MODUL 1

Pengantar Algoritma dan Pemrograman

Kani, M.Kom.

PENDAHULUAN

Modul ini akan menyampaikan informasi dan menjelaskan pengertian algoritma, kelebihan dan kelemahan penggunaan algoritma, mengklasifikasi algoritma, membuat algoritma, selain itu juga akan menjelaskan tentang *flowchart* serta kelebihan dan kelemahannya.

Secara khusus setelah mempelajari modul ini, mahasiswa diharapkan mampu:

1. mendefinisikan algoritma;
2. menyebutkan alasan menggunakan komputer;
3. mendefinisikan program komputer;
4. mendefinisikan dan menyebutkan kelompok besar program komputer;
5. mendefinisikan *programmer;*
6. mendefiniskan bahasa pemrograman;
7. membagi bahasa pemrograman berdasarkan fungsi;
8. mengekspresikan algoritma;
9. menyebutkan manfaat algoritma;
10. menyebutkan manfaat penggunaan algoritma;
11. menyusun algoritma;
12. Mengetahui dan mengklasifikasi algoritma-algoritma umum yang ada.

KEGIATAN BELAJAR 1

Pengantar Algoritma dan Pemrograman Komputer

Berkembangnya ilmu pengetahuan membuat manusia mampu menghasilkan karya yang semakin canggih. Dengan kemajuan teknologi, mereka selalu mencari formulasi yang tepat sehingga semakin algoritma optimal, sistematis dan logis. Saat ini manusia telah mampu menciptakan komputer canggih dengan kecerdasan buatan (*Artificial Intelegence*).

Komputer tidak otomatis mampu atau bisa menyelesaikan masalah begitu saja. Ada urutan langkah yang dimulai dari analisa masalah, penyusunan langkah penyelesaian (algoritma) oleh analis, kemudian diterjemahkan/menuliskan langkah-langkah ke dalam bahasa program komputer tertentu oleh programmer, kemudian dikompilasi dan dijalankan sehingga menemukan output yang diinginkan. Dalam dunia pemrograman, desain program yang tepat, bahkan sampai pada pemberian komentar pada setiap proses membantu *programmer* lain untuk menjaga dan memperbaiki fungsi program di masa depan.

A. PENGERTIAN ALGORITMA

Perpaduan antara perkembangan teknologi mikroelektronika dan desain *chip* memberikan sumbangsih berupa perangkat keras yang lebih cepat dan tepat guna. Penggunaan algoritma efisien (teroptimasi) dan disertai berbagai macam ilmu pengetahuan yang tercakup dalam suatu kecerdasan buatan telah memberikan efek revolusi komputer dan aplikasinya.

Efek revolusi komputer dan lonjakan kemajuan teknologi menjadi titik awal dimulainya beberapa pekerjaan manusia sudah mulai diambil alih oleh mesin. Banyak permasalahan yang bisa terselesaikan dengan teknologi terbaru saat ini. Dunia pendidikan tidak perlu memberikan khayalan kepada murid jika ingin mengambil objek beruang kutub, seorang dokter kandungan tidak perlu menebak jenis kelamin anak yang dikandung oleh pasiennya, seorang arsitek tidak perlu menggambar melalui kertas lagi, pembeli tak perlu datang ke supermarket untuk sekedar membeli kebutuhan rumah tangga dan banyak lagi hal yang lainnya. Semua bisa diselesaikan dengan teknologi komputer dengan

layanan-layanan aplikasi yang dibuat oleh pengembang. Layanan-layanan aplikasi seperti ini tentu diawali dengan kegiatan pembuatan desain hingga menjadi suatu algoritma yang kemudian direalisasikan dalam bentuk program komputer.

Definisi algoritma menurut beberapa pakar adalah sebagai berikut:

1. *Abu Ja'far Muhammad Ibnu Musa Al-Khawarizmi*: Algoritma adalah suatu metode khusus untuk menyelesaikan suatu masalah.
2. *Donald E. Knuth*: Algoritma adalah sekumpulan aturan-aturan berhingga yang memberikan sederetan proses-proses untuk menyelesaikan suatu masalah yang khusus.
3. *David Bolton*: Algoritma adalah deskripsi dari suatu prosedur yang berakhir dengan sebuah *output*.
4. *Stone dan Knuth*: Algoritma adalah suatu seperangkat aturan yang tepat mendefinisikan urutan operasi hingga sedemikian rupa sehingga setiap aturan yang efektif, jelas hingga sedemikian rupa sehingga urutan berakhir dalam waktu yang terbatas.
5. *Andrey Andreyevich Markov*: Algoritma adalah hal umum untuk dipahami sebagai suatu keputusan yang tepat untuk mendefinisikan proses komputasi yang mengarahkan dari data awal hingga hasil yang diinginkan.

Jika dikolaborasi dari definisi algoritma di atas maka definisi dari algoritma adalah suatu upaya dengan urutan operasi yang disusun secara logis dan sistematis untuk menyelesaikan suatu masalah untuk menghasilkan suatu *output* tertentu.

Dalam komputasi, algoritma sangat penting karena berfungsi sebagai prosedur sistematis yang diperlukan komputer. Algoritma yang baik adalah bagaikan menggunakan alat yang tepat di bengkel. Penggunaan algoritma yang salah adalah bagaikan mencoba memotong sepotong kayu dengan menggunakan gunting yang tentu tidak efektif. Penggunaan gunting tersebut juga akan membutuhkan waktu yang lama dalam menyelesaikan permasalahan.

Untuk memperluas pemahaman kita tentang konsep algoritma dengan cara yang lebih baik, kita cermati kasus berikut: Katakanlah Anda mau melakukan sebuah perjalanan dari Jakarta ke Bogor, Anda diperhadapkan dengan berbagai pilihan untuk mencapai tujuan, misalnya dengan naik kereta, naik taksi atau naik angkot. Ketiga pilihan yang ada mempunyai kelebihan dan kelemahan baik dari sisi waktu dan biaya.

Naik Kereta
- ✓ Ke stasiun terdekat
- ✓ Membeli tiket kartu Jakarta - Bogor
- ✓ Gesek kartu untuk membuka pintu masuk
- ✓ Menunggu keberangkatan kereta
- ✓ Jika kursi penuh maka berdiri
- ✓ Sampai tujuan Bogor

Naik Taksi
- ✓ Memesan taksi lewat Android atau telepon langsung
- ✓ Menunggu taksi untuk menjemput
- ✓ Naik taksi
- ✓ Lewat jalur biasa atau lewat jalan tol
- ✓ Sampai tujuan
- ✓ Bayar taksi sesuai argo

Naik Angkot
- ✓ Ke terminal atau pangkalan angkot
- ✓ Menunggu angkot penuh
- ✓ Melalui jalur biasa
- ✓ Sampai Bogor
- ✓ Bayar sesuai tarif angkot.

Ketiga algoritma di atas mencapai tujuan yang sama, namun masing-masing algoritma melakukannya dengan cara yang berbeda. Setiap algoritma juga memiliki biaya yang berbeda dan waktu perjalanan yang berbeda. Dengan naik taksi misalnya, adalah cara tercepat, tapi juga yang paling mahal. Naik angkot, jauh lebih murah akan tetapi memakan waktu lebih banyak dari naik taksi, begitu juga dengan naik kereta, menunggu jadwal kereta, singgah di setiap stasiun dan walaupun bebas hambatan. Setiap pilihan punya kelebihan dan kekurangan.

B. ALASAN MENGGUNAKAN ALGORITMA

Istilah algoritma digunakan dalam ilmu komputer atau informatika untuk mendeskripsikan metode pemecahan masalah yang terbatas, deterministik, dan efektif sesuai tujuan implementasi suatu program komputer. Algoritma

merupakan salah satu bidang/bagian dari ilmu komputer dan objek penelitian utama di lapangan sampai saat ini. Algoritma adalah prosedur pemecahan masalah dalam bahasa sangat alami (bahasa manusia), langkah-langkah pemecahan masalah dalam algoritma nantinya akan dituangkan menjadi program komputer untuk mempercepat/mengotomasi penyelesaian masalah. Sedangkan untuk bahasa pemrograman yang digunakan untuk menuangkan algoritma kedalam bahasa program sangat tergantung selera dan penguasaan pada individu *programmer*.

Salah satu alasan utama mempelajari algoritma dilihat dari kacamata disiplin ilmu adalah bahwa algoritma adalah sebuah keterampilan yang memberi potensi untuk memecahkan masalah serumit apapun dengan waktu peneyelesaian proses/eksekusi singkat, bahkan mungkin bisa meringkas langkah kerja yag tidak efisien menjadi otomatis. Dalam sebuah aplikasi bisa saja memproses jutaan objek, fungsi atau *procedure* (langkah-langkah kerja) penyelesaian masalah. Program yang dirancang dengan menggunakan algoritma yang tepat sangat mungkin membuat program jutaan kali lebih cepat dibanding dengan program dengan sebuah algoritma dengan desain asal-asalan. Pada modul-modul selanjutnya akan diberikan beberapa contoh kecil efisiensi pengerjaan suatu masalah yang bisa diselesaikan secara dengan logika algoritma yang tepat. Algoritma yang tepat akan memberikan efek yang signifikan terhadap waktu dan tenaga. Tidak sedikit perusahaan yang saat ini mau dan rela menginvestasikan uang tambahan untuk membeli dan memasang komputer baru untuk mempercepat pekerjaan mereka, tidak hanya perangkat keras, mereka mau menggunakan/membeli (membayar lisensi) sebuah algoritma untuk kepentingan perusahaan.

Dilihat dari disiplin ilmu maka berikut alasan mengadopsi atau menggunakan algoritma sebagai berikut:

1. *Efisiensi*: Untuk mengukur sebuah algoritma yang efisien harus mempertimbangkan efisiensi waktu-CPU dan memori. Terkadang *programmer* hanya berhenti kepada hasil tepat, akan tetapi tidak mempertimbangkan waktu dan memori yang terkuras oleh algoritma yang digunakan. Basis untuk membuat algoritma adalah efisiensi waktu, memori dan keluaran yang tepat. Walaupun tidak bisa dielakkan bahwa bahwa setiap orang akan memiliki cara berpikir/logika dalam menyelesaikan masalah yang berbeda-beda walau menghasilkan solusi

yang sama. Dalam algoritma kecepatan dan ruang memori harus mampu diseimbangkan untuk menghasilkan solusi cepat dan tepat.

2. *Abtraksi*: Kelebihan dari pada algoritma adalah mampu memperlihatkan sebuah permasalahan yang tingkat kerumitannya besar lalu kemudian dapat diurai menjadi kelihatan mudah dan sederhana, gambaran kerumitan terkikis dengan alur algoritma yang tersusun baik dan jelas.
3. *Reusability*: Algoritma adalah metode bukan program, artinya bahwa algoritma harus mampu digunakan tanpa melihat bahasa pemrograman yang digunakan, dapat digunakan kembali dan dan bahkan berkali-kali pada pada berbagai situasi untuk menerapkan dalam bahasa pemrograman apapun.

C. PENGERTIAN PROGRAM DAN BAHASA PEMROGRAMAN

Program dan Bahasa Pemrograman adalah sebuah istilah yang tidak bisa dipisahkan. Program adalah set intruksi dan bahasa pemrograman adalah intruksi standar dalam membuat program.

1. Program

Program adalah satu set intruksi yang berkode yang dapat dimengerti oleh komputer untuk memecahkan masalah atau menghasilkan hasil yang diinginkan. Terdapat dua macam kelompok besar program komputer, yaitu:

a. Sistem Operasi Komputer (*Computer Operating System* atau lebih di kenal dengan *OS*), yakni program komputer yang menyediakan intruksi paling mendasar yang digunakan komputer dalam operasinya. OS merupakan perangkat lunak sistem yang mengelola perangkat keras komputer, sumber daya perangkat lunak, dan menyediakan layanan umum untuk program komputer lainnya. Contoh Sistem Operasi *Windows*, *Linux*, *MacOS*.
b. Program Aplikasi, yang berjalan pada sistem operasi dan melakukan pekerjaan sesuai tujuan kehendak kita misal pengolah kata, perhitungan (olah data), presentasi video, suara dan sebagainya. Suatu program umumnya ditulis dengan menggunakan suatu bahasa pemrograman tingkat tinggi, seperti: Java, C/C++, Python, PHP dan sebagainya. Pada awal kehadirannya program komputer dibuat dengan bahasa tingkat rendah, seperti: Bahasa *Assembly* atau bahasa mesin. Kehadiran generasi bahasa pemrograman tingkat tinggi menjadikan bahasa pemrograman

tingkat rendah menjadi kurang diminati, penyebab utamanya adalah bahasa pemrograman tingkat tinggi lebih menyerupai bahasa manusia.

2. Bahasa Pemrograman Pertama

Bahasa Pemrograman pertama di dunia sudah berusia lebih dari 100 tahun. Ditulis oleh seorang wanita bernama Augusta Ada Byron yang lahir pada 10 Desember 1815. Dia adalah putri penyair ternama, Lord Byron. Lima pekan setelah Ada Lahir, ibunya minta cerai dari ayahnya dan hak asuh jatuh kepada Lady Byron. Alasan kuat perceraian kedua orang tuanya adalah karena ibunya tidak ingin anaknya kelak mengikuti jejaknya ayahnya menjadi seorang penyair. Berbagai upaya dilakukan oleh ibunya untuk mengalihkan anaknya dari dunia syair, diantaranya adalah dengan mengajaknya dan mengenalkannya kepada banyak ilmuwan dan ahli matematik, walaupun pada kenyataannya bahwa darah ayahnya tidak benar-benar hilang dari dirinya. Terbukti pada usia 30 tahun, dia mencampur adukkan ilmu matematika dengan imajinasi, dan dituangkan dalam bentuk tulisan-tulisan metafora.

Pada usia 17 tahun, Ada diperkenalkan ke Mary Somerville, seorang wanita luar biasa yang menerjemahkan karya LaPlace ke dalam bahasa Inggris. Mrs. Somerville memotivasi Ada untuk memperkuat ilmu matematikanya, mencoba membentuk pola pikir Ada yang menyatukan ilmu matematika dan teknologi yang layak digunakan oleh manusia. Pada suatu pesta makan malam pada bulan November 1834, Mrs. Somerville mendengar ide Babbage untuk membuat mesin penghitung baru yaitu mesin analitikal, dengan kemampuan bukan hanya bisa memprediksi akan tetapi mesin tersebut bisa memberikan misi futuristik dan hal ini memberi titik awal menemukan bahasa programnya yang pertama.

Babbage mengerjakan rencana mesin baru ini dan menseminarkan perkembangannya di Turin, Italia pada musim gugur tahun 1871. Seorang ahli matematik Italia bernama Menabrea, menulis sebuah rangkuman dari apa yang Babbage uraikan dan menerbitkan sebuah artikel berbahasa Perancis tentang perkembangan mesin analitikal tersebut. Ada menerjemahkan dan mengurai tulisan Menabrea tersebut, kemudian memperlihatkan hasilnya kepada Babbage. Babbage merasa kagum dengan tulisan Ada yang 3 kali lebih panjang dari tulisan Menabrea. Dari hasil tulisan Ada, Babbage menyarankan untuk menjadikan artikelnya sendiri dan kemudian mempublikasikan secara luas. Catatan ilmiah yang penuh fakta dan fantasi tersebut yang bakal menjadi penemuannya yang dituangkan kedalam mesin analitikal. Pada tahun 1843

presenter Lady Lovelace meramal bahwa kelak mesin semacam itu dapat digunakan untuk menyusun dan memainkan alat musik, dapat menghasilkan gambar ilmiah dan yang praktis digunakan. Hal tersebut terbukti tersedia pada saat ini.

Seiring berjalannya waktu, beberapa ilmuwan matematik menyarankan kepada Babbage dan Ada agar menyusun rencana agar mesinnya bisa menghitung angka Bernoulli. Dan rencana inilah, yang sekarang dianggap sebagai "Program Komputer" pertama, Babbage sebagai pembuat perangkat kerasnya dan Ada yang membuat dan menambahkan perangkat lunaknya. Pada tahun 1978, Departemen Pertahanan A.S. mengembangkan software pertamanya dan diberi nama "ADA" untuk menghormati Ada Byron.

3. *Programmer* dan Bahasa Pemrograman

Programmer adalah orang yang secara profesional bertanggung jawab atas perangkat lunak. Pada umumnya masyarakat berfikir bahwa semua *programmer* itu pada dasarnya melakukan hal dan pekerjaan yang sama, tapi sesungguhnya tidak demikian. Profesi *programmer* banyak memiliki spesialisasi yang terdefinisi dengan jelas, bahkan banyak bidang yang begitu berbeda satu dengan yang lain, sehingga tidak serupa sama sekali (kecuali persamaan orientasi, yaitu pada komputer).

Kedua kelompok terbesar adalah pemrograman sistem dan pemrograman aplikasi. Pemrograman sistem adalah aktivitasnya khusus pada pemrograman perangkat lunak sistem komputer (sistem operasi) dan jenis-jenis *system control program*. Pemrograman aplikasi adalah bertujuan menghasilkan perangkat lunak yang menyediakan layanan kepada pengguna secara langsung yang berjalan di atas sistem operasi.

Bahasa Pemrograman (*Programmming Language*) adalah bahasa formal yang terdiri set intruksi untuk komputer yang menghasilkan keluaran. Bahasa Pemrograman digunakan dalam pemrograman komputer untuk mengimplementasikan algoritma. Bahasa pemrograman memungkinkan seseorang *programmer* dapat menentukan secara presisi data mana yang akan diolah oleh komputer. Hanya ada satu bahasa pemrograman yang benar-benar dapat dipahami dan dijalankan oleh komputer apapun, yaitu kode biner aslinya '0' dan '1'.

Klasifikasi bahasa pemrograman tidaklah baku. Secara umum terdapat 3 (tiga) klasifikasi bahasa pemrograman, yaitu klasifikasi bahasa pemrograman tingkat rendah, tingkat menengah, dan tingkat tinggi.

a. Bahasa pemrograman tingkat rendah atau biasa disebut dengan bahasa mesin, satu-satunya bahasa yang langsung diolah tanpa kompilasi terlebih dahulu. Bahasa pemrograman ini ditulis dengan kode-kode mesin.
b. Bahasa pemrograman tingkat menengah atau biasa disebut dengan bahasa rakitan (*Assembly*), yaitu memberikan perintah untuk komputer dengan memakai kode-kode singkat (kode mnemonic), contohnya kode mesin: MOV, SUB, CMP, JMP, JGE, JL, LOOP. Contoh bahasa pemrograman ini adalah *Assembler*, *Microsoft Macro Assembler* (MASM).
c. Bahasa pemrograman tingkat tinggi di awali kemunculannya pada pemrograman generasi ke-3 dan hingga generasi ke-5. Perkembangan bahasa pemrograman dari generasi ke generasi mengalami kemajuan pesat ditandai dengan bahasa sudah lebih banyak menggunakan *keyword* bahasa manusia, pemrograman berorientasi obyek, pemrograman berbasis web bahkan dengan sistem *cloud*, pemrograman berasis data dan bahkan yang lebih maju lagi adalah pemrograman *mobile*. Contoh bahasa pemrograman tingkat tinggi adalah Visual Basic, Delphi, Pascal, PHP, dan Java.

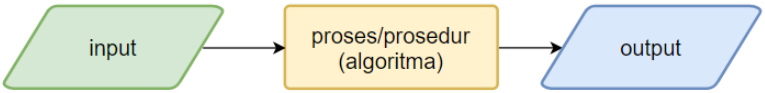
D. PENYELESAIAN MASALAH DENGAN PROGRAM

Tiga konsep penting dalam penyelesaian masalah, yaitu: (1) Menganalisa masalah dan membuat algoritma. (2) Menuangkan algoritma ke dalam bentuk program. (3) Mengeksekusi dan menguji program.

Penjelasan konsep di atas adalah sebagai berikut :

1. Menganalisa masalah dan membuat algoritma

Hal yang pertama yang harus dilakukan untuk memecahkan permasalahan adalah menganalisa dan mengidentifikasi suatu permasalahan, mengidentifikasi data yang menjadi masukan/keluaran, kemudian membuat proses yang mengolah semua data yang masuk menjadi suatu keluaran yang diinginkan. Semua proses harus berisi intruksi yang jelas, urut dan runtut sampai permasalahan bisa diurai. Apabila permasalahan tersebut cukup kompleks dan terdiri dari beberapa proses; maka kegiatan tersebut dapat dipecah menjadi beberapa sub-proses. Urutan penyelesaian suatu masalah secara runtut ini dikenal sebagai algoritma.



Gambar 1.1
Proses dalam Pemecahan Masalah (Algoritma)

2. Menuangkan algoritma dalam bentuk program

Konsep penyelesaian masalah, dalam bentuk urutan pemecahan masalah (algoritma) yang telah didesain, harus dapat dituangkan ke dalam bahasa program atau bahasa pemrograman oleh *programmer* dengan bahasa pemrograman yang dikuasainya.

3. Mengeksekusi dan menguji program

Program yang telah dibuat harus bisa di kompilasi menjadi suatu aplikasi untuk dapat di uji kebenarannya, dan apabila ditemukan bahwa telah terjadi kesalahan, maka proram tersebut harus diperbaiki sebelum diserahkan kepada pemakai.

E. EKSPRESI ALGORITMA

Algoritma dapat diekspresikan dalam banyak notasi berbeda-beda, termasuk bahasa alami, *flowchart*, *pseudocode*, atau diagram alur, dan bahasa pemrograman. Ekspresi algoritma dengan bahasa alami cenderung bertele-tele dan ambigu, dan jarang digunakan untuk algoritma kompleks. Teknik/cara *flowchart* dan *pseudocode* merupakan cara terstruktur untuk mengekspresikan algoritma untuk menghindari ambiguitas pernyataan bahasa alami, dan tetap independen dari bahasa implementasi tertentu (tidak terikat dengan bahasa pemrograman tertentu).

Bahasa pemrograman digunakan untuk mengekspresikan algoritma dalam bentuk kode program yang dapat dijalankan komputer. Jika suatu komunitas pengembangan sistem informasi hanya menggunakan satu bahasa pemrograman, algoritma yang dibuatnya boleh dibuat mendekati bahasa pemrograman yang digunakan.

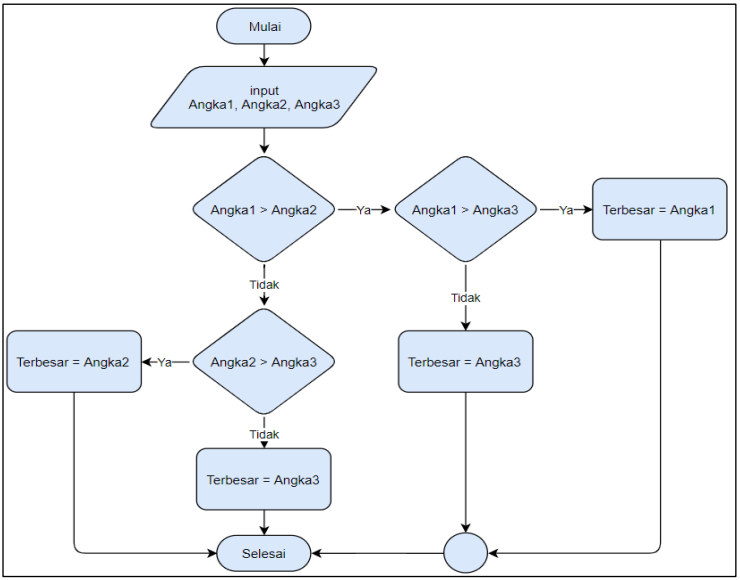
Algoritma yang digunakan dalam memecahkan masalah sangat bisa beragam karena sangat tergantung tingkat pemahaman dan logika pembuat algoritma. Perlu ditekankan bahwa algoritma yang baik adalah algoritma yang tidak banyak menggunakan sumber daya dan waktu.

Anda bisa membandingkan 2 contoh algoritma di bawah ini, yang keduanya mencari nilai terbesar pada 3 angka, yang satu dengan bahasa alami dan yang lainnya adalah menggunakan *pseudocode*.

Algoritma dengan bahasa alami:

1. Mulai
2. Masukkan *angka1*, *angka2*, dan *angka3*
3. Asumsikan terbesar adalah *angka1* untuk awal.
4. Jika *angka1* lebih besar dari terbesar maka terbesar adalah *angka1*
5. Jika tidak, apakah *angka2* lebih besar dari terbesar, Jika ya maka terbesar sama dengan *angka2*;
6. Jika tidak maka terbesar sama dengan *angka3*.
8. Selesai

Algoritma dengan *flowchart*:



Gambar 1.2
Contoh Proses Dalam Pemecahan Masalah Algoritma (*Flowchart*)

Algoritma dengan *pseudocode*:

Pseudocode Angka Terbesar (dalam bahasa Indonesia)	
1	Mulai
2	Masukkan angka1
3	Masukkan angka2
4	Masukkan angka3
5	terbesar ← angka1
6	Jika angka1 > terbesar, maka
7	terbesar ← angka1
9	Jika angka2 > terbesar
10	terbesar ← angka2
11	Tapi jika tidak, maka
12	terbesar ← angka3
13	Angka terbesar ditemukan adalah terbesar
16	Selesai

Jika dibandingkan antara algoritma bahasa alami dengan *pseudocode*, maka lebih mudah memahami algoritma dengan *pseudocode*. Tapi untuk lebih praktisnya adalah dengan *pseudocode* karena dapat mengatur keterkaitan antar bloknya, serta mudah memahaminya. Contoh penggalan *pseudocode* berikut:

```
1 ...
2 Jika angka1 > terbesar, maka
3   terbesar <- angka1
4 Jika angka2 > terbesar, maka
5 ...
```

Baris yang menjorok kedalam *terbesar <- angka1* (baris 3) memberikan informasi bahwa baris ini akan dikerjakan jika memenuhi syarat pada baris di atasnya (baris 2), dan jika tidak memenuhi, maka akan melakukan perbandingan/menguji pada baris di bawahnya (baris 4), begitu seterusnya hingga perbandingan angka3. Jika pada umumnya bahasa pemrograman menggunakan Bahasa Inggris untuk *keyword*, maka dapat ditransformasikan sebagai berikut:

PSMDL-2 : Pseudocode Angka Terbesar (dalam bahasa Inggris)

```
1   Mulai
2   Input angka1
3   Input angka2
4   Input angka3
5   terbesar ← angka1
6   if angka1 > terbesar then
7      terbesar ← angka1
9   else if angka2 > terbesar then
10     terbesar ← angka2
11  Else
12     terbesar ← angka3
13  Print "Angka terbesar :" +Terbesar
16  Selesai
```

Harus diakui bahwa pendekatan dengan *keyword* berbahasa Inggris efektif jika acuan bahwa bahasa pemrograman pada umumnya menggunakan Bahasa Inggris. Jika algoritma di atas ditransformasi atau pendekatan kedalam bahasa pemrograman tertentu, misalnya bahasa pemrograman Java, maka hasilnya sebagai berikut:

ProgramMDL-3: Mencari Angka/Nilai Terbesar (Java)

```
1   //Nama Kelas (Program)
2    public class AngkaTerbesar {
3
4      public static void main(String[] arg) {
5        int angka1 = 10;
6        int angka2 = 15;
7        int angka3 = 5;
8        int terbesar = angka1;
9        if (angka1 > terbesar)
10          terbesar = angka1;
```

```
11      else if (angka2 > terbesar)
12         terbesar = angka2;
13      else
14         terbesar = angka3;
15
        System.out.println("Bilangan terbesar adalah = " +
terbesar);
       }
      }
```

Mengapa hal ini bisa ditransformasi ke dalam bahasa pemrograman? Pada dasarnya analis dan *programmer* yang berpengalaman mempunyai naluri yang kuat membaca algoritma dengan pendekatan logika berpikir untuk menyelesaikan masalah.

F. CIRI-CIRI DAN MANFAAT MEMPELAJARI ALGORITMA

Ciri-ciri dari algoritma adalah sebagai berikut:

1. Ada *input*/masukan dan *output*/keluaran.
2. Memiliki proses tertentu.
3. Prosesnya merupakan pola pikir dan logis dalam menghasilkan *output*.
4. Prosesnya memiliki intruksi yang jelas dan tidak ambiguitas
5. Memiliki *stopping role* atau jika pada keadaan tertentu mengalami proses iterasi yang berlebihan maka ada proses pemberhentian.

Adapun manfaat belajar algoritma adalah sebagai berikut :

1. Meningkatkan kemampuan berfikir secara logis. Logika dan algoritma pemrograman menjadi suatu hal yang sangat penting dalam membuat atau mengembangkan sebuah produk. Kesalahan logika yang digunakan, tentu akan berakibat fatal terhadap produk yang akan dikembangkan, selain *error*, tentu produk yang dikembangkan tidak akan sesuai dengan apa yang kita inginkan walaupun *programmer* telah berhasil membuat dalam program.
2. Mengembangkan cara berfikir dengan sistemastis. Dalam membuat sebuah algoritma harus secara urut dan sistematis begitu juga dengan program hasil penerapan dari algoritma, seseorang akan dihadapkan pada

urutan-urutan yang disusun secara sistematis. Urutan-urutan harus terstruktur dan tidak boleh terbolak-balik baik penyusunannya maupun penulisannya, agar program yang dibangun dapat berjalan dengan baik dan benar.

3. Mempertajam analisis ketika pembuatan program. Ketika membuat algoritma maupun program terkadang muncul kesalahan-kesalahan dalam penyelesaiannya, misalnya program yang dibangun *error* saat diverifikasi atau di-*build*. Permasalahan ini akan memerlukan sedikit ketelitian untuk mengatasinya yaitu dengan melakukan pengecekan ulang kode program yang dibuat, pengecekan yang berulang-ulang akan membawa pada pelatihan menganalisis permasalahan dan meningkatkan ketelitian dalam membuat sebuah program dan membuahkan hasil program yang baik.
4. Meningkatkan kemampuan dalam mengatasi masalah. Tujuan utama algoritma adalah menyelesaikan masalah, jadi kita akan dilatih untuk menyelesaikan sebuah permasalahan, bahkan sampai pada memprediksi masalah yang akan muncul dan bagaimana mengelolanya. Secara tidak sadar, pola ini akan terbawa dalam kehidupan sehari-hari untuk menghadapi berbagai macam permasalahan yang terjadi. Kita secara tidak sadar akan berpikir secara logis dan sistematis.

G. PEDOMAN PENYUSUNAN ALGORITMA

Tidak ada pedoman baku tentang teknik / cara baku untuk pembuatan dan penyusunan algoritma, namun diberikan syarat keterpenuhan. Menurut Ellis Horowitz dan Sartaj Sahni dalam bukunya berjudul "*Fundamentals of Data Structures*", syarat ketercapaian suatu algoritma adalah apabila memenuhi syarat berikut:

1. *Input*: boleh nol atau lebih masukan dalam satu algoritma;
2. *Output*: dalam satu algoritma, dipersyaratkan memiliki satu keluaran, boleh lebih;
3. *Definiteness*: setiap intruksi harus jelas, tidak boleh ambigu (bermakna ganda atau lebih sehingga membingungkan);
4. *Finiteness*: menyatakan bahwa setelah melakukan proses maka apapun kondisinya suatu algoritma harus memiliki akhir;
5. *Effectiveness*: algoritma bekerja secara efektif, yaitu semua operasi yang dilakukan bersifat sederhana dan dapat diselesaikan dengan waktu yang singkat.

LATIHAN

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

1) Apa yang dimaksud dengan algoritma?
2) Apa yang dimaksud dengan program?
3) Sebutkan dan jelaskan dua macam kelompok besar program komputer.
4) Apa yang dimaksud dengan bahasa pemrograman dan *programmer*?
5) Sebutkan dan jelaskan bahasa pemrograman berdasarkan fungsi kerja pada mesin komputer.
6) Sebutkan dan jelaskan tiga konsep penyelesaian masalah dengan program komputer.
7) Sebutkan dan jelaskan tiga alasan menggunakan algoritma?
8) Sebutkan manfaat dari menggunakan algoritma.

Petunjuk Jawaban Latihan

Jawaban untuk latihan di atas dapat ditemukan pada teori yang diberikan pada Kegiatan Belajar 2 tentang Pengantar Algoritma dan Pemrograman Komputer. Jawaban dari soal-soal di atas adalah sebagai berikut:

1) Algoritma adalah prosedur pemecahan masalah dalam bahasa alami manusia yang tidak tergantung kepada bahasa pemroraman tertentu.
2) Program adalah satu set intruksi yang berkode yang dapat dimengerti oleh komputer untuk memecahkan masalah atau menghasilkan hasil yang diinginkan.
3) Kelompok besar program komputer adalah:
   a. Sistem Operasi Komputer atau *Operating System* (OS): Program komputer yang menyediakan intruksi paling mendasar yang digunakan komputer dalam operasinya, OS merupakan perangkat lunak yang bisa mengelola perangkat keras, sumber daya perangkat lunak dan penyedia layanan untuk komputer lainnya. Contoh: Sistem Operasi Windows, Linux, MacOS.
   b. Program Aplikasi: Aplikasi yang berjalan di atas sistem operasi dan melakukan pekerjaan sesuai tujuan kehendak kita, misalnya pengolah

kata, perhitungan, aplikasi pemutar video, aplikasi penampil gambar, aplikasi grafis.

4) Bahasa Pemrograman (*Programmming Language*) adalah bahasa formal yang terdiri set intruksi untuk komputer yang menghasilkan keluaran. Bahasa Pemrograman digunakan dalam pemrograman komputer untuk mengimplementasikan algoritma. *Programmer* adalah orang yang profesional bertanggung jawab atas perangkat lunak. Profesi ini banyak memiliki spesialisasi yang terdefinisi dengan jelas, bahkan banyak bidang yang begitu berbeda satu dengan yang lain, sehingga tidak serupa sama sekali (kecuali persamaan orientasi, yaitu pada komputer).
5) Berdasarkan fungsi kerja pada mesin komputer bahasa pemrograman terdiri dari:
   a. Bahasa Mesin: Instruksi komputer dengan memakai kode bahasa biner 0 dan 1.
   b. Bahasa Tingkat Rendah: Bahasa Rakitan (*Assembly*), memberikan intruksi-intruksi singkat diluar bahasa manusia (bahasa mesin).
   c. Bahasa Tingkat Menengah: Bahasa komputer yang memadukan bahasa manusia dengan bahasa simbolik.
   d. Bahasa Tingkat Tinggi: Bahasa komputer dengan perintah bahasa manusia (bahasa Inggris).
6) Tiga konsep penyelesaian dengan program komputer yaitu:
   a. Menganalisa (analisa) dan membuat algoritma: Tahapan analisa adalah mengenali, mengidentifikasi suatu masalah, mengidentifikasi adalah seberapa besar masalah yang ingin dipecahkan, jika permasalahan cukup besar, maka bisa dipecah menjadi sub-sub proses.
   b. Menuangkan Algoritma kedalam bentuk program: Algoritma yang dibuat harus jelas prosesnya. Urutan antara proses atau sub proses harus sesuai, sehingga *programmer* dengan mudah menuangkan kedalam bentuk program.
   c. Mengeksekusi dan Menguji Program: Program komputer yang telah dibuat harus di eksekusi dan diuji. Eksekusi adalah mengkompilasi kode-kode program yang sudah dibuat . Uji adalah menguji program apakah sudah sesuai dengan yang dinginkan atau sudah tercapai pemecahan yang dinginkan.

7) Tiga alasan kenapa menggunakan alasan menggunakan algoritma, sebagai berikut:
   a. Efisiensi: Untuk mengukur sebuah algoritma yang efisien harus mempertimbangkan yaitu efisiensi waktu-CPU dan memori. Terkadang *programmer* hanya berhenti kepada hasil tepat, akan tetapi tidak mempertimbangkan waktu dan memori yang terkuras oleh algoritma yang digunakan. Basis untuk membuat algoritma adalah efisiensi waktu, memori dan keluaran yang tepat.
   b. Abtraksi: Kelebihan dari pada algoritma adalah mampu memperlihatkan sebuah permasalahan yang tingkat kerumitannya besar dapat diurai menjadi kelihatan mudah dan sederhana, gambaran kerumitan terkikis dengan alur algoritma yang tersusun baik dan jelas dengan pendekatan-pendekatan umum.
   c. *Reusability*: Algoritma adalah metode, bukan program itu sendiri, artinya bahwa algoritma harus mampu digunakan tanpa melihat bahasa pemrograman yang digunakan, dapat digunakan kembali dan dan bahkan berkali-kali pada pada berbagai situasi untuk menerapkan dalam bahasa program.
8) Adapun manfaat dari menggunakan algoritma adalah sebagai berikut:
   a. Meningkatkan kemampuan berfikir secara logis, logika dan algoritma pemrograman menjadi suatu hal yang sangat penting dalam membuat atau mengembangkan sebuah produk. Kesalahan logika yang digunakan, tentu akan berakibat fatal terhadap produk yang akan dikembangkan. Selain *error*, tentu produk yang dikembangkan tidak akan sesuai dengan apa yang kita inginkan.
   b. Mengembangkan cara berfikir dengan sistemastis. Dalam membuat sebuah algoritma harus secara urut dan sistematis begitu juga dengan program hasil penerapan dari algoritma, seseorang akan dihadapkan pada urutan-uruan yang disusun secara sistematis. Urutan-urutan harus terstruktur dan tidak boleh terbolak-balik baik penyusunannya maupun penulisannya, agar program yang dibangun dapat berjalan tanpa permasalahan.
   c. Mempertajam analisis ketika pembuatan program, ketika membuat algoritma maupun program terkadang muncul kesalahan-kesalahan dalam penyelesaiannya, misalnya program yang dibangun *error* saat diverifikasi atau di-*build*. Permasalahan ini akan memerlukan sedikit ketelitian untuk mengatasinya yaitu dengan mengecek ulang kode

program yang dibuat, pengecekan yang berulang-ulang akan membawa pada pelatihan menganalisis permasalahan dan meningkatkan ketelitian dalam membuat sebuah program.

d. Meningkatkan kemampuan dalam mengatasi masalah: Tujuan utama algoritma adalah menyelesaikan masalah, jadi kita akan dilatih untuk menyelesaikan sebuah permasalahan, bahkan sampai pada memprediksi masalah yang akan muncul dan bagaimana mengelolanya. secara tidak sadar, pola ini akan terbawa dalam kehidupan sehari-hari untuk menghadapi berbagai macam permasalahan yang terjadi. Kita secara tidak sadar akan berpikir secara logis dan sistematis.

RANGKUMAN

Algoritma adalah suatu upaya dengan urutan operasi yang disusun secara logis dan sistematis untuk menyelesaikan suatu masalah untuk mengahsilkan suatu *output* tertentu.

Dilihat dari disiplin ilmu maka berikut alasan mengadopsi atau menggunakan algoritma sebagai berikut:

1. Efisiensi: Untuk mengukur sebuah algoritma yang efisien harus mempertimbangkan yaitu efisiensi waktu-CPU dan memori. Terkadang *programmer* hanya berhenti kepada hasil tepat, akan tetapi tidak mempertimbangkan waktu dan memori yang terkuras oleh algoritma yang digunakan. Basis untuk membuat algoritma adalah efisiensi waktu, memori dan keluaran yang tepat. Walaupun tidak bisa dielakkan bahwa bahwa setiap orang akan memiliki cara menyelesaikan masalah yang berbeda-beda walau menghasilkan solusi yang sama. Dalam algoritma kecepatan dan ruang memori harus mampu diseimbangkan untuk menghasilkan solusi cepat dan tepat.
2. Abtraksi: Kelebihan dari pada algoritma adalah mampu memperlihatkan sebuah permasalahan yang tingkat kerumitannya besar dapat diurai menjadi kelihatan mudah dan sederhana, gambaran kerumitan terkikis dengan alur algoritma yang tersusun baik dan jelas dengan pendekatan-pendekatan umum.
3. *Reusability*: Algoritma adalah metode, bukan program itu sendiri, artinya bahwa algoritma harus mampu digunakan tanpa melihat bahasa pemrograman yang digunakan, dapat digunakan kembali dan dan bahkan berkali-kali pada pada berbagai situasi untuk menerapkan dalam bahasa program.

Program adalah satu set intruksi yang berkode yang dapat dimengerti oleh komputer untuk memecahkan masalah atau menghasilkan hasil yang diinginkan. Dua jenis program komputer adalah (1) Sistem Operasi, yang menyediakan intruksi paling mendasar yang digunakan komputer dalam operasinya, misalkan: Sistem Operasi Windows, Linux, MacOS dan (2) Program Aplikasi, yang berjalan pada sistem operasi dan melakukan pekerjaan seperti pengolah kata, perhitungan (olah data).

Programmer adalah orang yang profesional bertanggung jawab atas perangkat lunak. Profesi ini banyak memiliki spesialisasi yang terdefinisi dengan jelas, bahkan banyak bidang yang begitu berbeda satu dengan yang lain, sehingga tidak serupa sama sekali (kecuali persamaan orientasi, yaitu pada komputer).

Tiga konsep penting dalam penyelesaian masalah, yaitu: (1) Menganalisa masalah dan membuat algoritma, (2) Menuangkan algoritma kedalam bentuk program, (3) Mengeksekusi dan menguji program. Dan tiga alasan kenapa algoritma penting digunakan dalam menyelesaikan masalah yaitu: efisiensi, abstraksi dan *reusability*.

Manfaat mempelajari algoritma adalah sebagai berikut:

- Meningkatkan pengambilan keputusan dan kemampuan berfikir secara logis.
- Mengembangkan cara berfikir dengan sistematis.
- Mempertajam analisis ketika pembuatan program.
- Meningkatkan kemampuan dalam mengatasi masalah.

Syarat ketercapaian suatu algoritma adalah apabila memenuhi syarat berikut:

1. *Input*: boleh nol atau lebih masukan dalam satu algoritma;
2. *Output*: dalam satu algoritma, dipersyaratkan memiliki satu keluaran, boleh lebih;
3. *Definiteness*: setiap intruksi harus jelas, tidak boleh ambigu (bermakna ganda atau lebih sehingga membingungkan);
4. *Finiteness*: menyatakan bahwa setelah melakukan proses maka apapun kondisinya suatu algoritma harus memiliki akhir;
5. *Effectiveness*: algoritma bekerja secara efektif, yaitu semua operasi yang dilakukan bersifat sederhana dan dapat diselesaikan dengan waktu yang singkat.

TES FORMATIF 1

Pilihlah satu jawaban yang paling tepat!

1) Suatu upaya dengan urutan operasi yang disusun secara logis dan sistematis untuk menyelesaikan suatu masalah untuk menghasilkan suatu output tertentu definisi dari....
   A. Logika pemrograman
   B. Algoritma
   C. Program komputer
   D. Logika informatika

2) Algoritma adalah suatu metode khusus untuk menyelesaikan suatu masalah, definisi ini menurut....
   A. Abu Ja'far Muhammad Ibnu Musa Al-Khawarizmi
   B. Donald E. Knuth
   C. David Bolton
   D. Andrey Andreyevich Markov

3) Algoritma adalah hal umum untuk dipahami sebagai suatu keputusan yang tepat untuk mendefinisikan proses komputasi yang mengarahkan dari data awal hingga hasil yang diinginkan, definisi ini menurut....
   A. Abu Ja'far Muhammad Ibnu Musa Al-Khawarizmi
   B. Donald E. Knuth
   C. David Bolton
   D. Andrey Andreyevich Markov

4) Pengembang tidak mempertimbangkan waktu dan memori yang terkuras oleh algoritma yang digunakan. Narasi di atas bertentangan dengan alasan adopsi penggunaan algoritma pada poin....
   A. Abtraksi
   B. Efisiensi
   C. *Reusability*
   D. Semua benar

5) Bahasa Pemrograman yang pertama di dunia adalah….
   A. Visual Basic
   B. ADA
   C. Java
   D. Delphi

6) Program yang berjalan pada sistem operasi dan melakukan pekerjaan sesuai tujuan kehendak kita misal pengolah kata, perhitungan (olah data), presentasi video, suara dan sebagainya. Suatu program umumnya ditulis dengan menggunakan suatu bahasa pemrograman tingkat tinggi. Kalimat di atas mewakili dari definisi….
   A. Program Sistem Operasi
   B. Bahasa pemrograman
   C. Program aplikasi
   D. Algoritma

7) Tiga alasan kenapa harus menggunakan algoritma, yaitu....
   A. Efisiensi, abstraksi, dan *reusability*
   B. Efektif, hemat waktu, minim biaya
   C. Fleksibel, efektif, dan normatif
   D. Fleksibel, abtraksi, dan normatif

8) Mempertajam analisis ketika pembuatan program, adalah bagian dari....
   A. Manfaat mempelajari algoritma
   B. Ciri sebuah algoritma
   C. Ekspresi algoritma
   D. Salah satu konsep penyelesaian masalah

9) Mengidentifikasi data yang menjadi masukan/keluaran, kemudian membuat proses yang mengolah semua data yang masuk menjadi suatu keluaran yang diinginkan. Semua proses harus berisi intruksi yang jelas, urut dan runtut sampai permasalahan bisa diurai, narasi di atas lebih tepat pada penyelesaian masalah pada bagian….
   A. Analisa dan membuat algoritma
   B. Menuamgkan algoritma dalam bentuk program
   C. Mengeksekusi dan menguji program
   D. Semua benar

10) *Effectiveness*, artinya....
   A. Boleh nol atau lebih masukan dalam satu algoritma;
      Output: dalam satu algoritma, dipersyaratkan memiliki satu keluaran, boleh lebih
   B. Setiap intruksi harus jelas, tidak boleh ambigu (bermakna ganda atau lebih sehingga membingungkan)
   C. Setelah melakukan proses maka apapun kondisinya suatu algoritma harus memiliki akhir
   D. algoritma bekerja secara efektif, yaitu semua operasi yang dilakukan bersifat sederhana dan dapat diselesaikan dengan waktu yang singkat

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 1 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 1.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali
80 - 89% = baik
70 - 79% = cukup
< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 2. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 1, terutama bagian yang belum dikuasai.

KEGIATAN BELAJAR 2

Struktur Dasar Algoritma

Ketika ingin membangun bangunan rumah hal yang paling pertama yang harus dipikirkan adalah kontruksi dasar, seperti pondasi, balok beton dan konstruksi dinding, hal sama juga dengan membangun sebuah algoritma harus tahu struktur dasarnya. Ada tiga struktur dasar dalam algoritma yaitu: skuesial (*squenstial*), seleksi (*selection*), dan perulangan (*looping*).

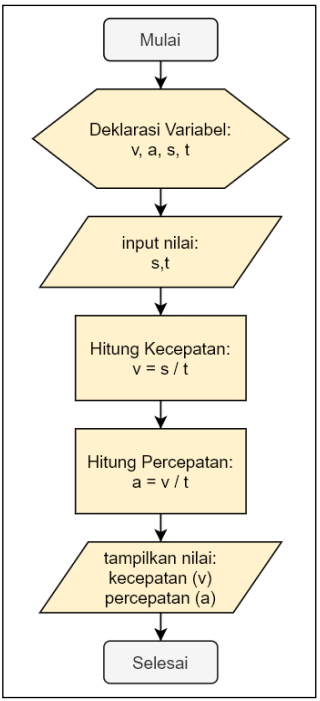
A. SKUENSIAL (*SQUENTIAL*)

Struktur dasar skuensial adalah sebuah algoritma dibangun dengan langkah-langkah (instruksi/perintah) dikerjakan secara berurutan, tidak boleh melompati satu langkah perintah pun. Misal dalam sebuah algoritma terdapat 20 langkah, maka semua langkah tersebut dikerjakan berurutan mulai dari langkah 1 sampai pada langkah 20 tanpa melewatkan satu langkah pun.

Cara kerja skuensial juga nantinya berlaku dalan bahasa pemrograman, ketika instruksi bahasa pemrograman yang kita tulis diproses oleh komputer, maka komputer akan memproses dan menterjemahkan baris demi baris intruksi-instruksi bahasa pemrograman tersebut secara beruntun dari awal hingga akhir dimulai dari instruksi pada baris awal hingga baris akhir.

Dengan struktur skuensial ini, pembuat algoritma harus mampu menganalisa dan menentukan intruksi mana yang harus ditulis lebih awal dan seterusnya dan yang mana harus paling akhir.

Pada Gambar 1.3 adalah contoh algoritma dengan skuensial untuk mencari kecepatan dan percepatan, tidak ada satupun proses yang terlewatkan atau melompati proses yang ada di bawahnya, semua dikerjakan secara berurutan. Demikian juga dengan intruksi yang ada di dalam prosesnya, misalnya rumus hitung *pecepatan (a)* tidak mendahului proses perhitungan *kecepatan (v)*. hal ini karena untuk mencari percepatan harus mencari atau menghitung kecepatan terlebih dahulu.



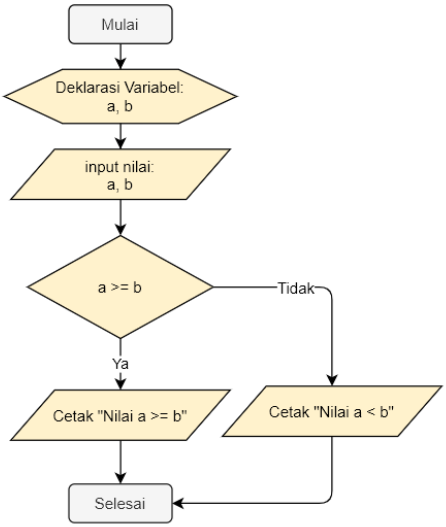
Gambar 1.3
Algoritma dengan Cara Skuensial Mencari Kecepatan dan Percepatan

Jika struktur dasar skuensial direprestasikan menjadi algoritma *pseudocode,* urutan pekerjaannya akan sama. Perhatikan *pseudocode* mencari kecepatan dan percepatan di bawah ini:

Pseudocode cari kecepatan dan percepatan	
1	`Mulai`
2	`Deklarasi variabel: v, a, s, t`
2	`Input/masukkan nilai: s, a`
3	`v ← s/t`
4	`a ← v/t`
5	`Tampilkan nilai kecepatan: v`
6	`Tampilkan nilai percepatan: a`
7	`Selesai`

B. SELEKSI (*SELECTION*)

Pada kenyataannya banyak algoritma setidaknya akan mengandung proses seleksi (*selection*) pada intruksi-intruksi pada tubuh algoritma, instruksi seleksi digunakan apabila menemukan/memiliki kasus dua atau lebih alternatif penyelesaian/keputusan.



Gambar 1.4
Flowchart dengan Struktrur Algoritma dengan Seleksi

Gambar 1.4 adalah contoh algoritma penyelesaian dengan seleksi dua keputusan. Ada dua angka yang akan diseleksi dengan membandingkan keduanya mana yang terbesar/sama dengan atau terkecil. Misal nilai $a = 6$ dan nilai $b = 5$, maka seleksinya adalah akan tercetak "*Nilai a >= b*" dan jika sebaliknya maka keputusan yang lain adalah tercetak "*Nilai a < b*".

Pseudocode menentukan nilai terbesar	
1	`Mulai`
2	`Deklarasi variabel: a, b`
3	`Input/masukkan nilai: a, b`
4	`if a >= b`

```
5    Cetak “Nilai a >= b”
6    ke langkah 9
7   else //lainnya
8    Cetak “Nilai a < b”
9   Selesai
```

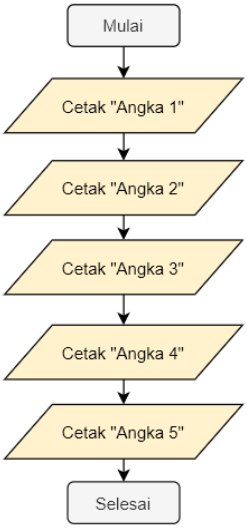
Dalam dunia algoritma dan pemrograman akan sering diperhadapkan kondisi seleksi seperti contoh di atas, baik tipe kasusnya sama maupun berbeda.

C. PERULANGAN (*LOOPING*)

Struktur algoritma yang ketiga adalah perulangan. Banyak hal terjadi di dunia ini berulang-ulang, kita bisa menghafal sesuatu salah satu cara adalah mengulang, materi yang diberikan dikelas oleh guru atau dosen mugkin tidak serta merta langsung bisa dimengerti, akan tetapi melakukan pengulangan materi yang pernah diberikan sebelumnya bisa menjadi efektif untuk memahaminya/menghafalnya.

Perulangan dalam algoritma sangat dibutuhkan untuk menyelesaikan banyak masalah. Pengurutan data yang banyak dalam program tertentu itu karena andil sebuah algoritma perulangan. Bayangkan jika harus menuliskan angka 1 sampai 100 dengan manual dalam sebuah program, akan memakan waktu, akan tetapi dengan menggunakan algoritma perulangan dalam progam hanya terdiri dari 4 langkah intruksi dalam algoritma sudah bisa menyelesaikan/menampilkan angka 1 sampai 100 bahkan lebih. Anda pernah menggunakan aplikasi/teknologi kecerdasan buatan pengenalan wajah atau sidik jari dari smartphone anda? Yakinlah bahwa didalamnya terdapat banyak perulangan.

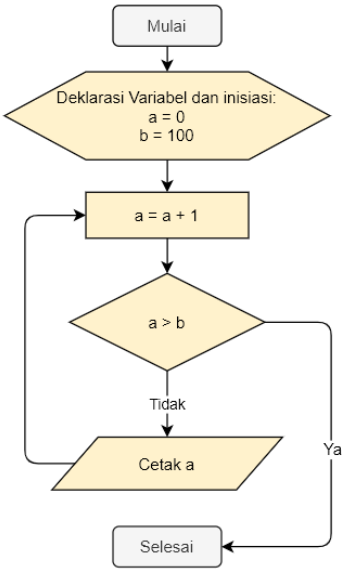
Di atas sudah saya sebutkan bahwa algoritma sendiri untuk mengatasi kasus pengulangan data, memiliki intruksi tersendiri, dengan intruksi tersebut pengulangan akan lebih mudah ditulis secara singkat dan praktis daripada harus di tulis satu-persatu. Akan saya berikan satu contoh *flowchart* untuk menuliskan angka 1 sampai dengan angka 5, sebagai berikut:



Gambar 1.5
*Flowchart d*engan Struktur Algoritma Skuesial untuk Cetak Angka 1 Sampai 5

Gambar 1.5 secara sekilas tampak tidak ada masalah, kenapa? Sebab angka yang dicetak masih sedikit, tapi coba dibayangkan jika harus mencetak angka 1 sampai angka 100 dengan mengikuti pola pada Gambar 1.5, tentu cara seperti ini tidak efektif, kenapa? Berapa anda harus membuat 102 simbol (termasuk simbol *mulai* dan *selesai*) dalam *flowchart* hanya untuk mencetak angka 1 sampai 100. Tentu *flowchart* di atas sangat tidak efektif.

Lalu apakah ada cara yang efektif? Cara yang efektif adalah dengan algoritma perulangan. Di bawah disajikan contoh *flowchart* mencetak angka 1 sampai 100 dengan kolaborasi antara algoritma perulangan dengan algoritma seleksi:



Gambar 1.6
Flowchart dengan Algoritma untuk Perulangan dan Seleksi Cetak Angka 1 Sampai 100

Sekarang bandingkan Gambar 1.5 dan Gambar 1.6. Gambar 1.5 hanya bisa mencetak angka 1 sampai 5, jika ingin cetak angka lebih dari 4 maka wajib menambahkan simbol *flowchart* baru. Sedangkan untuk Gambar 1.6 bisa mencetak angka 1 sampai angka 100, lalu bagaimana kalau kita ingin mencetak angka sampai 1000 untuk Gambar 1.6? jawabannya adalah tidak perlu menambah simbol *flowchart* baru, akan tetapi cukup mengubah b = 1000, cukup efektifkan?

Pseudocode cetak angka 1 - 100	
1	`Mulai`
2	`Deklarasi variabel dan inisiasi : a=0, b=100`
3	`a ← a + 1`
4	`if a > b`
5	`Cetak a`
6	`ke langkah 3`

```
7  else //lainnya
8  Selesai
```

Bagaimana proses perulangan berjalan pada *flowchart* Gambar 1.6 atau *pseudocode* perulangan? Berikut penjelasannya. Tahap awal variabel *a* diisi 1 dan variabel *b* diisi 100, (Perulangan Pertama) kemudian variabel *a* diisi dengan $a = a + 1$, sehingga $a = 1$, karena $a = a + 1$ sama dengan $a = 0 + 1$. Langkah berikutnya *a* diuji dengan *b*, apakah $a > b$ atau $1 > 100$, tentu jawabannya salah atau "Tidak", karena hasil selekasi adalah "Tidak" maka variabel a dicetak. (Perulangan Kedua) Dan kemudian kembali ke $a = a + 1$ atau $a = 1 + 1$ sehingga $a = 2$, proses ini akan berulang dan mencetak angka 100 hingga sampai memenuhi kondisi $a > b$ benar atau "Ya" dan algoritma perulangan selesai.

LATIHAN

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

1) Sebutkan dan jelaskan 3 struktur dasar pada algoritma?

Petunjuk Jawaban Latihan

1) Tiga struktur dasar algoritma adalah:
   - Skuensial: Struktur dasar algoritma yang mengerjakan semua proses perintah dalam sebuah.
   - Seleksi: Algoritma yang mengandung lebih dari satu kondisi pilihan dalam satu algoritma.
   - Perulangan: Algoritma yang mengandung perulangan-perulangan dalam proses intruksinya.

RANGKUMAN

Ada tiga struktur dasar algoritma, yaitu: skuensial (*squential*), seleksi (*selection*), dan perulangan (*looping*). Struktur dasar skuensial adalah sebuah algoritma dibangun dengan langkah-langkah (instruksi/perintah) dikerjakan secara berurutan, tidak boleh melompati satu langkah perintah pun. Struktur dasar algoritma seleksi adalah digunakan apabila dalam tubuh algoritma menemukan kasus dengan 2 atau lebih alternatif penyelesaian. Struktur dasar algoritma yang terakhir adalah perulangan, banyak hal dalam dalam dunia algoritma bisa diselesaikan dengan perulangan.

TES FORMATIF 2

Pilihlah satu jawaban yang paling tepat!

1) Struktur dasar algoritma yang menyelesaikan semua langkah dari setiap proses dalam algoritma adalah....
   A. Perulangan
   B. Skuensial
   C. Seleksi
   D. Semuanya benar

2) Sebuah algoritma yang mengandung 2 atau lebih alternatif solusi termasuk dalam struktur dasar....
   A. Seleksi
   B. Skuensial
   C. Perulangan
   D. Skuensial dan seleksi

3) Jika kita ingin mencetak angka 1 sampai 100, lebih cocok menggunakan struktur dasar algoritma....
   A. Skuensial
   B. Seleksi
   C. Skuensial dan Seleksi
   D. Perulangan

4) Algoritma yang terdapat didalamnya ada yang dibandingkan dan ada pembanding, struktur algoritma tersebut disebut dengan….
   A. Seleksi
   B. Perulangan
   C. Skuensial
   D. Tidak ada yang benar

5) Jika sebuah algoritma mengandung seleksi maka bisa dipastikan bahwa algoritma tersebut tidak mengandung struktur dasar….
   A. Skuensial dan Seleksi
   B. Seleksi
   C. Perulangan
   D. Skuensial

6) Struktur dasar yang bisa digabung dan saling mengisi adalah struktur dasar algoritma....
   A. Seleksi dan Perulangan
   B. Seleksi dan skuensial
   C. Skuensial dan Perulangan
   D. Tidak ada yang benar

7) Jika ada struktur dasar algoritma skuensial, bisa dipastikan bahwa struktur dasar algoritma yang tidak ada adalah...
   A. Perulangan dan seleksi
   B. Perulangan
   C. Seleksi
   D. Tidak ada yang benar

8) Jika sebuah variabe c = *5*, kemudian dalam sebuah perulangan dibuat c = c + *1*, dan dilakukan perulangan selama 5 kali, berapakah nilai c di akhir perulangan?
   A. 9
   B. 10
   C. 11
   D. 12

9) Jika sebuah variabe c = 5, kemudian dalam sebuah perulangan dibuat kondisi jika c = 7 maka operasi c = c + 2, jika tidak operasi c = c + 1, perulangan dilakukan perulangan selama 5 kali, berapakah nilai c di akhir perulangan?

A. 9
B. 10
C. 11
D. 12

10) Jika sebuah variabel k = 10, kemudian dalam sebuah perulangan dibuat kondisi jika k = 11 maka operasi k = k + 1, jika tidak maka operasi k = k + 2, perulangan dilakukan perulangan selama 6 kali, berapakah nilai k di akhir perulangan?
A. 17
B. 18
C. 19
D. 20

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 2 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 2.

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan: 90 - 100% = baik sekali
80 - 89% = baik
70 - 79% = cukup
< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Modul selanjutnya. **Bagus!** Jika masih di bawah 80%, Anda harus mengulangi materi Kegiatan Belajar 2, terutama bagian yang belum dikuasai.

Kunci Jawaban Tes Formatif

Tes Formatif 1

1) B
2) A
3) D
4) B
5) B
6) C
7) A
8) A
9) A
10) D

Tes Formatif 2

1) B
2) A
3) D
4) A
5) D
6) C
7) C
8) B
9) C
10) D

Glosarium

CPU (Central Processing Unit)	:	Perangkat keras komputer yang memiliki tugas untuk menerima dan melaksanakan perintah dan data dari perangkat lunak. Karena merupakan pusat pengolahan data dalam sebuah komputer, CPU sering disebut juga sebagai *processor*
GPS	:	Sistem untuk menentukan letak di permukaan bumi dengan bantuan penyelarasan (*synchronization*) sinyal satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke Bumi. Sinyal ini diterima oleh alat penerima di permukaan, dan digunakan untuk menentukan letak, kecepatan, arah, dan waktu.
Kecerdasan Buatan	:	Kkecerdasan yang ditambahkan kepada suatu sistem yang bisa diatur dalam konteks ilmiah atau bisa disebut juga intelegensi artifisial (*Artificial Intelligence*) disingkat AI, didefinisikan sebagai kecerdasan *entitas* ilmiah. Andreas Kaplan dan Michael Haenlein mendefinisikan kecerdasan buatan sebagai "kemampuan sistem untuk menafsirkan data eksternal dengan benar, untuk belajar dari data tersebut, dan menggunakan pembelajaran tersebut guna mencapai tujuan dan tugas tertentu melalui adaptasi yang fleksibel".
Mnemonik	:	Teknik yang memudahkan penyimpanan, atau penyandian dan pengingat terhadap informasi dalam memori.
Pseudocode	:	Deskripsi tingkat tinggi informal dan ringkas atas algoritme pemrograman komputer yang menggunakan konvensi struktural atas suatu bahasa pemrograman, dan ditujukan untuk dibaca oleh manusia dan bukan oleh mesin.

Daftar Pustaka

Christodoulou, M. & Szczygiel, E. (2018). *Algoritmic and programming.* P.T.E.A. Wszechnica Sp. Z.o.o.

Erickson, J. (2019). *Algoritms.* xxx.

Hermin, F. & Widyati, R. (2004). *Komputer II.* Penerbit Universitas Terbuka.

Horwits, E. & Sahni S. (1993). *Fundamental of data structure in C++, source DBLP.*

Laaksonen, A. (2018). *Competitive programmer's handbook.*

Nugroho, E. (2010). *Pengantar aplikasi komputer.* Penerbit Universitas Terbuka.

Rinaldi, M. (1999). *Algoritma dan pemrograman Jilid1.* Penerbit IPB.

Wimatra, A. dkk. (2008). *Dasar-dasar komputer.* Civil Aviation Safety and Technics Academy. Medan.

Purbasari, I. Y., "*Desain & analisis algoritma*". Yogjakarta: Graha Ilmu