Belajar Pemrograman untuk *Damis*

Oleh :

dwi sakethi

PENGRAJIN SISTEM INFORMASI

0816-403 432

dwijim@unila.ac.id

http://www.dwijim.wordpress.com

tulisan niki dipun serat nganggé LaTeX

mBandar Lampung 2011

Daftar Isi

Belajar Pemrograman untuk *Nyubis*

dwi sakethi
dwijim@unila.ac.id,

18 Pebruari 2012

Bagian I

Pendahuluan

Bab 1

Komputer dan Program

1.1 Membuat Program : Gampang atau Tidak Sulit ?

Membuat program ...? Sama seperti naik sepeda atau permainan tradisional egrang. Untuk orang yang sudah terbiasa naik sepeda atau naik egrang, maka pekerjaan itu bukan merupakan sesuatu yang sulit. Tapi untuk orang yang baru belajar, alangkah sulitnya membuat program komputer. Membuat program merupakan proses mentransfer proses yang ada di dalam benak manusia ke dalam perintah-perintah yang dimengerti oleh komputer. Kesulitannya adalah bahwa manusia dan komputer mempunyai pola pikir, kemampuan yang berbeda. Apa yang sangat mudah dikerjakan oleh manusia, ketika proses itu diterjemahkan ke dalam perintah-perintah yang bisa dimengerti oleh komputer, menjadi sesuatu yang perlu latihan dan pemikiran. Contoh : Terdapat suatu bilangan yang terdiri dari 3 digit angka, misal : 137. Baliklah bilangan itu! Berapa angka yang diperoleh ? Dengan sangat mudah, bisa diperoleh bahwa jawabannya adalah 731. Nah kalau ada seseorang yang bisa mendapatkan jawaban tersebut, tentu saja orang itu tahu dan mengerti prosesnya. Masalahnya sekarang, bagaimana proses-proses itu diberikan kepada komputer dan komputer bisa menjalankan proses itu. Pada akhirnya komputer akan memberikan hasil proses pembalikan angka tadi.

Contoh lain misalnya mencari bilangan terbesar dari sekumpulan bilangan. Misal ada bilangan-bilangan : 11, 0, 6, 68, 3, 40, 50. Dengan mudah juga, bisa dijawab untuk masalah tersebut maka bilangan terbesarnya adalah 68. Sama seperti pada masalah sebelumnya, jika ada seseorang yang bisa memecahkan masalah ini, berarti orang tersebut tahu prosesnya sehingga bisa mendapatkan jawaban yang diinginkan. Bagaimana proses yang ada di otaknya bisa dipindah ke komputer.

Pembelajaran pemrograman komputer biasanya lebih banyak ditekankan pada pengajaran suatu bahasa pemrograman, misalnya Pascal. Dengan kata lain, orang mempelajari perintah-perintah yang ada pada suatu bahasa pemrograman. Model seperti ini, mungkin lebih cocok untuk orang-orang sudah sudah terbiasa membuat program. Sehingga tinggal *memberikan tahu* (memberitahukan) perintah-perintah yang bisa digunakan untuk menyelesaikan masalah-masalah yang ada. Pada pembelajaran pemrograman komputer ini, diberikan dengan nuansa yang sedikit berbeda. Pada buku ini, mahasiswa ditekankan untuk memahami proses transfer dari pola pikir manusia ke dalam perintah-perintah yang dipahami oleh komputer. Jadi pembelajaran bukan kepada mengajarkan Pascal atau C atau bahasa pemrograman lainnya. Pascal atau C menjadi alat (*tool*) yang digunakan saja, bukan sebagai fokus pembelajaran.

Sekian kali memberikan materi pemrograman (apalagi ini menjadi materi awal untuk pemrograman), hasilnya masih perlu dikembangkan lagi. Mereka yang belajar pemrograman masih kesulitan menterjemahkan proses yang ada di dalam pola pikirnya kepada sesuatu yang

bisa dimengerti oleh komputer. Oleh karena itu, pada materi ini dicoba untuk menggunakan pendekatan yang berbeda. Semoga berhasil ... berhasil ... (Dora Emon).

1.2 Ketrampilan yang Dibutuhkan dalam *Programming*

Menulis atau membuat program bukanlah pekerjaan yang mudah, akan tetapi juga tidak cukup alasan untuk mengatakannya sebagai sesuatu yang sulit. Dalam menulis program, tidak membutuhkan bakat alam atau ketrampilan bawaan, seperti misalnya menyanyi atau melukis. Dalam menyanyi atau melukis, seseorang yang tidak memiliki bakat bawaan akan terasa sulit untuk melakukannya. Orang yang belajar menyanyi dengan membawa bakat alami akan lebih baik hasilnya dibandingkan dengan orang yang tidak memiliki bakat alam. Namun tidak demikian halnya dengan dengan menulis program komputer. Untuk bisa menulis program dengan baik, tidak dibutuhkan adanya bakat alam. Ada 4 ketrampilan yang dibutuhkan di sini, yaitu :

1. Perhatian terhadap hal-hal yang rinci.
   Komputer itu barang yang bodoh tidak dapat dipercaya (*incredibly stupid*). Pemrogram tidak dapat hanya menjelaskan 3/4 bagian dari proses dan kemudian berkata, "Kamu ngerti kan dengan apa yang aku mauin ?". Selanjutnya komputer akan mengerjakan dengan sendirinya apa-apa yang kurangnya. Jadi program harus ditulis sampai dengan hal-hal yang rinci. Misalnya jika dalam penulisan program mesti ada keluaran yang ditampilkan ke layar, maka komputer harus diperintahkan untuk melakukan hal itu. Karena variabel harus dideklarasikan, maka variabel pun harus dideklarasikan. Komputer tidak dapat dengan sendirinya mendeklarasikan variabel yang digunakan. Ibaratnya ketika komputer berkunjung ke rumah seseorang, jika komputer tidak diberi perintah untuk menekan bel, maka komputer tidak akan pernah menekan bel, meskipun belnya sudah ada di depan mata.

2. Bersikap bodoh.
   Sebagaimana tadi sudah disebutkan bahwa tadi komputer bodoh, tidak dapat dipercaya. Komputer hanya mengerjakan apa yang betul-betul diperintahkan kepada mereka, tidak lebih dan tidak kurang. Mungkin komputer bisa diibaratkan seperti malaikat, dimana malaikat tidak pernah melawan apa yang diperintahkan dan tidak pernah mengerjakan apa yang tidak diperintahkan. Komputer tidak bisa diberikan sebungkus mie rebus dan kemudian disuruh membaca petunjuk yang ada dibungkusnya, maka kemudian komputer akan membuat mie rebus.

3. Ingatan yang baik.
   Banyak hal yang harus diingat ketika membuat program. Mulai dari aturan penulisan, fungsi dan perintah yang digunakan, parameter untuk suatu fungsi yang sudah dibuat, nama-nama variabel yang digunakan dan lain sebagainya. Dengan ingatan yang baik, tentu saja akan sangat membantu dalam penulisan program. Misalkan perintah untuk mencetak apakah dengan `writeln`, `print` atau `echo` ... ?

4. Kemampuan untuk berfikir secara abstrak dan mampu memikirkan suatu hal dalam beberapa model tingkatan.
   Ini merupakan kemampuan yang paling penting di dalam melakukan aktifitas pemrograman. Membayangkan sesuatu yang sepertinya abstrak atau memikirkan sesuatu yang belum terjadi. Misalnya dalam proses melakukan pengurutan bilangan. Proses pengurutan bilangan merupakan hal yang sangat mudah dilakukan secara manual (jika datanya sedikit). Nah ... bagaimana proses yang bisa dilakukan sendiri oleh manusia

ini, sekarang dialihkan dalam perintah-perintah yang bisa dipahami oleh komputer. Sebagai contoh lain, ketika dengan pola fikir manusia, bagaimanan seseorang bisa dengan mudah membalikkan suatu angka. Untuk angka 137, komputer tidak akan memahami perintah, "Ambil angka terakhir!". Akan tetapi komputer dapat memahami jika diberikan perintah ambil digit ketiga. Atau juga jika diberikan perintah "satuan = 137 mod 10". Jadi untuk mendapatkan angka atau karakter 7, paling tidak ada dua perintah yang bisa diberikan kepada komputer.

1.3 Kemampuan Komputer

Sebelum membuat atau mengembangkan program, tentu penting untuk memahami sifat-sifat yang ada pada komputer. Karena program itu sendiri nantinya akan dikerjakan oleh komputer. Karena program itu akan dikerjakan oleh komputer, maka perintah-perintah yang akan diberikan haruslah perintah-perintah yang dimengerti oleh komputer.

Beberapa kemampuan yang dimiliki oleh komputer :

1. Menyimpan suatu nilai.
   Komputer mampu menyimpan suatu nilai bahkan kemudian mengganti nilai itu dengan nilai lain dan kemudian memberi tahu kepada pemakai (manusia) tentang nilai yang disimpannya. Misalkan dalam suatu percakapan terjadi dialog seperti ini :
   Syah : Siapa namamu ?
   Bejo : Nama saya Bejo Sutaqwa
   Syah : Alamatmu di mana ?
   Bejo : Tanah Miring, Kotabumi.
   Salah satu bentuk penterjemahan dari dialog ini ke dalam bahasa yang dimengerti oleh komputer :

   ```
   nama   = 'Bejo Sutaqwa';
   alamat = 'Tanah Miring, Kotabumi';
   ```

   Pada satu saat, untuk satu tempat maka hanya ada satu nilai yang bisa disimpan. Jika suatu tempat menyimpan nilai, kemudian diisi dengan nilai lain, maka nilai sebelumnya akan tertimpa. Misalkan jika kemudian terjadi dialog :
   Syah : Sekarang kamu tinggal dimana ?
   Bejo : Kali Cinta, Kotabumi
   Dalam instruksi komputer, perintahnya misalnya seperti ini :

   ```
   alamat = 'Kali Cinta, Kotabumi';
   ```

   Maka komputer tidak tahu lagi dimana alamat lama jika komputer tidak diberikan perintah untuk menyimpan alamat yang lama.

2. Melakukan perhitungan matematika.
   Kemampuan mendasar dari komputer adalah melakukan penghitungan matematis. Dari angka yang kecil sampai dengan angka yang besar (pada tahapan ini belum dibahas berapa angka terbesar yang bisa dihitung oleh komputer). Penghitungan matematis di sini, sementara untuk penghitungan dasar saja yaitu penjumlahan, pengurangan, pembagian dan perkalian. Bagaimana dengan pangkat misalnya ? Ya ... ini belum dibahas di sini.

3. Melakukan pembandingan dua buah nilai.
   Apakah suatu nilai dibandingkan nilai satunya lebih kecil, sama atau lebih besar? Pembandingan ini bisa dilakukan baik untuk bilangan ataupun tulisan. Contoh :
   Ada dua orang anak kembar. Masing-masing ditanya dengan dialog seperti ini :
   Ibu : Nama kamu siapa ? (Kepada anak 1)
   Yasmin : Nama saya Yasmin Nurul Barkah
   Ibu : Nama kamu siapa ? (Kepada anak 2)
   Wardah : Nama saya Wardah Nurul Rahmah
   Jika Anda ditanya apakah nama anak 1 dengan nama anak 2 sama? Tentu saja kita bisa dengan mudah menjawab nama kedua anak itu berbeda. Jika diterjemahkan ke dalam bahasa yang bisa dimengerti oleh komputer, maka bisa dituliskan dalam perintah seperti ini :

```
nama_anak_1 = 'Yasmin Nurul Barkah';
nama_anak_2 = 'Wardah Nurul Rahmah';
if (nama_anak_1==nama_anak_2)
   { printf("Nama sama ");}
else
   { printf("Nama berbeda ");}
```

Itulah kemampuan-kemampuan dasar yang dimiliki oleh komputer. Kemampuan lain dibangun dari kemampuan dasar ini. Kehebatan komputer adalah bahwa komputer bisa melaksanakan kemampuan tadi tanpa rasa jenuh atau akan salah. Jika manusia diberikan perintah untuk melakukan penjumlahan selama 3 hari 3 malam, mungkin lama-lama hasil penghitungannya akan salah. Tidak demikian halnya dengan komputer.

Bagian II

Membuat Program

Bab 2

Dasar-Dasar Membuat Program

2.1 Masalah Mendasar

Dalam banyak proses pembelajaran, ada masalah mendasar yang selalu muncul. Masalah tersebut adalah kemauan untuk mencoba sendiri dan melakukan kesalahan. Dalam proses belajar, melakukan kesalahan merupakan hal biasa dan wajar, atau bahkan suatu keharusan. Oleh karena itu, satu hal yang amat penting dalam proses belajar ini adalah selalu dan selalu mencoba sendiri mencari penyelesaian dari masalah yang diberikan. Hal ini akan semakin melatih kemampuan dari siapa saja yang sedang belajar. Kualitas tidak muncul dengan tiba-tiba. Kualitas diawali dengan kuantitas yang jelek. Seorang pemain sepakbola profesional, sebelum memiliki tendangan melengkung yang aneh, tentu saja diawali dengan melakukan ratusan atau ribuan tendangan yang salah sasaran terus. Namun seiring dengan kuantitas tendangan yang salah tadi, kualitas akan semakin meningkat. Tulisan seseorang yang begitu bagus, setidaknya diawali dengan sekian banyak tulisan-tulisan yang gagal. Namun sekali lagi, itu akan semakin mengasah kemampuannya, hingga pada akhirnya melahirkan kemampuan menulis yang sangat memikat. Gambaran lain, misalnya ada seseorang yang mampu mengetik menggunakan papan ketik dengan begitu cepat tanpa melihat papan ketik tentu dimulai dengan banyak kesalahan di dalam pengetikannya. Kemampuan-kemampuan itu tidak demikian saja muncul. Jadi . . . perlu sekali lagi disampaikan tentang betapa pentingnya kemauan untuk selalu mencoba mencari sendiri solusi dari masalah-masalah yang diberikan.

Membuat program seperti seni. Sangat terbuka dengan berbagai banyak hal. Tentunya akan terasa aneh, ketika ada dua orang diminta menggambar, ternyata hasilnya gambarnya sama persis!

2.2 Memberikan Instruksi kepada Komputer

Perintah seperti apa *sih* yang bisa dimengerti oleh komputer? Komputer tidak bisa diperintah dengan perintah yang global, misalnya tolong buatkan saya secangkir teh manis. Mengapa? Komputer tidak memahami teh manis itu isinya berapa gelas air putih, berapa sendok gulanya, mengambilnya dimana? Untuk contoh masalah membalikkan angka, jika dengan bahasa manusia, bisa saja instruksinya adalah pindah angka di belakang ke depan dan angka di depan dipindah ke belakang. Bisa? Ingat komputer, tidak tahu mana yang di depan dan mana yang di belakang?

Komputer tidak bisa diperintah dengan instruksi yang sifatnya global atau tindak rinci. Komputer harus diberikan perintah sedetail-detailnya. Sebagai contoh, misalkan ada dua buah wadah yang mempunyai isi masing-masing. Wadah pertama berupa gelas, berisi solar. Kemudian wadah kedua, berupa ember berisi bensin. Kemudian isi dari wadah itu

akan ditukar. Jika proses ini diberikan kepada manusia, tentu saja akan sangat mudah untu dikerjakan. Cukup dengan kata-kata, "Tolong isi kedua wadah itu ditukar!", maka manusia bisa menterjemahkan perintah itu ke dalam suatu proses untuk memindah isi wadah pertama akan pindah ke wadah kedua, dan demikian juga sebaliknya. Sebaliknya, jika perintah itu diberikan kepada komputer, maka komputer itu akan bertanya, "Menukar itu apa?" Ingat bahwa dari kemampuan yang dimiliki oleh komputer, komputer tidak mengenal istilah menukar. (Untuk materi ini, perlu alat peraga 3 buah gelas dan 2 buah cairan yang berbeda).

Supaya komputer bisa menyelesaikan masalah menukar isi wadah tersebut, maka perintah yang diberikan bukanlah instruksi menukar isi wadah tersebut, tetapi perintahnya menjadi seperti berikut :

1. Siapkan 3 wadah.
2. Wadah 1 diisi degan solar.
3. Wadah 2 diisi dengan bensin.
4. Wadah 3 diisi dengan isi wadah 1 (solar).
5. Wadah 1 diisi dengan isi wadah 2 (bensin).
6. Wadah 2 diisi dengan isi wadah 3 (solar).

Maka sekarang wadah 1 berisi bensin dan wadah 2 berisi solar. Kondisi seperti ini, artinya isi wadah pertama sudah ditukar dengan isi wadah 2. Namun jika melihat instruksi yang diberikan kepada komputer, di sana tidak ada sama sekali perintah menukar.

Contoh lain misalnya dalam masalah menentukan nomor cantik. Dalam masalah ini, suatu nomor yang terdiri dari 3 digit angka dan tidak diawali dengan angka 0 (nol), akan disebut sebagai nomor cantik jika memenuhi salah satu dari kriteria berikut :

1. Selisih antara satu digit dengan digit berikutnya adalah 1. Contoh : 123, 678, atau 987.
2. Ketiga digit bilangan bernilai sama. Misalnya : 222, 666, 999 dan sebagainya.
3. Digit pertama dan digit ketiga bernilai sama. Misalnya : 212, 121, 909 dan sebagainya.

Dengan ketentuan-ketentuan seperti tersebut, akan sangat mudah jika masalah penggolongan nomor cantik ini diserahkan kepada makhluk yang bernama manusia. Misalkan apakah nomor 345 adalah nomor cantik? Tentu dengan mudah jawabannya adalah nomor cantik. 306 - bukan nomor cantik. 595 - nomor cantik. Demikian seterusnya. Nah ... bagaimana jadinya jika sekarang apa yang ada di otak manusia itu ditransfer pola pikirnya ke dalam proses yang dimengerti oleh komputer. Sekali lagi perlu ditekankan bahwa komputer tidak bisa memahami istilah bilangan paling belakang, bilangan yang paling depan, atau bilangan yang di tengah? Komputer akan menanyakan apa itu bilangan yang paling belakang. Oleh karena itu, komputer harus dibimbing dengan perintah yang rinci sehingga pada akhirnya komputer memahami apa itu bilangan yang paling belakang. OK ... kita lihat di TKP ... !

Kemampuan komputer salah satunya adalah melakukan perhitungan. Ibaratnya komputer tidak akan pernah salah dalam melakukan perhitungan ini. Kemudian komputer bisa menyimpan nilai-nilai yang didapat dari hasil perhitungan tadi. Kemampuan inilah yang akan dieksplorasi untuk menyelesaikan masalah yang sudah diberikan, yaitu pengecekan nomor cantik. Ada satu operasi matematis yaitu mod yang bermakna sisa pembagian. Misalnya 11 mod 3 = 2, 1567 mod 1000 = 567 demikian seterusnya.

Nama lain untuk digit paling belakang adalah angka satuan. Angka satuan ini lebih dimengerti oleh komputer dari pada digit paling belakang. Bagaimana cara mendapatkan angka satuan ini ... Dengan operasi sisa pembagian 10 maka angka satuan bisa didapatkan. Misalkan 345 mod 10 = 5 (karena 10x34=340, jadi sisanya tinggal 5).

Selain perintah harus detail, perhatikan bahwa pemakai mesti mengganggap komputer merupakan makhluk yang tidak mampu melakukan apa-apa, akan tetapi bisa diberikan perintah (asal sesuai), lihat kembali kemampuan komputer.

2.3 Perhitungan Matematis Biasa

Linudin (Udin yang suka Linux) pada suatu hari kedatangan seorang klien yang ingin berkonsultasi, bernama Dolarudin (Udin yang banyak uangnya). Pak Dolarudin ingin menjual suatu berkas (*file*). Harga dihitung berdasarkan ukuran bahwa 1 byte disepakati harganya sebesar Rp. 11.600,-. Pak Dolarudin hanya mengetahui bahwa berkas yang akan dijualnya berukuran 68 MB. Apakah itu berarti harganya 11600 x 68 x 1000 x 1000 ? Baiklah ... lihat saja ke TKP ...!

```
Linu : Gimana kabarnya Dol ?
Dola : Alhamdulillah ... biasanya ... aku masih seperti yang dulu.
Linu : Bapak kamu kemarin makan-makan di Kampung Nyenyet ya ... ?
Dola : Kok kamu tahu sih ...
Linu : Karena beliau telah membayariku billing-ku di sana ...
       He he he ...
Linu : Ngomong-ngomong ... Kalau boleh tidak tahu nih ...
       Apa gerangan maksud kedatangannya ke sini ?
Dola : Aku mau menjual satu berkas.
Linu : Harganya ?
Dola : 1 byte harganya Rp. 11.600,-
Linu : Nah ... berkas itu ukuranya berapa ?
Dola : 68 MB.
Linu : Ehm ... 1 MB = 1024 KB dan 1 KB = 1024 B
       Jadi harga berkasnya adalah 68 x 1024 x 1024 x Rp. 11.600.
       Hasil akhir adalah Rp. 827.116.748.800 (827 milyar 116 juta
       748 ribu 800 rupiah).
Linu : Berkas apaan sih kok harganya sampai segitunya ?
Dola : Gambar tumpukan apel Washington dan apel Malang !!!!!!!!!
```

Kira-kira dialog yang terjadi bakal seperti itu. Proses ini masih belum bisa diterjemahkan langsung ke dalam instruksi yang bisa dimengerti sistem komputer. Mengapa ? Karena dialognya masih ada yang perlu untuk dihaluskan lagi. Jawaban yang diberikan oleh Pak Dolarudin harus merupakan jawaban singkat. Penghalusan dialog menjadi seperti ini :

```
Linu : Berapa harga per byte-nya ?
Dola : 11600
Linu : Berapa ukuran berkasnya (MB) ?
Dola : 68
Dola : ... melakukan proses perhitungan dalam pikirannya.
       Harganya adalah Rp. 827116748800
```

Dari bentuk dialog terakhir ini, dialog bisa dikonversi ke dalam bentuk perintah-perintah yang bisa dimengerti oleh komputer.

2.4 Melakukan Perbandingan

Bab 3

Dasar-Dasar Algoritma

3.1 Pengertian Algoritma

Langkah-langkah detail atau rinci yang ditujukan untuk bisa dipahami oleh komputer guna menyelesaikan suatu masalah disebut dengan algoritma. Dari mana muncul istilah algoritma? Istilah algoritma berasal dari nama ilmuwan muslim Arab bernama Abu Ja'far Mohammed ibn Musa al Khowarizmi (tahun 790-840). Beliau juga dikenal sebagai Bapak Aljabar. Perhatikan contoh dialog berikut:

```
Linu : Gimana kabarnya Dol ?
Dola : Alhamdulillah ... biasanya ... aku masih seperti yang dulu.
Linu : Ehm ngomong-ngomong ... Bapak kamu kemarin makan-makan di
       Kampung Nyenyet ya ... ?
Dola : Kok kamu tahu sih ...
Linu : Karena beliau telah membayariku billing-ku di sana ...
       He he he ...
Linu : Btw ... Kalau boleh tidak tahu nih ...
       Apa gerangan maksud kedatangan ki sanak ke sini ?
Dola : Aku mau menjual satu berkas.
Linu : Harganya ?
Dola : 1 byte harganya Rp. 11.600,-
Linu : Nah ... berkas itu ukuranya berapa ?
Dola : 68 MB.
Linu : Ehm ... 1 MB = 1024 KB dan 1 KB = 1024 B
       Jadi harga berkasnya adalah 68 x 1024 x 1024 x Rp. 11.600.
       Hasil akhir adalah Rp. 827.116.748.800 (827 milyar 116 juta
       748 ribu 800 rupiah).
Linu : Berkas apaan sih kok harganya sampai segitunya ?
Dola : Gambar tumpukan apel Washington dan apel Malang !!!!!!!!!
```

Jika inti dari masalah tersebut dituangkan ke dalam bentuk algoritma, maka akan didapatkan urutan proses seperti berikut:

1. Menanyakan harga per byte.
2. Menanyakan ukuran berkasnya.
3. Menghitung konversi dari ukuran MB ke byte.
4. Menghitung harga dengan nilai harga adalah harga per byte dikalikan dengan hasil konversi ukuran berkas dalam byte.

Algoritma terkadang disampaikan dalam bentuk pseudokode. Pseudokode adalah bentuk algoritma yang dituliskan dengan berbagai notasi untuk menyederhanakan bentuk kalimat yang ditulis dalam bahasa manusia. Inti dari masalah yang disampaikan dalam tulisan sebelumnya adalah:

```
Linu : Berapa harga per byte-nya ?
Dola : 11600
Linu : Berapa ukuran berkasnya (MB) ?
Dola : 68
Dola : ... melakukan proses perhitungan dalam pikirannya.
       Harganya adalah Rp. 827116748800
```

Penyelesaian dari masalah ini bisa dilihat pada contoh program berikut:

```
/* ------------------------------------------------------------------------
  program untuk menghitung harga berkas
  nama file : harga-berkas.cpp
  editing   : 13 oktober 2011
   ------------------------------------------------------------------------ */

#include <iostream>
// menyertakan berkas yang berisi fungsi-fungsi tertentu dan sudah dikompilasi

using namespace std;
// ciri penulisan c++ pada compiler modern, std padanan dari global

int main()
// program dalam bahasa C++ paling tidak terdiri dari satu fungsi
// yaitu fungsi main()

  {

  int harga_per_byte;
  int ukuran_berkas;
  // mendeklarasikan pengenal (variable) bertipe integer

  double harga_total;
  // mendeklarasikan pengenal (variable) bertipe long integer

  cout  << "Harga per byte Rp.    : ";
  // menampilkan ke layar pertanyaan ke pemakai

  cin  >> harga_per_byte;
  // memasukkan data ke variable

  cout  << "Berapa ukuran berkas (MB) : ";
  // menampilkan ke layar pertanyaan ke pemakai

  cin  >> ukuran_berkas;
  // memasukkan data ke variable

  harga_total = harga_per_byte*1024*1024*ukuran_berkas;
```

```
  // menghitung harga

  cout  << "Harga berkas Rp. " << harga_total << "\n";
  // menampilkan ke layar hasil perhitungan

  cout  << "\n";
  return 11;
  // karena int main() jadi harus return bilangan bulat
}
```

3.2 Struktur Dasar Algoritma

Struktur dasar algoritma ada 3 (Kadir, 2005) yaitu :

1. Sekuensial (runtunan).
   Dalam struktur algoritma yang bersifat sekuensial maka langkah-langkah yang ada di dalam algoritma akan dilakukan secara berurutan dari awal sampai akhir.

2. Seleksi.
   Pada struktur seleksi ini, suatu langkah akan dikerjakan jika suatu syarat terpenuhi, atau sebaliknya bahwa suatu langkah akan dikerjakan jika suatu syarat tidak terpenuhi.

3. Pengulangan.
   Struktur pengulangan adalah suatu langkah yang sama atau serupa (misal hanya berbeda nilai) yang kemudian dilakukan beberapa kali (berulang-ulang).

Suatu proses yang rumit (misalkan jika membayangkan proses mencari kebalikan dari suatu matrik misalnya), pada dasarnya terdiri dari ketiga struktur dasar tadi.

3.2.1 Struktur Sekuensial

Ibarat orang berjalan, maka dalam struktur algoritma yang sekuensial, orang tersebut berjalan lurus ke depan, tidak ada belok akan, belok kiri, atau berputar kembali ke jalan sebelumnya. Dalam kehidupan nyata, tentu saja langkah-langkah yang seperti ini bisa dikatakan tidak ada. Untuk (memaksakan contoh), misalnya ketika akan menghitung umur seseorang hanya berdasarkan tahun lahir dan tahun sekarang. Algoritma untuk proses ini:

1. Memasukkan tahun lahir.

2. Memasukkan tahun sekarang.

3. Menghitung umur adalah tahun sekarang dikurang tahun lahir.

4. Mencetak umur.

Langkah perhitungan seperti ini, tentu saja tidak adil. Mengapa? Karena orang yang lahir pada tanggal 1 Januari dengan orang yang lahir pada tanggal 30 Desember pada tahun yang sama, akan memiliki usia yang sama. Pada sebenarnya kedua orang tersebut memiliki umur yang berselisih satu tahun.

3.2.2 Struktur Seleksi

Dengan melihat ketimpangan pada contoh penghitungan umur, maka akan lebih baik jika penghitungan umur tidak hanya berdasarkan tahun lahir saja, akan tetapi berdasarkan bulan lahir dan bulan sekarang. Dengan adanya tambahan bulan lahir dan bulan sekarang, maka masalah yang perlu diperhatikan adalah ketika bulan sekarang lebih kecil dari bulan lahir. Contoh program untuk masalah ini bisa dilihat seperti berikut.

```
/* ------------------------------------------------------------------------
  program untuk menghitung umur berdasarkan bulan dan tahun
  nama file : umur-bulan.cpp
  penulis   : www.dwijim.wordpress.com
  editing   : 15 oktober 2011

  menghitung umur dalam kondisi yang mudah caranya adalah dengan
  mengurangkan antara tahun dan bulan sekarang dengan
  tahun dan bulan lahir. diperlukan perhatian khusus ketika
  bulan sekarang nilainya lebih kecil dari bulan lahir.
  misalnya sekarang bulan 6 dan lahir bulan 7.
  jika 6-7 = -1 ? tidak mungkin nilainya negatif
   ------------------------------------------------------------------------ */

#include <iostream>
// menyertakan berkas yang berisi fungsi-fungsi tertentu dan sudah dikompilasi

#define hasil_fungsi 11;
// untuk menghindari adanya magic number

using namespace std;
// ciri penulisan c++ pada compiler modern, std padanan dari global

int main()
// program dalam bahasa C++ paling tidak terdiri dari satu fungsi
// yaitu fungsi main() dan fungsi main() ini menghasilkan nilai integer

  {

  int bulan_lahir, tahun_lahir;
  // mendeklarasikan pengenal (variable) bertipe integer
  // untuk menyimpan tahun dan bulan lahir

  int bulan_sekarang, tahun_sekarang;
  // mendeklarasikan pengenal (variable) bertipe integer
  // untuk menyimpan tahun dan bulan sekarang

  int selisih_bulan, selisih_tahun;
  // mendeklarasikan pengenal (variable) bertipe integer
  // untuk menyimpan selisih tahun dan bulan

  cout  << "Tahun lahir : ";
  // menampilkan ke layar pertanyaan ke pemakai
```

```
  cin  >> tahun_lahir;
  // memasukkan data ke variable tahun_lahir

  cout  << "Bulan lahir : ";
  // menampilkan ke layar pertanyaan ke pemakai

  cin  >> bulan_lahir;
  // memasukkan data ke variable bulan_lahir

  cout  << "Tahun sekarang : ";
  // menampilkan ke layar pertanyaan ke pemakai

  cin  >> tahun_sekarang;
  // memasukkan data ke variable tahun_sekarang

  cout  << "Bulan sekarang : ";
  // menampilkan ke layar pertanyaan ke pemakai

  cin  >> bulan_sekarang;
  // memasukkan data ke variable bulan_sekarang

  if (bulan_sekarang<bulan_lahir)
     {
      // berarti bulan sekarang ditambah 12 dan tahun sekarang dikurangi 1
      bulan_sekarang = bulan_sekarang + 12;
      tahun_sekarang = tahun_sekarang - 1;
     }
  selisih_bulan = bulan_sekarang - bulan_lahir;
  selisih_tahun = tahun_sekarang - tahun_lahir;

  cout << "Umur tahun : " << selisih_tahun << "\n";
  cout << "     bulan : " << selisih_bulan;
  cout  << "\n";
  return hasil_fungsi;
  // karena int main() jadi harus return bilangan bulat

} // akhir program
```

3.2.3 Struktur Pengulangan

Pengulangan adalah suatu proses yang menjalankan perintah-perintah yang sama secara terus-menerus sampai ditemukan suatu kondisi yang membuat proses tersebut berhenti. Dengan adanya struktur pengulangan ini, maka program akan menjadi lebih singkat dan ringkas. Contoh berikut adalah program yang melakukan suatu proses tanpa perulangan :

```
/* ----------------------------------------------------------------------
  program untuk mencetak tulisan ilmu komputer universitas lampung
  sebanyak 27 kali tanpa proses pengulangan.
  nama file     : cetak-ilkom-27x.cpp
  penulis       : www.dwijim.wordpress.com
```

```
  first editing : 18 februari 2012
  last editing  : 18 februari 2012
  compiler g++ (Ubuntu/Linaro 4.6.1-9ubuntu3) 4.6.1
  --------------------------------------------------------------------------- */

#include <iostream>
// menyertakan berkas yang berisi fungsi-fungsi tertentu dan sudah dikompilasi

#define hasil_fungsi 11;

using namespace std;
// ciri penulisan c++ pada compiler modern, std padanan dari global

int main()
// program dalam bahasa C++ paling tidak terdiri dari satu fungsi
// yaitu fungsi main() dan fungsi main() ini menghasilkan nilai integer

  {
       // mencetak tulisan sebanyak 27 kali
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";
       cout << "ilmu komputer universitas lampung \n";

  return hasil_fungsi;
  // karena int main() jadi harus return bilangan bulat
```

```
} // akhir program
```

Setidaknya ada dua masalah yang bisa dilihat dari program tersebut:

1. Jumlah keluaran apakah sudah tepat ?
   Apakah bisa diyakinkan bahwa jumlah keluaran memang ada 27 kali. Jika kemudian ternyata jumlahnya juga berubah maka harus dihitung kembali. Tentu saja menjadi tidak praktis lagi ketika jumlah baris yang akan dicetak harus selalu dihitung untuk meyakinkan kesesuai jumlah.

2. Program menjadi tidak praktis jika jumlah baris yang akan dicetak selalu berubah-ubah.
   Perubahan jumlah baris yang akan dicetak, menyebabkan program itu harus diubah lagi, demikian seterusnya. Belum lagi jika ternyata jumlahnya banyak.

Dalam masalah ini terdapat perintah yang sama dan diulang-ulang, yaitu mencetak suatu tulisan. Karena ada proses yang berulang-ulang, oleh karena itu akan lebih tepat jika program ini menggunakan proses perulangan, bukan proses yang berurutan seperti pada contoh program tersebut. Dengan demikian, program tersebut diperbaiki sehingga menjadi seperti berikut:

```
/* ------------------------------------------------------------------------
  program untuk mencetak tulisan ilmu komputer universitas lampung
  sebanyak n kali.
  nama file     : cetak-n.cc
  penulis       : www.dwijim.wordpress.com
  first editing : 9 november 2011
  last editing  : 18 februari 2012
  compiler g++ (Ubuntu/Linaro 4.6.1-9ubuntu3) 4.6.1
   ------------------------------------------------------------------------ */

#include <iostream>
// menyertakan berkas yang berisi fungsi-fungsi tertentu dan sudah dikompilasi

#define hasil_fungsi 11;

// using namespace global;
//  khusus pada compiler turbo c versi 3x

using namespace std;
// ciri penulisan c++ pada compiler modern, std padanan dari global

int main()
// program dalam bahasa C++ paling tidak terdiri dari satu fungsi
// yaitu fungsi main() dan fungsi main() ini menghasilkan nilai integer

  {

  int jumlah_baris;
  // mendeklarasikan pengenal (variable) bertipe integer
  // untuk menyimpan tahun dan bulan lahir
```

```
  int proses_looping;
  // mendeklarasikan pengenal (variable) bertipe integer
  // untuk menyimpan tahun dan bulan sekarang

  cout  << "Jumlah baris : ";
  // menampilkan ke layar pertanyaan ke pemakai

  cin  >> jumlah_baris;
  // memasukkan data ke variable tahun_lahir

  for (proses_looping=1;proses_looping<=jumlah_baris;proses_looping++)
       cout << "ilmu komputer universitas lampung \n";

  return hasil_fungsi;
  // karena int main() jadi harus return bilangan bulat

} // akhir program
```

Contoh keluaran dari program:

```
Jumlah baris : 5
ilmu komputer universitas lampung
ilmu komputer universitas lampung
ilmu komputer universitas lampung
ilmu komputer universitas lampung
ilmu komputer universitas lampung
```

Suatu program yang rumit, pada dasarnya terdiri atas ketiga dasar-dasar proses tersebut. Masalahnya bagaimana meramu perintah-perintah yang ada, sehingga bisa dihasilkan suatu program yang dapat menyelesaikan suatu masalah.

3.3 Contoh Penyelesaian Masalah - Membuat Penanggalan

Hampir kebanyakan dari pembaca pernah memperhatikan suatu penanggalan. Jika selama ini, penanggalan merupakan sesuatu yang sudah ada dan tinggal digunakan, maka sekarang problemnya adalah bagaimana membuat penanggalan atau kalender. Untuk mempermudah masalah, maka penanggalan yang akan dibuat hanya pada satu bulan tertentu.

Ketika seseorang akan membuat penanggalan, hal apakah yang harus diperhatikan? *Kan* tidak mungkin langsung membuat penanggalan.

```
/* ------------------------------------------------
   program untuk mencetak penanggalan pada suatu bulan
   dibuat dengan bahasa C++ berbasis teks
   dibuat oleh  : dwi sakethi
   firt editing : 1 november 2011
   last editing : 1 november 2011
   ------------------------------------------------ */
#include <iostream>
// berkas standar untuk input dan output
```

```
#include <iomanip>
// karena ada perintah setw(n) untuk mengatur banyaknya
// digit tampilan angka yang akan dicetak

using namespace std;

// program dalam C++ minimal terdiri dari satu fungsi main()
int main()
{

    short int tanggal;
    // deklarasi variabel tanggal 1 .. 30

    short int hari_ke;
    // deklarasi variabel posisi hari atau hari ke 1 .. 7

    short int jumlah_hari;
    // deklarasi variabel banyaknya hari pada suatu bulan

    short int posisi;
    // deklarasi variabel posisi 1, 2, ... 7

    cout << "Tanggal satu jatuh pada hari ke berapa : ";
    cin >> hari_ke;
    // memasukkan tanggal satu pada hari apa ... ?

    cout << "Jumlah hari pada bulan ini : ";
    cin >> jumlah_hari;

    // menulis kosong dari posisi satu sampai dengan
    // posisi - 1

    // cetak nama-nama hari
    // karena tanggal dicetak dengan 3 digit (setw(3))
    // maka penulisan nama-nama dari juga 3 karakter
    cout << "  M  S  S  R  K  J  S\n";
    for (posisi=1;posisi<=hari_ke-1;posisi++)
        {
            cout << "   ";
    // mencetak karakter kosong sebelum posisi
    // tanggal 1
        } // akhir looping for posisi

    // menulis tanggal 1 sampai dengan tanggal terakhir
    for (tanggal=1;tanggal<=jumlah_hari;tanggal++)
        {
            cout << setw(3) << tanggal;
    // menulis tanggal selebar 3 digit

            posisi = posisi + 1;
```

```
    // posisi hari bergeser ke kanan (tambah 1)

            if (posisi>7) { posisi=1; cout << endl;}
    // jika sudah hari ke-7, maka ganti baris
    // atau kembali ke posisi pertama
        } // akhir looping for tanggal

    return 0;
    // karena int main, maka mesti ada nilai yang dikembalikan
}
// akhir program
```