

**1. Deskriptif**

Algoritma bertipe deskriptif maksudnya adalah algoritma yang ditulis dalam bahasa manusia sehari-hari (misalnya bahasa Indonesia atau bahasa Inggris) dan dalam bentuk kalimat. Setiap langkah algoritmanya diterangkan dalam satu atau beberapa kalimat.

Sebagai contoh misalnya algoritma menentukan bilangan terbesar dari 3 bilangan berikut ini:

**Algoritma Menentukan\_bilangan\_terbesar\_dari\_3\_bilangan**

1. Meminta input 3 bilangan dari user, misalkan bilangan a, b, dan c.

2. Apabila bilangan a lebih besar dari b maupun c, maka bilangan a merupakan bilangan terbesar

3. Jika tidak (bilangan a tidak lebih besar dari b atau c) berarti bilangan a sudah pasti bukan bilangan terbesar. Kemungkinannya tinggal bilangan b atau c. Apabila bilangan b lebih besar dari c, maka b merupakan bilangan terbesar. Sebaliknya apabila bilangan b tidak lebih besar dari c, maka bilangan c merupakan yang terbesar.

4. Selesai.

**Contoh:**

1). Kepala algoritma:

Algoritma Luas\_Lingkaran { Menghitung luas lingkaran dengan ukuran jejari tertentu. Algoritma menerima masukan jejari lingkaran, menghitung luasnya, dan menyajikan hasilnya ke piranti keluaran } Perhatian, dalam menulis nama-nama dalam algoritma harus mempunyai makna yang mencerminkan proses, sifat atau identitas lainnya yang melekat dengan suatu proses, tipe,

konstanta, variabel, sub-program dan lain-lainnya. Nama-nama yang bermakna disebut mnemonic.

2) Deklarasi algoritma:

DEKLARASI { nama konstanta }

const PHI = 3.14; { Nilai phi = 22/7 }

{ nama peubah } var R : real; { input jejari lingkaran bilangan riil }

l\_Lingkaran : real; { luas lingkaran bilangan riil }

{ nama sub program }

procedure TUKAR(input/output A:integer, input/output B:integer)

{ Mempertukarkan nilai A dan B.Parameter A dan B sudah terdefinisi nilainya.Setelah pertukaran, A berisi nilai B

dan B berisi nilai A }

3) Deskripsi algoritma:

Bagian ini merupakan bagian inti algoritma yang berisikan uraian

langkah-langkah penyelesaian suatu masalah. Setiap langkah algoritma dibaca dari atas ke

bawah.

Urutan penulisan menentukan urutan pelaksanaan perintah.

{ Baca data jejari lingkaran R.Jika R <= 0 tulis pesan data salah, selain itu hitung luas lingkaran. Tampilkan luas

lingkaran. }

baca(R);

jika R <= 0 then tulis("Data salah !")

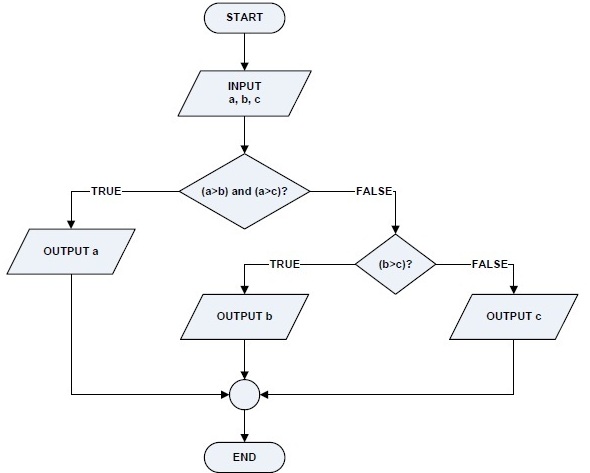
selain itu l\_Lingkaran = PHI x R x R;

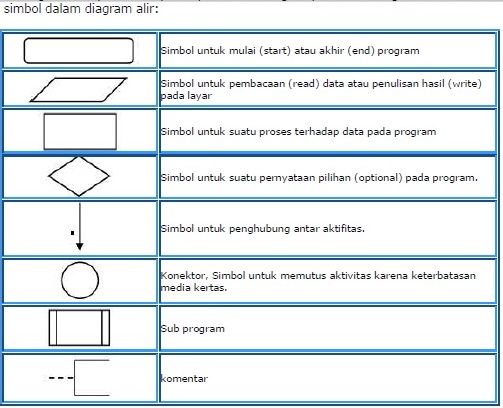
tulis(l\_Lingkaran);

**2. Flow Chart (Diagram Alir)**

Selain dalam bentuk tulisan, algoritma juga dapat ditulis dalam bentuk diagram diagram dengan anak panah sebagai penunjuk urutan langkah algoritmanya. Algoritma yang ditulis dengan simbol-simbol demikian yang dinamakan **flow chart**.

Mengenai lambang-lambang yang digunakan akan dibahas pada bagian selanjutnya. Sekarang diberikan suatu contoh algoritma menentukan bilangan terbesardari 3 bilangan seperti yang dicontohkan sebelumnya, tetapi ditulis dalam bentuk flow

****



Walaupun tidak ada kaidah-kaidah yang baku dalam penyusunan *flowchart*, namun ada beberapa anjuran:

􀂾 Hindari pengulangan proses yang tidak perlu dan logika yang berbelit sehingga jalannya proses menjadi singkat.

􀂾 Jalannya proses digambarkan dari atas ke bawah dan diberikan tanda panah untuk memperjelas.

􀂾 Sebuah *flowchart* diawali dari satu titik START dan diakhiri dengan END.

**3. Pseudocode**

**Pseudo** berarti imitasi dan **code** berarti kode yang dihubungkan dengan instruksi yang ditulis dalam bahasa komputer (kode bahasa pemrograman). Apabila diterjemahkan secara bebas, maka pseudocode berarti tiruan atau imitasi dari kode bahasa pemrograman. Pada dasarnya, pseudocode merupakan suatu bahasa yang memungkinkan programmer untuk berpikir terhadap permasalahan yang harus dipecahkan tanpa harus memikirkan *syntax* dari bahasa pemrograman yang tertentu. Tidak ada aturan penulisan *syntax* di dalam pseudocode. Jadi pseudocode digunakan untuk menggambarkan logika urut-urutan dari program tanpa memandang bagaimana bahasa pemrogramannya. Walaupun pseudocode tidak ada aturan penulisan syntax, di dalam buku ini akan diberikan suatu aturan-aturan penulisan syntax yang cukup sederhana agar pembaca dapat lebih mudah dalam mempelajari algoritma-algoritma yang ada di dalam buku ini. Pseudocode yang ditulis di dalam buku ini akan menyerupai (meniru) syntax-syntax dalam bahasa Pascal. Namun dibuat sesederhana mungkin sehingga tidak akan ada kesulitan bagi pembaca untuk memahami algoritma-algoritma dalam buku ini walaupun pembaca belum pernah mempelajari bahasa Pascal. Contoh algoritma menentukan bilangan terbesar dari tiga bilangan yang ditulis dalam bentuk pseudocode bergaya buku ini.

**01 ALGORITMA Menentukan\_terbesar\_dari\_3\_bilangan**

**02 Deklarasi:**

**03 a,b,c, terbesar : intege**

**05 Deskripsi:**

**06 Read(a,b,c)**

**07 If (a>b) and (a>c) then**

**08 Terbesar a**

**09 Else**

**10 If b>c then**

**11 Terbesar b**

**12 Else**

**13 Terbesar c**

**14 Endif**

**15 Endif**

**16 Write(terbesar)**

**C. Flow Char t (Diagram Al ir )**

**1. Pengertian**

Diagram alir atau flow chart adalah suatu bagan yang menggambarkan arus logika dari data yang akan diproses dalam suatu program dari awal sampai akhir. Diagram alir terdiri dari simbol-simbol yang mewakili fungsi-fungsi langkah program dan garis alir (flow lines) menunjukkan urutan dari simbol-simbol yang akan dikerjakan.

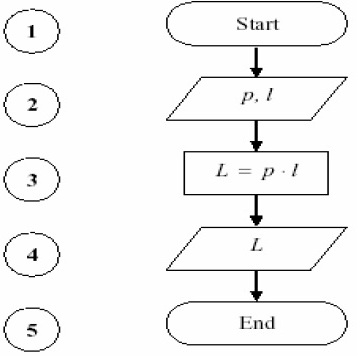
**2. Simbol-simbol Flow Chart**

Simbol terminal (terminator) digunakan untuk menunjukkan awal dan akhir algoritma Simbol persiapan (preparation)digunakan untuk memberikan nilai awal suatu variabel atau suatu counter Simbol proses (process)digunakan untuk proses perhitungan aritmatika atau proses

pemindahan data Simbol Data (data) digunakan untuk menunjukkan proses input maupun output data.

Simbol Keputusan (decision) digunakan untuk pengambilan keputusan dua jalur atau lebih dalam flow chart. Simbol Penghubung (on-page reference) digunakan untuk menunjukkan hubungan arus flow chart yang terputus, tetapi masih dalam halaman yang sama.

Simbol Penghubung Halaman Lain (off-page reference)digunakan untuk menunjukkan hubungan arus flow chart yang terputusyang berada di halaman lain.



**3. Bentuk-bentuk Dasar Struktur Logika Flow Chart**

Runtunan (Sequence Structure)

Pemilihan/Percabangan IF (Selection Structure)

Pengulangan FOR (FOR Loop Structure)

Pengulangan WHILE-DO

(WHILE-DO Structure)

**TRUE**

**FALSE**

Pengulangan REPEAT-UNTIL

(REPEAT-UNTIL Structure)

**TRUE**

**FALSE**

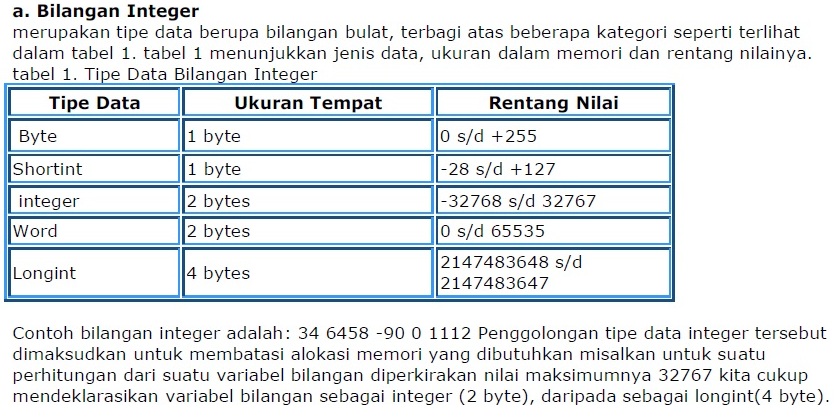
**D. Tipe Data**

Secara umum, tipe data dalam pemrograman Pascal adalah:

**– Tipe data sederhana**

– Tipe data standard (standard data type)

--> Integer, Real, Boolean, Char, dan String



**Contoh deklarasi:**

Var A, B, C : Integer;

D, E, F : Longint;

Operator-operator yang berlaku pada tipe data integer, yaitu :

Operator arithmatik : +, - , \*, /, div, mod

Operator logic : <, = , > , <= , >= , <>

**Derajat Operator**

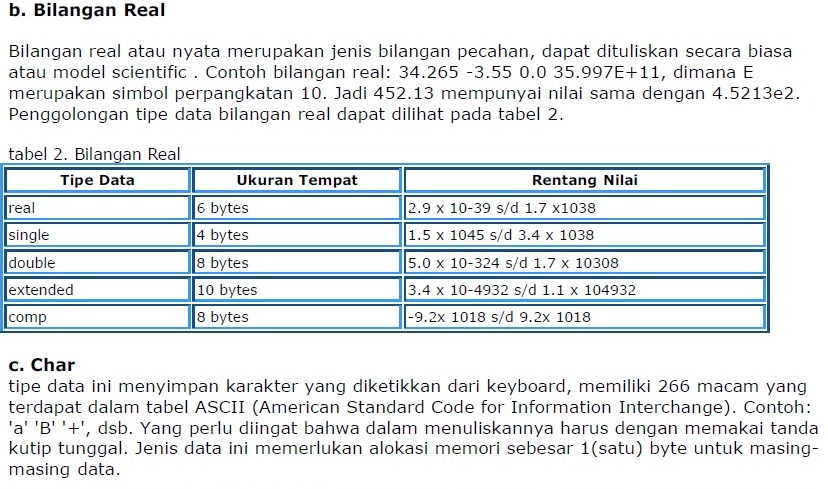
Anda sudah mengenal beberapa tipe data di dalam Pascal dan operatornya. Di dalam Pascal, operator – operator tersebut memiliki derajat atau tingakatan. Adapun kegunaandari derajat ini adalah Pascal dapat menentukan operator mana yang akan di jalankan terlebih dahulu.

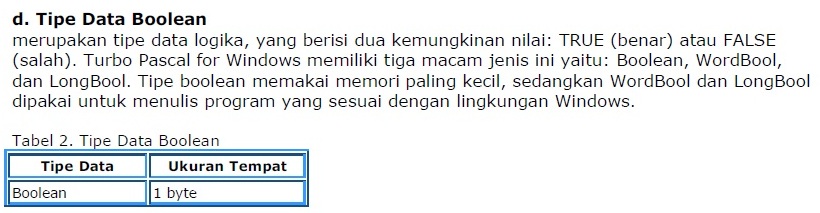
Misal: **x := 4 + 5 \* 3**

Menurut anda, berapakah nilai dari x? Apakah 27 (hasil dari 9 \* 3) atau 19 (hasil dari 4 +15) Berikut ini urutan operator berdasarkan derajatnya:

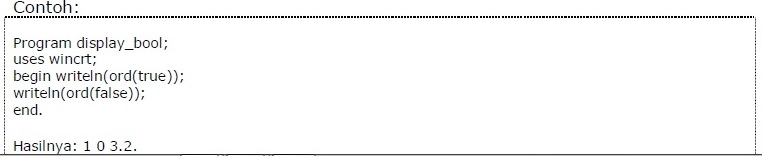
**@, not, \*, /, div, mod, as, and, shl, shr, +, -, or, xor, :=, <>, <, >, <=, >=,**

Berdasarkan dari urutan operator di atas, maka nilai x pada x := 4 + (5 \* 3) adalah 19, karena operator \* lebih dulu dikerjakan daripada operator +.









– Tipe data didefinisikan pemakai (user defined data type)

--> subrange dan enumerasi

**– Char**

Char adalah semua tombol yang terdapat pada keyboard atau lebih lengkapnya semua karakter yang terdapat pada kode ASCII.

Catatan : Apabila char ingin dijadikan sebagai konstanta maka karakter yang dimasukkan harus diapit dengan tanda kutip satu. Dan apabila karakter itu berupa tanda kutip satu maka harus diapit dengan dua tanda kutip satu

**Beberapa fungsi untuk memanipulasi tipe daa char:**

– Ord(x) dengan x adalah data bertipe char. Fungsi ini digunakan untuk memperoleh nilai urutan dalam kode ASCII yang digunakan untuk melambangkan karakter tersebut. Contoh: Ord('C') adalah 67.

– Char(x) dengan x adalah data bertipe byte. Fungsi ini adalah kebalikan dari fungsi Ord. Nilai yang diperoleh merupakan karakter ASCII yang dinyatakan dengan urutan ke x. Contoh: Char(67) adalah 'C'.

– Pred(x) dengan x adalah data bertipe char. Fungsi ini digunakan untuk mengetahui karakter yang mendahului x. Contoh: Pred('h') adalah 'g'.

– Succ(x) dengan x adalah data bertipe char. Fungsi ini digunakan untuk mengetahui karakter sesudah x. Contoh: Succ('h') adalah 'i'. Jika x tidak mempunyai penerus, maka nilai fungsi ini tidak terdefinisikan.

– Upcase(x) dengan x adalah data bertipe char. Jika x merupakan huruf kecil maka

akan dikembalikan huruf kapitalnya.

Contoh: upcase('a') adalah 'A'. – String String adalah data yang berisi sederet karakter yang terletak diantara tanda kutip satu. Jika karakter kutip merupakan bagian dari konstanta string, maka ditulis dengan menggunakan dua buah tanda kutip satu berurutan. Nilai data string akan menempati memori sebesar maksimla jumlah karakter yang dapa ditampung

ditambah denga 1 byte (index ke-0) untuk menyimpan panjang string yang sebenarnya. Jika panjang string tidak ditulis, maka panjang string dianggap 255 karakter. Panjang string yang diijinkan antara 1 sampai 255.

Bentuk umum dari deklarasi tipe string adalah: Var pengenal : string[panjang];

Dimana pengenal : nama variabel

panjang : bilangan bulat yang menunjukkan banyaknya karakter (1 – 255).

Untuk tipe data string, operator yang berlaku adalah operator penggabungan (+).

Tipe data didefinisikan pemakai (user-defined data type)

**– Subrange**

Tipe data subrange adalah suatu range yang menunjukkan nilai terkecil dan nilai

terbesar yang dapat dipergunakan. Deklarasi tipe data subrange mempunyai bentuk:

Type pengenal = konstanta1 .. konstanta2;

dimana:

pengenal : nama tipe data yang dideklarasikan

konstanta1 : batas bawah nilai data

konstanta2 : batas atas nilai data

Kedua konstanta di dalam subrange harus bertipe ordinal yang sama, di mana nilai

konstanta pertama lebih kecil atau sama dengan nilai konstanta kedua. Tipe data

real tidak dapat digunakan sebagai nilai subrange, karena buka tipe ordinal. Contoh:

Type tanggal = 1 .. 31;

bulan = 1 .. 12;

– **Enumerated**

Tipe data enumerated (skalar) menunjukkan kumpulan dari nilai yang urutannya

sudah pasti. Nilai dari tipe yang dideklarasikan ini akan diwakili dengan pengenal –

pengenal (identifiers) yang akan menjadi nilai suatu konstanta.

Contoh:

Type bahasa = (delphi, java, c, pascal, basic);

bulan = (maret, april, mei, juni, juli);

Tipe data bulan mempunya 5 elemen dari maret sampai juli. Dari urutannya, maret adalah identifier berupa konstanta bernilai 0 dan juli bernilai 4. Yang perlu diperhatikan dengan tipe data skalar ini adalah tipe data ini sudah berbeda dengan tipe standar yang ada dan pascal tidak mengijinkan operasi dengan tipe data yamg berbeda.

– Tipe data terstruktur

--> Array, record, set, file

merupakan suatu data yang menyimpan array (larik), sebagai contoh 'ABCDEF' merupakan sebuah konstanta string yang berisikan 6 byte karakter. Ukuran Tempat untuk tipe data ini adalah 2 s/d 256 byte, dengan jumlah elemen 1 s/d 255. String dideklarasikan dengan string [ konstanta ] atau string. Bila ukuran string tidak didefinisikan maka akan banyak memakan ruang, karena ukuran string menyesuaikan dengan defaultnya. Misalkan var kata: string [20]; atau var kata: string; karena string merupakan array dari karakter.

Maka kata[1] merupakan karakter pertama dari string, kemudian kata[2], merupakan elemen

kedua, dst.

Program hal\_string;

Uses wincrt;

var s : string;

begin s:='Hello';

writeln(s);

writeln('panjang dari string adalah: ',ord(s[0]));

end.

Contoh :

Program hal\_string;

Uses wincrt;

var s : string;

begin s:='Hello';

writeln(s);

writeln('panjang dari string adalah: ',ord(s[0]));

end.

– Tipe data penunjuk

--> Pointer

routines pada Pascal untuk manipulasi string :

-**Length** : menghasilkan panjang string.

Syntax : length(s)

Contoh : n:=length(s);

Misal s:='pemrograman'; n akan sama dengan 11.

-**Copy** : mengkopi suatu ekstrak karakter dari sebuah string.

Syntax : copy(s,dari,banyaknya)

Contoh : st:=copy(s,5,3); mengekstrak 3 karakter dari s, dimulai dari karakter kelima.

Misal s:='Who are you ?'; st adalah 'are'.

catatan: bila indek lebih bear dari panjang string maka hasilnya akan kosong:

Contoh : st:=copy(s,15,4); { kosong} jika banyaknya string yang akan diekstrak lebih besar

dari s, maka hasilnya adalah bagian yang tersisa dari string.

Contoh : st:=copy(s,9,10); st hasilnya adalah 'you ?'

-**Pos** : menghasilkan posisi dari suatu substring dari suatu string.

Syntax : Pos(substr,s)

Contoh : n:=pos('are','Who are you ?'); { n:=5; }

Jika substring tidak ditemukan, maka hasilnya 0.

-**Val** : mengkonversi string ke numerik.

Syntax : val(strvar,numvar,errorcode)

strvar adalah variabel string yang akan dikonversi, numvar variabel numerik ( bisa integer atau real, dan errorcode adalah variabel integer yang menyimpan error code (kode kesalahan). Jika errorcode adalah 0, konversi berhasil. Sebaliknya, hal ini akan menunjukan posisi dimana pada strvar yang menyebabkan kegagalan.

Contoh :

program konversiangka;

uses wincrt;

var s : string;

e : integer; r : real;

begin

write('masukkan angka : ');

readln(s);

val(s,r,e);

if e<>0 then writeln('salah pada posisi : ',e)

else writeln('itu adalah : ',r:4:3);

end.

-**Str** : mengkonversi numerik ke string.

Syntax : str(numvar,strvar)

program konversiangka\_1;

uses wincrt;

var s : string;

i : integer;

begin

write('masukkan angka: ');

readln(i);

str(i,s);

writeln('itu adalah: ',s);

end.

-**Concat** : Menggabungkan dua atau lebih string.

Syntax : concat(s1,s2,...,sn)

Contoh : st:=concat(s1,s2); Jikas1='ABC' dan s2='DEF',

st akan menjadi 'ABCDEF' st:=concat('Borland ','Pascal ','ver. ','7.0'); akan menjadi 'Borland

Pascal ver. 7.0'

-**Insert** : Menyisipkan suatu string ke dalam string yang lain.

Syntax : insert(asal,target,indek)

program sisip ;

uses wincrt;

var s1, s2 : string;

begin

s1:='tidak ';

s2:='saya bekerja';

insert(s1,s2,6);

writeln(s2); {saya tidak bekerja }

end.

-**Fillchar** : mengisi string s dengan karakter c sampai s sama dengan n-1 panjang karakternya.

Syntax : fillchar(s,n,c); Hati-hati : s[0] akan tertumpuki, jangan lupa menambahkan s[0]:=chr(n-1); untuk menormalkannnya.

program isikarakter;

uses wincrt;

var s : string;

begin

fillchar(s,51,'=');

s[0]:=chr(50);

end.

**b. Tipe Data Set**

Sebuah set merupakan suatu himpunan yang berisi nilai (anggota). set merupakan Tipe data yang khusus untuk Pascal. Set dalam pemrograman sangat mirip dengan himpunan dalamilmu matematika,

contoh: A = { 1, 2, 3, 4, 5 }

Syntax: set of contoh:

type Digits = set of 0..9;

Letters = set of 'A'..'Z';

type Day = (Sun, Mon, Tue, Wed, Thu, Fri, Sat);

CharSet = set of Char;

Digits = set of 0..9;

Days = set of Day;

const EvenDigits: Digits = [0, 2, 4, 6, 8];

Vowels: Letters = ['A', 'E', 'I', 'O', 'U', 'Y'];

Kita tidak bisa menulis atau membaca isi dari set, tetapi kita bisa melakukan operasi yang lain

dengan data yang ada pada set (mis. relasional).

Program contoh\_set;

Uses wincrt;

type hari = (ahad, sen, sel, rab, kam,jum, Sab);

var semua\_hari : set of hari;

hari\_kerja : set of sen .. jum;

hari\_ini : hari;

begin

hari\_ini:=sen;

if hari\_ini in hari\_kerja then

writeln('HARI INI HARI KERJA')

else

writeln('HARI LIBUR');

end.

**-Tipe Data Pointer**

pointer merupakan variabel khusus yang berisi suatu address (alamat) di lokasi lain didalam memory. Suatu variabel yang points(menunjuk) ke sesuatu sehingga disebut pointer. Ada dua macam pointer:

-typed(tertentu): merupakan pointer yang menunjuk pada tipe data tertentu pada variable.

-generic(umum): merupakan pointer yang tidak menunjuk pada tipe data tertentu pada variable.

Contoh deklarasi pointer:

var p : ^integer;

Contoh di atas merupakan deklarasi variabel p sebagai pointer dari integer. Dengan menambahkan tanda caret (^) di depan nama tipe data variabel yang dideklarasikan:

contoh :

program contoh\_pointer1;

uses wincrt;

var p : ^integer;

m, n : integer;

begin

m:=10;

n:=15;

p:=@m; { p sekarang menunjuk ke m }

p^:=12; { hal ini sama dengan m:=12; }

p:=@n; { p sekarang menunjuk ke n }

p^=m; { { hal ini sama dengan n:=m; }

writeln('m = ',m,', n = ',n); { m = 12, n = 12 }

end.

Program contoh\_pointer2;

uses wincrt;

Type Pemployee = ^Temployee;

Temployee = record Name : string[10];

Position : char;

Salary : longint;

end;

var p : pemployee;

begin

new(p);

p^.name:='Saddam';

p^.position:='S';

p^.salary:=3000;

writeln(p^.name,' ',p^.position,' ',p^.salary);

dispose(p);

end.

**4. Operator**

Beberapa operator yang disediakan oleh PASCAL:

1. Aritmatika

2. Boolean

3. Relasional

4. Set

