1
Pengantar ke Algoritma

1.1 Pendahuluan

Komputer (*hardware*) dibuat sebagai alat bantu untuk menyelesaikan masalah. Permasalahan apa pun dapat diselesaikan oleh komputer asalkan langkah-langkah penyelesaiannya disediakan oleh oleh manusia (*brainware*). Dengan kata lain, manusia menulis **program** (*software*) yang berisi urutan langkah-langkah penyelesaian masalah, lalu proram tersebut "dimasukkan" ke dalam komputer. Dalam hal ini, komputer hanya bertindak menjalankan perintah-perintah yang tertulis di dalam program tersebut. Sebenarnya manusia sendiri mampu melaksanakan perintah- perintah tersebut, tetapi komputer mempunyai kelebihan dibandingkan manusia. Komputer adalah benda mati, jadi ia tidak mengenal lelah dan bosan. Komputer mampu mengerjakan perintah yang banyak sekalipun, selain itu ia juga mampu mengerjakan suatu perintah yang sama berulang kali, 100 kali, sejuta kali, atau berapa kalipun yang manusia perintahkan. Manusia suka pelupa, sedangkan komputer tidak. Komputer memiliki memori yang besar sehingga ia mampu menyimpan data dan informasi dalam volume yang banyak.

Perhatikan paragraf di atas kembali. Ketiga kata yang disebutkan: *hardware*, *software*, dan *brainware*, adalah tiga komponen utama *sistem komputer*. Komputer baru bermanfaat jika ia diprogram. Sebagai ilustrasi sederhana saja, misalkan anda ingin menggunakan komputer untuk mengurutkan sekumpulan nilai. Permasalahan pengurutan (*sorting*) banyak ditemukan dalam kehidupan sehari-hari. Kita lebih mudah membaca data yang terurut daripada data yang tersusun secara acak. Data yang terurut memudahkan kita mencari data tertentu (misalnya mencari nama di dalam katalog buku telepon). Data yang terurut juga memudahkan kita menentukan *ranking* (misalnya dalam menyeleksi siswa baru berdasarkan nilai tes masuk). Jika data yang diurutkan hanya sedikit (misalnya hanya 10 buah data), tentu komputer tidak terlalu diperlukan. Manusia sendiri mampu mengurutkannya secara manual. Bagiamana jika data yang diurutkan jumlahnya sangat banyak (misalnya 10.000 sampai satu juta data)? Tentu manusia

mampu juga melakukannya, tetapi mungkin dibutuhkan waktu yang cukup lama, ketelitian, kesabaran, dan sebagainya. Disinilah peran komputer membantu manusia untuk melakukan pengurutan tersebut. Seperti yang disebutkan di paragraf pertama, komputer mampu melakukan perintah yang diberikan tanpa mengenal lelah dan bosan namun tetap memberikan hasil pekerjaan yang teliti dalam waktu yang relatif sangat cepat.

Coba anda pikirkan, bagaimana cara anda mengurutkan sejumlah data? Taruhlah anda punya setumpuk kartu yang diberi nomor 1 sampai 50. Kartu tersebut semula tersusun acak, dan sekarang anda perlu mengurutkannya kembali dari nomor kecil ke nomor besar (nomor kecil di atas dan nomor besar di bawah). Secara tradisional, langkah-langkah yang anda lakukan adalah mencari kartu dengan nomor terbesar (50), lalu meletakkannya paling bawah. Selanjutnya, cari kartu dengan nomor terbesar kedua (49), lalu letakkan di atas kartu pertama. Begitu seterusnya, anda mencari kartu dengan nomor terbesar ketiga, terbesar keempat, sampai akhirnya seluruh kartu sudah terurut dengan nomor kecil di atas dan nomor besar di bawah. Sebagai catatan, anda juga melakukan cara yang berkebalikan, yaitu cari kartu dengan nomor terkecil (1), lalu letakkan pada posisi pertama. Selanjutnya kartu dengan nomor terkecil kedua (2), lalu letakkan di bawah kartu pertama. Begitu seterusnya, anda mencari kartu dengan nomor terkecil ketiga, terkecil keempat, sampai akhirnya seluruh kartu sudah terurut dengan nomor kecil di atas dan nomor besar di bawah.

Sekarang, mari kita tuliskan langkah-langkah pengurutan 50 buah kartu tersebut sebagai berikut:

Langkah 1: Cari kartu dengan nomor terbesar
Langkah 2: Tempatkan kartu tersebut pada posisi paling bawah (posisi 50)
Langkah 3: Cari kartu dengan nomor terbesar kedua
Langkah 4: Tempatkan kartu tersebut pada posisi 49
Langkah 5: Cari katu dengan nomor terbesar ketiga
Langkah 6: Tempatkan kartu tersebut pada posisi 48
Langkah 8: Cari katu dengan nomor terbesar keempat
Langkah 9: Tempatkan kartu tersebut pada posisi 47

... (dst)

Perhatikanlah bahwa sebenarnya di dalam langkah-langkah penyelesaian di atas hanya ada dua perintah, yaitu mencari kartu dengan nomor terbesar dan menempatkan kartu tesrebut pada posisi yang benar. Kedua perintah ini diulang berkali-kali tetapi dengan acuan yang berbeda (kedua, ketiga, dan seterusnya). Langkah-langkah penyelesaian di atas dapat kita tulis lebih sederhana sebagai berikut:

Langkah 1: Cari kartu dengan nomor terbesar
Langkah 2: Tempatkan nilai terbesar tersebut pada posisi yang tepat
Langkah 3: Ulangi Langkah 1 untuk 49 buah kartu yang lain

Secara umum, jika kita mempunyai N buah kartu, maka langkah-langkah pengurutannya dapat dibuat lebih umum sebagai berikut:

Langkah 1: Cari kartu dengan nomor terbesar
Langkah 2: Tempatkan nilai terbesar tersebut pada posisi yang tepat
Langkah 3: Ulangi Langkah 1 untuk $N - 1$ buah kartu yang lain

Perhatikan bahwa Langkah 1 masih perlu dirinci lagi menjadi langkah-langkah yang lebih primitif. Hal ini akan dibahas di dalam Bab 2 nanti.

Ide pengurutan tumpukan kartu dengan cara di atas mengilahmi orang untuk menulis program pengurutan sekumpulan data dengan bantuan komputer. Data yang akan diurutkan harus dimasukkan dulu ke dalam memori komputer dengan cara "pembacaan", selanjutnya data tersebut diurut dengan langkah-langkah yang sudah dijelaskan di atas. Agar program dapat dilaksanakan oleh komputer, maka program tersebut harus ditulis dalam bahasa komputer khusus. Bahasa komputer yang digunakan dalam menulis program dinamakan **bahasa pemrograman**. Orang yang membuat program komputer disebut **pemrogram**, dan kegiatan merancang dan menulis program disebut **pemrograman**.

Salah satu dari sekian banyak bahasa pemrograman yang tersedia saat ini adalah Bahasa Pascal. Algoritma 1.1 di bawah ini adalah program yang ditulis dalam Bahasa Pascal untuk melakukan pengurutan sekumpulan nilai ujian mahasiswa. Data nilai ujian mahasiswa dibaca dari papan ketik (*keyboard*), lalu diurutkan, dan hasil pengurutan dicetak ke layar peraga (monitor). Untuk sementara ini anda tidak perlu memikirkan kenapa programnya demikian, karena inilah materi yang akan anda pelajari di dalam buku algoritma ini. Jadi, bersabar dululah, yang penting saat ini adalah anda mengetahui contoh rupa sebuah program komputer.

```
program PENGURUTAN;
{ Program untuk mengurutkan nilai ujian sejumlah mahasiswa }
const
   Nmaks = 1000;     { jumlah maksimum data }
var
   Nilai : array[1..Nmaks] of integer; { tempat menyimpan data }
   j,k,temp, N, Imaks : integer;

begin
   {baca data nilai ujian N orang mahasiswa}
   read(N);
   for j:=1 to N do
     readln(Nilai[j]);
   {endfor}

   {urutkan data dengan langkah-langkah berikut:}
   for j:=1 to N-1 do  { ulangi sebanyak N – 1 kali }
    begin
      {cari nilai terbesar }
      Imaks:=j;
      for k:=j+1 to N do
        if Nilai[k] > Nilai[j] then
          Imaks:=k;
        {endif}
      {endfor}

      {tempatkan nilai terbesar pada posisi yang tepat }
      temp:=Nilai[j];
      Nilai[j]:=Nilai[Imaks];
      Nilai[Imaks]:=temp;
    end; {for}

    { tuliskan nilai yang sudah terurut!}
    for j:=1 to N do
       writeln(Nilai[j]);
    {endfor}
end.
```

Algoritma 1.1 *Program pengurutan nilai ujian N orang mahasiswa*

1.2 Program Komputer dan Algoritma

Anda sudah melihat contoh sebuah program komputer untuk menyelesaikan masalah pengurutan. Program komputer berisi urutan langkah-langkah penyelesaian masalah secara sistematis dan ditulis dalam bahasa pemrograman tertentu (pada contoh di atas Bahasa Pascal). Urutan langkah-langkah penyelesaian masalah inilah yang dinamakan **algoritma**. Ada banyak definisi algoritma yang dijumpai di berbagai literatur, namun definisi yang umum adalah sebagai berikut:

Definisi:

Algoritma adalah urutan logis langkah-langkah penyelesaian masalah.

Jadi, program komputer pada hakikatnya adalah realisasi teknis dari sebuah algoritma. Disebut realisasi teknis karena algoritma dikodekan ke dalam bahasa pemroraman tertentu. Kita dapat menuliskan kembali algoritma dari program pengurutan di atas sebagai berikut ini (tidak termasuk pembacaan data dan penulisan hasi pengurutan):

```
PROGRAM Pengurutan
Program untuk mengurutkan nilai ujian sejumlah mahasiswa

ALGORITMA:
 1. Cari nilai terbesar di antara N buah data
 2. Tempatkan nilai terbesar tersebut pada posisi yang tepat
 3. Ulangi dari langkah 1 untuk N – 1 buah data yang lain
```

Sejarah kata "algoritma"

Ditinjau dari asal usul kata, kata algoritma sendiri mempunyai sejarah yang aneh. Kata ini tidak muncul di dalam kamus Webster sampai akhir tahun 1957. Orang hanya menemukan kata *algorism* yang berarti proses menghitung dengan angka Arab [KNU73]. Anda dikatakan *algorist* jika anda menggunakan angka Arab. Para ahli bahasa berusaha menemukan asal kata *algorism* ini namun hasilnya kurang memuaskan. Akhirnya para ahli sejarah matematika menemukan asal mula kata tersebut. Kata *algorism* berasal dari nama penulis buku Arab yang terkenal, yaitu Abu Ja'far Muhammad ibnu Musa al-Khuwarizmi (al-Khuwarizmi dibaca orang Barat menjadi *algorism*). Al-Khuwarizmi menulis buku yang berjudul *Kitab al jabar wal-muqabala*, yang artinya "Buku pemugaran dan pengurangan" (*The book of restoration and reduction*). Dari judul buku itu kita juga memperoleh akar kata "aljabar" (*algebra*). Perubahan dari kata *algorism* menjadi *algorithm* muncul karena kata *algorism* sering dikelirukan dengan *arithmetic*, sehingga akhiran *-sm* berubah menjadi *-thm*. Karena perhitungan dengan angka Arab sudah menjadi hal yang sudah biasa/ lumrah, maka lambat laun kata *algorithm* berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna aslinya [PAR95]. Dalam bahasa Indonesia, kata *algorithm* diserap menjadi *algoritma*.

(a)

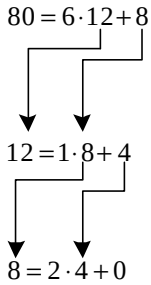
(b)

Gambar 1.1
(a) Sketsa wajah al-Khuwarizmi, (b) al-Khuwarizmi dalam sebuah perangko (?).

Pada tahun 1950, kata algoritma sering dihubungkan dengan "algoritma Euclidean" (*Euclid's algorithm*), yaitu proses untuk menemukan pembagi bersama terbesar (*common greatest divisor* atau *gcd*), dari dua buah bilangan bulat, *m* dan *n* [KNU73]. Pembagi bersama terbesar dari dua buah bilangan bulat tak-negatif adalah bilangan bulat positif terbesar yang habis membagi kedua bilangan tersebut. Misalnya, $m = 80$ dan $n = 12$. Faktor pembagi 80 adalah 1, 2, **4**, 5, 8, 10, 16, 20, 40, 80, dan faktor pembagi 12 adalah 1, 2, 3, **4**, 6, 12, maka *gcd*(80,12) = 4. Langkah-langkah mencari *gcd*(80, 12) dengan algoritma Euclidean adalah sebagai berikut:

80/12 = 6, sisa 8
12/8 = 1, sisa 4
8/4 = 2 sisa 0

Karena pembagian yang terakhir menghasilkan 0, maka sisa pembagian terakhir sebelum 0, yaitu 4, menjadi *gcd*(80,12). Jadi, *gcd*(80,12) = *gcd*(12,8) = *gcd*(8,4) = *gcd*(4,0)= 4. Proses mencari *gcd* dari 80 dan 12 juga dapat diilustrasikan dalam diagram berikut:



Ada beberapa versi algoritma Euclidean, salah satu versinya dituliskan di bawah ini.

```
PROGRAM Euclidean
Diberikan dua buah bilangan bulat tak-negatif m dan n (m ³ n). Algoritma
Euclidean mencari pembagi bersama terbesar, gcd, dari kedua bilangan
tersebut, yaitu bilangan bulat positif terbesar yang habis membagi m dan n.

ALGORITMA:
1. Jika n = 0 maka
     m adalah jawabannya;
     stop.
   tetapi jika n ≠ 0,
     lanjutkan ke langkah 2.
2. Bagilah m dengan n dan misalkan r adalah sisanya.
3. Ganti nilai m dengan nilai n dan nilai n dengan nilai r, lalu ulang
   kembali ke langkah 1.
```

Algoritma 1.2 *Algoritma Euclidean untuk menentukan gcd dari dua buah bilangan bulat*

Dengan menggunakan $m = 80$ dan $n = 12$, maka eksekusi algoritma Euclidean untuk mencari *gcd*(80,12) adalah sebagai berikut:

1(i) Karena $n = 12 \neq 0$, maka lanjutkan ke langkah 2(1)
2(i) Hitung $m/n = 80/12 = 6$, sisanya $r = 8$
3(i) Nilai $m_{baru} = n_{lama} = 12$ dan $n_{baru} = r = 8$. Lanjut ke langkah 1(2).
1(ii) Karena $n = 8 \neq 0$, maka lanjutkan ke langkah 2(2)
2(ii) Hitung $m/n = 12/8 = 1$, sisanya $r = 4$.
3(ii) Nilai $m_{baru} = n_{lama} = 8$ dan $n_{baru} = r = 4$. Lanjut ke langkah 1(3).
1(iii) Karena $n = 4 \neq 0$, maka lanjutkan ke langkah 2(3)
2(iii) Hitung $m/n = 8/4 = 2$, sisanya $r = 0$.
3(iii) Nilai $m_{baru} = n_{lama} = 4$ dan $n_{baru} = r = 0$. Lanjut ke langkah 1(4).
1(iv) Karena $r = 0$, maka $n = 4$ adalah jawabannya. Stop.
Jadi, *pbt*(80,12) = 4.

Keterangan: angka romawi di dalam kurung, yaitu (i), (ii), (iii) dan iv), masing-masing menyatakan perulangan yang ke-1, ke-2, ke-3, dan ke-4 dari instruksi yang dijalankan.

Satu hal yang perlu dicatat, langkah-langkah penyelesaian masalah di dalam algoritma haruslah menyatakan urutan yang logis. Langkah-langkah yang logis berarti bahwa hasil dari urutan langkah-langkah tersebut menghasilkan penyelesaian yang benar. Langkah-langkah yang tidak logis dapat memberikan hasil yang salah. Contoh 1.1 dan Contoh 1.2 berikut ini memperlihatkan bahwa urutan langkah yang logis menentukan kebenaran algoritma.

Contoh 1.1 (Mempertukarkan isi dua buah ember)
Tinjau sebuah persoalan sederhana yaitu persoalan mempertukarkan isi dari dua buah ember, sebut ember *A* dan *B*. Ember *A* berisi air yang berwarna merah, sedangkan ember *B* berisi air berwarna biru (Gambar 1.2). Kita ingin mempertukarkan isi kedua ember itu sedemikian sehingga ember *A* akan berisi air berwarna biru dan ember *B* berisi air berwarna merah.

Ember *A* Ember *B*

Gambar 1.2
Dua buah ember berisi air yang berbeda warna.
Ember A berisi air berwarna merah, ember B berisi air berwarna biru.

Penyelesaian:
Misalkan seseorang menuliskan langkah-langkah pertukaran isi kedua ember tersebut ke dalam program `Tukar_Isi` dengan algoritma sebagai berikut:

```
PROGRAM Tukar_Isi
Diberikan dua buah ember, A dan B; ember A berisi air berwarna merah,
ember B berisi air berwarna biru. Pertukarkan isi kedua ember itu
sedemikian sehingga ember A akan berisi air berwarna biru dan ember B
berisi air berwarna merah.

ALGORITMA:
1. Tuangkan air dari ember A ke dalam ember B.
2. Tuangkan air dari ember B ke dalam ember A.
```

Perhatikan bahwa algoritma `Tukar_Isi` di atas tidak akan menghasilkan pertukaran yang benar. Langkah-langkahnya tidak logis. Alih-alih mempertukarkan, yang terjadi adalah percampuran keduanya. Dengan kata lain, algoritma `Tukar_Isi` tersebut salah!

Agar dapat melakukan pertukaran yang benar, kita memerlukan sebuah ember tambahan yang diperlukan sebagai tempat penampungan sementara. Namakan ember tambahan tersebut ember *C*. Dengan menggunakan ember *C* ini, algoritma mempertukarkan isi dua buah ember yang benar adalah seperti Algoritma 1.3 berikut ini:

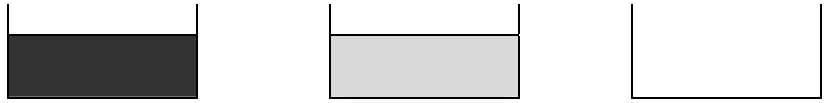
```
PROGRAM Tukar_Isi
Diberikan dua buah ember, A dan B; ember A berisi air berwarna merah,
ember B berisi air berwarna biru. Pertukarkan isi kedua ember itu
sedemikian sehingga ember A berisi air berwarna biru dan ember B berisi
air berwarna merah.

ALGORITMA:
1. Tuangkan air dari ember A ke dalam ember C.
2. Tuangkan air dari ember B ke dalam ember A.
3. Tuangkan air dari ember C ke dalam ember B.
```

Algoritma 1.3 *Mempertukarkan isi dari dua buah ember*

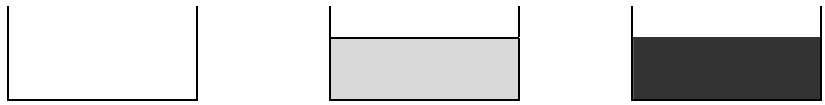
Sekarang, dengan algoritma `Tukar_Isi` yang sudah diperbaiki ini, isi ember *A* dan ember *B* dapat dipertukarkan dengan benar. Pada akhir algoritma, ember *A* berisi ai berwarna biru dan ember *B* berisi air berwarna merah. Proses pertukaran tersebut diilustrasikan pada Gambar 1.3.

Keadaan awal sebelum pertukaran:



Proses Pertukaran:

1. Tuangkan air dari ember *A* ke dalam ember *C*



2. Tuangkan air dari ember *B* ke dalam ember *A*

3. Tuangkan air dari ember *C* ke dalam ember *B*

Keadaan Akhir Setelah Pertukaran:

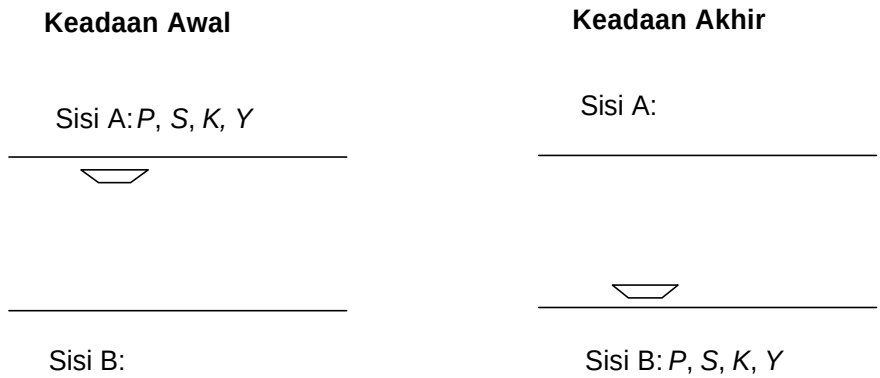
Gambar 1.2
Proses pertukaran isi ember A dan ember B, menggunakan ember bantu C.
Setelah pertukaran, ember A berisi air dari ember B semula, demikian pula sebaliknya.

Contoh 1.2 (Menyeberangkan barang bawaan)

Misalkan seorang pemuda tiba di tepi sebuah sungai. Pemuda tersebut membawa seekor kambing, seekor srigala, dan sekeranjang sayur. Mereka bermaksud hendak menyeberangi sungai. Pemuda itu menemukan sebuah perahu kecil yang hanya dapat memuat satu bawaannya setiap kali menyeberang. Situasinya dipersulit dengan kenyataan bahwa srigala tidak dapat ditinggal berdua dengan kambing (karena srigala akan memangsa kambing) atau kambing tidak dapat ditinggal berdua dengan sekeranjang sayur (karena kambing akan memakan sayur). Buatlah algoritma untuk menyeberangkan pemuda dan seluruh bawaannya itu sehingga mereka sampai ke seberang sungai dengan selamat.

Penyelesaian:

Misalkan sisi sungai kita namakan *A* dan sisi sungai seberangnya kita namakan *B*. Keadaan awalnya, di sisi *A* ada pemuda (*P*), srigala (*S*), kambing (*K*), dan sayur (*Y*). Keadaan akhir yang kita inginkan adalah di sisi *B* ada pemuda (*P*), srigala (*S*), kambing (*K*), dan sayur (*Y*). Gambar 1.3 memperlihatkan keadaan yang dimaksud.



Gambar 1.3

Keadaan awal dan keadaan akhir proses penyeberangan pemuda (P) dan bawaannya yang terdiri dari srigala (S), kambing (K), dan sekeranjang sayur (Y). Perahu hanya dapat memuat satu bawaan saja pada setiap kali menyeberang. Srigala tidak dapat ditinggalkan bersama kambing, begitu pula kambing tidak dapat ditinggalkan bersama sayur.

Algoritma menyeberangkan seluruh bawaan tersebut adalah berupa runtunan langkah seperti dirunjukkan pada Algoritma 1.4.

```
PROGRAM Menyeberangi_Sungai
Program untuk menyeberangkan pemuda (P) dan tiga barang bawaannya: srigala
(S), kambing (K), dan sekeranjang sayur (Y) dengan sebuah perahu kecil
yang hanya dapat memuat 2 item (pemuda dan salah satu bawaannya). Srigala
tidak dapat ditinggal berdua dengan kambing (karena srigala akan memangsa
kambing) atau kambing tidak dapat ditinggal berdua dengan sekeranjang
sayur (karena kambing akan memakan sayur).

ALGORITMA:
   { sisi A: (P,S,K,Y)   sisi B: (-,-,-,-) }
1. Pemuda menyeberangkan kambing dari sisi A ke sisi B.
   { sisi A: (-,S,K,Y)   sisi B: (P,-,K,-) }
2. Pemuda menyeberang sendiri dari B ke A.
   { sisi A: (P,S,-,Y)   sisi B: (-,-,K,-) }
3. Pemuda menyeberangkan srigala dari sisi A ke sisi B.
   { sisi A: (-,-,-,Y)   sisi B: (P,S,K,-) }
4. Pemuda menyeberangkan kambing dari sisi B ke sisi A.
   { sisi A: (P,-,K,Y)   sisi B: (-,S,-,-) }
5. Pemuda menyeberangkan sayur dari sisi A ke sisi B.
   { sisi A: (-,-,K,-)   sisi B: (P,S,-,Y) }
6. Pemuda menyeberang sendiri dari B ke A.
   { sisi A: (P,-,K,-)   sisi B: (-,S,-,Y) }
3. Pemuda menyeberangkan kambing dari sisi A ke sisi B.
   { sisi A: (-,-,-,-)   sisi B: (P,S,K,Y) }
```

Algoritma 1.4 *Menyeberangkan tiga barang bawaan dengan perahu kecil*

Keterangan:

Kalimat di dalam pasangan kurung kurawal { dan } disebut komentar (*comment*). Komentar bukan bagian dari langkah penyelesaian di dalam algoritma. Namun, komentar diperlukan untuk mempermudah membaca dan memahami algoritma. Komentar berisi penjelasan tambahan tentang segala sesuatu di dalam algoritma.

Ketiga contoh algoritma yang sudah kita bicarakan: algoritma Euclidean, algoritma mempertukarkan isi dua buah ember, dan algoritma menyeberangkan barang bawaan memberikan dua pesan penting. Pertama, sebuah algoritma harus benar. Kedua, algoritma harus berhenti, dan setelah berhenti, algoritma memberi hasil yang benar.

Menurut Donald E. Knuth dalam bukunya yang berjudul *The Art of Computer Programming* [KNU73], algoritma harus mempunyai lima ciri penting:

1. Algoritma harus berhenti setelah mengerjakan sejumlah langkah terbatas. Sebagai contoh, tinjau kembali algoritma `Euclidean`. Pada langkah 1, jika $n = 0$, algoritma berhenti. Jika $n \neq 0$, maka nilai n selalu berkurang sebagai akibat langkah 2 dan 3, dan pada akhirnya nilai $n = 0$. Program yang tidak pernah berhenti mengindikasikan bahwa program tesrebut berisi algoritma yang salah.
2. Setiap langkah harus didefinisikan dengan tepat dan tidak berarti-dua (*ambiguous*). Pembaca harus mengerti apa yang dimaksud dengan "m dan n adalah bilangan bulat tak-negatif". Contoh lainnya, pernyataan "bagilah p dengan sejumlah beberapa buah bilangan bulat positif" dapat bermakna ganda. Berapakah yang dimaksud dengan "berapa"? Algoritma menjadi jelas jika langkah tersebut ditulis "bagilah p dengan 10 buah bilangan bulat positif".

3. Algoritma memiliki nol atau lebih masukan (*input*). Masukan ialah besaran yang diberikan kepada algoritma sebelum algoritma mulai bekerja. Algoritma `Euclidean` mempunyai dua buah masukan, *m* dan *n*, sedangkan algoritma `Tukar_Isi` memiliki masukan berupa air di dalam ember *A* dan air di dalam ember *B*.
4. Algoritma mempunyai nol atau lebih keluaran (*output*). Keluaran ialah besaran yang memiliki hubungan dengan masukan. Algoritma `Euclidean` mempunyai satu keluaran, yaitu *n* pada langkah 2, yang merupakan pembagi bersama terbesar dari kedua masukannya. Algoritma `Tukar_Isi` tidak memiliki keluaran sama sekali.
5. Algoritma harus sangkil (*effective*). Setiap langkah harus sederhana sehingga dapat dikerjakan dalam sejumlah waktu yang masuk akal.

1.3 Algoritma Merupakan Jantung Informatika

Algoritma adalah *jantung* ilmu komputer atau informatika. Banyak cabang ilmu komputer yang diacu dalam terminologi algoritma. Dalam kehidupan sehari-haripun banyak terdapat proses yang digambarkan dalam suatu algoritma. Cara-cara membuat kue atau masakan, misalnya dinyatakan dalam suatu resep. Resep masakan adalah suatu algoritma, misalnya resep membuat *Otak-otak Ikan Bandeng* (dikutip dari Tabloid *Nova* 25 Agustus 1996) seperti dapat anda baca pada Gambar 1.4.

Pada setiap resep selalu ada urutan langkah-langkah membuat masakan. Ibu-ibu yang mencoba resep suatu masakan akan membaca satu per satu langkah pembuatannya, lalu ia mengerjakan proses sesuai yang ia baca. Secara umum, pihak (benda) yang mengerjakan proses disebut *pemroses (processor)*. Pemroses tersebut dapat berupa manusia, komputer, robot, atau alat-alat mekanik/elektronik lainnya. Pemroses melakukan suatu proses dengan melaksanakan atau mengeksekusi algoritma yang menjabarkan proses tersebut. Melaksanakan algoritma berarti mengerjakan langkah- langkah di dalam algoritma tersebut.

Resep *Otak-otak Ikan Bandeng*

Bahan:
- 2 ekor ikan bandeng ukuran kecil
- 2 buah kentang kukus
- 2 butir telur
- 2 tangkai daun bawang, dirajang halus
- 75 cc santan kental
- 3 butir kemiri sangrai
- 1 sendok teh ketumbar halus
- 2 siung bawang putih
- 3 butir bawang merah
- 2 cabe merah tanpa biji
- garam dan merica secukupnya
- minyak goreng secukupnya

Cara membuat:
1. Haluskan kemiri, ketumbar, bawang putih, bawang merah, cabe merah, garam, dan merica. Sisihkan.
2. Ambil ikan bandeng yang telah dibersihkan. Pukul-pukul dengan anak lumpang hingga daging ikan hancur di dalam.

```
3. Tekuk ekor ikan ke arah atas hingga terdengar bunyi "klik" sebagai
   tanda tulang telah patah.
4. Tarik tulang ke arah atas melalui bagian kepala. Keluarkan pula isi
   dagingnya. Sisihkan tulang-tulang halus yang masih tersisa.
5. Giling daging ikan bersama bumbu halus. Tuangkan santan. Campur
   dengan kentang kukus. Aduk rata.
6. Isikan adonan ikan ke dalam kulit bandeng. Kukus selama 20 menit.
7. Goreng hingga matang
```

Gambar 1.4
Resep membuat Otak-otak Ikan Bandeng adalah contoh sebuah algoritma. Urutan langkah-langkah membuat masakan diberi nomor 1 sampai 7.

Contoh-contoh lain algoritma dalam kehidupan sehari-hari misalnya pola pakaian, panduan praktikum, papan not balok, dan pengisian *voucher* pada telepon seluler ditunjukkan pada Tabel 1.1.

Pemroses mengerjakan proses sesuai dengan algoritma yang diberikan kepadanya. Juru masak membuat kue berdasarkan resep yang diberikan kepadanya, pianis memainkan lagu berdasarkan papan not balok, teknisi merakit mobil berdasarkan panduan merakit. Karena itu suatu algoritma harus dinyatakan dalam bentuk yang dapat dimengerti oleh pemroses. Seorang pianis tidak dapat memainkan musik bila ia tidak mengerti not balok. Menurut [GOL88], suatu pemroses harus:

1. Mengerti setiap langkah dalam algoritma,
2. Mengerjakan operasi yang bersesuaian dengan langkah tersebut.

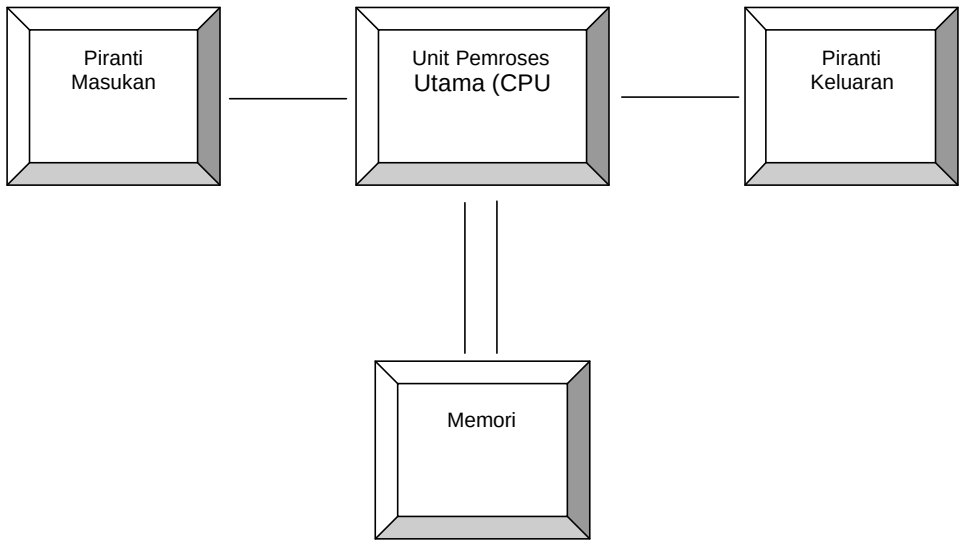
Tabel 1.1
Contoh-contoh Algoritma dalam Kehidupan Sehari-hari.

Proses	Algoritma	Contoh Langkah dalam Algoritma
1. Membuat kue	resep kue	masukkan telur ke dalam wajan,kocok sampai mengembang
2. Membuat pakaian	pola pakaian	gunting kain dari pinggir kiri bawah ke arah kanan sejauh 5 cm
3. Praktikum reaksi kimia	panduan praktikum	campurkan 10 ml H_2SO4 ke dalam 15 ml NaOH
4. Merakit mobil	panduan merakit	sambungkan komponen *A* dengan komponen *B*
5. Kegiatan sehari-hari	jadwal harian	pukul 15.00 : tidur siang pukul 16.00 : membuat PR
6. Memainkan musik	papan not balok	not balok
7. Mengisi *voucher* telepon genggam (HP)	panduan pengisian	tekan nomor 888 masukkan nomor *voucher* 14 digit

1.4 Mekanisme Pelaksanaan Program oleh Komputer

Secara garis besar komputer tersusun atas empat komponen utama: piranti masukan, piranti keluaran, unit pemroses utama, dan memori (Gambar 1.5). Unit pemroses utama (*Central Processing Unit - CPU*) adalah "otak" komputer, yang berfungsi mengerjakan operasi-operasi dasar seperi operasi perbandingan, operasi perhitungan, operasi membaca, dan operasi menulis. Memori adalah komponen yang berfungsi menyimpan atau mengingat-ingat. Yang disimpan di dalam memori adalah program (berisi operasi-operasi yang akan dikerjakan oleh *CPU*) dan data atau informasi (sesuatu yang diolah oleh operasi-operasi). Piranti masukan dan keluaran (*I/O devices*) adalah alat yang memasukkan data atau program ke dalam memori, dan alat yang digunakan komputer untuk mengkomunikasikan hasil-hasil aktivitasnya. Contoh piranti masukan antara lain papan ketik (*keyboard*), pemindai (*scanner*), tetikus (*mouse*), *joystick*, dan cakram (*disk*). Contoh piranti keluaran adalah layar peraga (*monitor*), pencetak (*printer*), perajah (*plotter*), dan cakram.

Mekanisme kerja keempat komponen di atas dapat dijelaskan sebagai berikut. Mula-mula program dimasukkan ke dalam memori komputer. Ketika program dieksekusi (*execute*), setiap perintah di dalam program yang telah tersimpan di dalam memori dikirim ke *CPU*. *CPU* mengerjakan operasi-operasi yang bersesuaian dengan perintah tersebut. Bila suatu perintah di dalam program meminta data masukan, maka data dibaca dari piranti masukan, lalu dikirim ke *CPU* untuk operasi yang memerlukannya. Bila program menghasilkan keluaran, maka keluaran tersebut ditulis ke piranti keluaran (misalkan dengan mencetaknya ke layar peraga).



Gambar 1.5
Komponen-komponen utama komputer.
Komputer terdiri atas Unit Pemroses Utama ,
Memori, Piranti Masukan, dan Piranti Keluaran.

1.5 Belajar Memprogram dan Belajar Bahasa Pemrograman

Belajar memprogram tidak sama dengan belajar bahasa pemrograman [LIE96]. Belajar memprogram adalah belajar tentang metodologi pemecahan masalah, kemudian menuangkan algoritma pemecahan masalah dalam suatu notasi tertentu. Sedangkan belajar bahasa pemrograman berarti belajar memakai suatu bahasa, aturan tata bahasanya, instruksi-instruksinya, tata cara pengoperasian *compiler*-nya, dan memanfaatkan instruksi-instruksi tersebut untuk membuat program yang ditulis hanya dalam bahasa itu saja [LIE96].

Dalam pelajaran pemrograman, kita lebih memikirkan pada cara penyelesaian masalah yang akan diprogram dengan menekankan pada **desain** atau rancangan yang mewakili pemecahan masalah tersebut. Desain ini dibuat sedemikian sehingga independen dari bahasa pemrograman yang kelak digunakan dan komputer yang akan menjalankan program tesrebut. Desain berisi urutan langkah-langkah pencapaian solusi yang ditulis dalam notasi-notasi deskriptif (notasi ini kelak kita sebut **notasi algoritmik**). Setiap orang yang berbeda mungkin menghasilkan rancangan program yang berbeda pula. Karena belajar memprogram bukan belajar membuat program yang asal jadi, perlu dipikirkan membuat program dengan menggunakan skema yang benar. Inilah titik berat dari pelajaran pemrograman.

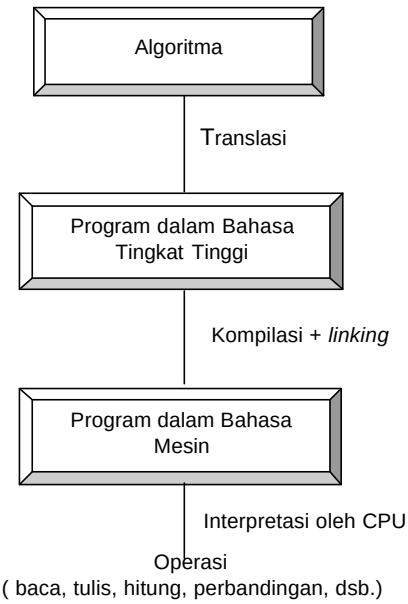
Bila desain program sudah dibuat dengan skema yang benar, maka desain tersebut siap dikodekan dengan notasi bahasa pemrograman agar program bisa dieksekusi oleh komputer. Disinilah perlunya kita belajar bahasa pemrograman. Ada banyak bahasa pemrograman yang tersedia, namun desain program dapat diterjemahkan ke bahasa apapun. Sampai saat ini terdapat puluhan bahasa pemrograman. Kita dapat menyebutkan antara lain bahasa rakitan (*assembly*), *Fortran*, *Cobol*, *Ada*, *PL/I*, *Algol*, *Pascal*, *C*, *C++*, *Basic*, *Prolog*, *LISP*, *PRG*, bahasa-bahasa simulasi seperti *CSMP*, *Simscript*, GPSS, *Dinamo*, dan banyak lagi. Belakangan juga muncul bahasa pemrograman baru seperti *Perl* dan *Java*. Berdasarkan aplikasi kegunaanya, bahasa pemrograman dapat digolongkan menajdi dua kelompok:

1. **Bahasa pemrograman bertujuan khusus (*specific purpose programming language*).** Yang termasuk kelompok ini adalah *Cobol* (untuk terapan bisnis dan administrasi), *Fortran* (aplikasi komputasi ilmiah), bahasa *assembly* (aplikasi pemrograman mesin), *Prolog* (aplikasi kecerdasan buatan), bahasa-bahasa simulasi, dan sebagainya.
2. **Bahasa pemrograman bertujuan umum (*general purpose programming language*)** yang dapat digunakan untuk berbagai aplikasi. Yang termasuk kelompok ini adalah bahasa *Pascal*, *Basic*, dan *C*, *C++*.

Tentu saja pembagian ini tidak kaku. Bahasa-bahasa bertujuan khusus tidak berarti tidak bisa digunakan untuk aplikasi lain. *Cobol* misalnya, dapat juga digunakan untuk terapan ilmiah, hanya saja kemampuannya tentu saja terbatas. Yang jelas, bahasa-bahasa pemrograman yang berbeda dikembangkan untuk bermacam-macam kegunaan yang berbeda pula.

Berdasarkan "kedekatan" bahasa pemrograman apakah lebih condong ke bahasa mesin atau ke bahasa manusia, maka bahasa pemrograman juga dapat dikelompokkan atas dua macam:

1. **Bahasa tingkat rendah**. Bahasa jenis ini dirancang agar setiap instruksinya langsung dikerjakan oleh komputer, tanpa harus melalui *penerjemah (translator).* Contohnya adalah bahasa mesin (*machine language*). Bahasa mesin adalah sekumpulan kode biner (0 dan 1). Setiap perintah dalam bahasa mesin langsung "dimengerti" oleh mesin dan langsung dikerjakan. Bahasa tingkat rendah bersifat primitif, sangat sederhana, dan relatif sulit dipahami manusia. Bahasa *assembly* dimasukkan ke dalam kelompok ini karena notasi yang dipakai dalam bahasa ini merupakan bentuk "manusiawi" dari bahasa mesin, dan untuk melaksanakan instruksinya masih diperlukan penerjemahan (oleh *assembler*) ke dalam bahasa mesin. Bahasa tingkat rendah merupakan bahasa pemrograman generasi pertama yang pernah ditulis orang.
2. **Bahasa tingkat tinggi.** Bahasa jenis ini membuat program menjadi lebih mudah dipahami, lebih "manusiawi", dan lebih dekat ke bahasa manusia (bahasa Inggris terutama). Kelemahannya, program dalam bahasa tingkat tinggi tidak dapat langsung dilaksanakan oleh komputer. Ia perlu diterjemahkan terlebih dahulu oleh sebuah *translator bahasa* (yang disebut kompilator atau *compiler*) ke dalam bahasa mesin sebelum akhirnya dieksekusi oleh CPU. Tahapan pemrograman dan pelaksanaan program oleh komputer digambarkan pada Gambar 1.6. Contoh bahasa tingkat tinggi adalah *Pascal*, *PL/I*, *Ada*, *Cobol*, *Basic*, *Fortran*, *C*, *C++*, dan sebagainya.



Gambar 1.6
Tahapan pelaksanaan program oleh komputer.

Catatlah bahwa batas-batas penggolongan bahasa pemrograman itu tidak selalu jelas. Pengertian tentang apa yang dimaksud dengan bahasa tingkat tinggi seringkali berbeda pada beberapa buku. Ada buku yang mendefinisikan bahasa tingkat tinggi dari sudut pandang kemudahan pemakaiannya serta orientasinya yang lebih dekat ke bahasa manusia. Les Goldschlager [GOL88] menuliskan spektrum bahasa mulai dari bahasa tingkat tinggi (*Pascal*, *Ada*, *PL/I*, *Cobol*), bahasa tingkat menengah (Bahasa *Assembly*, *Basic*, *Fortran*),

sampai bahasa tingkat rendah (bahasa mesin). Kita tidak mendebatkan perbedaan cara pengelompokan bahasa pemrograman itu di sini, karena topik buku ini bukanlah mempelajari bahasa pemrograman tetapi belajar memprogram.

1.6 Notasi Algoritmik

Di dalam upabab 1.5 sudah dinyatakan bahwa notasi algoritmik dibuat independen dari spesifikasi bahasa pemrograman dan komputer yang mengeksekusinya. Notasi algoritmik ini dapat diterjemahkan ke dalam berbagai bahasa pemrograman. Analoginya sama dengan resep membuat kue. Sebuah resep dapat ditulis dalam bahasa manapun, bahasa Inggris, Perancis, Indonesia, Jepang, dan sebagainya. Apapun bahasanya, kue yang dihasilkan tetap sama, sebab algoritmanya sama (dengan catatan semua aturan pada resep diikuti). Mengapa bisa demikian? Karena setiap juru masak (yang merupakan pemroses) mampu melakukan operasi dasar yang sama, seperti mengocok telur, menimbang berat gula, dan sebagainya. Jadi, resep membuat kue tidak terikat pada bahasa dan juru masak yang mengerjakannya.

Demikian pula halnya komputer. Meskipun setiap komputer berbeda teknologinya, tetapi secara umum semua komputer dapat melakukan operasi-operasi dasar dalam pemrograman seperti operasi pembacaan data, operasi perbandingan, operasi aritmetika, dan sebagainya. Perkembangan teknologi komputer tidak mengubah operasi-operasi dasar itu, yang berubah hanyalah kecepatan, biaya, atau tingkat ketelitian. Pada sisi lain, setiap program dalam bahasa tingkat tinggi selalu diterjemahkan kedalam bahasa mesin sebelum akhirnya dikerjakan oleh *CPU*. Setiap instruksi dalam bahasa mesin menyajikan operasi dasar yang sesuai, dan menghasilkan efek netto yang sama pada setiap komputer.

Yang perlu dicatat adalah bahwa notasi algoritmik bukan notasi bahasa pemrograman, sehingga siapa pun dapat membuat notasi algoritmik yang berbeda. Hal yang penting mengenai notasi tersebut adalah ia mudah dibaca dan dimengerti. Selain itu, meskipun notasi algoritmik bukan notasi baku sebagaimana pada notasi bahasa pemrograman, namun ketaatasasan terhadap notasi perlu diperhatikan untuk menghindari kekeliruan.

Di bawah ini saya kemukakan beberapa notasi yang digunakan untuk menulis algoritma. Masalah yang dijadikan contoh adalah menghitung pembagi bersama terbesar dengan algoritma Euclidean (lihat bab 1.2).

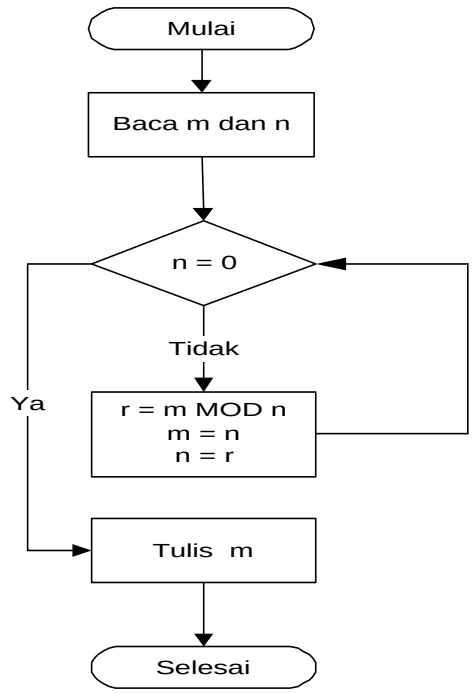
1. Notasi I: menyatakan langkah-langkah algoritma dengan untaian kalimat deskriptif.

```
PROGRAM Euclidean
Diberikan dua buah bilangan bulat tak-negatif m dan n (m ³ n). Algoritma
Euclidean mencari pembagi bersama terbesar, gcd, dari kedua bilangan
tersebut, yaitu bilangan bulat positif terbesar yang habis membagi m dan n.

ALGORITMA:
1. Jika n = 0 maka
     m adalah jawabannya;
     stop.
   tetapi jika n ≠ 0,
     lanjutkan ke langkah 2.
2. Bagilah m dengan n dan misalkan r adalah sisanya.
3. Ganti nilai m dengan nilai n dan nilai n dengan nilai r, lalu ulang
   kembali ke langkah 1.
```

Dengan notasi bergaya kalimat ini, deskripsi setiap langkah dijelaskan dengan bahasa yang gamblang. Proses diawali dengan kata kerja seperti 'baca', 'hitung', 'bagi', 'ganti', dan sebagainya, sedangkan pernyataan kondisional dinyatakan dengan 'jika … maka …'.
Secara umum, notasi ini ini relatif sukar diterjemahkan langsung ke dalam notasi bahasa pemrograman.

2. Notasi II: menggunakan diagram alir (*flow chart*)



Keterangan:

1. MOD adalah operator pembagian bilangan bulat yang menghasilkan sisa hasil pembagian. Contohnya, 9 MOD 2 = 1 karena 9 dibagi 2 = 4 dan memberikan sisa = 1.

2. Di dalam diagram alir di atas ditambahkan instruksi pembacaan nilai *m* dan *n*.
   Diagram alir populer pada awal era pemrograman dengan komputer (terutama dengan bahasa *Basic*, *Fortran*, dan *Cobol*). Sampai saat ini diagram alir masih banyak digunakan orang untuk menjelaskan proses. Namun, diagram alir lebih mengambarkan aliran instruksi di dalam program secara visual ketimbang memperlihatkan struktur program. Kotak empat persegi panjang menyatakan proses, sedangkan pernyataan kondisional dinyatakan dengan bentuk intan (*diamond*). Notasi algoritmik dengan diagram alir cocok untuk masalah yang kecil, namun tidak cocok untuk masalah yang besar karena membutuhkan berlembar halaman kertas. Selain itu, pengkonversian notasi algoritma ke notasi bahasa pemrograman juga cenderung relatif sukar.

3. Notasi III: menggunakan *pseudo-code*
   Pseudocode (*pseudo* artinya semu atau tidak sebenarnya) adalah notasi yang menyerupai notasi bahasa pemrograman tingkat tinggi, khususnya Bahasa *Pascal* dan *C*. Hasil pengamatan memperlihatkan bahwa bahasa pemrograman umumnya mempunyai notasi yang hampir mirip untuk beberapa instruksi, seperti notasi *if-then-else*, *while-do*, *repeat-until*, *read*, *write*, dan sebagainya. Berdasarkan pengamatan tersebut, maka beberapa penulis buku algoritma, termasuk penulis buku ini, mendefinisikan notasi algoritma yang disebut *pseudo-code* itu. Tidak seperti bahasa pemrograman yang direpotkan dengan tanda titik koma (*semicolon*), indeks, format keluaran, kata-kata khusus, dan sebagainya, sembarang versi *pseudocode* dapat diterima asalkan perintahnya tidak membingungkan pembaca. Keuntungan menggunakan notasi *pseudo-code* adalah kemudahan mengkonversinya (lebih tepat disebut mentranslasi) ke notasi bahasa pemrograman, karena terdapat korespondensi antara setiap *pseudo-code* dengan notasi bahasa pemrograman. Korespondensi ini dapat diwujudkan dengan tabel translasi dari notasi algoritma ke notasi bahasa pemrograman apa pun.

Algoritma Euclidean kita tulis kembali dengan menggunakan notasi *pseudo-code* yang didefinisikan oleh penulis buku ini.

```
PROGRAM Euclidean
Program untuk mencari gcd dari dua buah bilangan bulat tak-negatif m dan n (m
³ n). gcd dari m dan n adalah bilangan bulat positif terbesar yang habis
membagi m dan n.

DEKLARASI:
  m, n : integer   { bilangan bulat yang akan dicari pbt-nya}
  r    : integer   { sisa hasil bagi }

ALGORITMA:
   read(m,n)       { m ³ n}
   while n ≠ 0 do
     r ← m MOD n    { hitung sisa hasil pembagian }
     m ← n
     n ← r
   endwhile
   { kondisi selesai pengulangan: n = 0, maka gcd(m,n) = m }

   write(m)
```

Kata-kata yang digarisbawahi menyatakan kata-kata kunci untuk setiap notasi *pseudo-code* yang digunakan. Arti dari setiap notasi *pseudo-code* di atas akan anda temukan pada bab-bab selanjutnya di dalam buku ini.

1.7 Pemrograman Prosedural

Algoritma berisi urutan langkah-langkah penyelesaian masalah. Ini berarti algoritma adalah menggambarkan proses yang prosedural.

Definisi prosedur menurut Kamus Besar Bahasa Indonesia (KBBI) :

prosedur:
1. tahap tahap kegiatan untuk menyelesaikan suatu aktivitas;
2. metode langkah demi langkah secara eksak dalam memecahkan suatu masalah (KBB1 1988).

Pada pemrograman prosedural, program dibedakan antara bagian data dengan bagian instruksi. Bagian instruksi terdiri atas runtunan (*sequence*) instruksi yang dilaksanakan satu per satu secara berurutan oleh sebuah pemroses. Alur pelaksanaan instruksi dapat berubah karena adanya pencabangan/kondisional. Data yang disimpan di dalam memori dimanipulasi oleh instruksi secara beruntun. Kita katakan bahwa tahapan pelaksanaan program mengikuti pola beruntun atau prosedural. Paradigma pemrograman seperti ini dianamakan **pemrograman prosedural**.

Bahasa-bahasa tingkat tinggi seperti *Cobol*, *Basic*, *Pascal*, *Fortran*, dan *C* mendukung kegiatan pemrograman prosedural, karena itu mereka dinamakan juga bahasa prosedural.

Selain paradigma pemrograman prosedural, ada lagi paradigma yang lain yaitu **pemrograman berorientasi objek** (*Object Oriented Programming* atau OOP). Paradigma pemrograman yang disebutkan terakhir ini merupakan *trend* baru dan sangat populer akhir-akhir ini. Pada paradigma OOP, data dan instruksi dibungkus (*encapsulation*) menjadi satu. Kesatuan ini disebut **kelas** (*class*) dan instansiasi kelas pada saat *run-time* disebut **objek** (*object*). Data di dalam objek hanya dapat diakses oleh instruksi yang ada didalam objek itu saja.

Paradigma pemrograman yang lain adalah **pemrograman fungsional**, **pemrograman deklaratif**, dan **pemrograman konkuren**. Buku ini hanya menyajikan paradigma pemrograman prosedural saja. Paradigma pemrograman yang lain di luar cakupan buku.

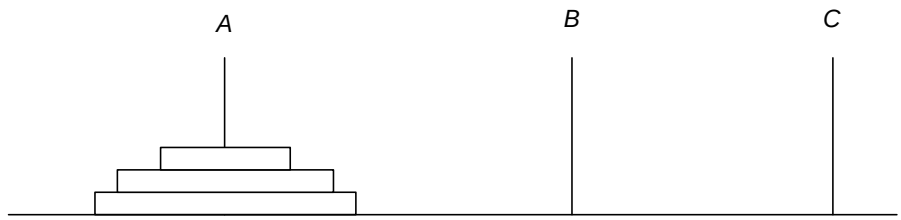
Soal Latihan Bab 1

1. Tuliskan beberapa contoh algoritma yang lain dalam kehidupan sehari-hari. Tuliskan juga beberapa contoh langkah di dalam algoritmanya.

2. Tiga pasang suami istri yang sedang menempuh perjalanan sampai ke sebuah sungai. Di situ mereka menemukan sebuah perahu kecil yang hanya bisa membawa tidak lebih dari dua orang setiap kali menyeberang. Penyeberangan sungai dirumitkan oleh kenyataan bahwa para suami sangat pencemburu dan tidak mau meninggalkan istri-istri mereka jika ada lelaki lain. Tulislah algoritma untuk menunjukkan bagaimana penyeberangan itu bisa dilakukan.

3. Misalkan terdapat dua buah ember, masing-masing mempunyai volume 5 liter dan 3 liter. Tuliskan algoritma untuk memperoleh air sebanyak 1 liter dengan hanya menggunakan kedua ember tersebut.

4. Tiga buah cakram yang masing-masing berdiameter berbeda mempunyai lubang di titik pusatnya. Ketiga cakram tersebut dimasukkan pada sebuah batang besi *A* sedemikan sehingga cakram yang berdiameter lebih besar selalu terletak di bawah cakram yang berdiameter lebih kecil (Gambar 1.5). Tulislah algoritma untuk memindahkan seluruh cakram tersebut batang besi B; setiap kali hanya satu cakram yang boleh dipindahkan, tetapi pada setiap perpindahan tidak boleh ada cakram yang lebih besar berada di atas cakram kecil. Batang besi *C* dapat dipakai sebagai tempat peralihan dengan tetap memegang aturan yang telah disebutkan.



5. Pada peristiwa pemilihan kepala desa (kades), setiap warga yang mempunyai hak pilih memilih satu di antara 4 calon kades. Kartu suara memuat foto dan nomor urut kades. Warga mencoblos calon kades yang dipilihnya, lalu memasukkan kartu suara ke dalam sebuah kotak. Setelah pemungutan suara usai, kegiatan selanjutnya adalah menghitung jumlah suara untuk masing-masing calon. Untuk menghitungnya, panitia tidak menggunakan tabel *cayley* seperti yang biasa dlakukan orang, tetapi menyediakan empat buah kotak kosong (yang merepresentasikan 4 calon kades). Satu per satu kartu suara diambil dan dibaca. Setiap kali kartu suara berisi coblosan nomor satu, maka sebutir batu kecil dimasukkan ke dalam kotak 1. Begitu pula setiap kali kartu suara berisi coblosan nomor dua, maka sebutir batu kecil dimasukkan ke dalam kotak 2. Hal yang sama juga dilakukan untuk kartu yang berisi coblosan nomor 3 dan empat. Demikian seterusnya sampai semua kartu suara habis dibaca. Akhirnya, jumlah batu di dalam setiap kotak menyatakan jumlah suara yang diraih oleh setiap calon kades. Tulislah algoritma untuk menghitung jumlah suara untuk masing-masing calon kades dengan metode perhitungan yang unik ini. Asumsikan bahwa semua suara adalah sah (tidak ada golput).

6. Di manakah letak kesalahan lojik algoritma memutar kaset *tape recorder* di bawah ini:

```
PROGRAM Memutar Kaset Tape Recorder
Program memutar jaset dengan tape recorder.

ALGORITMA:
1.  Pastikan tape recorder berada dalam keadaan POWER ON.
2.  Tekan tombol PLAY.
3.  Masukkan kaset ke dalam tape recorder.
```