

MATERI DASAR PEMROGRAMAN

BAB I

ALGORITMA PEMROGRAMAN

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 1 ini siswa diharapkan dapat :

- 1) Memahami Konsep Algoritma
- 2) Memahami Struktur Algoritma
- 3) Memahami Algoritma menggunakan bahasa natural Algoritma
- 4) Memahami Pseudocode
- 5) Memahami Flowchart dan penggunaan Tool Flowchart
- 6) Pengenalan Variabel
- 7) Memahami Pengenalan tipe data
- 8) Memahami Pengenalan operator

B. Uraian Materi

1. Pengantar Algoritma Pemrograman

Belajar memprogram adalah belajar tentang strategi pemecahan masalah, metodologi dan sistematika pemecahan masalah tersebut kemudian menuangkannya dalam suatu notasi yang disepakati bersama.
“lebih bersifat pemahaman persoalan, analisis, sintesis”

Belajar bahasa pemrograman adalah belajar memakai suatu bahasa, aturan sintaks (tatabahasa), setiap instruksi yang ada dan tata cara pengoperasian kompilator atau interpreter bahasa yang bersangkutan pada mesin tertentu.

Jadi :

”

1.1 Algoritma dan pemrograman Dasar



Perangko dari Rusia pada Gambar di samping ini bergambar seorang pria dengan nama Muhammad bin Musa al-Khwarizmi. Bagi kalian yang sedang berkecimpung dalam dunia komputer maka seharusnya mengetahui siapa orang di samping ini. Dia adalah seorang ilmuwan Islam yang karya karyanya dalam bidang matematika, astronomi, astrologi dan geografi banyak menjadi dasar perkembangan ilmu modern. Dan dari namanya istilah yang akan kita pelajari dalam bab ini muncul. Dari Al-Khwarizmi

kemudian berubah menjadi **algorithm** dalam Bahasa Inggris dan diterjemahkan menjadi **algoritma** dalam bahasa Indonesia.

1.2 Definisi Algoritma

1. **Algoritma** adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis.

Algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki tingkat kerumitan yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan suatu masalah membutuhkan tingkat kerumitan yang tinggi.

1.3 Struktur Algoritma

Perhatikan algoritma sederhana berikut :

Jika seseorang ingin mengirim surat kepada kenalan di tempat lain, langkah yang harus dilakukan adalah:

1. Menyiapkan Peralatan Tulis
2. Menulis surat
3. Surat dimasukkan ke dalam amplop tertutup
4. Amplop ditempel perangko secukupnya.
5. Pergi ke Kantor Pos terdekat untuk mengirimkannya

Algoritma menghitung luas persegi panjang:

1. Masukkan panjang (P)
2. Masukkan lebar (L)
3. Luas $P * L$
4. Tulis Luas

Pembuatan algoritma mempunyai banyak keuntungan di antaranya:

- a) Pembuatan atau penulisan algoritma tidak tergantung pada bahasa pemrograman manapun, artinya penulisan algoritma independen dari bahasa pemrograman dan komputer yang melaksanakannya.
- b) Notasi algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman.
- c) Apapun bahasa pemrogramannya, output yang akan dikeluarkan sama karena algoritmanya sama.

Beberapa hal yang perlu diperhatikan dalam membuat algoritma:

- a) Teks algoritma berisi deskripsi langkah-langkah penyelesaian masalah. Deskripsi tersebut dapat ditulis dalam notasi apapun asalkan mudah dimengerti dan dipahami.
- b) Tidak ada notasi yang baku dalam penulisan teks algoritma seperti notasi bahasa pemrograman. Notasi yang digunakan dalam menulis algoritma disebut notasi algoritmik.
- c) Setiap orang dapat membuat aturan penulisan dan notasi algoritmik sendiri. Hal ini dikarenakan teks algoritma tidak sama dengan teks program. Namun, supaya notasi algoritmik mudah ditranslasikan ke dalam notasi bahasa pemrograman tertentu, maka sebaiknya notasi algoritmik tersebut berkorespondensi dengan notasi bahasa pemrograman secara umum.
- d) Notasi algoritmik bukan notasi bahasa pemrograman, karena itu pseudocode dalam notasi algoritmik tidak dapat dijalankan oleh komputer. Agar dapat dijalankan oleh komputer, pseudocode dalam notasi algoritmik harus ditranslasikan atau diterjemahkan ke dalam notasi bahasa pemrograman yang dipilih. Perlu diingat bahwa orang yang menulis program sangat terikat dalam aturan tata bahasanya dan spesifikasi mesin yang menjalannya. *Pseudocode* adalah kode yang mirip dengan instruksi kode program sebenarnya.
- e) Algoritma sebenarnya digunakan untuk membantu kita dalam mengkonversikan suatu permasalahan ke dalam bahasa pemrograman.
- f) Algoritma merupakan hasil pemikiran konseptual, supaya dapat dilaksanakan oleh komputer, algoritma harus ditranslasikan ke dalam notasi bahasa pemrograman

Perhatikan algoritma sederhana berikut :

Algoritma menghitung luas segitiga

1. Start
2. Baca data alas dan tinggi.
3. Luas adalah alas kali tinggi kali 0.5
4. Tampilkan Luas
5. Stop

Penjelasan :

Algoritma di atas adalah algoritma yang sangat sederhana, hanya ada lima langkah. Pada algoritma ini tidak dijumpai perulangan ataupun pemilihan. Semua langkah dilakukan hanya satu kali.

Sekilas algoritma di atas benar, namun apabila dicermati maka algoritma ini mengandung kesalahan yang mendasar, yaitu tidak ada pembatasan pada nilai data untuk alas dan tinggi.

Hasil perbaikan algoritma perhitungan luas segitiga

1. Start
2. Baca data alas dan tinggi.
3. Periksa data alas dan tinggi, jika nilai data alas dan tinggi lebih besar dari nol maka lanjutkan ke langkah ke 4 jika tidak maka stop
4. Luas adalah alas kali tinggi kali 0.5
5. Tampilkan Luas
6. Stop

Dari penjelasan di atas dapat diambil kesimpulan pokok tentang algoritma. Pertama, algoritma harus benar. Kedua algoritma harus berhenti, dan setelah berhenti, algoritma memberikan hasil yang benar.

Contoh : Algoritma Berangkat Sekolah

Mulai

Bangun dari tempat tidur
Mandi Pagi
Sarapan Pagi
Pergi Ke Sekolah
Cari Ruang Kelas
Masuk kelas untuk Belajar

Selesai

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.

Program = Algoritma + Bahasa (Struktur Data)

Penerjemah Bahasa Pemrograman

Untuk menterjemahkan bahasa pemrograman yang kita tulis maka diperlukan *Compiler* dan *interpreter*.

Compiler adalah suatu program yang menterjemahkan bahasa program (Source code) ke dalam bahasa obyek (object code) secara keseluruhan program.

Interpreter berbeda dengan *Compiler*, *interpreter* menganalisis dan mengeksekusi setiap baris dari program secara keseluruhan. Keuntungan dari interpreter adalah dalam eksekusi yang bisa dilakukan dengan segera. Tanpa melalui tahap kompilasi, untuk alasan ini interpreter digunakan pada saat pembuatan program berskala besar.

Perbedaan Compiler dan interpreter.

| Compiler | Interpreter |
|---|--|
| Menterjemahkan secara keseluruhan | Menterjemahkan Instruksi per instruksi |
| Bila terjadi kesalahan kompilasi maka source program harus diperbaiki dan dikompilasi ulang | Bila terjadi kesalahan interpretasi dapat Diperbaiki |
| Dihasilkan Object program | Tidak dihasilkan obyek program |
| Dihasilkan Executable program | Tidak dihasilkan Executable program |
| Proses pekerjaan program lebih cepat | Proses pekerjaan program lebih lambat |
| Source program tidak dipergunakan hanya bila untuk perbaikan saja | Source program terus dipergunakan |
| Keamanan dari program lebih terjamin | Keamanan dari program kurang terjamin |

1.4 Jenis-Jenis Bahasa Pemrograman

- Bahasa Pemrograman Tingkat rendah (Bahasa mesin, Biner)
- Bahasa Pemrograman Tingkat tinggi

Contoh-contoh Bahasa Pemrograman yang ada :

1. Prosedural : Algol, Pascal, Fortran, Basic, Cobol, C
2. Fungsional : LOGO, APL, LISP
3. Deklaratif : Prolog

Object oriented murni: Smalltalk, Eifel, Java, PHP

Cara penulisan algoritma

Ada tiga cara penulisan algoritma, yaitu :

1. Structured English (SE)

SE merupakan alat yang cukup baik untuk menggambarkan suatu algoritma. Dasar dari SE adalah Bahasa Inggris, namun kita dapat memodifikasi dengan Bahasa Indonesia sehingga kita boleh menyebutnya sebagai Structured Indonesian (SI).

"SE atau SI lebih tepat untuk menggambarkan suatu algoritma yang akan dikomunikasikan kepada pemakai perangkat lunak"

2. Pseudocode

Pseudocode adalah kode yang mirip dengan instruksi kode program sebenarnya. Pseudocode didasarkan pada bahasa pemrograman yang sesungguhnya seperti BASIC,

FORTRAN atau PASCAL. Pseudocode yang berbasis bahasa PASCAL merupakan pseudocode yang sering digunakan.

“Pseudo berarti imitasi atau tiruan atau menyerupai, sedangkan code menunjuk pada kode program”

Contoh Pseudocode :

1. Start
2. READ alas, tinggi
3. Luas = $0.5 * \text{alas} * \text{tinggi}$
4. PRINT Luas
5. Stop

Pada Contoh diatas tampak bahwa algoritma sudah sangat mirip dengan bahasa BASIC. Pernyataan seperti READ dan PRINT merupakan keyword yang ada pada bahasa BASIC yang masing-masing menggantikan kata “baca data” dan “tampilkan”. Dengan menggunakan pseudocode seperti di atas maka proses penterjemahan dari algoritma ke kode program menjadi lebih mudah.

1.5 Membuat Alur Logika Pemrograman

A. Penyajian atau Penulisan Algoritma

Penyajian algoritma secara garis besar bisa dalam 2 bentuk penyajian yaitu tulisan dan gambar. Algoritma yang disajikan dengan tulisan yaitu dengan struktur bahasa tertentu (misalnya bahasa Indonesia atau bahasa Inggris) dan pseudocode.

Pseudocode adalah kode yang mirip dengan kode pemrograman yang sebenarnya seperti Pascal, atau C, sehingga lebih tepat digunakan untuk menggambarkan algoritma yang akan dikomunikasikan kepada pemrogram. Sedangkan algoritma disajikan dengan gambar, yaitu dengan Flowchart

B. Flowchart (Diagram Alir)

Flowchart atau bagan alir adalah skema/bagan (chart) yang menunjukkan aliran (flow) di dalam suatu program secara logika.

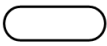


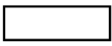


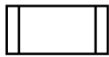


Flowchart merupakan alat yang banyak digunakan untuk menggambarkan algoritma dalam bentuk notasi-notasi tertentu. Flowchart merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta pernyataannya. Gambaran ini dinyatakan dengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu. Sedangkan antara proses digambarkan dengan garis penghubung. Dengan menggunakan flowchart akan memudahkan kita untuk melakukan pengecekan bagian-bagian yang terlupakan dalam analisis masalah. Di samping itu flowchart juga berguna sebagai fasilitas untuk berkomunikasi antara pemrogram yang bekerja dalam tim suatu proyek.

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

- 1) Hindari pengulangan proses yang tidak perlu dan logika yang berbelit sehingga jalannya proses menjadi singkat.
- 2) Jalannya proses digambarkan dari atas ke bawah dan diberikan tanda panah untuk memperjelas.




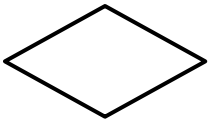

3) Sebuah flowchart diawali dari satu titik START dan diakhiri dengan END.

Berikut merupakan beberapa contoh simbol flowchart yang disepakati oleh duniapemrograman:

| Keterangan | Lambang |
|---|---|
| Mulai/selesai (terminator) |  |
| Aliran data |  |
| Input/Output |  |
| Proses |  |
| Percabangan (Decision) |  |
| Pemberian nilai awal suatu variabel (Preparation) |  |
| Memanggil prosedur/fungsi (Call) |  |
| Connector (di halaman yg sama) |  |
| Off page Connector (halaman lain) |  |

Penjelasan lebih lanjut :

Simbol-simbol bagan alir program (Flowchart)

-  Notasi Membuat algoritma sederhana untuk menyelesaikan permasalahan menggunakan bahasa natural, flowchart dan pseudocode
-  Notasi ini disebut Data yang digunakan untuk mewakili data input atau output atau menyatakan operasi pemasukan data dan pencetakan hasil
-  Notasi ini disebut Process yang digunakan untuk mewakili suatu proses
-  Notasi ini disebut Decision yang digunakan untuk suatu pemilihan, penyeleksian kondisi di dalam suatu program
-  Notasi ini disebut Preparation yang digunakan untuk memberi nilaiawal, nilai akhir, penambahan/pengurangan bagi suatu variabel counter.



Notasi ini disebut Predefined Process yang digunakan untuk menunjukkan suatu operasi yang rinciannya ditunjukkan ditempat lain (prosedur, sub-prosedur, fungsi)

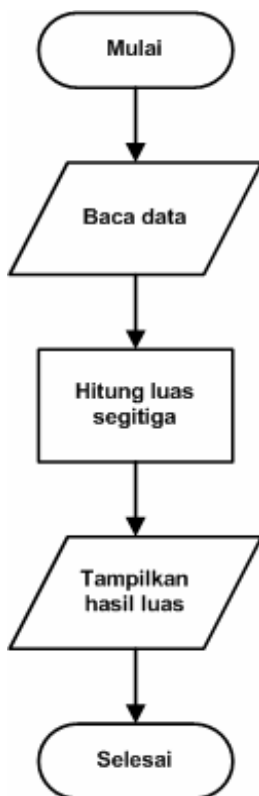


Notasi ini disebut Connector yang digunakan untuk menunjukkan sambungan dari flowchart yang terputus di halaman yang sama atau halaman berikutnya.

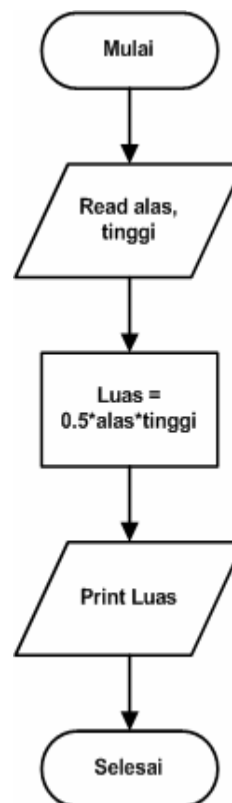


Notasi ini disebut Arrow yang digunakan untuk menunjukkan arus data atau alir data dari proses satu ke proses lainnya.

Contoh program Flowchart



Bagan alir logika program



Bagan alir program komputer terinci

C. Struktur Dasar Algoritma

Algoritma berisi langkah-langkah penyelesaian suatu masalah. Langkah-langkah tersebut dapat berupa runtunan aksi (sequence), pemilihan aksi (selection), pengulangan aksi (iteration) atau kombinasi dari ketiganya. Jadi struktur dasar pembangunan algoritma ada tiga, yaitu:

1. Struktur Runtunan / Beruntun : Digunakan untuk program yang pernyataannya sequential atau urutan.
2. Struktur Pemilihan / Percabangan : Digunakan untuk program yang menggunakan pemilihan atau penyeleksian kondisi.
3. Struktur Perulangan : Digunakan untuk program yang pernyataannya akan dieksekusi berulang-ulang.

1. Struktur Algoritma Runtunan / Berurutan :

Ada tiga struktur dasar yang digunakan dalam membuat algoritma yaitu struktur berurutan (sequencing), struktur pemilihan/keputusan/percabangan (branching) dan struktur pengulangan (looping). Sebuah algoritma biasanya akan menggabungkan ketiga buah struktur ini untuk menyelesaikan masalah.

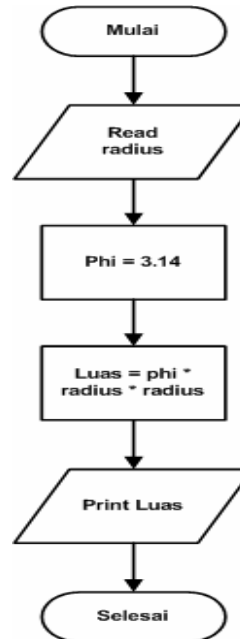
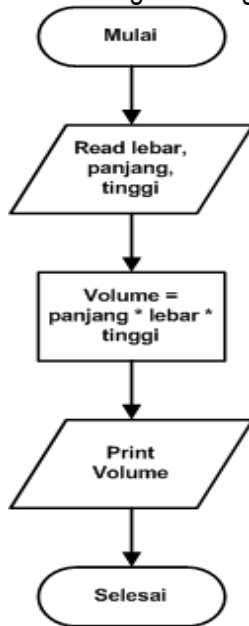
Struktur berurutan dapat kita samakan dengan mobil yang sedang berjalan pada jalur lurus yang tidak terdapat persimpangan seperti tampak pada Gambar disamping Mobil tersebut akan melewati kilometer demi kilometer jalan sampai tujuan tercapai. *Struktur berurutan terdiri satu atau lebih instruksi.*

Tiap instruksi dikerjakan secara berurutan sesuai dengan urutan penulisannya, yaitu sebuah instruksi dieksekusi setelah instruksi sebelumnya selesai dieksekusi. Urutan instruksi menentukan keadaan akhir dari algoritma. Bila urutannya diubah, maka hasil akhirnya mungkin juga berubah.

Menurut Goldshlager dan Lister (1988) struktur berurutan mengikuti ketentuan-ketentuan sebagai berikut:

1. Tiap instruksi dikerjakan satu persatu
2. Tiap instruksi dilaksanakan tepat sekali, tidak ada yang diulang
3. Urutan instruksi yang dilaksanakan pemroses sama dengan urutan aksi sebagaimana yang tertulis di dalam algoritmanya
4. Akhir dari instruksi terakhir merupakan akhir algoritma.

Contoh bagan alir logika program berurutan (sequencing)



1. Struktur Algoritma Percabangan

Sebuah program tidak selamanya akan berjalan dengan mengikuti struktur berurutan, kadang-kadang kita perlu merubah urutan pelaksanaan program dan menghendaki agar pelaksanaan program meloncat ke baris tertentu. Peristiwa ini kadang disebut sebagai percabangan/pemilihan atau keputusan. Hal ini seperti halnya ketika mobil/motor berada dalam persimpangan

Pada struktur percabangan, program akan berpindah urutan pelaksanaan jika suatu kondisi yang disyaratkan dipenuhi. Pada proses seperti ini simbol flowchart Decision harus digunakan. Simbol decision akan berisi pernyataan yang akan diuji kebenarannya. Nilai hasil pengujian akan menentukan cabang mana yang akan ditempuh.

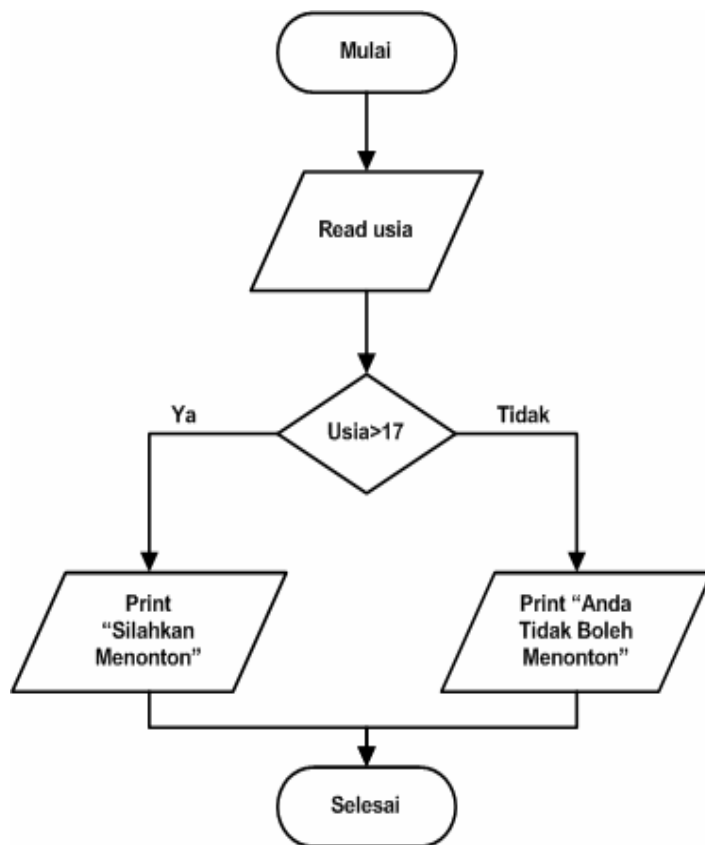
Contoh Struktur percabangan untuk masalah batasan umur.

Sebuah aturan untuk menonton sebuah film tertentu adalah sebagai berikut, jika usia penonton lebih dari 17 tahun maka penonton diperbolehkan dan apabila kurang dari 17 tahun maka penonton tidak diperbolehkan nonton. Buatlah flowchart untuk permasalahan tersebut.

Penyelesaian:

Permasalahan diatas merupakan ciri permasalahan yang menggunakan struktur percabangan. Hal ini ditandai dengan adanya pernyataan jika ..maka ...(atau If ... Then dalam Bahasa Inggris).

Bagan alir logika (Flowchart) penyelesaian masalah nonton film



2. Struktur Algoritma Perulangan / Pengulangan

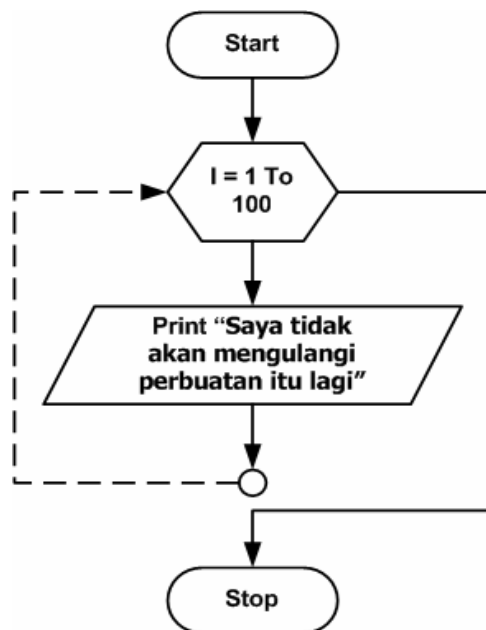
Dalam banyak kasus seringkali kita dihadapkan pada sejumlah pekerjaan yang harus diulang berkali-kali. Salah satu contoh yang gampang kita jumpai adalah balapan mobil

Struktur pengulangan terdiri dari dua bagian :

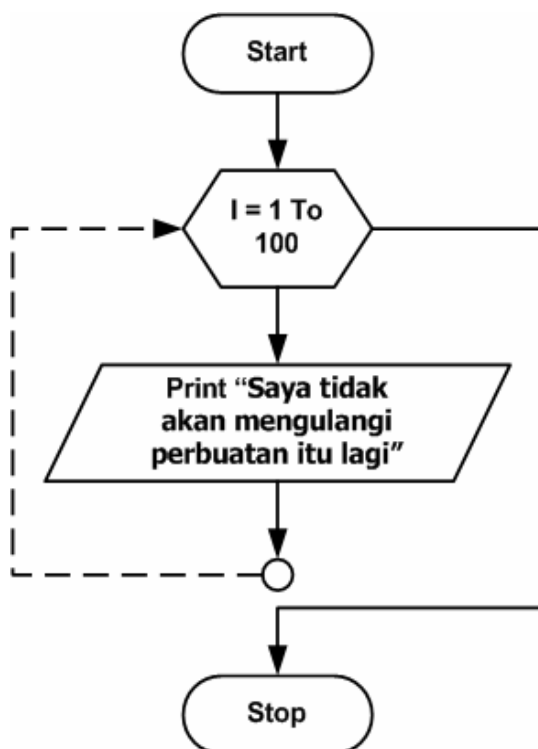
- Kondisi pengulangan, yaitu syarat yang harus dipenuhi untuk melaksanakan pengulangan. Syarat ini biasanya dinyatakan dalam ekspresi Boolean yang harus diuji apakah bernilai benar (true) atau salah (false)
- Badan pengulangan (loop body), yaitu satu atau lebih instruksi yang akan diulang

Pada struktur pengulangan, biasanya juga disertai bagian *inisialisasi* dan bagian *terminasi*. **Inisialisasi** adalah instruksi yang dilakukan sebelum pengulangan dilakukan pertama kali. Bagian inisialisasi umumnya digunakan untuk memberi nilai awal sebuah variabel. Sedangkan **terminasi** adalah instruksi yang dilakukan setelah pengulangan selesai dilaksanakan. Ada beberapa bentuk pengulangan yang dapat digunakan, masing-masing dengan syarat dan karakteristik tersendiri. Beberapa bentuk dapat dipakai untuk kasus yang sama, namun ada bentuk yang hanya cocok untuk kasus tertentu saja. Pemilihan bentuk pengulangan untuk masalah tertentu dapat mempengaruhi kebenaran algoritma. Pemilihan bentuk pengulangan yang tepat bergantung pada masalah yang akan diprogram.

Bagan alir logika (flowchart) untuk mencetak pernyataan sebanyak 100 kali



Bagan alir logika (Flowchart) untuk mencetak anggota suatu himpunan.



▮ Struktur pengulangan dengan For

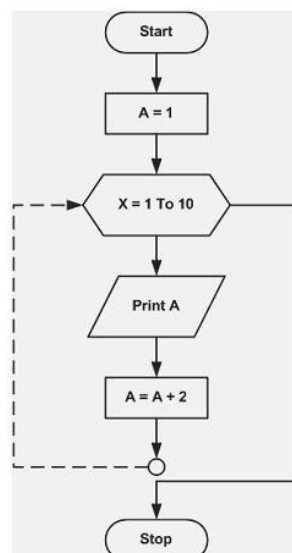
Pengulangan dengan menggunakan For, merupakan salah teknik pengulangan yang paling tua dalam bahasa pemrograman. Hampir semua bahasa pemrograman menyediakan metode ini, meskipun sintaksnya mungkin berbeda. Pada struktur For kita harus tahu terlebih dahulu seberapa banyak badan loop akan diulang. Struktur ini menggunakan sebuah variable yang biasa disebut sebagai loop s counter, yang nilainya akan naik atau turun selama proses pengulangan.

Contoh :

Diketahui sebuah himpunan A yang beranggotakan bilangan 1, 3, 5, ..., 19. Buatlah flowchart untuk mencetak anggota himpunan tersebut.

Penyelesaian:

Pada contoh ini, kita mencoba menentukan hasil dari sebuah flowchart . Bagaimana menurut kalian jawabannya? Marilah kita uraikan jalannya flowchart tersebut. Pada flowchart, setelah Start, kita meletakkan satu proses yang berisi pernyataan $A = 1$. Bagian inilah yang disebut inisialisasi . Kita memberi nilai awal untuk $A = 1$. Variabel counter-nya adalah X dengan nilai awal 1 dan nilai akhir 10, tanpa increment (atau secara default increment-nya adalah 1). Ketika masuk ke badan loop untuk pertama kali maka akan dicetak langsung nilai variabel A. Nilai variabel A masih sama dengan 1. Kemudian proses berikutnya adalah pernyataan $A = A + 2$. Pernyataan ini mungkin agak aneh, tapi ini adalah sesuatu yang pemrograman. Arti dari pernyataan ini adalah gantilah nilai A yang lama dengan hasil penjumlahan nilai A lama ditambah 2. Sehingga A akan bernilai 3. Kemudian dilakukan pengulangan yang ke-dua. Pada kondisi ini nilai A adalah 3, sehingga yang tercetak oleh perintah print adalah 3. Baru kemudian nilai A kita ganti dengan penjumlahan $A + 2$. Nilai A baru adalah 5. Demikian seterusnya. Sehingga output dari flowchart ini adalah 1, 3, 5, 7, ..., 19.

**Struktur pengulangan dengan While**

Pada pengulangan dengan For, banyaknya pengulangan diketahui dengan pasti karena nilai awal (start) dan nilai akhir (end) sudah ditentukan diawal pengulangan. Bagaimana jika kita tidak tahu pasti harus berapa kali mengulang? Pengulangan dengan While merupakan jawaban dari permasalahan ini. Seperti halnya For, struktur pengulangan dengan While juga merupakan struktur yang didukung oleh hampir semua bahasa pemrograman namun dengan sintaks yang berbeda.

Struktur While akan mengulang pernyataan pada badan loop sepanjang kondisi pada While bernilai benar. Dalam artian kita tidak perlu tahu pasti berapa kali diulang. Yang penting sepanjang kondisi pada While dipenuhi maka pernyataan pada badan loop akan diulang.

Penyelesaian: Perhatikan Gambar. bisakah kalian menentukan hasil dari flowchart tersebut? Perhatikan tahapan eksekusi flowchart berikut ini.

- Pada flowchart ini ada dua variabel yang kita gunakan yaitu A dan B. Kedua variabel tersebut kita inisialisasi nilai awalnya ($A = 1$ dan $B = 0$) sebelum proses loop terjadi. Variabel A adalah variabel counter.
- Pada simbol decision, nilai A akan diperiksa apakah memenuhi kondisi ($A < 10$). Jika Ya maka perintah berikutnya dieksekusi, jika tidak maka program akan berhenti. Pada awal eksekusi ini kondisi akan terpenuhi karena nilai $A = 1$.
- Jalankan perintah Print B.
- Nilai variabel A kemudian diganti dengan nilai A lama (1) ditambah 2. Sehingga nilai variabel A baru adalah 3. Sedangkan nilai variabel B = 9 (hasil perkalian $A = 3$).
- Program akan berputar kembali untuk memeriksa apakah nilai variabel A masih lebih kecil dari 10. Pada kondisi ini nilai $A = 3$, sehingga kondisi masih terpenuhi. Kemudian langkah berulang ke langkah ke 3. Begitu seterusnya sampai nilai variabel A tidak lagi memenuhi syarat kurang dari 10.

BAB II BAHASA PEMROGRAMAN

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 3 ini siswa diharapkan dapat :

- 1) Memahami Tipe Data
- 2) Memahami Variabel
- 3) Memahami Operator
- 4) Memahami Ekspresi

B. Uraian Materi

Tipe Data, Variabel Konstanta, Operator, dan Ekspresi

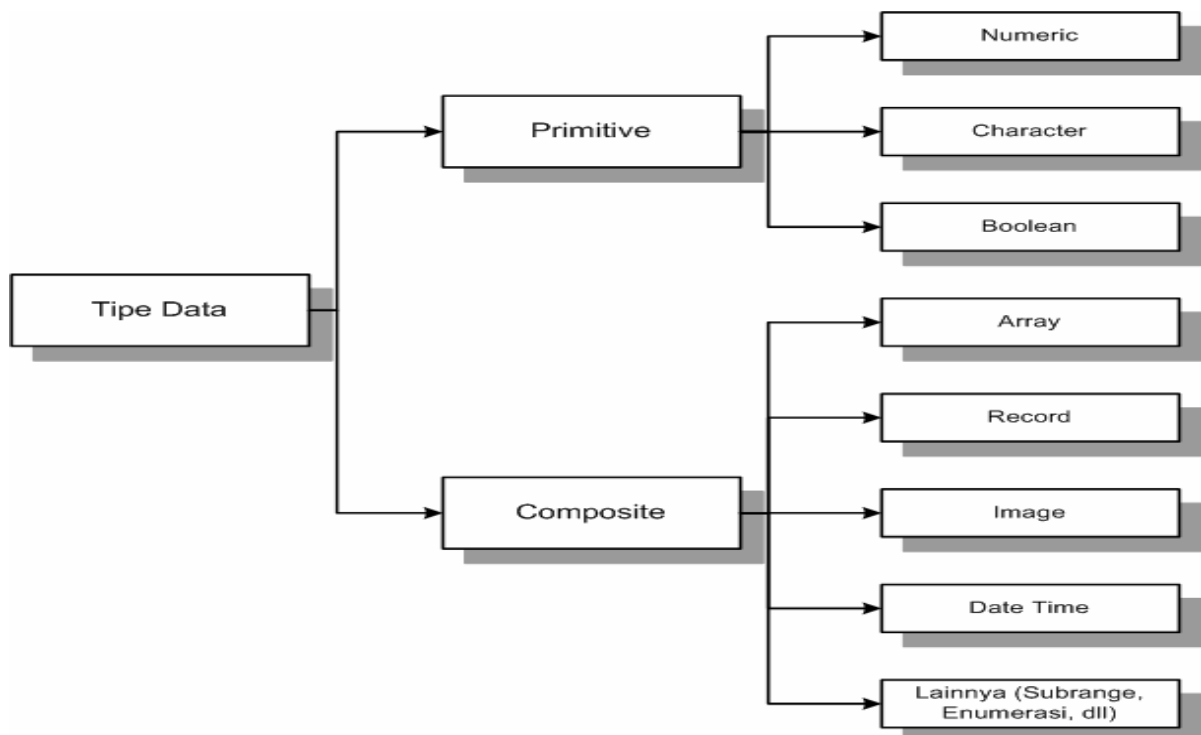
Variabel, konstanta dan tipe data merupakan tiga hal yang akan selalu kita jumpai ketika kita membuat program. Bahasa pemrograman apapun dari yang paling sederhana sampai yang paling kompleks, mengharuskan kita untuk mengerti ketiga hal tersebut.

1. Tipe Data

Tipe data adalah jenis data yang dapat diolah oleh komputer untuk memenuhi kebutuhan dalam pemrograman komputer.

Setiap variabel atau konstanta yang ada dalam kode program, sebaiknya kita tentukan dengan pasti tipe datanya. Ketepatan pemilihan tipe data pada variabel atau konstanta akan sangat menentukan pemakaian sumberdaya komputer (terutama memori komputer). Salah satu tugas penting seorang programmer adalah memilih tipe data yang sesuai untuk menghasilkan program yang efisien dan berkinerja tinggi.

Ada banyak tipe data yang tersedia tergantung jenis bahasa pemrograman yang dipakai. Namun secara umum dapat dikelompokkan seperti pada Gambar dibawah ini



Ada 2 jenis tipe data :

1. Tipe data primitive adalah tipe data dasar yang tersedia secara langsung pada suatu bahasa pemrograman.
2. Tipe data composite adalah tipe data bentukan yang terdiri dari dua atau lebih tipe data primitive.

▮ **Tipe data numeric**

Tipe data numeric digunakan pada variabel atau konstanta untuk menyimpan nilai dalam bentuk bilangan atau angka. Semua bahasa pemrograman menyediakan tipe data numeric, hanya berbeda dalam jenis numeric yang diakomodasi.

2. Variabel

Variabel adalah tempat dimana kita dapat mengisi atau mengosongkan nilainya dan memanggil kembali apabila dibutuhkan. Setiap variabel akan mempunyai nama (identifier) dan nilai.

Contoh Nama variabel dan nilai.

username = "joni"

Nama = "Udin"

Harga = 2500

HargaTotal = 34000

Pada sebagian besar bahasa pemrograman, variabel harus dideklarasikan lebih dulu untuk mempermudah compiler bekerja. Apabila variabel tidak dideklarasikan maka setiap kali compiler bertemu dengan variabel baru Pemberian nama variabel harus mengikuti aturan yang ditetapkan oleh bahasa pemrograman yang kita gunakan. Namun secara umum ada aturan yang berlaku untuk hampir semua bahasa pemrograman. Aturan-aturan tersebut yaitu :

1. Nama variabel harus diawali dengan huruf.
2. Tidak boleh menggunakan spasi pada satu nama variabel. Spasi bisa diganti dengan karakter underscore (_).
3. Nama variabel tidak boleh mengandung karakter-karakter khusus, seperti : ., +, -, *, /, <, >, &, (,) dan lain-lain.
4. Nama variabel tidak boleh menggunakan kata-kata kunci d bahasa pemrograman

Contoh penamaan variabel

| Penamaan yang benar | Penamaan yang salah |
|---------------------|--|
| namasiswa XY12 | nama siswa (salah karena menggunakan spasi) 12X (salah karena dimulai dengan angka) |
| harga_total | harga.total (salah karena menggunakan karakter) |
| JenisMotor | Jenis Motor (salah karena menggunakan spasi) |
| alamatRumah | for (salah karena menggunakan kata kunci bahasa pemrograman) |

Jenis-jenis Variabel

1) Variabel Numerik

Variabel numerik ini dibagi menjadi menjadi 3 (tiga) macam :

- Bilangan Bulat
- Bilangan Desimal Berpresisi Tunggal atau Floating Point.
- Bilangan Desimal Berpresisi Ganda atau Double Precision.

2) Variabel Text

- Character (Karakter Tunggal)
- String (Untuk Rangkaian Karakter)

3. Konstanta

Konstanta adalah variabel yang nilai datanya bersifat tetap dan tidak bisa diubah. Jadi konstanta adalah juga variabel bedanya adalah pada nilai yang disimpannya. Jika nilai datanya sepanjang program berjalan tidak berubah-ubah, maka sebuah variabel lebih baik diperlakukan sebagai konstanta.

Sebagai contoh, jika kita membuat program perhitungan matematik yang menggunakan nilai pi (3.14159) yang mungkin akan muncul dibanyak tempat pada kode program, kita dapat membuat pi sebagai konstanta. Penggunaan konstanta pi akan lebih memudahkan penulisan kode program dibanding harus mengetikkan nilai 3.14159 berulang-ulang.

□ Character

Bersama dengan tipe data numeric, character merupakan tipe data yang paling banyak digunakan. Tipe data character kadang disebut sebagai char atau string. Tipe data string hanya dapat digunakan menyimpan teks atau apapun sepanjang berada dalam tanda petik dua ("...") atau petik tunggal ('...'). Perhatikan contoh berikut.

□ Boolean

Tipe data Boolean digunakan untuk menyimpan nilai True/False (Benar/Salah). Pada sebagian besar bahasa pemrograman nilai selain 0 menunjukkan True dan 0 melambangkan False. Tipe data ini banyak digunakan untuk pengambilan keputusan pada struktur percabangan dengan IF ... THEN atau IF ... THEN ... ELSE.

Contoh :

Program Pascal

```
If Nilai >= 60 Then
    writeln('Lulus Ujian');
Else
    Writeln('Tidak lulus');
End if
```

□ Array

Array atau sering disebut sebagai larik adalah tipe data yang sudah terstruktur dengan baik, meskipun masih sederhana. Array mampu menyimpan sejumlah data dengan tipe yang sama (homogen) dalam sebuah variabel. Setiap lokasi data array diberi nomor indeks yang berfungsi sebagai alamat dari data tersebut.

Contoh:

Penggunaan Array

Var

X: array[1..100] of integer;

Cara mengisi data pada elemen larik dalam pemrograman adalah seperticontoh berikut

:

X[1]:= 4;

X[2]:= 3;

X[3]:= 2;

X[4]:= 1;

▢ **Record atau Struct**

Seperti halnya Array, Record atau Struct adalah termasuk tipe data komposit. Record dikenal dalam bahasa Pascal/Delphi sedangkan Struct dikenal dalam bahasa C++. Berbeda dengan array, tipe data record mampu menampung banyak data dengan tipe data berbeda-beda (heterogen).

Sebagai ilustrasi array mampu menampung banyak data namun dengan satu tipe data yang sama, misalnya integer saja. Sedangkan dalam record, kita bisa menggunakan untuk menampung banyak data dengan tipe data yang berbeda, satu bagian integer, satu bagian lagi character, dan bagian lainnya Boolean. Biasanya record digunakan untuk menampung data suatu obyek. Misalnya, siswa memiliki nama, alamat, usia, tempat lahir, dan tanggal lahir. Nama akan menggunakan tipe data string, alamat bertipe data string, usia bertipe data single (numeric), tempat lahir bertipe data string dan tanggal lahir bertipe data date.

▢ **Image**

Image atau gambar atau citra merupakan tipe data grafik

▢ **Date Time**

Nilai data untuk tanggal (Date) dan waktu (Time) secara internal disimpan dalam format yang spesifik. Variabel atau konstanta yang dideklarasikan dengan tipe data Date dapat digunakan untuk menyimpan baik tanggal maupun jam. Tipe data ini masuk dalam kelompok tipe data composite karena merupakan bentukan dari beberapa tipe data.

Berikut ini contoh tipe data dalam Visual Basic.

```
Dim WaktuLahir As Date
WaktuLahir = "01/01/1997"
WaktuLahir = "13:03:05 AM"
WaktuLahir = "02/23/1998 13:13:40 AM"
WaktuLahir = #02/23/1998 13:13:40 AM#
```

▢ **Subrange**

Tipe data subrange merupakan tipe data bilangan yang mempunyai jangkauan nilai tertentu sesuai dengan yang ditetapkan programmer. Biasanya tipe data ini mempunyai nilai batas minimum dan nilai batas maksimum. Tipe data ini didukung dengan sangat baik dalam Delphi.

Berikut ini contoh deklarasi tipe data subrange dalam Delphi.

```
Type
    BatasIndeks = 1..20
    RentangTahun = 1950..2030
Var
    Indeks : BatasIndeks
    Tahun : RentangTahun
```

▢ **Enumerasi**

Tipe data ini merupakan tipe data yang mempunyai elemen-elemen yang harus disebut satupersatu dan bernilai konstanta integer sesuai dengan urutannya. Nilai konstanta integer

elemen ini diwakili oleh suatu nama variable yang ditulis di dalam kurung. Tipe data ini juga dijumpai pada Delphi dan bahasa pemrograman deklaratif seperti SQL.

Berikut ini contoh deklarasi tipe data enumerasi dalam Delphi.

Type

Hari_dlm_Minggu = (NoI, Senin, Selasa, Rabu, Kamis, Jumat, Sabtu, Minggu)

Nama_Bulan = (NoI, Januari, Pebruari, Maret, April, Mei, Juni, Juli, Agustus, September, Oktober, Nopember, Desember)

Var

No_Hari : Hari_dlm_Minggu

No_Bulan : Nama_Bulan

▢ Object

Tipe data object digunakan untuk menyimpan nilai yang berhubungan dengan obyek-obyek yang disediakan oleh Visual Basic, Delphi dan bahasa pemrograman lain yang berbasis GUI. Sebagai contoh, apabila kita mempunyai form yang memiliki control Command button yang kita beri nama Command1, kita dapat mendeklarasikan variabel sebagai berikut :

Contoh Penggunaan tipe data object.

```
Dim A As CommandButton
Set A = Command1
A.Caption = "HEY!!!"
A.FontBold = True
```

4. Operator

Operator merupakan simbol atau karakter yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi, seperti penjumlahan, pengurangan dan lain lain.

Operator mempunyai sifat sebagai berikut :

▢ Unary

Sifat Unary pada operator adalah hanya melibatkan sebuah operand pada suatu operasi aritmatika

Contoh : -5

▢ Binary

Sifat Binary pada operator adalah melibatkan dua buah operand pada suatu operasi aritmatika

Contoh : 4 + 8

▢ Ternary

Sifat Ternary pada operator adalah melibatkan tiga buah operand pada suatu operasi aritmatika

Contoh : (10 % 3) + 4 + 2

Operator Aritmatika

Operator untuk operasi aritmatika yang tergolong sebagai operator binary adalah : Tabel Operator Aritmatika

| Operator | Keterangan | Contoh |
|----------|----------------|--------|
| * | Perkalian | 4 * 5 |
| / | Pembagian | 8 / 2 |
| % | Sisa Pembagian | 5 % 2 |
| + | Penjumlahan | 7 + 2 |
| - | Pengurangan | 6 - 2 |

Tabel Operator Unary

| Operator | Keterangan | Contoh |
|----------|-------------|--------|
| + | Tanda Plus | -5 |
| - | Tanda Minus | +6 |

Operator Penambah dan Pengurang

Masih berkaitan dengan operator pemberi nilai, Bahasa C menyediakan operator penambah dan pengurang. Dari contoh penulisan operator pemberi nilai sebagai penyederhanaannya dapat digunakan operator penambah dan pengurang.

Tabel Operator Penambah dan Pengurang

| Operator | Keterangan |
|----------|-------------|
| ++ | Penambahan |
| -- | Pengurangan |

$A = A + 1$ atau $A = A - 1$; disederhanakan menjadi :

$A += 1$ atau $A -= 1$; masih dapat disederhanakan menjadi $A ++$ atau $A --$

Notasi " ++ " atau " -- " dapat diletakan didepan atau di belakang variabel

Contoh : $A ++$ atau $++A$ / $A --$ atau $--A$

Kedua bentuk penulisan notasi ini mempunyai arti yang berbeda.

- ▢ **Jika diletakan didepan variabel**, maka proses penambahan atau pengurangan akan dilakukan sesaat sebelum atau langsung pada saat menjumpai ekspresi ini, sehingga nilai variabel tadi akan langsung berubah begitu ekspresi ini ditemukan, *sedangkan*
- ▢ **Jika diletakan dibelakang variabel**, maka proses penambahan atau pengurangan akan dilakukan setelah ekspresi ini dijumpai atau nilai variabel akan tetap pada saat ekspresi ini ditemukan.

5. Komentar Program

Komentar program hanya diperlukan untuk memudahkan pembacaan dan pemahaman suatu program (untuk keperluan dokumentasi program). Dengan kata lain, komentar program hanya merupakan keterangan atau penjelasan program. Untuk memberikan komentar atau penjelasan dalam bahasa C digunakan pembatas `/*` dan `*/` atau menggunakan tanda `//` untuk komentar yang hanya terdiri dari satu baris. Komentar program tidak akan ikut diproses dalam program (akan diabaikan).

BAB III

STRUKTUR KONTROL PERCABANGAN

A. Tujuan Pembelajaran

Setelah mengikuti kegiatan belajar 3 ini siswa diharapkan dapat :

1. Memahami Percabangan 1 kondisi
2. Memahami Percabangan 2 kondisi
3. Memahami Percabangan lebih dari 2 kondisi
4. Memahami Percabangan bersarang

B. Uraian Materi

Penyeleksian kondisi digunakan untuk mengarahkan perjalanan suatu proses. Penyeleksian kondisi dapat diibaratkan sebagai katup atau kran yang mengatur jalannya air. Bila katup terbuka maka air akan mengalir dan sebaliknya bila katup tertutup air tidak akan mengalir atau akan mengalir melalui tempat lain. Fungsi penyeleksian kondisi penting artinya dalam penyusunan bahasa C, terutama untuk program yang kompleks.

1. Percabangan 1 kondisi (Tunggal) atau Struktur Kondisi IF....

Struktur if dibentuk dari pernyataan if dan sering digunakan untuk menyeleksi suatu kondisitunggal. Bila proses yang diseleksi terpenuhi atau bernilai benar, maka pernyataan yang adadi dalam blok if akan diproses dan dikerjakan.

Bentuk umum:

```
if(kondisi)  
    pernyataan;
```

1. Percabangan Ganda

Percabangan ganda apabila terdapat 2 alternatif instruksi yang dijalankan. Logika ini memungkinkan kompiler menjalankan salah satu dari 2 alternatif instruksi yang ada, dan salah satu instruksi pasti dijalankan.

Notasi algoritmik yang digunakan :

```
    If (kondisi) {  
        Instruksi pertama  
    }else{  
        Instruksi kedua  
    }
```

2. Percabangan Lebih dari 2

Pada dasarnya hanya terdapat 2 jenis seleksi dalam struktur algoritma pemrograman, namun bukan berarti hanya bisa dikembangkan pada 2 jenis tersebut saja. Struktur seleksi dapat dikembangkan menjadi bentuk yang tidak terbatas dan dapat dikombinasikan kedalam bentuk perulangan selama notasi penulisannya tidak terdapat kesalahan.

Kemampuan logika seseorang dalam merancang program dan mengamati dari permasalahan yang ada menjadi bagian yang paling penting dalam melakukan pengembangan dari bentuk seleksi ini.

Dibawah ini akan diberikan contoh seleksi menggunakan kondisi lebih dari 2.

3. Percabangan Bersarang Struktur Case

Struktur case sebenarnya memiliki fungsi yang sama dengan struktur if yang telah kita pelajari diatas. Struktur case ini dapat meringkaskan alur logika yang terjadi apabila diaplikasikan pada alur seleksi yang memiliki lebih dari 2 kondisi.

Struktur logika seleksi menggunakan struktur case ini jauh lebih ringkas apabila diaplikasikan pada struktur seleksi yang memiliki kondisi lebih dari 2. Kompiler program akan menjalankan instruksi dari struktur case dan memeriksa setiap kondisi yang ada, apabila belum ada kondisi yang bernilai benar maka kompiler akan terus menjalankan instruksi dibawahnya sampai ditemukan kondisi yang bernilai benar. Namun apabila hingga kondisi terakhir diperiksa dan tidak ditemukan kondisi yang bernilai benar maka kondisi default yang akan dijalankan.

Perbedaan yang paling jelas antara struktur if dengan struktur case adalah :

- ▢ Struktur if dapat menerima kondisi yang berupa operasi logika. Sedangkan struktur case tidak.
- ▢ Struktur case lebih efektif apabila digunakan untuk logika seleksi lebih dari 2 kondisi.
- ▢ Struktur case dan struktur if dapat dikombinasikan kedalam satu bagian, dengan catatan tata cara penulisan notasi tidak terdapat kesalahan.
- ▢ Struktur case tidak dapat melakukan pengecekan terhadap tipe data string / kalimat.

BAB IV

ALGORITMA PERULANGAN

Struktur Perulangan

Struktur perulangan terdiri dari 2 bagian, yaitu :

1. Kondisi perulangan, yaitu ekspresi yang dilakukan sebelum perulangan dilakukan pertama kali.
2. Body atau tubuh perulangan, yaitu satu atau lebih instruksi yang diulang.

Selain itu biasanya di perulangan juga terdapat 2 hal dibawah ini, antara lain :

- Inisialisasi : aksi yang dilakukan sebelum perulangan dilakukan pertama kali.
- Terminasi : aksi yang dilakukan untuk membuat perulangan berakhir. Biasanya berupa sebuah kondisi.

Dalam setiap bahasa pemrograman pada umumnya biasanya terdapat 3 jenis perulangan, antara lain :

1. Struktur WHILE – DO
2. Struktur Do – WHILE / REPEAT – UNTIL
3. Struktur FOR.

Ketiga jenis diatas hanyalah sebuah metode dan pada implementasinya, notasi penulisannya (sintaks) sangat tergantung dari setiap bahasa pemrograman yang digunakan.

A. Struktur WHILE – DO

Ciri khas dari struktur ini adalah :

1. Dilakukan pengecekan di awal pada kondisi sebelum menjalankan instruksi di tubuh perulangan.
2. Ada kemungkinan tubuh perulangan tidak dijalankan sama sekali.
3. Setiap kali hendak melakukan perulangan berikutnya, selalu memeriksa kondisi perulangan. Apabila kondisi perulangan telah
4. memberikan nilai false / salah. Maka perulangan akan dihentikan.

Notasi algoritmiknya adalah :

```
while (KONDISI){  
    tubuh perulangan yang berisi instruksi untuk dijalankan.  
}
```

Dalam struktur perulangan ini, ada 2 hal yang harus diperhatikan untuk menghindari terjadinya kesalahan logika pada program.

1. Inisialisasi variabel awal.
Ini dimaksudkan agar ketika kompiler program melakukan pemeriksaan terhadap kondisi awal, ditemukan kondisi yang benar. Pada beberapa bahasa pemrograman tertentu, apabila sebuah variabel tidak diinisialisasikan maka nilainya bisa berupa random ataupun nol. (lihat contoh dibawah)

2. Manipulasi variabel awal.

Banyak terjadi kesalahan pada programmer ketika mereka membuat program perulangan, memanipulasi variabel kondisi sangat penting untuk menjaga program tetap sesuai dengan yang diinginkan. Ketika kita lupa memanipulasi variabel awal, ada kemungkinan program mengulang terus menerus (looping forever) karena kondisi yang diinginkan tercapat terus tanpa ada perubahan. (lihat contoh dbawah)

B. Struktur Do – WHILE / REPEAT – UNTIL

Struktur Do = WHILE / REPEAT – UNTIL hampir mirip dengan struktur WHILE – DO. Berikut adalah ciri khas dari struktur perulangan ini.

1. Tidak dilakukan pengecekan kondisi perulangan di awal eksekusi program.
2. Minimal perulangan yang terjadi di tubuh program sebanyak 1 kali (Kerena tidak ada pengecekan kondisi perulangan di awal).
3. Setiap kali hendak melakukan perulangan berikutnya, selalu memeriksa kondisi perulangan. Apabila kondisi perulangan telah memberikan nilai false / salah. Maka perulangan akan dihentikan.

Perbedaan paling mendasar sebenarnya terletak pada pengecekan kondisi perulangan, struktur ini melakukan pengecekan kondisi perulangan di akhir tubuh perulangan (bukan di awal seperti struktur WHILE – DO) sehingga mengakibatkan instruksi dijalankan minimal 1 kali.

Notasi algoritmiknya adalah :

```
do {   repeat
Tubuh perulangan   atau
}while (KONDISI); until KONDISI
```

Pada implemenitasnya notasi penulisan struktur perulangan ini juga bergantung pada bahasa pemrograman yang digunakan. Pembahasan ini menggunakan bahasa pemrograman Turbo C dan lebih ditekankan kepada konsep – konsep perulangannya.

Kapan menggunakan WHILE – DO atau Do – WHILE ?

Pemilihan antara kedua struktur ini sangat tergantung pada permasalahan yang dihadapi. Apabila sebuah program memerlukan instruksi dijalankan dahulu dan baru diperiksa kondisinya maka struktur DO – WHILE harus digunakan namun apabila sebuah program harus memeriksa kondisi perulangan terlebih dahulu dan baru menjalankan tubuh perulangan, maka kondisi WHILE – DO harus digunakan.

C. Struktur FOR

Struktur perulangan for ini digunakan untuk perulangan yang tidak perlu memeriksa kondisi apapun dan hanya melaksanakan perulangan sejumlah kali tertentu.

Struktur perulangan ini paling cocok untuk proses perulangan yang telah diketahui batas akhirnya, karena kompiler akan mengeksekusi lebih cepat daripada 2 jenis struktur perulangan diatas.

Notasi algoritmiknya :

```
For (variabel awal = nilai awal; kondisi ;  
faktor penaik){Tubuh perulangan  
}
```

Inti dari struktur perulangan ini adalah :

1. Lebih cocok untuk jenis perulangan yang memiliki batas akhir yang sudah jelas.
2. Pemeriksaan kondisi awal akan dilakukan di awal. Apabila kondisi terpenuhi, maka tubuh perulangan akan dilakukan. Apabila tidak, maka tubuh perulangan tidak akan pernah dilakukan.
3. Ada kemungkinan tubuh perulangan tidak dijalankan sama sekali.
4. Memiliki proses yang lebih cepat dibandingkan bentuk DO – WHILE atau WHLE – DO dalam proses perhitungan matematika.

Algoritma pengulangan (While, While bersarang, Repeat)

STATEMEN/PERYATAAN WHILE

Pernyataan while digunakan untuk perulangan yang banyaknya perulangan tidak diketahui. Pernyataan while mirip dengan pernyataan if yang melakukan pemeriksaan ekspresi boolean sebelum sebuah atau serangkaian pernyataan dilakukan.

Bentuk umum:

```
while kondisi do  
Statemen
```

Kondisi adalah ekspresi boolean. Jika ekspresi bernilai true statemen dijalankan dan iperiksa kembali, dan keluar dari perulangan jika bernilai false.

Contoh_While1:

```
Program deretangka_1;  
uses crt;  
var i:integer;  
Begin
```

```
    clrscr;  
    i:=1;  
    while i <= 10 do  
        begin writeln(i);  
            i:=i+1;  
        end;  
    readln;
```

End.

Hasil :

12345678910

Contoh_While2:

Program deretangka_2;

uses crt;

var i:integer;

Begin

 clrscr;

 i:=10;

 while i > 0 do

 begin

 writeln(i); i:=i-

 1;

 end;

 readln;

End.

Hasil :

10987654321

Contoh_While3:

Program jumlahinteger;

uses crt;

var i,batas,hasil:integer;

Begin

 clrscr;

 write('Masukkan integer positif :');readln(batas);

 hasil:=0;

 i:=0;

 while i < batas do

 begin

 i:=i+1;

 hasil:=hasil+1;

 end;

 write('Jumlah 1 sampai ',batas,'=');

 write(hasil);

 readln;

End.

While Bersarang**Contoh_While4:**

Program bintang2;

uses crt;

var baris, kolom, jumbaris:integer;

Begin

 clrscr;

 write('Jumlah baris : ');readln(jumbaris);

 baris:=1;

 while baris <= jumbaris do

 begin

```
        write('*' :jumbaris+1-baris);
        kolom:=2;
begin   while kolom <= (2*baris-1) do

        write('*');
end;    kolom:=kolom+1;

writeln;
baris:=baris+1;
end;
readln;
End.
```

