Aunur R. Mulyanto

REKAYASA PERANGKAT LUNAK
JILID 1

SMK

Direktorat Pembinaan Sekolah Menengah Kejuruan
Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah
Departemen Pendidikan Nasional

REKAYASA PERANGKAT LUNAK JILID 1

Untuk SMK

Penulis : Aunur R. Mulyanto
Perancang Kulit : Tim

Ukuran Buku : 17,6 x 25 cm

MUL MULYANTO,Aunur R.
Rekayasa Perangkat Lunak Jilid 1 untuk SMK /oleh Aunur R. Mulyanto ---- Jakarta : Direktorat Pembinaan Sekolah Menengah Kejuruan, Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah, Departemen Pendidikan Nasional, 2008.
viii. 153 hlm
Daftar Pustaka : A1-A2
Glosarium : B1-B6
ISBN : 978-979-060-007-2
ISBN : 978-979-060-008-9

Diterbitkan oleh
Direktorat Pembinaan Sekolah Menengah Kejuruan
Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah
Departemen Pendidikan Nasional
Tahun 2008

KATA SAMBUTAN

Puji syukur kami panjatkan kehadirat Allah SWT, berkat rahmat dan karunia Nya, Pemerintah, dalam hal ini, Direktorat Pembinaan Sekolah Menengah Kejuruan Direktorat Jenderal Manajemen Pendidikan Dasar dan Menengah Departemen Pendidikan Nasional, telah melaksanakan kegiatan penulisan buku kejuruan sebagai bentuk dari kegiatan pembelian hak cipta buku teks pelajaran kejuruan bagi siswa SMK. Karena buku-buku pelajaran kejuruan sangat sulit di dapatkan di pasaran.

Buku teks pelajaran ini telah melalui proses penilaian oleh Badan Standar Nasional Pendidikan sebagai buku teks pelajaran untuk SMK dan telah dinyatakan memenuhi syarat kelayakan untuk digunakan dalam proses pembelajaran melalui Peraturan Menteri Pendidikan Nasional Nomor 45 Tahun 2008 tanggal 15 Agustus 2008.

Kami menyampaikan penghargaan yang setinggi-tingginya kepada seluruh penulis yang telah berkenan mengalihkan hak cipta karyanya kepada Departemen Pendidikan Nasional untuk digunakan secara luas oleh para pendidik dan peserta didik SMK.

Buku teks pelajaran yang telah dialihkan hak ciptanya kepada Departemen Pendidikan Nasional ini, dapat diunduh (*download*), digandakan, dicetak, dialihmediakan, atau difotokopi oleh masyarakat. Namun untuk penggandaan yang bersifat komersial harga penjualannya harus memenuhi ketentuan yang ditetapkan oleh Pemerintah. Dengan ditayangkan *soft copy* ini diharapkan akan lebih memudahkan bagi masyarakat khsusnya para pendidik dan peserta didik SMK di seluruh Indonesia maupun sekolah Indonesia yang berada di luar negeri untuk mengakses dan memanfaatkannya sebagai sumber belajar.

Kami berharap, semua pihak dapat mendukung kebijakan ini. Kepada para peserta didik kami ucapkan selamat belajar dan semoga dapat memanfaatkan buku ini sebaik-baiknya. Kami menyadari bahwa buku ini masih perlu ditingkatkan mutunya. Oleh karena itu, saran dan kritik sangat kami harapkan.

Jakarta, 17 Agustus 2008
Direktur Pembinaan SMK

PENGANTAR PENULIS

Dengan segala kerendahan hati, kami mengucapkan syukur kepada Allah SWT. Karena hanya dengan lindungan, rahmat dan karuniaNya-lah maka buku ini dapat diselesaikan.

Buku yang berjudul 'Rekayasa Perangkat Lunak' merupakan buku yang disusun untuk memenuhi kebutuhan buku pegangan bagi siswa Sekolah Menengah Kejuruan. Khususnya pada program keahlian Rekayasa Perangkat Lunak. Buku ini memuat uraian yang mengacu pada standar kompetensi dan kompetensi dasar Rekayasa Perangkat Lunak untuk siswa SMK mulai dari kelas X, XI sampai dengan kelas XII.

Tiap bab berisi teori yang harus dipahami secara benar oleh peserta didik dan disertai dengan contoh-contoh soal yang relevan dengan teori tersebut. Selain itu terdapat juga soal-soal yang didasarkan pada konsep dan teori yang dibahas sebagai alat uji untuk mengukur kemampuan peserta didik dalam penguasaan materi tersebut.

Dalam mengembangkan buku ini, penulis berupaya agar materi yang disajikan sesuai dengan kebutuhan kompetensi yang harus dicapai. Oleh karenanya, selain dari hasil pemikiran dan pengalaman penulis sebagai pengajar dan praktisi Rekayasa Perangkat Lunak, materi yang dikembangkan juga diperkaya dengan referensi-referensi lain yang sesuai.

Pada kesempatan ini penulis ingin menyampaikan rasa terima kasih kepada semua pihak yang mendukung buku ini dapat diterbitkan. Mudah-mudahan buku ini dapat bermanfaat bagi peserta didik dalam mengembangkan kemampuannya. Penulis menyadari bahwa buku ini masih perlu dikembangkan terus menerus, sehingga saran dari berbagai pihak pengguna buku ini sangat diharapkan.

Penulis

DAFTAR ISI

PETUNJUK PENGGUNAAN BUKU

A. Deskripsi Umum

Buku ini diberi judul "Rekayasa Perangkat Lunak",sama dengan salah satu program keahlian pada Sekolah Menengah Kejuruan (SMK). Meskipun demikian, sebenarnya isi dari buku ini tidak secara khusus membahas tentang Rekayasa Perangkat Lunak. Dari sisi pandang bidang Ilmu Komputer ada lima sub-bidang yang tercakup dalam dalam buku ini, yaitu sub-bidang Rekayasa Perangkat Lunak, Sistem Operasi, Algoritma dan Struktur Data, Bahasa Pemrograman dan Basis Data. Hal ini disesuaikan dengan kurikulum tingkat SMK untuk Program Keahlian Rekayasa Perangkat Lunak.

Pokok bahasan tentang Rekayasa Perangkat Lunak secara umum membahas dasar-dasar pengertian Rekayasa Perangkat Lunak, masalah dan pemecahan masalah, dan metode-metode pengembangan perangkat lunak. Pembahasan tentang sub-bidang Sistem Operasi berisi sistem computer, sistem operasi dan bekerja dalam jaringan computer. Cakupan materi algoritma meliputi algoritma dasar dan algoritma lanjutan. Sub bidang Bahasa Pemrograman mengambil porsi yang cukup besar, meliputi pemrograman GUI dengan VB & VB.Net, pemrograman Java, pemrograman C++, pemrograman berorientasi obyek dan Pemrograman berbasis web. Sub-bidang terakhir yang menjadi bagian dari buku ini adalah Basis Data dengan cakupan tentang system basis data, pemodelan konseptual, basis data relasional, Microsoft Access dan SQL.

B. Peta Kompetensi

Secara umum, buku ini mengacu pada Standar Kompetensi dan Kompetensi Dasar (SKKD) bagi SMK seperti berikut.

1. Menggunakan algoritma pemrograman tingkat dasar
2. Menggunakan algoritma pemrograman tingkat lanjut
3. Mengoperasikan aplikasi basis data
4. Membuat aplikasi berbasis Microsoft Access
5. Menguasai teknik elektronika dasar
6. Menguasai teknik elektronika digital
7. Membuat file dengan HTML sesuai spesifikasi
8. Menerapkan dasar-dasar pembuatan web statis tingkat dasar
9. Membuat program aplikasi menggunakan VB dan VB.NET
10. Membuat paket software aplikasi
11. Melakukan pemrograman data deskripsi (SQL – Stuctured Query Language) tingkat dasar
12. Mengoperasikan bahasa pemrograman data deskripsi (SQL) tingkat lanjut
13. Membuat halaman web dinamis tingkat dasar
14. Membuat halaman web dinamis tingkat lanjut
15. Membuat program aplikasi web menggunakan JSP

16. Membuat program aplikasi basis data menggunakan XML
17. Membuat program basis data menggunakan Microsoft (SQL Server)
18. Membuat program basis data menggunakan PL/SQL (Oracle)
19. Membuat program aplikasi menggunakan C++
20. Menjelaskan sistem peripheral
21. Membuat program dalam bahasa pemrograman berorientasi obyek
22. Membuat program aplikasi menggunakan Java
23. Mengoperasikan sistem operasi komputer berbasis teks dan GUI

Dalam penyajian buku ini, bab-bab tidak disusun berdasarkan SKKD, akan tetapi disusun berdasarkan urutan materi pokok bahasan. Sehingga di beberapa bab berisi gabungan dari beberapa standar kompetensi. Atau satu kompetensi dasar mungkin berada tidak pada kelompok standar kompetensi seperti pada daftar SKKD, tetapi berada pada sub bab yang lain.

Kesesuaian SKKD dan isi bab dapat dilihat pada table berikut ini.

Kode Kompetensi	Kompetensi	Bab Terkait
ELKA-MR.UM.001.A	Menguasai Teknik Dasar Elektronika	**3**
ELKA.MR.UM.004.A	Menguasai Dasar Elektronika Digital dan Komputer	**3**
TIK.PR02.001.01	Menggunakan algoritma pemograman tingkat dasar	**5**
TIK.PR02.002.01	Menggunakan algoritma pemograman tingkat lanjut	**6**
HDW.OPR.103.(1).A	Mengoperasikan sistem operasi jaringan komputer berbasis teks	**4**
HDW.OPR.104.(1).A	Mengoperasikan sistem operasi jaringan komputer berbasis GUI	**4**
TIK.PR02.020.01	Mengoperasikan aplikasi basis Data	**10 dan 11**
TIK.PR08.004.01	Membuat aplikasi Berbasis Microsoft Acces	**11**
TIK.PR08.024.01	Membuat dokumen dengan HTML sesuai spesifikasi	**13**
TIK.PR08.027.01	Menerapkan dasar-dasar pembuatan web statis tingkat dasar.	**13**
TIK.PR08.003.01	Membuat program aplikasi menggunakan VB & VB.NET	**7**
TIK.PR02.016.01	Membuat paket software Aplikasi	**7**
TIK.PR03.001.01	Mengoperasikan bahasa pemrograman data deskripsi (SQL) tingkat dasar	**12**
TIK.PR03.002.01	Mengoperasikan bahasa pemrograman data deskripsi (SQL) tingkat Lanjut	**12**
TIK.PR04.002.01	Membuat Halaman Web dinamis tingkat dasar	**13**
TIK.PR04.003.01	Membuat Halaman Web dinamis tingkat Lanjut.	**13**

Kode Kompetensi	Kompetensi	Bab Terkait
TIK.PR02.009.01	Mengoperasikan bahasa pemograman berorientasi obyek	**8**
TIK.PR08.012.01	Membuat program aplikasi menggunakan Java	**8**
TIK.PR08.001.01	Membuat program aplikasi menggunakan C++	**9**
TIK.PR06.003.01	Menjelaskan sistem Peripheral	**3**
TIK.PR08.005.01	Membuat program basis data menggunakan PL/SQL	**10 dan 12**
TIK.PR08.006.01	Membuat program basis data menggunakan SQL Server	**12**
TIK.PR08.008.01	Membuat program aplikasi web berbasis JSP	**14**

C. Cara Menggunakan Buku

Buku ini secara khusus ditujukan kepada siswa dan guru SMK untuk program keahlian RPL. Namun demikian, buku ini juga terbuka bagi pembaca umum yang berminat dalam dunia RPL, Algoritma dan Pemrograman, Basis Data dan Internet. Bagi siswa, buku ini dapat dijadikan buku pegangan, karena ini buku ini menyediakan bahan-bahan pelajaran yang cukup lengkap untuk mata pelajaran selama tiga tahun di bangku sekolah. Beberapa bagian dari buku ini mungkin memerlukan buku-buku bantu lainnya untuk lebih memperkaya wawasan dan peningkatan kemampuan. Sedangkan bagi guru, buku ini dapat digunakan sebagai buku referensi untuk menyusun modul-modul ajar bagi anak didiknya.

Buku ini disusun sedemikian rupa agar siswa dapat belajar secara mandiri dan terdorong untuk mencoba secara langsung. Oleh karena itu dalam buku ini, akan banyak dijumpai ilustrasi baik yang berupa gambar, skema maupun *listing program.* Hal ini dimaksudkan agar siswa dapat dengan mudah memahami penjelasan ataupun penerapan suatu konsep tertentu. Bahkan pada bagian akhir bab diakhiri dengan soal-soal latihan dari pokok bahasan pada bab tersebut.

BAB 1 PENDAHULUAN

Gambar 1.1. Tampilan desktop Microsoft Windows.

Coba kita perhatikan Gambar 1.1. di atas. Bagi pengguna komputer, gambar di atas merupakan tampilan yang sangat dikenal. Gambar ini merupakan tampilan desktop Sistem Operasi Microsoft Windows. Pada gambar tersebut, kita melihat sejumlah ikon-ikon tertentu. Apabila kita klik ganda pada satu ikon maka suatu perangkat lunak (software) tertentu akan terbuka dan dapat kita gunakan untuk menyelesaikan suatu tugas tertentu.

Pada masa sekarang, rasanya hampir semua bidang kehidupan tersentuh penggunaan perangkat lunak atau software. Beberapa perangkat lunak mungkin sudah terbiasa kita gunakan atau kita lihat seperti perangkat lunak untuk memainkan atau membuat musik, perangkat lunak untuk membantu kasir dalam penjualan barang, perangkat lunak untuk mengetik dokumen, dan lain-lain. Perangkat lunak ini merupakan hasil dari serangkaian proses atau kegiatan yang dikenal sebagai Rekayasa Perangkat Lunak. Apakah sebenarnya Rekayasa Perangkat Lunak itu? Bab ini akan memberi jawaban atas pertanyaan ini.

TUJUAN

Setelah mempelajari bab ini diharapkan kalian akan mampu :

- Menjelaskan pengertian perangkat lunak, program, prosedur dan rekayasa perangkat lunak
- Memahami tujuan rekayasa perangkat lunak
- Memahami ruang lingkup rekayasa perangkat lunak
- Memahami posisi bidang rekayasa perangkat lunak pada disiplin ilmu komputer dan keterkaitannya dengan bidang ilmu lain
- Mengetahui perkembangan ilmu rekayasa perangkat lunak
- Mengetahui profesi dan sertifikasi dalam bidang rekayasa perangkat lunak
- Menjelaskan prinsip-prinsip pemecahan masalah dalam rekayasa perangkat lunak

1.1. PENGERTIAN REKAYASA PERANGKAT LUNAK

Istilah Rekayasa Perangkat Lunak (RPL) secara umum disepakati sebagai terjemahan dari istilah *Software Engineering*. Istilah *Software Engineering* mulai dipopulerkan tahun 1968 pada *Software Engineering Conference* yang diselenggarakan oleh NATO. Sebagian orang mengartikan RPL hanya sebatas pada bagaimana membuat program komputer. Padahal ada perbedaan yang mendasar antara perangkat lunak (*software*) dan program komputer.

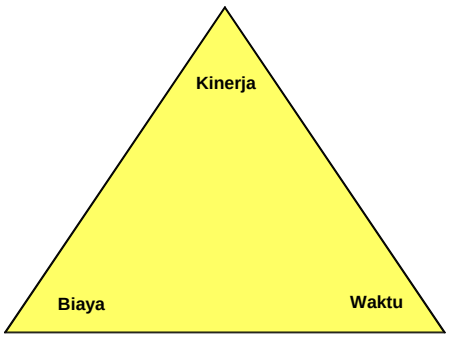
Perangkat lunak adalah *seluruh perintah yang digunakan untuk memproses informasi.* Perangkat lunak dapat berupa program atau prosedur. **Program** adalah kumpulan perintah yang dimengerti oleh komputer sedangkan **prosedur** adalah perintah yang dibutuhkan oleh pengguna dalam memproses informasi (O'Brien, 1999). Pengertian RPL sendiri adalah sebagai berikut:

> Suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, disain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan.

Jelaslah bahwa RPL tidak hanya berhubungan dengan cara pembuatan program komputer. Pernyataan "*semua aspek produksi*" pada pengertian di atas, mempunyai arti semua hal yang berhubungan dengan proses produksi seperti manajemen proyek, penentuan personil, anggaran biaya, metode, jadwal, kualitas sampai dengan pelatihan pengguna merupakan bagian dari RPL.

1.2. TUJUAN REKAYASA PERANGKAT LUNAK

Secara umum tujuan RPL tidak berbeda dengan bidang rekayasa yang lain. Mari kita perhatikan Gambar 1.2. berikut ini.



Gambar 1.2. Tujuan RPL.

Dari Gambar 1.2 dapat diartikan bahwa bidang rekayasa akan selalu berusaha menghasilkan output yang kinerjanya tinggi, biaya rendah dan waktu penyelesaian yang tepat. Secara lebih khusus kita dapat menyatakan tujuan RPL adalah :

a. Memperoleh biaya produksi perangkat lunak yang rendah.

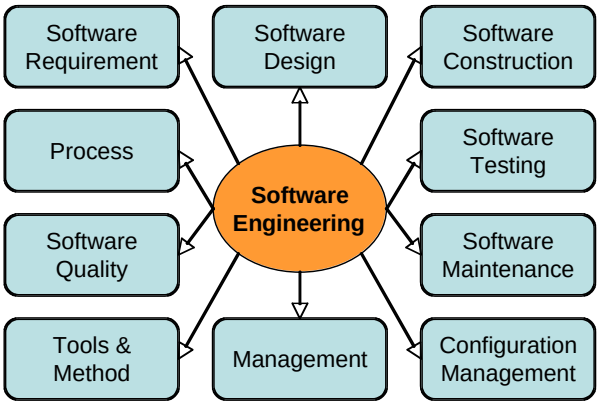
b. Menghasilkan perangkat lunak yang kinerjanya tinggi, andal dan tepat waktu.

c. Menghasilkan perangkat lunak yang dapat bekerja pada berbagai jenis *platform*.

d. Menghasilkan perangkat lunak yang biaya perawatannya rendah.

1.3. RUANG LINGKUP

Sesuai definisi yang telah disampaikan sebelumnya, maka ruang lingkup RPL dapat digambarkan sebagai berikut.



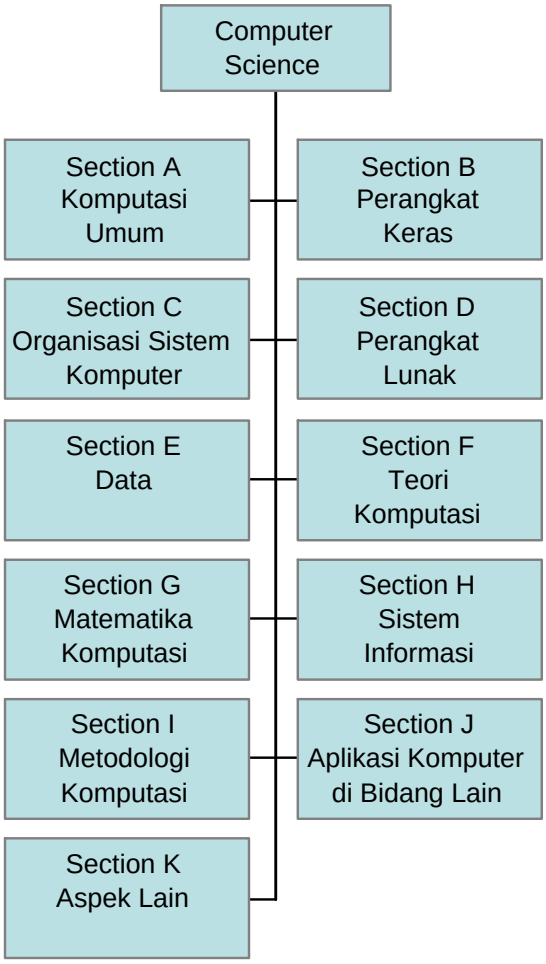
Gambar 1.3. Ruang lingkup RPL (Abran et.al., 2004).

- *Software requirements* berhubungan dengan spesifikasi kebutuhan dan persyaratan perangkat lunak.
- *Software design* mencakup proses penentuan arsitektur, komponen, antarmuka, dan karakteristik lain dari perangkat lunak.
- *Software construction* berhubungan dengan detil pengembangan perangkat lunak, termasuk algoritma, pengkodean, pengujian, dan pencarian kesalahan.
- *Software testing* meliputi pengujian pada keseluruhan perilaku perangkat lunak.
- *Software maintenance* mencakup upaya-upaya perawatan ketika perangkat lunak telah dioperasikan.
- *Software configuration management* berhubungan dengan usaha perubahan konfigurasi perangkat lunak untuk memenuhi kebutuhan tertentu.
- *Software engineering management* berkaitan dengan pengelolaan dan pengukuran RPL, termasuk perencanaan proyek perangkat lunak.
- *Software engineering tools and methods* mencakup kajian teoritis tentang alat bantu dan metode RPL.
- *Software engineering process* berhubungan dengan definisi, implementasi, pengukuran, pengelolaan, perubahan dan perbaikan proses RPL.
- *Software quality* menitikberatkan pada kualitas dan daur hidup perangkat lunak.

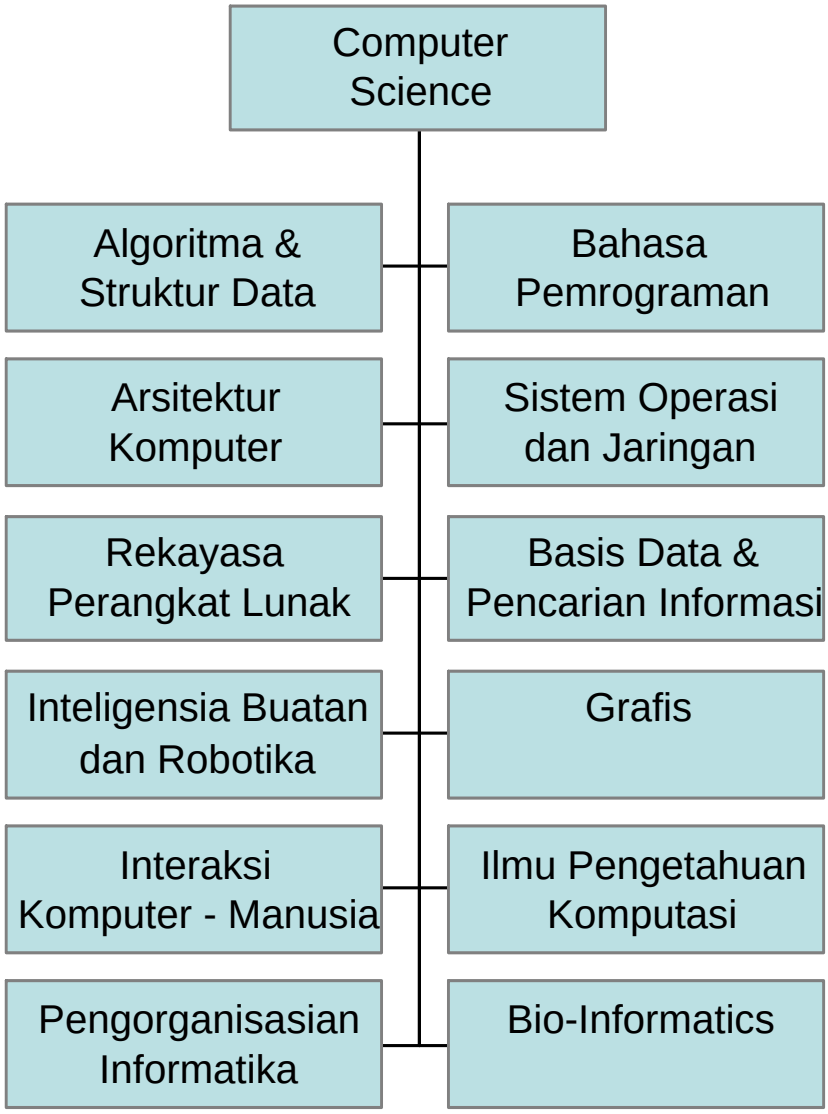
1.4. REKAYASA PERANGKAT LUNAK DAN DISIPLIN ILMU KOMPUTER

Disiplin ilmu komputer (*Computer Science*) lahir pada awal-awal tahun 1940-an yang merupakan integrasi dari teori algoritma, logika matematika dan ditemukannya cara penyimpanan program secara elektronik pada komputer. Sejak itu ilmu komputer mengalami perkembangan yang terus menerus sehingga cakupannya menjadi semakin meluas.

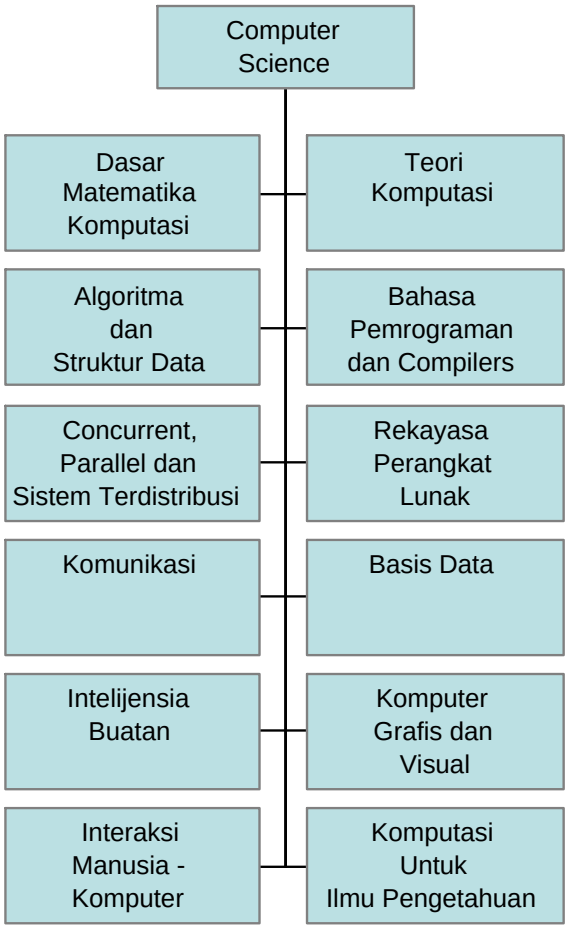
Cakupan pengetahuan dalam ilmu komputer seringkali didiskripsikan *sebagai suatu studi sistematis pada proses-proses algoritma yang menjelaskan dan mentransformasikan informasi* (Denning, 2000). Termasuk di sini adalah teori, analisis, disain, efisiensi, penerapan dan aplikasinya. Ada beberapa model pengelompokkan sub-bidang ilmu dalam disiplin ilmu komputer seperti terlihat pada Gambar 1.4, 1.5 dan 1.6.



Gambar 1.4. Klasifikasi disiplin ilmu komputer menurut ACM (1998).



Gambar 1.5. Klasifikasi disiplin ilmu komputer menurut Denning (2000).

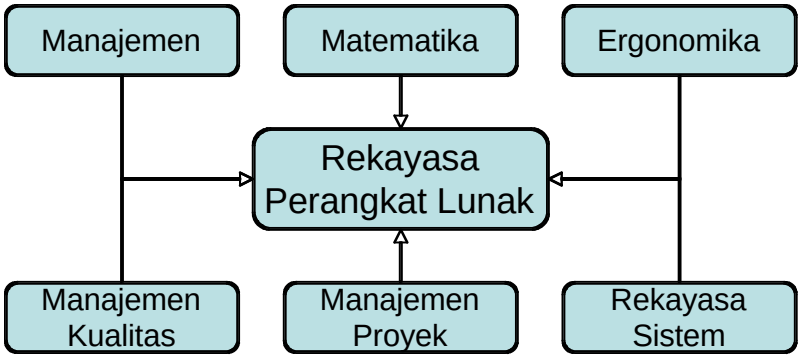


Gambar 1.6. Klasifikasi disiplin ilmu komputer menurut Wikipedia (2007).

Berdasarkan pengelompokkan Denning (2000) dan Wikipedia (2007), RPL merupakan sub-bidang ilmu komputer yang setara dengan sub-bidang lainnya. Sedangkan menurut ACM (*Association for Computing Machinery*), RPL merupakan bagian dari Section D (Perangkat Lunak). Meskipun terlihat terpisah-pisah, namun dalam penerapannya, sub-bidang RPL selalu membutuhkan dukungan dari sub-bidang lain, terutama sub-bidang Algoritma dan Struktur Data, Bahasa Pemrograman, Basis Data, Sistem Operasi dan Jaringan, dan Sistem Informasi.

1.5. REKAYASA PERANGKAT LUNAK DAN DISIPLIN ILMU LAIN

Cakupan ruang lingkup yang cukup luas, membuat RPL sangat terkait dengan disiplin bidang ilmu lain. Tidak saja dengan sub-bidang dalam disiplin ilmu komputer namun dengan beberapa disiplin ilmu lain di luar ilmu komputer. Hubungan keterkaitan RPL dengan ilmu lain dapat dilihat pada Gambar 1.7.



Gambar 1.7. Keterkaitan RPL dengan bidang ilmu lain.

- Bidang ilmu manajemen meliputi akutansi, finansial, pemasaran, manajemen operasi, ekonomi, analisis kuantitatif, manajemen sumber daya manusia, kebijakan dan strategi bisnis.
- Bidang ilmu matematika meliputi aljabar linier, kalkulus, peluang, statistik, analisis numerik dan matematika diskrit.
- Bidang ilmu manajemen proyek meliputi semua hal yang berkaitan dengan proyek, seperti ruang lingkup proyek, anggaran, tenaga kerja, kualitas, manajemen resiko, dan penjadwalan proyek.
- Bidang ilmu manajemen kualitas meliputi pengembangan sistem kualitas, manajemen resiko dan keandalan, perbaikan kualitas, dan metode-metode kuantitatif.
- Bidang ilmu ergonomika menyangkut hubungan (interaksi) antara manusia dengan komponen-komponen lain dalam sistem komputer.
- Bidang ilmu rekayasa sistem meliputi teori sistem, analisis biaya-keuntungan, pemodelan, simulasi, proses dan operasi bisnis.

1.6. PERKEMBANGAN REKAYASA PERANGKAT LUNAK

Meskipun baru dicetuskan pada tahun 1968, namun RPL telah memiliki sejarah yang cukup panjang. Gambar 1.8 menyajikan intisari perkembangan RPL. Dari sisi disiplin ilmu, RPL masih relatif muda dan akan terus berkembang. Arah perkembangan yang saat ini sedang dikembangkan antara lain meliputi :

Agile Software Development, Experimental Software Development, Model-Driven Software Development dan Software Product Lines.

Tahun	Kejadian
1940an	Komputer pertama yang membolehkan pengguna menulis kode program langsung
1950an	Generasi awal interpreter dan bahasa macro Generasi pertama compiler
1960an	Generasi kedua compiler Komputer mainframe mulai dikomersialkan Pengembangan perangkat lunak pesanan Konsep Software Engineering mulai digunakan
1970an	Perangkat pengembang perangkat lunak Perangkat minicomputer komersial
1980an	Perangkat Komputer Personal (PC) komersial Peningkatan permintaan perangkat lunak
1990an	Pemrograman berorientasi obyek (OOP) Agile Process dan Extreme Programming Peningkatan drastis kapasitas memori Peningkatan penggunaan internet
2000an	Platform interpreter modern (Java, .Net, PHP, dll) Outsourcing

Gambar 1.8. Perkembangan RPL.

1.7. PROFESI DAN SERTIFIKASI

Profesi sebagai seorang *Software Engineer* mungkin masih terasa asing di telinga orang Indonesia. Sebagian besar orang Indonesia mungkin lebih familiar dengan sebutan **Ahli Teknologi Informasi, Analis Sistem Informasi, Programmer, Operator** atau sebutan profesi lainnya. Hal ini karena adanya kerancuan tentang istilah RPL seperti telah disebutkan di awal bab. Namun di negara-negara yang maju dalam bidang teknologi informasi, sebutan *Software Engineer* telah mulai banyak digunakan.

Sertifikasi kompetensi dalam bidang RPL, saat ini masih menjadi perdebatan di kalangan ahli dan penyedia perangkat lunak. Sebagian besar sertifikasi dalam industri perangkat lunak biasanya sangat spesifik untuk perangkat lunak tertentu. Sebagai contoh, perusahaan perangkat lunak seperti Redhat Linux Inc., Adobe Inc., Oracle, atau Microsoft, memberikan sertifikasi

kemampuan pada seseorang yang menguasai perangkat lunak yang diproduksinya.

ACM (*Association for Computing Machinery*) pernah menyelenggarakan sertifikasi untuk program *Software Engineer* pada tahun 1980an, namun dihentikan karena kurangnya peminat. IEEE (*Institute of Electrical and Electronics Engineers*) telah mengeluarkan lebih dari 500 sertifikat profesi perangkat lunak. Di Canada, telah dikeluarkan sebuah sertifikat legal untuk RPL yang disebut sebagai ISP (*Information Systems Profesional*).

Saat ini, sertifikasi untuk RPL di Indonesia juga belum tersedia, namun telah disusun **Standar Kompetensi Kerja Nasional Indonesia untuk Bidang Programmer Komputer**. Meskipun belum memenuhi cakupan bidang RPL secara keseluruhan, namun paling tidak dapat digunakan sebagai pendekatan sertifikasi bidang RPL.

1.8. REKAYASA PERANGKAT LUNAK DAN PEMECAHAN MASALAH

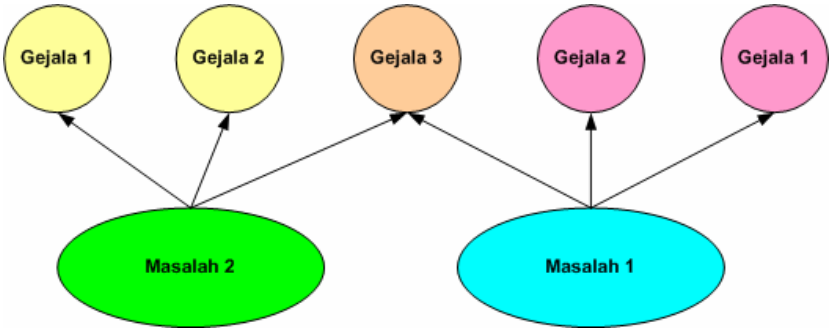
Secara konsep, rekayasa perangkat lunak memiliki kedekatan dengan prinsip-prinsip pemecahan masalah. Pemahaman tentang masalah, strategi dan proses pemecahan masalah, serta pendekatan sistem pada pemecahan masalah akan sangat membantu proses rekayasa perangkat lunak.

1.8.1. Masalah dan Gejala

Masalah (problem) adalah *perbedaan antara kondisi yang terjadi dan kondisi yang diharapkan* atau *boleh juga diartikan sebagai perbedaan antara kondisi sekarang dengan tujuan yang diinginka.* Sebagai contoh seorang siswa berharap memperoleh nilai di atas 80 untuk ujian mata pelajaran Pemrograman C++, namun pada kenyataannya dia hanya memperoleh nilai 60. Adanya perbedaan ini menunjukkan adanya masalah.

Gambar 1.9. Profesi dokter.

Seringkali kita kesulitan membedakan antara gejala dan masalah. **Gejala** adalah *tanda/petunjuk terjadinya suatu masalah.* Perhatikan seorang yang berprofesi sebagai dokter pada Gambar 1.9. Seorang dokter dalam usaha mengobati penyakit pasien selalu bertanya dulu tentang gejala-gejala yang dirasakan pasien kemudian menyimpulkan bahwa pasien menderita penyakit tertentu dan menentukan obat yang tepat. Pusing, demam, batuk, dan pilek merupakan gejala atau tanda dari penyakit flu. Apabila dokter hanya memberi obat sakit kepala, maka penyakit flu tidak akan sembuh. Satu masalah mungkin memiliki satu gejala tetapi mungkin juga lebih (perhatikan Gambar 1.10).

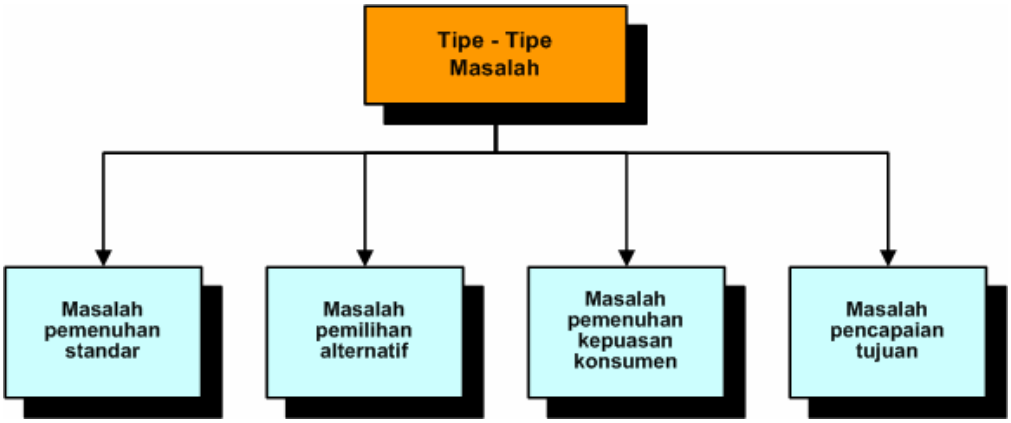


Gambar 1.10. Gejala dan masalah.

Mungkin kita bertanya-tanya apa hubungan masalah dan gejala dengan RPL. Seperti telah disampaikan di awal bab, perangkat lunak yang merupakan hasil dari RPL merupakan alat bantu yang digunakan untuk menyelesaikan tugas / masalah tertentu. Apabila kita tidak mengetahui dengan benar masalahnya mustahil kita dapat menentukan bagaimana menyelesaikannya. Dan, untuk mengetahui dengan baik masalah, maka pengetahuan tentang gejala dari masalah menjadi sangat penting.

1.8.2. Tipe-tipe Masalah

Masalah dapat dikelompokkan seperti pada Gambar 1.11.



Gambar 1.11. Tipe-tipe masalah (Deek et al, 2005).

- Masalah pemenuhan standar

  Tipe masalah dalam kelompok ini adalah masalah-masalah yang berhubungan dengan pencapaian standar yang telah ditentukan dalam

sebuah organisasi. Biasanya tujuan seperti ini berlaku dalam jangka yang relative panjang.

- Masalah pemilihan alternative

  Masalah dalam kelompok ini berhubungan dengan bagaimana memilih solusi terbaik dari berbagai alternative berdasarkan kriteria-kriteria tertentu. Permasalahan ini seringkali kita jumpai dalam kehidupan sehari-hari, seperti bagaimana memilih sekolah yang tepat, memilih lokasi tempat tinggal, memilih bidang pekerjaan. Masing-masing alternatif dan kriteria memiliki bobot yang telah disepakati.

- Masalah pemenuhan kepuasan konsumen

  Pada organisasi-organisasi yang bersifat profit (mencari keuntungan), masalah-masalah pada kelompok ini merupakan tipe yang seringkali muncul. Konsumen memiliki berbagai macam keinginan yang satu sama lain berbeda. Memenuhi seluruh keinginan konsumen sangat tidak mungkin dan sangat memberatkan sebuah organisasi. Oleh karena itu perlu dicari pemecahan yang sama-sama menguntungkan, baik bagi konsumen maupun organisasi tersebut.

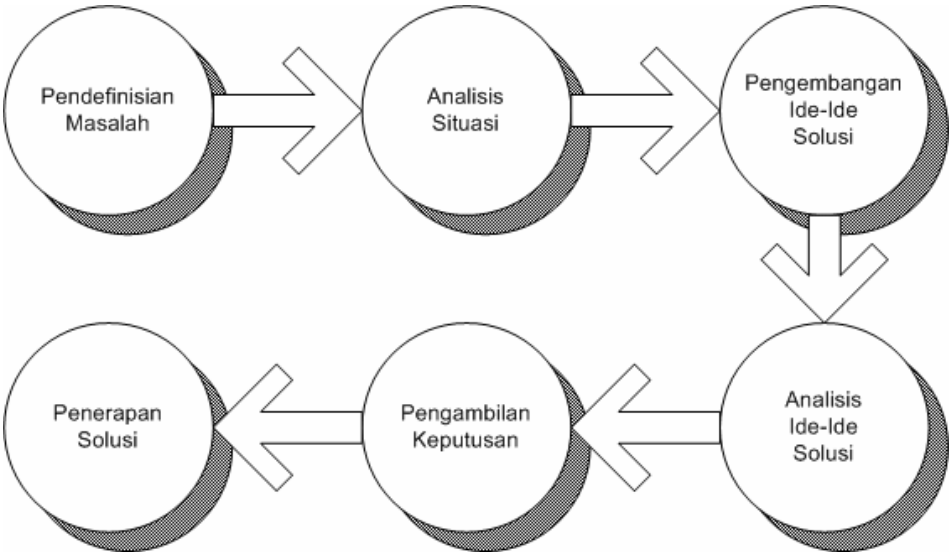
- Masalah pencapaian tujuan

  Tipe ini mirip dengan tipe pertama (masalah pemenuhan standar). Yang berbeda adalah, pada tipe ini tujuan yang ingin dicapai dapat berubah-ubah dan bersifat jangka pendek.

1.8.3. Pemecahan Masalah

Pemecahan masalah adalah *sebuah proses dimana suatu situasi diamati kemudian bila ditemukan ada masalah dibuat penyelesaiannya dengan cara menentukan masalah, mengurangi atau menghilangkan masalah atau mencegah masalah tersebut terjadi.* Ada banyak urutan proses pemecahan masalah yang diajukan oleh para ahli, salah satunya seperti terlihat pada Gambar 1.12.

Pada gambar 1.12 terlihat serangkaian tahapan proses yang berbeda yang dapat digunakan dalam berbagai tingkatan, tergantung dari tipe dan sifat masalahnya. Masalah yang berbeda membutuhkan penggunaan cara yang berbeda, bahkan mungkin urutan yang berbeda. Tahapan kritis dari proses pemecahan masalah adalah **Pendefinisian Masalah**. Apabila masalah tidak cukup jelas didefinisikan maka tahapan-tahapan berikut sulit untuk dijalankan. Bahkan apabila dipaksakan, kemungkinan besar penyelesaian yang tepat tidak akan diperoleh.



Gambar 1.12. Proses pemecahan masalah (diadopsi dari Deek et al, 2005)

Secara umum proses pemecahan masalah dapat dilakukan dengan empat tahapan utama yaitu :

- Memahami dan mendefinisikan masalah

  Bagian ini merupakan bagian yang sangat penting karena menjadi awal dari seluruh proses pemecahan masalah. Tujuan pada bagian ini adalah memahami masalah dengan baik dan menghilangkan bagian-bagian yang dirasa kurang penting.

- Membuat rencana untuk pemecahan masalah

  Pada bagian ini ada dua kegiatan penting yaitu :

  a) mencari berbagai cara penyelesaian yang mungkin diterapkan

  b) membuat rencana pemecahan masalah

  Penyelesaian suatu masalah biasanya tidak hanya satu tapi mungkin bisa beberapa macam. Sebagai ilustrasi, apabila kita berada di kota Surabaya dan ingin pergi ke Jakarta, maka banyak cara yang mungkin bisa dilakukan, misalnya kita bisa menempuh dengan angkutan darat, laut atau udara. Dengan angkutan darat kita bisa menggunakan kereta api, bus atau angkutan yang lain. Jalurnya pun kita bisa lewat jalur utara, tengah atau selatan. Jadi banyak sekali cara penyelesaian yang bisa kita kembangkan. Masing-masing mempunyai karakteristik sendiri-sendiri. Dari sekian banyak penyelesaian ini kita harus memilih satu yang berdasarkan persyaratan tertentu merupakan cara yang paling baik untuk menyelesaikan permasalahan. Setelah terpilih, maka kita dapat membuat rencana kasar (*outline*) penyelesaian masalah dan membagi masalah dalam bagian-bagian

yang lebih kecil. Rencana kasar (*outline*) penyelesaian masalah hanya berisi tahapan-tahapan utama penyelesaian masalah.

- Merancang dan menerapkan rencana untuk memperoleh cara penyelesaian

  Pada bagian ini rencana kasar penyelesaian masalah diperbaiki dan diperjelas dengan pembagian dan urutan rinci yang harus ditempuh dalam penyelesaian masalah.

- Memeriksa dan menyampaikan hasil dari pemecahan masalah

  Bagian ini bertujuan untuk memeriksa apakah akurasi (ketepatan) hasil dari cara yang dipilih telah memenuhi tujuan yang diinginkan. Selain itu juga untuk melihat bagaimana daya guna dari cara yang dipilih yang dipilih.

1.9. RINGKASAN

- **Perangkat lunak** adalah seluruh perintah yang digunakan untuk memproses informasi
  - **Program** adalah kumpulan perintah yang dimengerti oleh komputer
  - **Prosedur** adalah perintah yang dibutuhkan oleh pengguna dalam memproses informasi
- RPL adalah suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, disain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan.
- Tujuan RPL adalah menghasilkan perangkat lunak dengan kinerja tinggi, tepat waktu, berbiaya rendah, dan multiplatform.
- RPL merupakan sub bidang ilmu komputer yang dalam penerapannya membutuhkan dukungan baik dari sub bidang ilmu komputer lainnya maupun bidang-bidang ilmu lain.
- Sertifikasi untuk bidang RPL belum tersedia, namun mengacu pada bidang Programmer
- **Masalah (problem)** adalah perbedaan antara kondisi yang terjadi dan kondisi yang diharapkan dan **Gejala** adalah tanda/petunjuk terjadinya suatu masalah.
- Tipe-tipe masalah :
  - Masalah pemenuhan standar
  - Masalah pemilihan alternatif
  - Masalah pemenuhan kepuasan konsumen
  - Masalah pencapaian tujuan

- **Pemecahan masalah** adalah sebuah proses dimana suatu situasi diamati kemudian bila ditemukan ada masalah dibuat penyelesaiannya dengan cara menentukan masalah, mengurangi atau menghilangkan masalah atau mencegah masalah tersebut terjadi.
- Tahapan utama pemecahan masalah :
    - Memahami dan mendefinisikan masalah
    - Membuat rencana untuk pemecahan masalah
    - Merancang dan menerapkan rencana untuk memperoleh cara penyelesaian
    - Memeriksa dan menyampaikan hasil dari pemecahan masalah

1.10. SOAL-SOAL LATIHAN

1. Sebutkan pengertian dari perangkat lunak dan rekayasa perangkat lunak.
2. Apakah perbedaan antara program komputer dan prosedur?
3. Sebutkan lima sub bidang ilmu komputer berdasarkan pengelompokkan Denning.
4. Sebutkan lima bidang ilmu lain yang erat kaitannya dengan rekayasa perangkat lunak.
5. Apakah gejala dan masalah itu?

BAB 2 METODE REKAYASA PERANGKAT LUNAK

(Sumber: Clip Art Microsoft Office 2007)

Gambar 2.1. Bekerja dengan komputer.

Ketika kita bekerja dengan komputer seperti pada Gambar 2.1., kita membutuhkan serangkaian tahapan dan cara-cara tertentu agar dapat menghasilkan sesuatu yang menjadi harapan kita. Demikian juga dalam rekayasa perangkat lunak, diperlukan tahapan-tahapan kerja yang harus dilalui. Rekayasa perangkat lunak yang sukses tidak hanya membutuhkan kemampuan komputasi seperti algoritma, pemrograman, dan basis data yang kuat, namun juga perlu penentuan tujuan yang baik, identifikasi cara penyelesaian, metode pengembangan, urutan aktifitas, identifikasi kebutuhan sumberdaya, dan faktor-faktor lain. Hal-hal seperti ini terkait dengan apa yang disebut dengan metode rekayasa perangkat lunak.

Isi dari bab ini tidak termasuk dalam standar kompetensi bidang keahlian RPL. Namun penulis memandang perlu disampaikan agar kalian dapat mengetahui bagaimana sebenarnya rekayasa perangkat lunak dilakukan dan metode-metode apa saja yang biasa digunakan. Beberapa bagian dari bab ini mungkin agak sulit dipahami, sehingga peran guru dalam membantu menjelaskan akan sangat diperlukan. Rangkuman bab disampaikan di bagian akhir dari uraian isi.

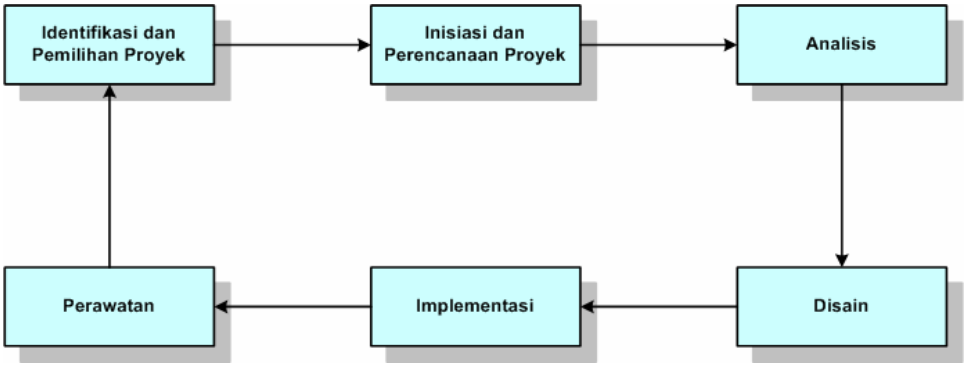
TUJUAN

Setelah mempelajari bab ini diharapkan kalian akan mampu :

- Memahami karakteristik umum model proses dalam rekayasa perangkat lunak.
- Menyebutkan beberapa model rekayasa perangkat lunak .
- Mengetahui prinsip-prinsip dari metode *waterfall*, *prototyping*, dan *unified process*.
- Memahami tahapan-tahapan dalam rekayasa perangkat lunak.

2.1. MODEL PROSES REKAYASA PERANGKAT LUNAK

Pada rekayasa perangkat lunak, banyak model yang telah dikembangkan untuk membantu proses pengembangan perangkat lunak. Model-model ini pada umumnya mengacu pada model proses pengembangan sistem yang disebut ***System Development Life Cycle (SDLC)*** seperti terlihat pada Gambar 2.2.



Gambar 2.2. System Development Life Cycle (SDLC).

Setiap model yang dikembangkan mempunyai karakteristik sendiri-sendiri. Namun secara umum ada persamaan dari model-model ini, yaitu:

- *Kebutuhan terhadap definisi masalah yang jelas*. Input utama dari setiap model pengembangan perangkat lunak adalah pendefinisian masalah yang jelas. Semakin jelas akan semakin baik karena akan memudahkan dalam penyelesaian masalah. Oleh karena itu pemahaman masalah seperti dijelaskan pada Bab 1, merupakan bagian penting dari model pengembangan perangkat lunak.
- *Tahapan-tahapan pengembangan yang teratur*. Meskipun model-model pengembangan perangkat lunak memiliki pola yang berbeda-beda, biasanya model-model tersebut mengikuti pola umum *analysis – design – coding – testing - maintenance*.
- *Stakeholder berperan sangat penting dalam keseluruhan tahapan pengembangan*. Stakeholder dalam rekayasa perangkat lunak dapat berupa pengguna, pemilik, pengembang, pemrogram dan orang-orang yang terlibat dalam rekayasa perangkat lunak tersebut.
- *Dokumentasi merupakan bagian penting dari pengembangan perangkat lunak*. Masing-masing tahapan dalam model biasanya menghasilkan sejumlah tulisan, diagram, gambar atau bentuk-bentuk lain yang harus didokumentasi dan merupakan bagian tak terpisahkan dari perangkat lunak yang dihasilkan.
- *Keluaran dari proses pengembangan perangkat lunak harus bernilai ekonomis*. Nilai dari sebuah perangkat lunak sebenarnya agak susah di-rupiah-kan. Namun efek dari penggunaan perangkat lunak yang telah dikembangkan haruslah memberi nilai tambah bagi organisasi. Hal ini dapat

berupa penurunan biaya operasi, efisiensi penggunaan sumberdaya, peningkatan keuntungan organisasi, peningkatan "image" organisasi dan lain-lain.

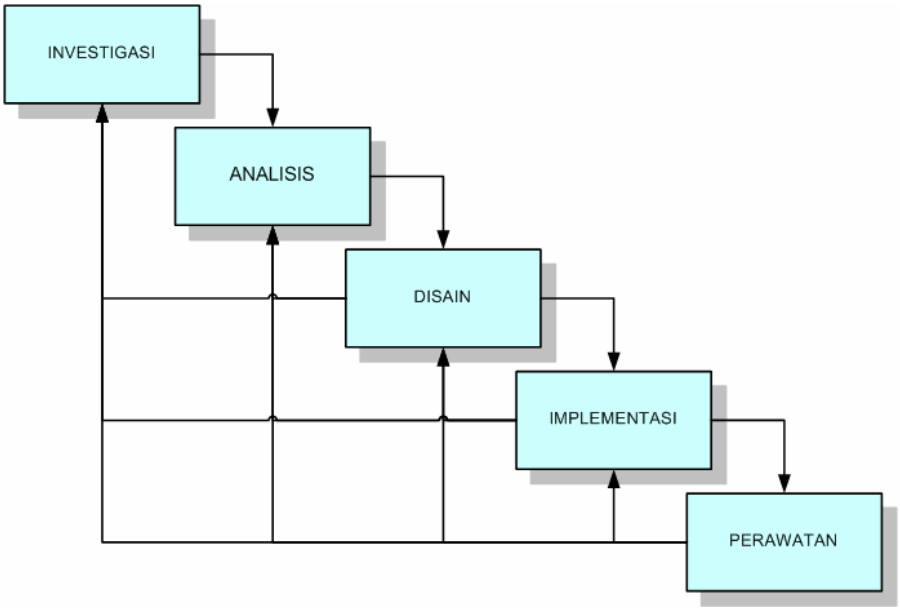
Ada banyak model pengembangan perangkat lunak, antara lain *The Waterfall Model, Joint Application Development (JAD), Information Engineering (IE), Rapid Application Development (RAD)* termasuk di dalamnya *Prototyping, Unified Process (UP), Structural Analysis and Design (SAD) dan Framework for the Application of System thinking (FAST).* Pada buku ini akan dibahas tiga model pengembangan yaitu *The Waterfall Model, Prototyping, dan Unified Processs (UP).*

2.1.1. The waterfall model

Model siklus hidup (*life cycle model*) adalah model utama dan dasar dari banyak model. Salah satu model yang cukup dikenal dalam dunia rekayasa perangkat lunak adalah *The Waterfall Model*. Ada 5 tahapan utama dalam *The Waterfall Model* seperti terlihat pada Gambar 2.3. Disebut *waterfall* (berarti air terjun) karena memang diagram tahapan prosesnya mirip dengan air terjun yang bertingkat.

Tahapan-tahapan dalam The Waterfall Model secara ringkas adalah sebagai berikut:

- Tahap investigasi dilakukan untuk menentukan apakah terjadi suatu masalah atau adakah peluang suatu sistem informasi dikembangkan. Pada tahapan ini studi kelayakan perlu dilakukan untuk menentukan apakah sistem informasi yang akan dikembangkan merupakan solusi yang layak
- Tahap analisis bertujuan untuk mencari kebutuhan pengguna dan organisasi serta menganalisa kondisi yang ada (sebelum diterapkan sistem informasi yang baru).
- Tahap disain bertujuan menentukan spesifikasi detil dari komponen-komponen sistem informasi (manusia, *hardware, software, network* dan data) dan produk-produk informasi yang sesuai dengan hasil tahap analisis.
- Tahap implementasi merupakan tahapan untuk mendapatkan atau mengembangkan *hardware* dan *software* (pengkodean program), melakukan pengujian, pelatihan dan perpindahan ke sistem baru.
- Tahapan perawatan (maintenance) dilakukan ketika sistem informasi sudah dioperasikan. Pada tahapan ini dilakukan monitoring proses, evaluasi dan perubahan (perbaikan) bila diperlukan.

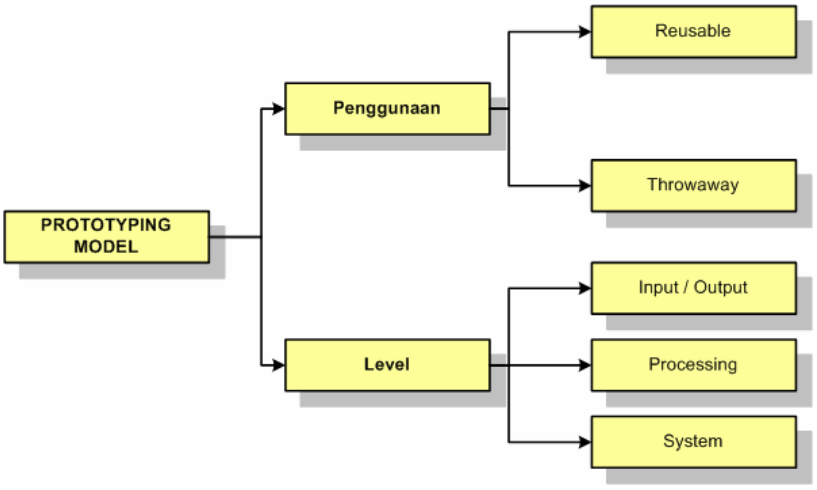


Gambar 2.3. The Waterfall Model

2.1.2. Prototyping model

Prototyping adalah s*alah satu pendekatan dalam rekayasa perangkat lunak yang secara langsung mendemonstrasikan bagaimana sebuah perangkat lunak atau komponen-komponen perangkat lunak akan bekerja dalam lingkungannya sebelum tahapan konstruksi aktual dilakukan* (Howard, 1997).

Prototyping model dapat diklasifikasikan menjadi beberapa tipe seperti terlihat pada gambar 2.4.



Gambar 2.4. Klasifikasi prototyping model (Harris, 2003)

- *Reusable prototype :*

  Prototype yang akan ditransformasikan menjadi produk final.

- *Throwaway prototype :*

  Prototype yang akan dibuang begitu selesai menjalankan maksudnya.

- *Input/output prototype :*

  Prototype yang terbatas pada antar muka pengguna (user interface).

- *Processing prototype :*

  Prototype yang meliputi perawatan file dasar dan proses-proses transaksi.

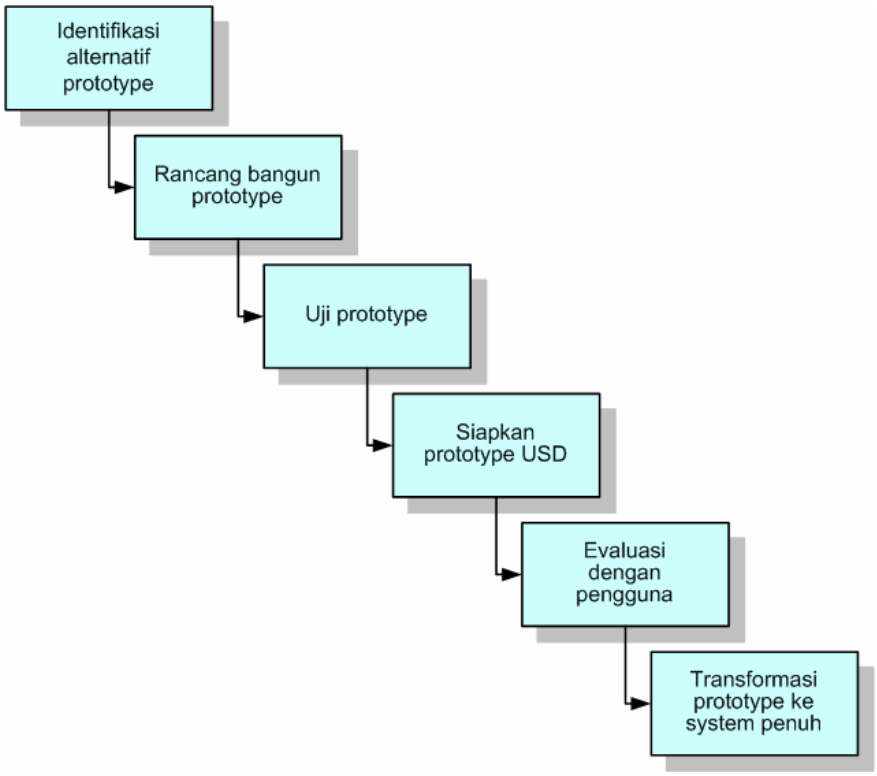
- *System prototype :*

  Prototype yang berupa model lengkap dari perangkat lunak.

Tahap-tahap dalam *prototyping* boleh dikata merupakan tahap-tahap yang dipercepat. Strategi utama dalam *prototyping* adalah kerjakan yang mudah terlebih dahulu dan sampaikan hasil kepada pengguna sesegera mungkin. Harris (2003) membagi prototyping dalam enam tahapan seperti terlihat pada gambar 2.5.

Tahapan-tahapan secara ringkas dapat dijelaskan sebagai berikut:

- *Identifikasi kandidat prototyping*. Kandidat dalam kasus ini meliputi *user interface* (menu, dialog, input dan output), file-file transaksi utama, dan fungsi-fungsi pemrosesan sederhana.
- *Rancang bangun prototype dengan bantuan software* seperti *word processor, spreadsheet, database*, pengolah grafik, dan software CASE (*Computer-Aided System Engineering*).
- *Uji prototype* untuk memastikan prototype dapat dengan mudah dijalankan untuk tujuan demonstrasi.
- *Siapkan prototype USD (User's System Diagram)* untuk mengidentifikasi bagian-bagian dari perangkat lunak yang di-*prototype*-kan.
- *Evaluasi dengan pengguna* untuk mengevaluasi *prototype* dan melakukan perubahan jika diperlukan.
- *Transformasikan prototype menjadi perangkat lunak yang beroperasi penuh* dengan melakukan penghilangan kode-kode yang tidak dibutuhkan, penambahan program-program yang memang dibutuhkan dan perbaikan dan pengujian perangkat lunak secara berulang.

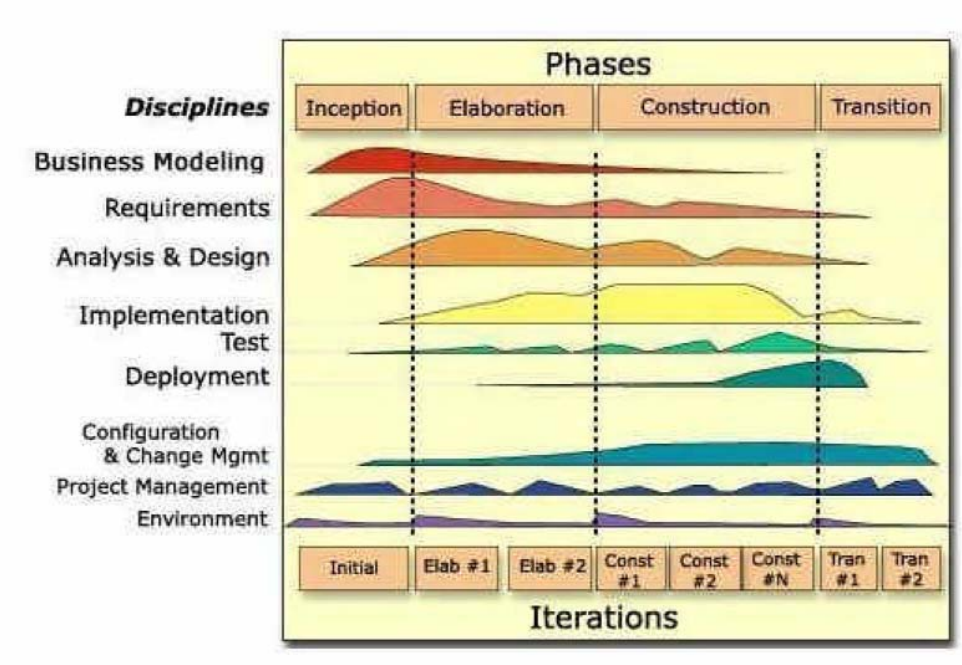


Gambar 2.5. Tahapan-tahapan prototyping model (Harris, 2003)

2.1.3. Unified Process dan Unified Modeling Language

Unified Process (UP) atau kadang disebut sebagai *Unified Software Development Process* (USDP) adalah kerangka proses pengembangan yang bersifat *use-case-driven*, berpusat pada arsitektur perangkat lunak, interatif dan tumbuh-kembang (Alhir, 2005). Kerangka pengembangan ini termasuk baru dalam metodologi pengembangan perangkat lunak. UP dapat diaplikasikan pada berbagai skala proyek, mulai dari skala kecil sampai dengan skala besar.

Daur hidup UP secara umum akan tampak seperti pada bagan di Gambar 2.6. Bagan ini biasa disebut sebagai "*hump chart*". Pada bagan ini terlihat ada empat tahap pengembangan yaitu *inception, elaboration, construction* dan *transition*. Selain itu tampak pula sejumlah aktivitas (*disciplines*) yang harus dilakukan sepanjang pengembangan perangkat lunak, yaitu, *business modeling, requirements, analysis and design, implementation, test*. Tahap dan aktivitas tersebut akan dilakukan secara iteratif (Ambler, 2005).



Gambar 2.6. RUP Life Cycle (Ambler, 2005).

Penjelasan singkat untuk empat tahapan dalam UP adalah sebagai berikut:

- *Inception*. Tahapan ini merupakan tahapan paling awal dimana aktivitas penilaian terhadap sebuah proyek perangkat lunak dilakukan. Tujuannya adalah untuk mendapatkan kesepakatan dari stakeholder sehubungan dengan tujuan dan dana proyek.
- *Elaboration*. Tujuan dari tahap ini adalah untuk mendapatkan gambaran umum kebutuhan, persyaratan dan fungsi-fungsi utama perangkat lunak. Hal ini penting untuk mengetahui secara lebih baik resiko-resiko proyek, baik meliputi resiko arsitektur perangkat lunak, perencanaan, maupun implementasi. Pada tahap ini telah dimulai rancang bangun perangkat lunak secara iterative melalui aktivitas-aktivitas seperti *business modeling, requirements, analysis* dan *design* meskipun baru pada tahap awal.
- *Construction*. Tujuan dari tahapan ini adalah membangun perangkat lunak sampai dengan saat perangkat lunak tersebut siap digunakan. Titik berat tahapan ini adalah pada penentuan tingkat prioritas kebutuhan / persyaratan, melengkapi spesifikasinya, analisis lebih dalam, disain solusi yang memenuhi kebutuhan dan persyaratan, pengkodean dan pengujian perangkat lunak. Jika dimungkinkan versi awal dari perangkat lunak diuji cobakan untuk mendapatkan masukan dari pengguna.

- *Transition.* Tahap ini difokuskan pada bagaimana menyampaikan perangkat lunak yang sudah jadi pada pengguna. Perangkat lunak akan secara resmi diuji oleh baik oleh penguji (tester) yang kompeten maupun oleh pengguna. Beberapa aktivitas seperti pemindahan pusat data dan pelatihan pengguna dan staf pendukung harus dilakukan pada tahap ini.

Dalam pengembangan perangkat lunak dengan menggunakan UP, maka tidak lepas dari penggunaan notasi-notasi yang biasa disebut sebagai **UML (*Unified Modeling Language*)**. Meskipun UP mensyaratkan penggunaan UML, namun UML sendiri dapat digunakan pada berbagai metodologi yang lain bahkan dapat digunakan pada bidang selain sistem informasi. UML adalah bahasa pemodelan standar atau kumpulan teknik-teknik pemodelan untuk men-spesifikasi, mem-visualisasi, meng-konstruksi dan mendokumentasi hasil kerja dalam pengembangan perangkat lunak (Fowler, 2004). UML lahir dari penggabungan banyak bahasa pemodelan grafis berorientasi obyek yang berkembang pesat pada akhir tahun 1980an dan awal 1990an.

Secara sederhana UML digunakan untuk menggambar sketsa sistem. Pengembang menggunakan UML untuk menyampaikan beberapa aspek dari sebuah perangkat lunak melalui notasi grafis. UML mendefinisikan notasi dan semantik. Notasi merupakan sekumpulan bentuk khusus yang memiliki makna tertentu untuk menggambarkan berbagai diagram piranti lunak dan semantik mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Ada beberapa jenis diagram yang disediakan dalam UML, antara lain adalah:

- *Use-case diagram*. Diagram ini berguna untuk menggambarkan interaksi antara pengguna dengan sebuah perangkat lunak
- *Activity diagram*. Diagram ini berguna untuk menggambarkan prosedur-prosedur perilaku perangkat lunak.
- *Class diagram*. Diagram ini berguna untuk menggambarkan class, fitur, dan hubungan-hubungan yang terjadi. Pada diagram ini pendekatan berorientasi obyek memegang peranan yang sangat penting.
- *Sequence diagram*. Diagram ini berguna untuk menggambarkan interaksi antar obyek dengan penekanan pada urutan proses atau kejadian.
- *State machine diagram*. Diagram ini digunakan untuk menggambarkan bagaimana suatu kejadian mengubah obyek selama masa hidup obyek tersebut.
- *Component diagram*. Diagram ini berguna untuk menggambarkan struktur dan koneksi komponen.

2.2. TAHAPAN REKAYASA PERANGKAT LUNAK

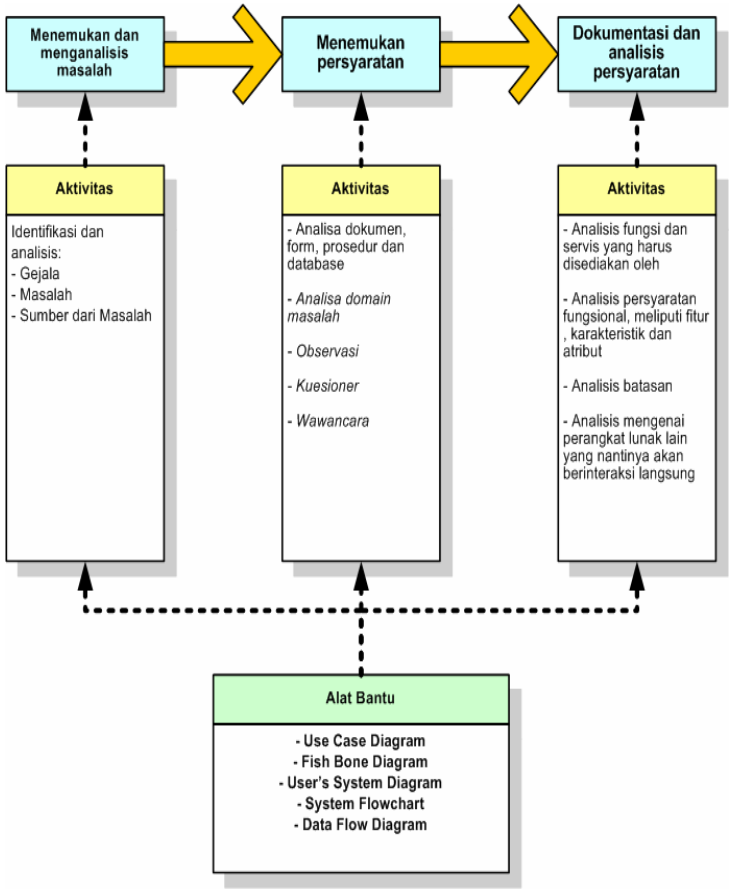
Seperti telah disebutkan, meskipun dalam pendekatan berbeda-beda, namun model-model di atas memiliki kesamaan, yaitu menggunakan pola tahapan *analysis – design – coding(construction) – testing – maintenance.*

2.2.1. Analisis

Analisis sistem adalah sebuah teknik pemecahan masalah yang menguraikan sebuah sistem menjadi komponen-komponennya dengan tujuan mempelajari seberapa bagus komponen-komponen tersebut bekerja dan berinteraksi untuk meraih tujuan mereka.

Analisis mungkin adalah bagian terpenting dari proses rekayasa perangkat lunak. Karena semua proses lanjutan akan sangat bergantung pada baik tidaknya hasil analisis. Tahapan-tahapan dalam analisis rekayasa perangkat lunak secara ringkas dapat dilihat pada Gambar 2.7.

Ada satu bagian penting yang biasanya dilakukan dalam tahapan analisis yaitu pemodelan proses bisnis. **Model proses** adalah model yang memfokuskan pada seluruh proses di dalam sistem yang mentransformasikan data menjadi informasi (Harris, 2003). Model proses juga menunjukkan aliran data yang masuk dan keluar pada suatu proses. Biasanya model ini digambarkan dalam bentu Diagram Arus Data (Data Flow Diagram / DFD). DFD meyajikan gambaran apa yang manusia, proses dan prosedur lakukan untuk mentransformasi data menjadi informasi.



Gambar 2.7. Tahapan dan aktifitas dalam analisis.

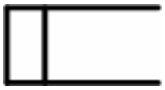
Umumnya ada empat notasi yang sering digunakan dalam DFD seperti tampak Gambar 2.8.

External Entitiy

External Entity melambangkan sumber data (dari mana data berasal) atau penerima informasi (tujuan akhir dari data). Contoh external entity antara lain konsumen yang memesan suatu produk, manajer yang mengevaluasi laporan penjualan mingguan, dan lain-lain.

Process

Proses adalah serangkaian langkah yang dilakukan untuk memanipulasi data, misalnya pengumpulan, pengurutan, pemilihan, pelaporan, peringkasan, analisis dan lain-lain.



Data Store

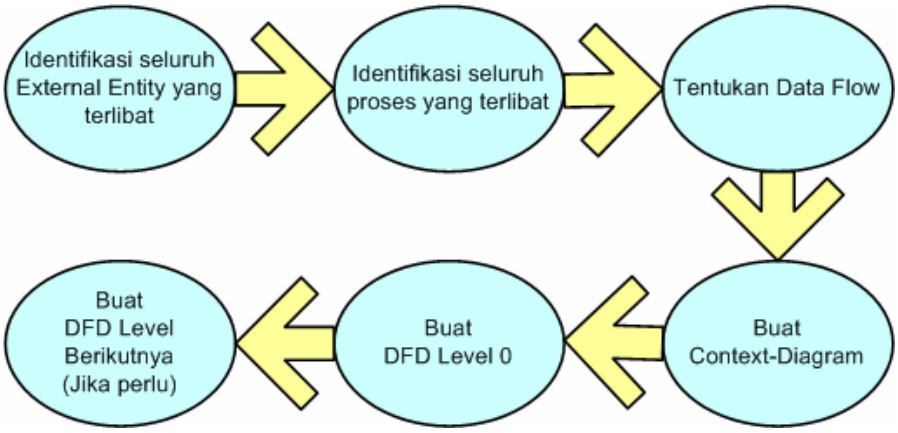
Data store adalah tempat untuk menyimpan data untuk digunakan kemudian. Nama yang pada data store ini merupakan abstraksi dari data yang disimpan. Namun detil / item data apa saja yang ada, bagaimana cara akses, atau bagaimana mengorganisasinya tidak dijelaskan dalam notasi ini.

Data Flow

Data flow menunjukkan aliran data dari satu tempat ke tempat lain. Perpindahan data ini dapat dari external entity ke proses, antar proses satu dengan yang lain, dari proses ke data store. Dalam penggambarannya setiap data flow harus diberi label yang menunjukkan data apa yang mengalir.

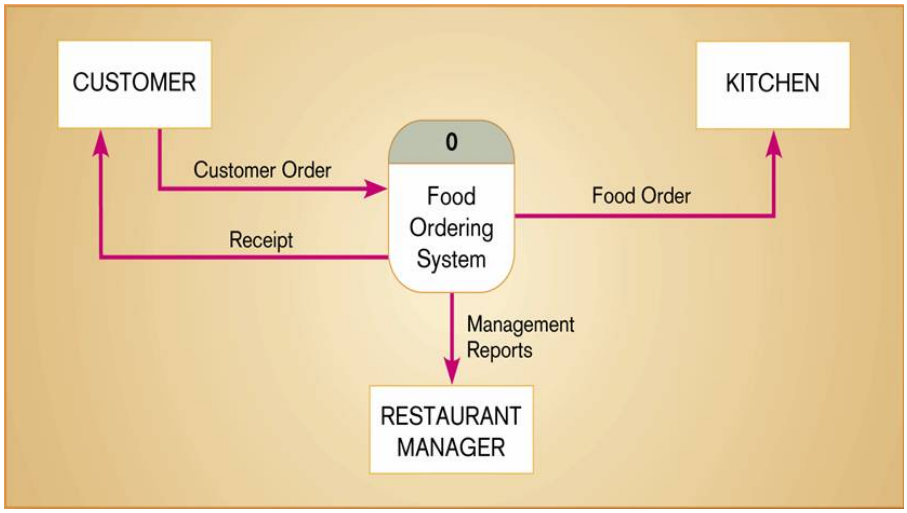
Gambar 2.8. Notasi pada DFD.

Dalam pembuatan DFD ada beberapa tahapan yang dilakukan secara berurutan. Gambar 2.9. menunjukkan urutan tahapan tersebut.



Gambar 2.9. Tahapan pembuatan DFD.

Context diagram adalah DFD ruang lingkup dari sistem yang menunjukkan batas-batas sistem, *external entitiy* yang berinteraksi dengan sistem dan aliran data utama antara *external entity* dengan sistem. *Context diagram* menggambarkan keseluruhan sistem dalam suatu proses tunggal. Gambar 2.10 menunjukkan sebuah contoh *context diagram.*

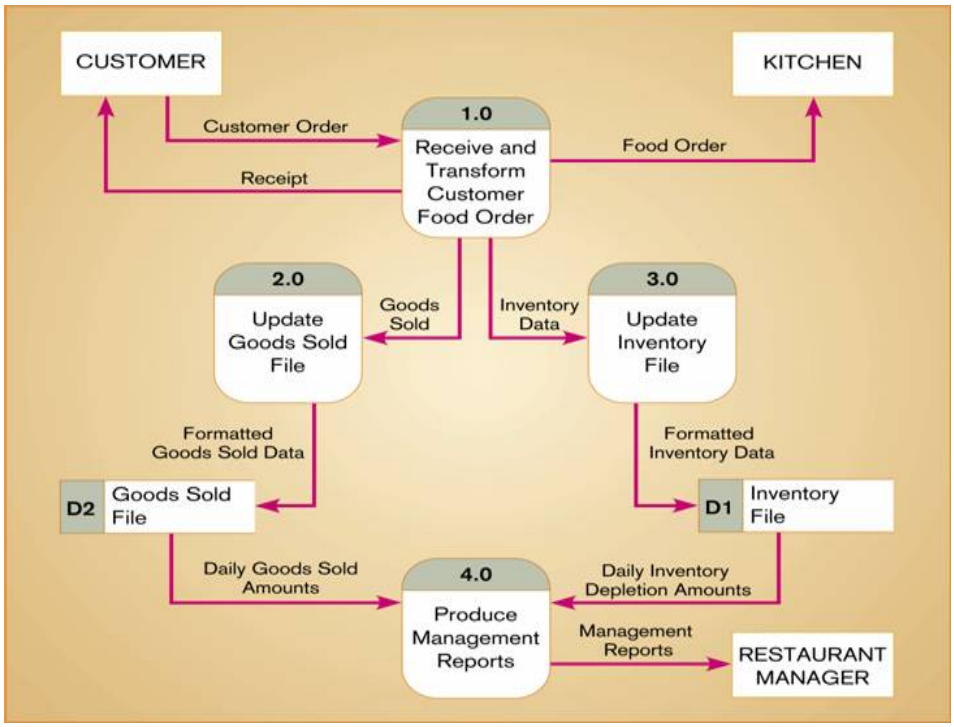


Gambar 2.10. *Context diagram* sistem pemesanan makanan (Hoffer et al., 2002).

Context diagram pada Gambar 2.10 tampak hanya ada satu proses tunggal yang merepresentasikan sistem yang dimodelkan. Pada proses ini diberi notasi angka 0 untuk menunjukkan ini adalah level paling abstrak dari sistem. Selain itu ada tiga *external entity* yaitu *customer, kitchen* dan *restaurant*

manager. Ketiganya dapat berperan sebagai sumber data (dalam contoh di atas adalah *customer*) atau sebagai penerima informasi (dalam contoh di atas *customer, kitchen*, dan *restaurant manager*). *Data flow* yang tampak pada gambar menunjukkan ada satu *data flow* yang masuk ke sistem dan ada tiga *data flow* yang keluar dari sistem. Masing-masing *data flow* diberi label yang menunjukkan data apa yang sedang mengalir.

Setelah context diagram terbentuk dengan benar maka langkah selanjutnya adalah merinci context diagram tersebut dalam DFD Level 0. DFD Level 0 adalah DFD yang merepresentasikan proses-proses, data flow dan data storage utama di dalam sistem. DFD Level 0 ini akan digunakan sebagai dasar untuk membangun DFD yang level dibawahnya (Level 1, 2, 3, .. dst) atau biasa disebut sebagai dekomposisi DFD. Gambar 3.10 merupakan DFD level 0 dari context-diagram pada gambar 2.11.



Gambar 2.11. DFD Level 0.

Pada gambar 2.11 tampak adanya pemecahan pada proses dari yang semula hanya satu menjadi empat. Masing-masing proses diberi nomor kode 1.0, 2.0, 3.0 dan 4.0. Jumlah *external entity* harus tetap yaitu 3 demikian pula *data flow* yang keluar dan masuk (input dan output) ke dalam sistem harus sama dengan pada context diagram. Sedangkan *data flow* yang berada di dalam sistem (yang mengalir antar proses dan atau data storage) tergantung pada proses dan *data storage* yang terlibat. Ada dua *data storage* yaitu *Goods Sold*

File dan *Inventory File*. Kedua *data storage* ini digunakan untuk menyimpan data dari suatu proses. Data ini juga akan dibaca / diakses oleh proses yang lain. Sebagai contoh *data storage Inventory File* berisi data hasil proses 3.0 (*Update Inventory File*). Data ini akan digunakan proses 4.0 (*Produce Management Reports*) untuk membuat laporan yang akan disampaikan pada *Restaurant Manager*.

DFD level berikutnya yaitu level 1, 2 dan seterusnya diperlukan apabila level sebelumnya dirasa kurang detil. Sebagai contoh apabila DFD level 0 (Gambar 14.12) dirasa belum cukup detil menunjukkan arus data yang mengalir, maka dapat dibuat detilnya pada DFD level 1. Bagian yang harus didetilkan biasanya adalah proses. Detil pada level berikutnya, mungkin pada semua proses atau hanya pada proses-proses tertentu saja. DFD pada level 0 maupun level di bawahnya memiliki kesamaan aturan yang tersaji berikut pada tabel berikut ini.

Tabel 2.1. Aturan-aturan dalam DFD

Kelompok	Aturan
Umum	• input-input ke suatu process akan selalu berbeda dengan output-outputnya • obyek obyek (External Entity, Process, Data Storage, dan Data Flow) yang ada pada suatu DFD selalu memiliki nama yang unik
External Entity	• nama yang dipakai pada External Entity selalu menggunakan kata benda • data tidak boleh mengalir secara langsung dari External Entity yang satu ke External Entity yang lain
Process	• nama yang dipakai pada Process selalu menggunakan kata kerja • tidak ada Process yang hanya menghasilkan output • tidak ada Process yang hanya menerima input
Data Storage	• nama yang dipakai pada Data Storage selalu menggunakan kata benda • data tidak boleh mengalir secara langsung dari Data Storage yang satu ke Data Storage yang lain • data tidak boleh mengalir secara langsung dari External Entity ke Data Storage demikian juga sebaliknya.
Data Flow	• nama yang dipakai pada Data Flow selalu menggunakan kata benda • Data Flow di antara dua notasi hanya memiliki satu arah aliran • Percabangan (fork) menunjukkan adanya data yang persis sama yang mengalir dari suatu tempat ke dua atau lebih tempat yang lain • Penggabungan (join) menunjukkan adanya data yang persis sama yang mengalir dua atau lebih tempat menuju satu tempat yang lain • Data Flow menuju Data Storage berarti terjadi update data • Data Flow dari Data Storage berarti terjadi pembacaan / pengambilan data

2.2.2. Disain

Disain perangkat lunak adalah tugas, tahapan atau aktivitas yang difokuskan pada spesifikasi detil dari solusi berbasis computer (Whitten et al, 2004).

Disain perangkat lunak sering juga disebut sebagai *physical design*. Jika tahapan analisis sistem menekankan pada masalah bisnis (business rule), maka sebaliknya disain perangkat lunak fokus pada sisi teknis dan implementasi sebuah perangkat lunak (Whitten et al, 2004).

Output utama dari tahapan disain perangkat lunak adalah spesifikasi disain. Spesifikasi ini meliputi spesifikasi disain umum yang akan disampaikan kepada stakeholder sistem dan spesifikasi disain rinci yang akan digunakan pada tahap implementasi. Spesifikasi disain umum hanya berisi gambaran umum agar stakeholder sistem mengerti akan seperti apa perangkat lunak yang akan dibangun. Biasanya diagram USD tentang perangkat lunak yang baru merupakan point penting dibagian ini. Spesifikasi disain rinci atau kadang disebut disain arsitektur rinci perangkat lunak diperlukan untuk merancang sistem sehingga memiliki konstruksi yang baik, proses pengolahan data yang tepat dan akurat, bernilai, memiliki aspek *user friendly* dan memiliki dasar-dasar untuk pengembangan selanjutnya.

Desain arsitektur ini terdiri dari desain database, desain proses, desain *user interface* yang mencakup desain *input, output form dan report,* desain hardware, software dan jaringan. Desain proses merupakan kelanjutan dari pemodelan proses yang dilakukan pada tahapan analisis.

2.2.3. Konstruksi

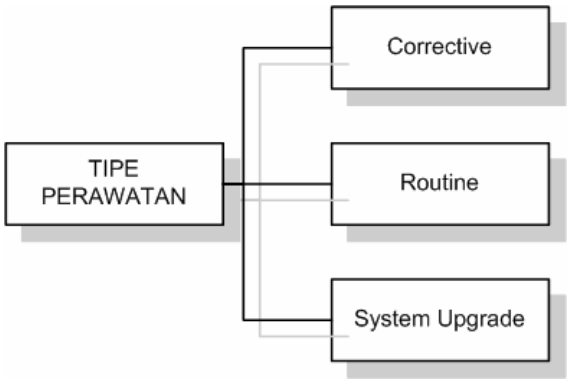
Konstruksi adalah tahapan menerjemahkan hasil disain logis dan fisik ke dalam kode-kode program computer. Buku ini sebagian besar berisi tentang bagian ini.

2.2.4. Pengujian

Pengujian sistem melibatkan semua kelompok pengguna yang telah direncanakan pada tahap sebelumnya. Pengujian tingkat penerimaan terhadap perangkat lunak akan berakhir ketika dirasa semua kelompok pengguna menyatakan bisa menerima perangkat lunak tersebut berdasarkan criteria-kriteria yang telah ditetapkan.

2.2.5. Perawatan dan Konfigurasi

Ketika sebuah perangkat lunak telah dianggap layak untuk dijalankan, maka tahapan baru menjadi muncul yaitu perawatan perangkat lunak. Ada beberapa tipe perawatan yang biasa dikenal dalam dunia perangkat lunak seperti terlihat pada diagram di Gambar 2.12.



Gambar 3.12. Tipe-tipe perawatan.

- Tipe perawatan *corrective* dilakukan jika terjadi kesalahan atau biasa dikenal sebagai bugs. Perawatan bisa dilakukan dengan memperbaiki kode program, menambah bagian yang dirasa perlu atau malah menghilangkan bagian-bagian tertentu.
- Tipe perawatan *routine* biasa juga disebut preventive maintenance dilakukan secara rutin untuk melihat kinerja perangkat lunak ada atau tidak ada kesalahan.
- Tipe perawatan *sistem upgrade* dilakukan jika ada perubahan dari komponen-komponen yang terlibat dalam perangkat lunak tersebut. Sebagai contoh perubahan platform sistem operasi dari versi lama ke versi baru menyebabkan perangkat lunak harus diupgrade.

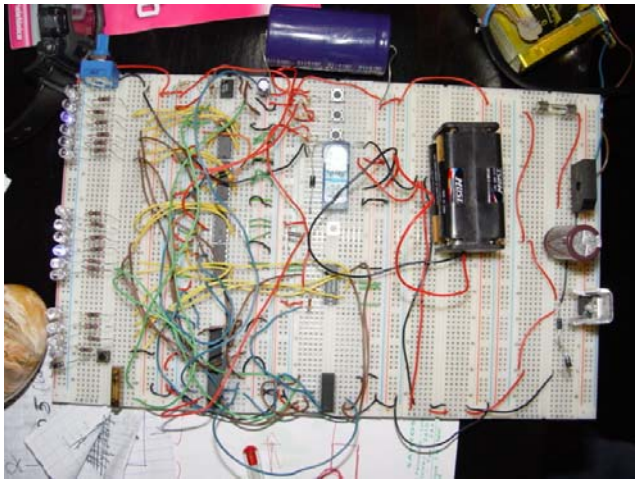
2.3. RINGKASAN

- Model-model rekayasa perangkat lunak pada umumnya mengacu pada model proses pengembangan sistem yang disebut ***System Development Life Cycle (SDLC).***
- Model pengembangan perangkat lunak yang sekarang umum digunakan adalah *The Waterfall Model, Prototyping,* dan *Unified Process (UP).*
- Tahapan-tahapan utama dalam rekayasa perangkat lunak meliputi : analisis, disain, konstruksi, pengujian dan perawatan.

2.4. SOAL-SOAL LATIHAN

1. Sebutkan tahapan-tahapan dalam System Developmen Life Cycle (SDLC)
2. Sebutkan persamaan karakteristik yang dimiliki oleh model-model pengembangan perangkat lunak.
3. Sebutkan lima model pengembangan perangkat lunak yang anda ketahui.
4. Apakah yang dimaksud tahapan konstruksi pada rekayasa perangkat lunak?
5. Gambarkan notasi-notasi dalam Data Flow Diagram yang anda ketahui.

BAB 3 ELEKTRONIKA DAN SISTEM KOMPUTER



Gambar 3.1. Rangkaian dan perangkat elektronik.

Kalau kalian pernah membuka atau melihat-lihat bagian dalam sebuah perangkat elektronik, maka kalian akan menjumpai kondisi yang mirip dengan Gambar 3.1. di atas. Ada papan circuit, kabel-kabel yang saling berhubungan, dan perangkat elektronik lainnya. Perangkat elektronik seperti inilah yang juga menyusun sebuah perangkat komputer. Sehingga pemahaman tentang elektronika, elektronika digital dan sistem komputer menjadi penting bagi kalian yang ingin berkecimpung dalam dunia rekayasa perangkat lunak.

Bab ini akan membahas dua standar kompetensi, yaitu teknik elektronika dasar dan teknik elektronika digital, terutama yang berhubungan dengan sistem komputer. Ada dua kompetensi dasar pada standar kompetensi teknik elektronika dasar, yaitu memahami prinsip-prinsip elektronika dasar dan mengetahui komponen-komponen elektronika dasar. Sedangkan standar kompetensi untuk teknik elektronika digital terdiri dari kompetensi dasar menguasai teknik elektronika digital dan menguasai teknik elektronika digital untuk komputer. Dalam penyajian pada buku ini, setiap kompetensi dasar memuat uraian materi. Ringkasan diletakkan pada akhir bab. Sebelum mempelajari kompetensi ini ingatlah kembali tentang teknik elektronika dasar dan materi-materi pendukung dari mata pelajaran matematika.

TUJUAN

Setelah mempelajari bab ini diharapkan kalian akan mampu :

- Menguasai konsep elektronika dasar.
- Mengetahui komponen-komponen elektronika.
- Menguasai konsep eletronika digital.
- Menguasai elektronika digital dan sistem komputer.

3.1. DASAR ELEKTRONIKA

3.1.1. Konsep Dasar Elektronika

Elektronika adalah ilmu yang mempelajari alat listrik **arus lemah** yang dioperasikan dengan cara mengontrol aliran elektron atau partikel bermuatan listrik dalam suatu alat seperti komputer, peralatan elektronik, termokopel, semikonduktor, dan lain sebagainya. Ilmu yang mempelajari alat-alat seperti ini merupakan cabang dari ilmu fisika, sementara bentuk desain dan pembuatan sirkuit elektroniknya adalah bagian dari teknik elektro, teknik komputer, dan ilmu/ teknik elektronika dan instrumentasi.

Alat-alat yang menggunakan dasar kerja elektronika ini biasanya disebut sebagai peralatan elektronik (*electronic devices*). Contoh peralatan/ piranti elektronik ini: Tabung Sinar Katoda (*Cathode Ray Tube*, CRT), radio, TV, perekam kaset, perekam kaset video (VCR), perekam VCD, perekam DVD, kamera video, kamera digital, komputer pribadi desk-top, komputer Laptop, PDA (komputer saku), robot, smart card, dll.

Seperti disebutkan di atas elektronika didasarkan pada pengetahuan tentang kelistrikan. **Listrik**, dapat diartikan sebagai berikut:

- Listrik adalah kondisi dari partikel subatomik tertentu, seperti elektron dan proton, yang menyebabkan penarikan dan penolakan gaya di antaranya.
- Listrik adalah sumber energi yang disalurkan melalui kabel. Arus listrik timbul karena muatan listrik mengalir dari saluran positif ke saluran negatif.

Ada 2 jenis muatan listrik: positif dan negatif. Melalui eksperimen, muatan-sejenis saling menolak dan muatan-lawan jenis saling menarik satu sama lain. Besarnya gaya menarik dan menolak ini ditetapkan oleh hukum Coulomb. **Hukum Coulomb** adalah hukum yang menjelaskan hubungan antara gaya yang timbul antara dua titik muatan, yang terpisahkan jarak tertentu, dengan nilai muatan dan jarak pisah keduanya. Satuan unit SI dari muatan listrik adalah coulomb, yang memiliki singkatan "C". Simbol *Q* digunakan dalam persamaan untuk mewakili kuantitas listrik atau muatan. Contohnya, "*Q*=0,5 C" berarti "kuantitas muatan listrik adalah 0,5 coulomb".

Jika listrik mengalir melalui bahan khusus, misalnya dari *wolfram* dan *tungsten*, cahaya pijar akan dipancarkan oleh logam itu. Bahan-bahan seperti itu dipakai dalam bola lampu (*bulblamp* atau *bohlam*). Setiap kali listrik mengalir melalui bahan yang mempunyai hambatan, maka akan dilepaskan panas.

Semakin besar arus listrik, maka panas yang timbul akan berlipat. Sifat ini dipakai pada elemen setrika dan kompor listrik.

Hambatan listrik adalah perbandingan antara tegangan listrik dari suatu komponen elektronik (misalnya *resistor*) dengan arus listrik yang melewatinya. Hambatan listrik dapat dirumuskan sebagai berikut:

$$\mathbf{R} = V/I$$

atau

$$\mathbf{R} = \delta V/I$$

di mana V adalah tegangan dan I adalah arus.

Tegangan listrik (kadang disebut sebagai **Voltase**) adalah perbedaan potensial listrik antara dua titik dalam rangkaian listrik, dan dinyatakan dalam satuan volt. Besaran ini mengukur energi potensial dari sebuah medan listrik yang mengakibatkan adanya aliran listrik dalam sebuah konduktor listrik. Tergantung pada perbedaan potensial listriknya, suatu tegangan listrik dapat dikatakan sebagai ekstra rendah, rendah, tinggi atau ekstra tinggi.

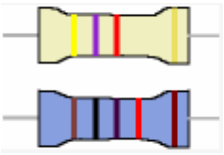
Arus listrik adalah banyaknya muatan listrik yang mengalir tiap satuan waktu. Muatan listrik bisa mengalir melalui kabel atau penghantar listrik lainnya.

$$I = \frac{Q}{t}$$

Pada zaman dulu, **Arus konvensional** didefinisikan sebagai aliran muatan positif, sekalipun kita sekarang tahu bahwa arus listrik itu dihasilkan dari aliran elektron yang bermuatan negatif ke arah yang sebaliknya. Satuan SI untuk arus listrik adalah **ampere** (**A**).

3.1.2. Komponen-Komponen Elektronika

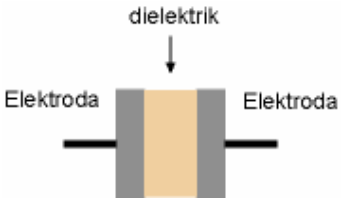
- **Resistor**



Gambar 3.2. Resistor.

Resistor adalah komponen dasar elektronika yang digunakan untuk membatasi jumlah arus yang mengalir dalam satu rangkaian. Sesuai dengan namanya resistor bersifat resistif dan umumnya terbuat dari bahan karbon. Satuan resistansi dari suatu resistor disebut Ohm atau dilambangkan dengan simbol W (Omega). Tipe resistor yang umum adalah berbentuk tabung dengan dua kaki tembaga di kiri dan kanan. Pada badannya terdapat lingkaran membentuk gelang kode warna untuk memudahkan pemakai mengenali besar resistansi tanpa mengukur besarnya dengan Ohmmeter.

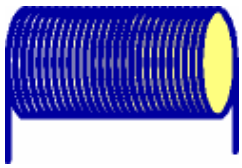
- **Kapasitor**



Gambar 3.3. Kapasitor.

Kapasitor adalah komponen elektronika yang dapat menyimpan muatan listrik. Struktur sebuah kapasitor terbuat dari 2 buah plat metal yang dipisahkan oleh suatu bahan dielektrik. Bahan-bahan dielektrik yang umum dikenal misalnya udara vakum, keramik, gelas dan lain-lain. Jika kedua ujung plat metal diberi tegangan listrik, maka muatan-muatan positif akan mengumpul pada salah satu kaki (elektroda) metalnya dan pada saat yang sama muatan-muatan negatif terkumpul pada ujung metal yang satu lagi.

- **Induktor**



Gambar 3.4. Induktor.

Induktor adalah komponen yang dapat menyimpan energi magnetik. Energi ini direpresentasikan dengan adanya tegangan emf (electromotive force) jika induktor dialiri listrik. Fungsi utama dari induktor di dalam suatu rangkaian adalah untuk melawan fluktuasi arus yang melewatinya. Aplikasinya pada rangkaian dc salah satunya adalah untuk menghasilkan tegangan dc yang konstan terhadap fluktuasi beban arus. Pada aplikasi rangkaian ac, salah satu gunanya adalah bisa untuk meredam perubahan fluktuasi arus yang tidak dinginkan. Akan lebih banyak lagi fungsi dari induktor yang bisa diaplikasikan pada rangkaian filter, tuner dan sebagainya.

3.2. ELEKTRONIKA DIGITAL

3.2.1. Pengertian Elektronika Digital

Elektronika digital adalah sistem elektronik yang menggunakan signal digital. Signal digital didasarkan pada signal yang bersifat terputus-putus. Biasanya dilambangkan dengan notasi aljabar 1 dan 0. Notasi 1 melambangkan terjadinya hubungan dan notasi 0 melambangkan tidak terjadinya hubungan. Contoh yang paling gampang untuk memahami pengertian ini adalah saklar lampu. Ketika kalian tekan ON berarti terjadi hubungan sehingga dinotasikan 1. Ketika kalian tekan OFF maka akan berlaku sebaliknya.

Elektronik digital merupakan aplikasi dari aljabar boolean dan digunakan pada berbagai bidang seperti komputer, telpon selular dan berbagai perangkat lain. Hal ini karena elektronik digital mempunyai beberapa keuntungan, antara lain: sistem digital mempunyai antar muka yang mudah dikendalikan dengan komputer dan perangkat lunak, penyimpanan informasi jauh lebih mudah dilakukan dalam sistem digital dibandingkan dengan analog. Namun sistem

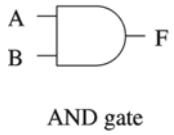
digital juga memiliki beberapa kelemahan, yaitu: pada beberapa kasus sistem digital membutuhkan lebih banyak energi, lebih mahal dan rapuh.

3.2.2. Gerbang Logika

Elektronik digital atau atau rangkaian digital apapun tersusun dari apa yang disebut sebagai gerbang logika. Gerbang logika melakukan operasi logika pada satu atau lebih input dan menghasilkan ouput yang tunggal. Output yang dihasilkan merupakan hasil dari serangkaian operasi logika berdasarkan prinsip-prinsip aljabar boolean. Dalam pengertian elektronik, input dan output ini diwujudkan dan voltase atau arus (tergantung dari tipe elektronik yang digunakan).

Setiap gerbang logika membutuhkan daya yang digunakan sebagai sumber dan tempat buangan dari arus untuk memperoleh voltase yang sesuai. Pada diagram rangkaian logika, biasanya daya tidak dicantumkan.

Dalam aplikasinya, gerbang logika adalah blok-blok penyusun dari perangkat keras elektronik. Gerbang logika ini dibuat dengan menggunakan transistor. Seberapa banyak transistor yang dibutuhkan, tergantung dari bentuk gerbang logika. Dasar pembentukan gerbang logika adalah tabel kebenaran (truth table). Ada tiga bentuk dasar dari tabel kebenaran yaitu AND, OR, dan NOT. Berikut adalah tabel-tabel dan bentuk gerbang logikanya.



AND gate

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

OR gate

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

NOT gate

A	F
0	1
1	0

Logic symbol

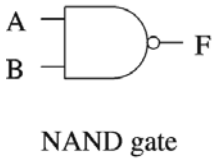
Truth table

Gambar 3.5. Tabel kebenaran dan representasinya dalam gerbang logika.

Penjelasan dari Gambar 3.5 di atas adalah sebagai berikut:

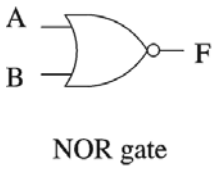
- Pada AND, bila ada dua buah input A dan B maka output atau signal hanya dihasilkan jika A = 1 dan B = 1.
- Pada OR, bila ada dua buah input A dan B maka output atau signal akan dihasilkan jika salah satu atau kedua input bernilai 1
- Pada NOT, bila ada satu input mempunyai nilai tertentu maka operasi NOT akan menghasilkan output / signal yang merupakan kebalikan dari nilai inputnya.

Selain bentuk dasar di atas, beberapa bentuk yang merupakan turunan dari bentuk dasar juga penting diketahui. Gambar 3.6. menampilkan bentuk tabel kebenaran dan gerbang logika NAND, NOR, dan XOR. NAND adalah hasil operasi NOT + AND, NOR adalah operasi NOT + OR sedangkan XOR adalah ekslusif OR. NAND dan NOR merupakan bentuk gerbang logika yang banyak sekali digunakan untuk membangun perangkat elektronik digital.



NAND gate

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



NOR gate

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

XOR gate

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

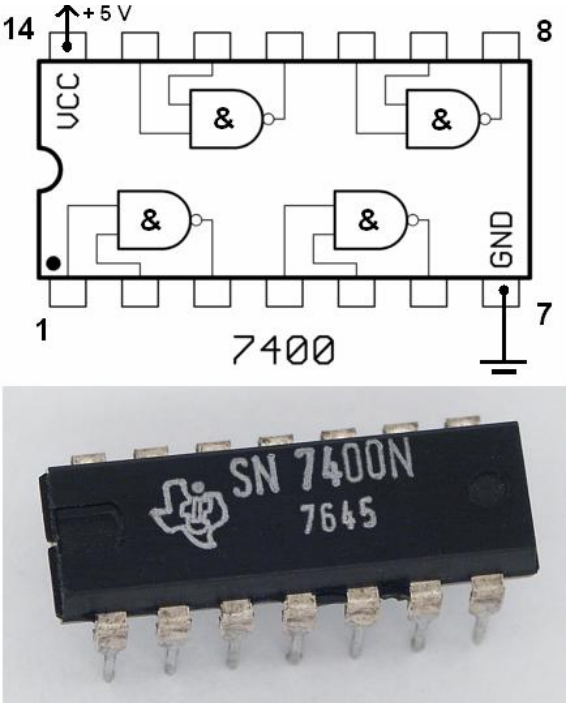
Logic symbol

Truth table

Gambar 3.6. Bentuk turunan tabel kebenaran dan representasinya dalam gerbang logika.

3.2.3. Rangkaian Digital

Pada sub bab di atas kita telah belajar tentang bentuk-bentuk gerbang logika berdasarkan tabel kebenaran. Sebuah rangkaian digital sebenarnya disusun dari satu atau lebih gerbang logika ini. Perhatikan contoh pada Gambar 3.7. berikut ini. Kalau kita perhatikan pada gambar tersebut, pada bagian atas terlihat ada empat notasi gerban logika NAND, satu pin untuk sumber daya 5 V dan satu pin untuk ground. Sedangkan pada bagian bawah adalah representasi dari rangkaian digital ini, yaitu sebuah chip 7400.



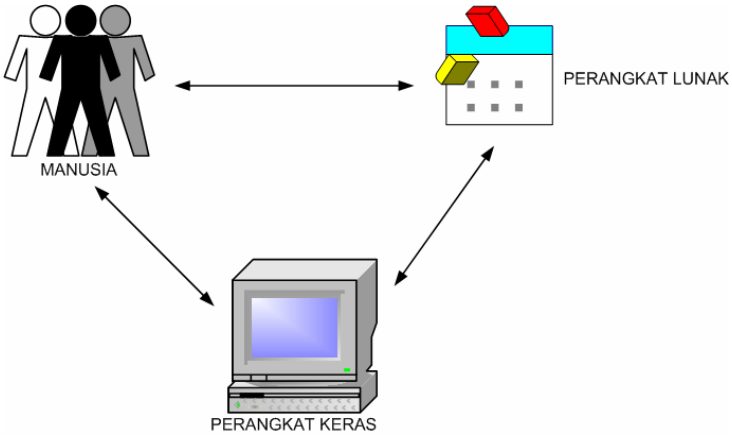
Gambar 3.7. Contoh rangkaian digital dan representasinya pada hardware.

3.3. SISTEM KOMPUTER

Istilah komputer berasal dari bahasa Latin "Computare" yang berarti menghitung. Oleh karena itu sebenarnya setiap alat yang berfungsi sebagai alat hitung seperti mesin penjumlah, kalkulator, atau bahkan simpoa (abacus) secara teknis dapat disebut sebagai komputer. Namun dalam perkembangannya, komputer mempunyai perkembangan arti yang berbeda.

Komputer adalah alat pengolah data elektronik yang bekerja dan dikontrol oleh sekumpulan instruksi (program) (Blissmer, 1985). **Sistem komputer** adalah kumpulan elemen-elemen yaitu manusia, perangkat keras, dan perangkat lunak yang saling berinteraksi untuk mencapai tujuan yaitu mendapatkan informasi yang berguna, kemudahan dalam bekerja, kecepatan dan tujuan lainnya.

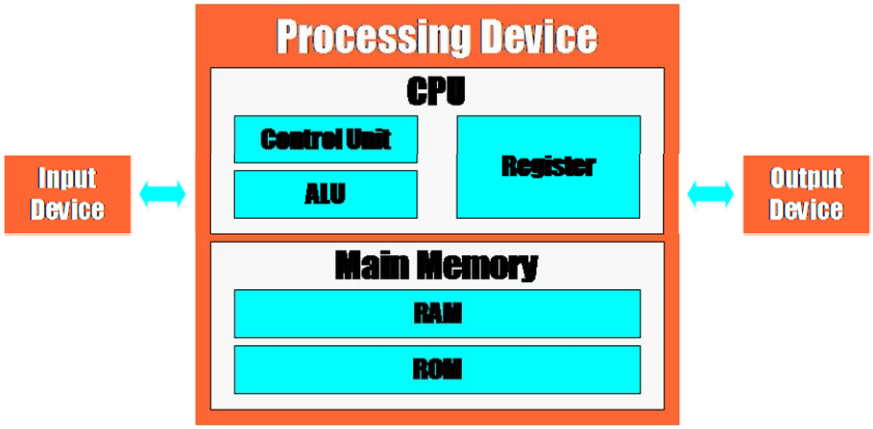
Ada tiga komponen utama dalam sistem komputer yaitu manusia sebagai pengguna, perangkat keras dan perangkat lunak (Gambar 3.8). Apabila satu tidak ada maka sistem komputer menjadi tidak bekerja. Sebagai contoh jika hanya ada manusia dan perangkat keras, maka sistem komputer tidak bekerja karena tidak program yang membantu manusia menjalankan perangkat keras.



Gambar 3.8. Sistem Komputer.

3.1.1 Perangkat keras

Perangkat keras adalah s*emua bagian fisik computer.* Perangkat keras dibedakan dengan data yang berada di dalamnya atau yang beroperasi di dalamnya, dan perangkat lunak yang menyediakan instruksi buat perangkat keras untuk menyelesaikan tugasnya. Secara umum ada empat komponen dasar pada komputer yang saling terkait (Lihat Gambar 3.9).



Gambar 3.9. Komponen dasar komputer

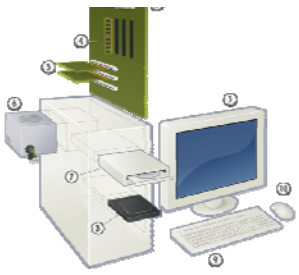
- Unit Masukan (Input), yaitu *perangkat yang memungkinkan pengguna memasukkan data atau perintah ke dalam komputer*. Contoh perangkat

yang termasuk dalam unit masukan adalah : keyboard, mouse, joystick, dan digitizer.

- Unit Keluaran (Output), yaitu *perangkat yang memungkinkan pengguna menerima informasi hasil pemrosesan oleh komputer*. Contoh perangkat yang termasuk dalam unit keluaran adalah : monitor, printer, dan plotter.
- Unit Memori Utama (Main memory), yaitu *perangkat yang digunakan untuk menyimpan data, program, dan informasi hasil pemrosesan komputer pada saat pemrosesan*. Unit memori utama terdiri dari banyak sel, yang masing-masing dapat menyimpan satu satuan informasi. Unit memori utama terdiri dari dua bagian, yaitu **ROM (Read Only Memory)** dan **RAM (Random Access Memory)**. ROM hanya dapat ditulisi sekali saja dan selanjutnya hanya dapat dibaca. RAM dapat ditulisi, dihapus dan dibaca berulang kali. Data, program, dan informasi yang sedang diproses disimpan dalam RAM ini, dan akan hilang apabila komputer dimatikan. Itu sebabnya data, program, dan informasi yang akan digunakan lagi disimpan dalam media penyimpanan tambahan (secondary storage) seperti, hard disk, disket, CD, tape dan lain-lain.
- Unit Pemrosesan Pusat (*Central Processing Unit*), yaitu bagian yang digunakan untuk memproses data, program, dan informasi pada komputer. Ada dua bagian penting dalam CPU yaitu *Arithmetic and Logical Unit* (ALU) dan *Control Unit*. Banyak orang menyebutkan ALU adalah jantung dari sebuah komputer. ALU bertanggung jawab pada dua operasi dasar yaitu operasi aritmatik dan perbandingan. Sedangkan *Control Unit* bertanggung jawab untuk menkoordinasi semua aktivitas unit-unit lain, misalnya bagaimana keyboard dapat dikenali dan bekerja sebagai unit input yang dimengerti aktivitasnya.

Secara fisik, arsitektur umum dari sebuah komputer yang biasa kita kenal (*Personal Computer/PC*) dapat dilihat pada Gambar 3.10. Sebuah PC merupakan rangkaian dari berbagai macam komponen yang memiliki fungsi masing-masing.

Gambar 3.10. Perangkat keras komputer.



1. *Display*
2. *Motherboard*
3. *CPU*
4. *Main Memory*
5. *Expansion Cards*
6. *Power Supply*
7. *Optical Disc Drive*
8. *Secondary Storage (Hard Disk)*
9. *Keyboard*
10. *Mouse*

Berikut ini penjelasan singkat tentang komponen-komponen fisik dalam sebuah komputer :

1. *Display*. Komponen *display* atau *monitor* termasuk dalam unit keluaran sebuah komputer. Sebuah kabel menghubungkan monitor dengan adapter video yang diinstal pada slot ekspansi *motherboard*. Komputer mengirimkan signal kepada adapter video, mengenai karakter, gambar atau grafik apa yang harus ditampilkan. Adapter video akan mengkonversi signal menjadi sekumpulan instruksi tentang bagaimana monitor harus menampilkan teks, atau gambar pada layarnya.

Gambar 3.11. Display atau monitor.

2. *Motherboard. Motherboard* atau dikenal juga sebagai *mainboard*, *system board* atau *logic board* (pada Apple Computer) dan kadang disingkat sebagai *mobo* adalah pusat dari papan sirkuit utama pada sebuah sistem elektronik, seperti perangkat komputer modern. Pada komponen ini akan diletakkan (ditancapkan) komponen-komponen lain seperti memori utama, *processor*, adapter video, adapter suara dan lain-lain, sehingga terbentuk sistem komputer yang komplit dan dapat bekerja.

Gambar 3.12. Motherboard sebuah komputer.

3. CPU. *Central Processing Unit* (CPU), atau sering disebut sebagai *Processor*, adalah komponen pada komputer digital yang menginterpretasi insktruksi dan memproses data pada suatu program komputer. CPU menyediakan bagian penting dari sustu sisitem digital yaitu kemampuan untuk diprogram. Komponen ini merupakan komponen yang harus ada pada setiap perangkat komputer.

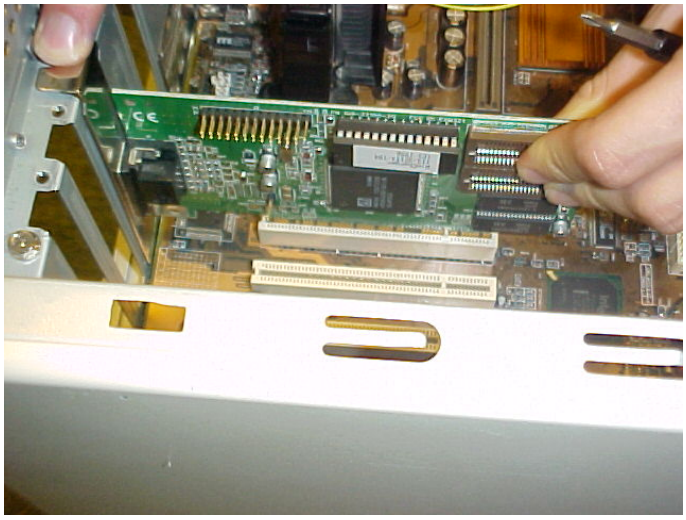
AMD Athlon processor Intel processor

Gambar 3.13. Central Processing Unit (CPU)

4. *Main Memory. Main Memory* atau kadang disebut sebagai *Primary Storage*, atau *Internal Memory*, adalah memori komputer yang secara langsung dapat diakses oleh CPU tanpa menggunakan jalur input/output komputer. Komponen ini digunakan untuk menyimpan data yang sedang aktif digunakan. *Primary storage* dapat terdiri dari beberapa tipe penyimpan seperti *main storage, cache memory*, dan *special registers*.

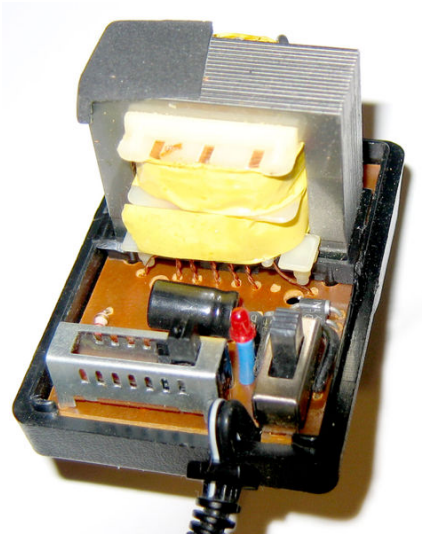
Gambar 3.14. Berbagai jenis main memory.

5. *Expansion Cards.* *Expansion card* (kartu ekspansi) adalah sebuah *printed circuit board* (PCB) yang dapat ditancapkan pada slot ekspansi yang tersedia pada *motherboard* komputer untuk menambah fungsionalitas dari komputer. Contoh expansion card antara lain kartu adapter video, kartu adapter audio, kartu adapter jaringan dan lain-lain.



Gambar 3.15. Pemasangan expansion card.

6. *Power Supply.* *Power supply* atau kadang-kadang disebut PSU (*Power Supply Unit*) adalah perangkat yang menyuplai energi listrik atau energi jenis lain pada komponen lain dalam komputer.



Gambar 3.16. Power Supply Unit.

7. *Optical Disc Drive.* *Optical Disc* adalah sebuah media penyimpanan sekunder yang berbentuk seperti piringan hitam, namun dalam ukuran yang lebih kecil. Data yang tersimpan dalam *Optical Disc* diakses ketika material yang spesifik pada *Optical Disc* disinari oleh sinar laser. Ada dua tipe utama dalam Optical Disc yaitu yang berbasis pada CD (*Compact Disc*) dan yang berbasis pada DVD (*Digital Versatile Disc*). Perangkat untuk membaca, menulis, atau menghapus disebut *Optical Disc Drive*.

Gambar 3.17. CD-RW Drive, salah satu contoh Optical Disc Drive.

8. *Secondary Storage (Hard Disk).* *Secondary Storage* adalah perangkat yang digunakan untuk membantu *Primary Storage (main memory),* terutama untuk menyimpan data, program, atau informasi yang akan digunakan lagi. Berbeda dengan *primary storage*, data, program dan informasi pada *secondary storage* tidak akan hilang meskipun komputer dimatikan, kecuali apabila memang sengaja dihapus. *Secondary storage* yang paling banyak ditemui dalam setiap komputer adalah *Hard Disk. Hard disk* akan menyimpan data dengan menggunakan material bersifat magnetic dalam pola-pola tertentu yang merepresentasikan data.

Gambar 3.18. Hard Disk

9. *Keyboard. Keyboard* atau papan kunci, perangkat yang digunakan untuk menginputkan teks dan karakter pada komputer. Perangkat ini juga dapat digunakan untuk mengontrol fungsi-fungsi khusus pada komputer. Gambar 3.19 menunjukkan lay out sebuah *keyboard* yang umum kita jumpai.

Gambar 3.19. Skema umum sebuah keyboard.

10. *Mouse. Mouse*, biasanya terdiri dari pointing device, yang digunakan untuk mendeteksi pergerakan relative dari dua permukaan secara dua dimensi yang kemudian ditampilkan pada display. Sebagai tambahan, pada mouse seringkali ditambahkan fungsi lain, seperti "*wheels*" atau roda. Selain mendeteksi pergerakan, mouse juga berperan dalam mengeksekusi perintah dengan cara menekan tombol pada *mouse* sekali (click) atau dua kali berurutan (double click).

Gambar 3.20. Berbagai jenis mouse.

3.1.2 Perangkat lunak

Perangkat lunak/piranti lunak adalah program komputer yang berfungsi sebagai sarana interaksi antara pengguna dan perangkat keras. Atau boleh juga diartikan sebagai 'penterjemah' perintah-perintah yang dijalankan pengguna komputer untuk diteruskan ke atau diproses oleh perangkat keras.

Program komputer ini, isinya dapat diubah dengan mudah. Pada komputer, perangkat lunak dimuat ke dalam RAM kemudian dieksekusi di

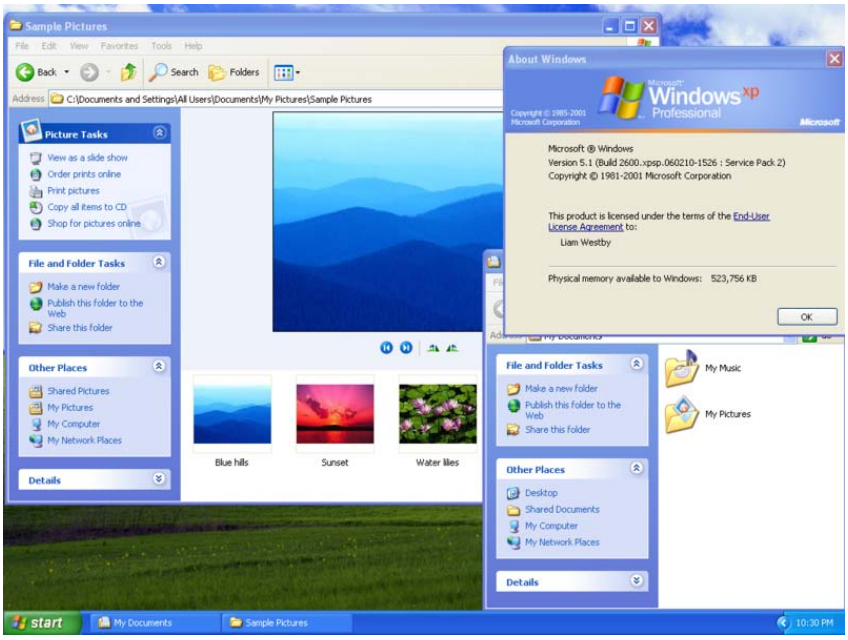
dalam CPU. Pada level paling bawah, perangkat lunak berisi bahasa mesin yang bersifat spesifik terhadap suatu processor.

Ada banyak model penggolongan perangkat lunak, namun secara umum perangkat lunak dapat dibagi menjadi tiga kelompok, yaitu :

1. System Software

System Software adalah perangkat lunak yang digunakan untuk membantu menjalankan perangkat keras dan sistem komputer. **Tujuan dari system software** adalah membatasi semaksimal mungkin programmer aplikasi dari kompleksitas sebuah komputer, terutama yang berhubungan dengan akses memori dan perangkat keras secara langsung.

Termasuk dalam kelompok ini adalah sistem operasi, *driver* perangkat keras, perangkat lunak pendiagnosa, *windowing system, utilities* dan lain-lain. Dari kelompok ini sistem operasi merupakan perangkat lunak yang paling penting. Perangkat lunak ini bekerja sebagai antar muka antara komputer dengan dunia luar. Pada bagian hardware, sistem operasi akan mendiskripsikan perangkat keras yang ada atau terhubung dengan komputer. Sistem operasi menyediakan antar muka pada perangkat keras ini menggunakan "driver" tertentu sehingga perangkat ini dapat dikenali dan bekerja sebagai mana mestinya. Penjelasan lebih detil tentang sistem operasi dapat dilihat pada Bab 4.



Gambar 3.21. Tampilan desktop sistem operasi Windows XP.

2. Programming Software

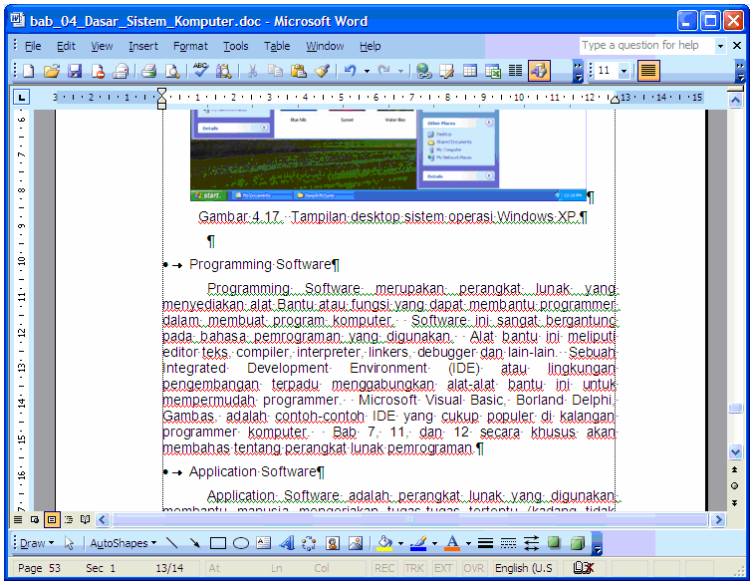
Programming Software adalah perangkat lunak yang menyediakan alat bantu atau fungsi yang dapat membantu programmer dalam membuat program komputer.

Software ini sangat bergantung pada bahasa pemrograman yang digunakan. Alat bantu ini meliputi editor teks, compiler, interpreter, linkers, debugger dan lain-lain. Sebuah Integrated Development Environment (IDE) atau lingkungan pengembangan terpadu menggabungkan alat-alat bantu ini untuk mempermudah programmer. Kita akan banyak mempelajari bagian ini pada buku ini.

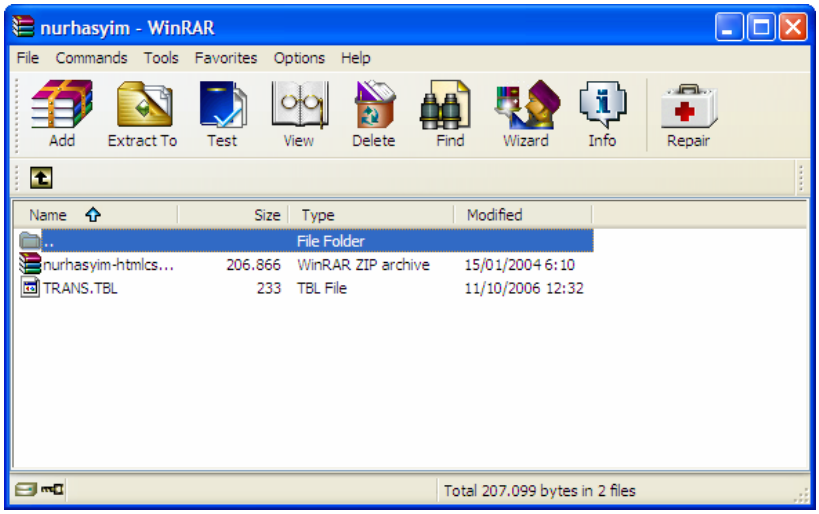
3. Application Software

Application Software adalah p*erangkat lunak yang digunakan membantu manusia mengerjakan tugas-tugas tertentu (kadang tidak berhubungan dengan komputer.*

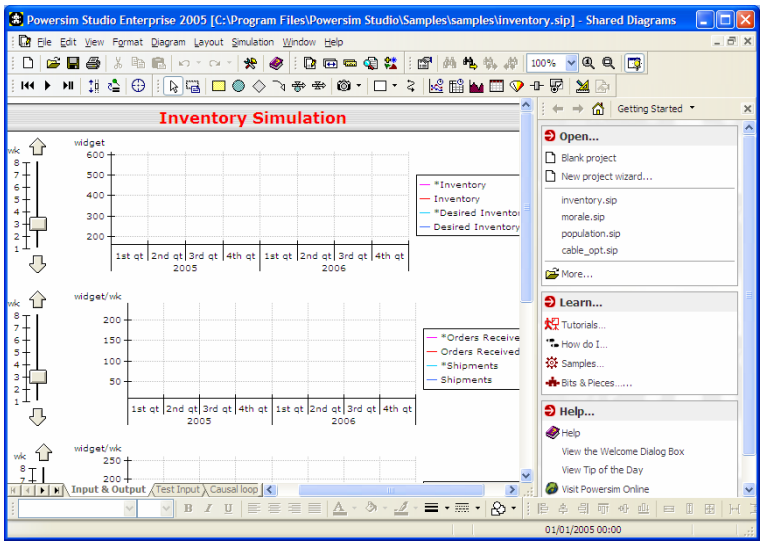
Tipe-tipe *application software* antara lain, perangkat lunak otomatisasi industri, perangkat lunak bisnis, perangkat lunak pendidikan, perangkat lunak software, database, dan game komputer. Beberapa contoh application software dapat dilihat pada Gambar-Gambar berikut ini.



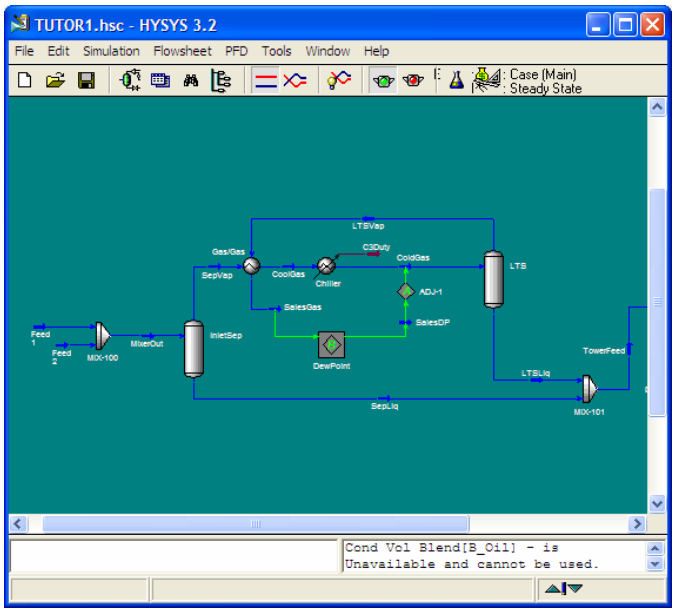
Gambar 3.22. *Application software* Microsoft Word (Software pengolah kota).



Gambar 3.23. *Application software* Winrar (Software kompresi dan ekstraksi file).



Gambar 3.24. *Application software* PowerSim (Software untuk simulasi sistem).



Gambar 3.25. *Application software* Hysis (Software untuk perancangan pabrik).

3.4. RINGKASAN

- Elektronika adalah ilmu yang mempelajari alat listrik arus lemah yang dioperasikan dengan cara mengontrol aliran elektron atau partikel bermuatan listrik.
- Elemen penting dalam teori kelistrikan adalah muatan listrik (Q), hambatan (R), tegangan (V) dan arus (I).
- Elektronika digital adalah sistem elektronik yang menggunakan signal digital dan tersusun dari apa yang disebut sebagai gerbang logika.
- Gerbang logika adalah blok-blok penyusun dari perangkat keras elektronik.
- Ada tiga bentuk dasar dari tabel kebenaran dan gerban logika yaitu AND, OR, dan NOT. Selain itu dikenal juga bentuk turunan yaitu NAND, NOR, dan XOR.
- Komputer adalah alat pengolah data elektronik yang bekerja dan dikontrol oleh sekumpulan instruksi (program).
- Sistem komputer adalah kumpulan elemen-elemen yaitu manusia, perangkat keras, dan perangkat lunak yang saling berinteraksi untuk mencapai tujuan yaitu mendapatkan informasi yang berguna, kemudahan dalam bekerja, kecepatan dan tujuan lainnya.
- Ada tiga komponen utama dalam sistem komputer yaitu manusia sebagai pengguna, perangkat keras dan perangkat lunak.

3.5. SOAL-SOAL LATIHAN

1. Jelaskan pengertian elektronika.
2. Sebutkan hubungan antara muatan listrik, hambatan, tegangan dan arus.
3. Sebutkan komponen-komponen elektronika yang anda ketahui.
4. Sebutkan pengertian elektronika digital.
5. Gambarkan bentuk-bentuk gerbang logika.
6. Gambarkan satu contoh rangkaian digital.
7. Sebutkan komponen-komponen dalam sistem komputer.
8. Jika satu dari komponen sistem komputer tidak tersedia, apa yang akan terjadi?

BAB 4 SISTEM OPERASI

Gambar 4.1. Menjalankan sistem operasi berbasis teks.

Kalau kita perhatikan sekilas Gambar 4.1 di atas mungkin kita akan bertanya-tanya baris-baris tulisan apakah yang tersaji pada gambar tersebut. Tapi kalau kita cermati kita akan dapat menduga teks di atas adalah baris perintah dan hasil eksekusi dari sistem operasi. Bagian perintah sistem operasi berbasis teks ini sering kita abaikan namun sebenarnya sangat penting dan berguna.

Bab ini akan membahas standar kompetensi mengoperasikan sistem operasi komputer berbasis teks dan GUI. Ada dua kompetensi dasar pada standar kompetensi ini yaitu menyiapkan pengoperasian PC, mengoperasikan PC yang tersambung ke jaringan, dan memutuskan koneksi jaringan. Dalam penyajian pada buku ini, setiap kompetensi dasar memuat uraian materi. Ringkasan diletakkan pada akhir bab. Sebelum mempelajari kompetensi ini ingatlah kembali tentang sistem komputer pada bab sebelumnya dan materi-materi pendukung dari mata pelajaran matematika.

TUJUAN

Setelah anda membaca Bab ini, diharapkan pembaca akan mampu :

- Menjelaskan pengertian sistem operasi.
- Menjalankan proses instalasi dan booting sistem operasi.
- Menjalankan sistem operasi dengan mode teks maupun GUI pada sistem operasi.
- Mengoperasikan PC yang tersambung ke jaringan

4.1. PENGERTIAN SISTEM OPERASI

Seperti telah disebutkan pada bab terdahulu, sistem operasi termasuk dalam kelompok *system software* yaitu perangkat lunak yang berperan dalam menjalankan perangkat keras komputer dan sistem komputer secara keseluruhan.

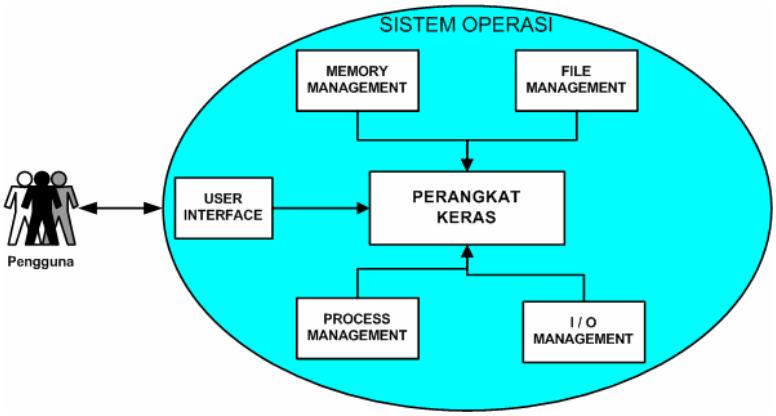
Sistem Operasi adalah perangkat lunak yang bertugas mengelola penggunaan sumberdaya dalam komputer dan menyediakan antarmuka bagi pengguna untuk mengakses sumberdaya tersebut.

FUNGSI

Fungsi-fungsi sebuah sistem operasi secara umum dapat dilihat pada gambar 4.2.

- **Antar muka pengguna**

  Fungsi ini merupakan fungsi yang paling mudah dikenali oleh pengguna karena melalui fungsi ini pengguna dapat berinteraksi dengan sistem operasi, perangkat keras maupun perangkat lunak yang lain. Sistem operasi pada dasarnya menunggu input atau instruksi dari pengguna dan kemudian menerjemahkan perintah-perintah tersebut dalam bahasa yang dimengerti oleh komputer. Antar muka pengguna menjadi tempat bagi pengguna untuk menuliskan atau menyampaikan perintah tersebut.



Gambar 4.2. Fungsi-fungsi sistem operasi.

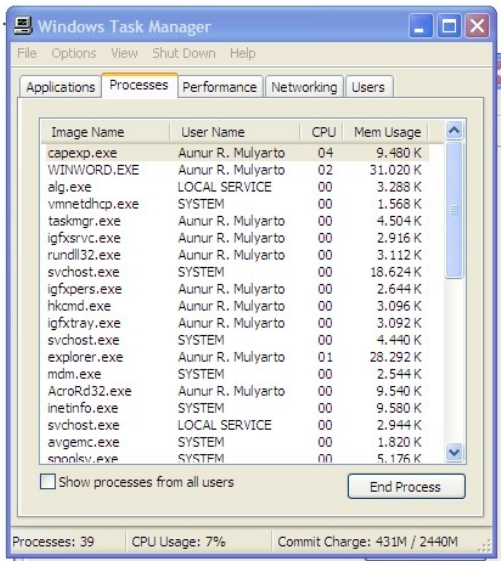
Secara garis besar ada dua model antar muka pengguna yaitu *Command Line Interface* (CLI) dan *Graphical User Interface* (GUI). CLI memberikan fasilitas bagi pengguna untuk memberikan perintah dalam bentuk teks sedangkan GUI lebih berbasis pada tampilan grafis. Dewasa ini hampir semua sistem operasi modern menyediakan model GUI sebagai antar muka pengguna. Beberapa menyediakan GUI yang terintegrasi dengan kernel sistem operasi, misalnya pada Microsoft Windows dan Apple Mac OS versi awal. Sedangkan yang lainnya menyediakan GUI yang bersifat modular, yaitu tidak terintegrasi langsung pada kernel sistem operasinya, seperti pada Unix, Linux dan Mac OS versi X ke atas.

- **Manajemen memori**

Memori utama atau lebih dikenal sebagai memori adalah sebuah *array* yang besar dari *word* atau *byte*, yang ukurannya mencapai ratusan, ribuan, atau bahkan jutaan. Setiap *word* atau byte mempunyai alamat tersendiri. Memori utama berfungsi sebagai tempat penyimpanan instruksi/data yang akses datanya digunakan oleh CPU dan perangkat Masukan/Keluaran. Memori utama termasuk tempat penyimpanan data yang yang bersifat *volatile* -- tidak permanen -- yaitu data akan hilang kalau komputer dimatikan.

Sistem operasi bertanggung-jawab atas aktivitas-aktivitas yang berkaitan dengan manajemen memori seperti:

- Menjaga *track* dari memori yang sedang digunakan dan siapa yang menggunakannya.
- Memilih program yang akan di-*load* ke memori.



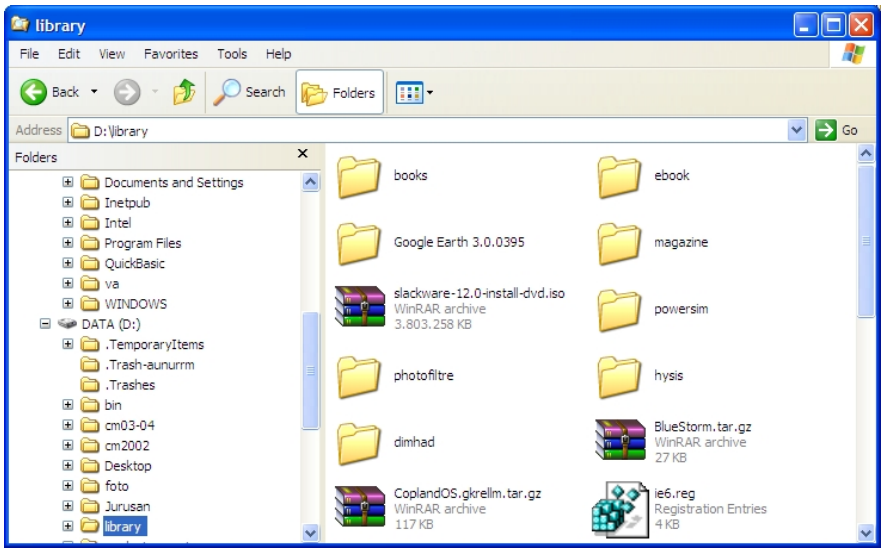
Gambar 4.3. Manajemen memori pada sistem operasi Microsoft Windows

- **Manajemen file**

File (berkas) adalah kumpulan informasi yang berhubungan, sesuai dengan tujuan pembuat berkas tersebut. Umumnya file merepresentasikan program dan data. File dapat mempunyai struktur yang bersifat hirarkis (direktori, volume, dll.). Sistem operasi mengimplementasikan konsep abstrak dari file dengan mengatur media penyimpanan massal, misalnya *tapes* dan *disk*.

Sistem operasi bertanggung-jawab dalam aktivitas yang berhubungan dengan manajemen file :

- Pembuatan dan penghapusan file.
- Pembuatan dan penghapusan direktori.
- Mendukung manipulasi berkas dan direktori.
- Memetakan berkas ke *secondary-storage*.
- Mem-*back-up* berkas ke media penyimpanan yang tidak permanen (*non-volatile*).



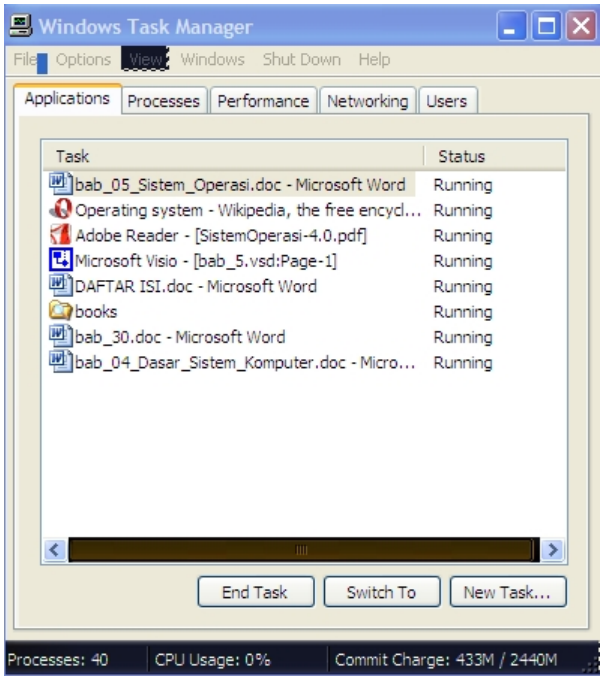
Gambar 4.4. Windows Explorer sebagai sarana pengelolaan file.

- **Manajemen proses**

Proses adalah sebuah program yang sedang dieksekusi. Sebuah proses membutuhkan beberapa sumber daya untuk menyelesaikan tugasnya. Alokasi sumber daya tersebut dikelola oleh Sistem Operasi. Misalnya, penggunaan memori oleh *CPU*, file-file yang terbuka, dan penggunaan oleh perangkat-perangkat input/output lain. Ketika proses tersebut berhenti dijalankan, sistem operasi akan mendapatkan kembali semua sumber daya yang bisa digunakan kembali.

Sistem operasi bertanggung-jawab atas aktivitas-aktivitas yang berkaitan dengan manajemen proses seperti:

- Membuat dan menghapus proses pengguna dan sistem proses.
- Menunda atau melanjutkan proses.
- Menyediakan mekanisme untuk sinkronisasi proses.
- Menyediakan mekanisme untuk komunikasi proses.
- Menyediakan mekanisme untuk penanganan *deadlock*.



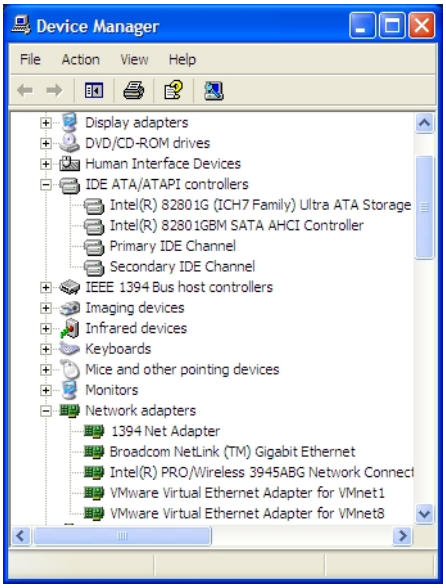
Gambar 4.5. Manajemen proses pada sistem operasi Microsoft Windows.

- **Manajemen sistem masukan dan keluaran (I / O)**

Sistem ini sering disebut dengan *device manager*. Menyediakan *device driver* yang umum sehingga operasi Masukan/Keluaran dapat seragam (membuka, membaca, menulis, menutup). Contoh: pengguna menggunakan operasi yang sama untuk membaca berkas pada perangkat keras, *CD-ROM* dan *floppy disk*.

Komponen Sistem Operasi untuk sistem Masukan/Keluaran:

- Penyangga: menampung sementara data dari/ke perangkat Masukan/Keluaran.
- *Spooling*: melakukan penjadwalan pemakaian Masukan/Keluaran sistem supaya lebih efisien (antrian dsb.).
- Menyediakan *driver*: untuk dapat melakukan operasi rinci untuk perangkat keras Masukan/Keluaran tertentu.



Gambar 4.6. Manajemen I / O pada sistem operasi Microsoft Windows.

3.1.3 BIOS

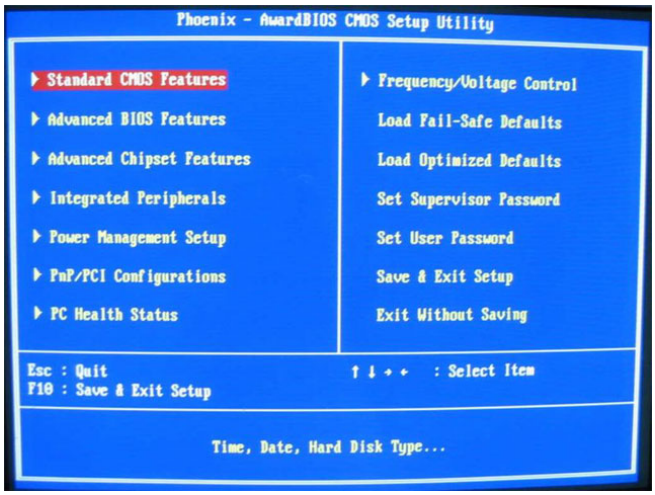
BIOS merupakan singkatan dari *Basic Input/Output System*. BIOS adalah kode-kode program yang pertama kali dijalankan ketika komputer dinyalakan (booting). Fungsi utama BIOS adalah untuk mengidentifikasi dan mengenali perangkat keras komputer. Biasanya BIOS akan tersimpan dalam ROM (*Read Only Memory*) yang ada pada motherboard suatu komputer.

Ketika komputer dinyalakan maka BIOS akan mencoba mengenali bagian-bagian komputer berikut ini:

- *clock generator.*
- *processors dan caches.*
- *chipset (memory controller and I/O controller).*
- *system memory.*
- *Semua perangkat PCI*
- *primary graphics controller.*
- *Mass storage controllers (*seperti *SATA and IDE controllers).*
- *Various I/O controllers (such keyboard/mouse and USB).*

Setelah dikenali maka BIOS akan memanggil program untuk boot suatu sistem operasi (*boot loader*).

Kita dapat melakukan setting BIOS dengan menggunakan fasilitas yang disediakan oleh BIOS. Biasanya dengan menekan tombol Del atau F2 (tergantung jenis komputernya) ketika komputer baru dinyalakan. Jika berhasil masuk maka kita akan disuguhi tampilan seperti pada Gambar 4.7. kita dapat melakukan serangkain pengaturan pada perangkat keras yang ada pada komputer.



Gambar 4.7. Tampilan BIOS utility.

4.2. JENIS-JENIS SISTEM OPERASI

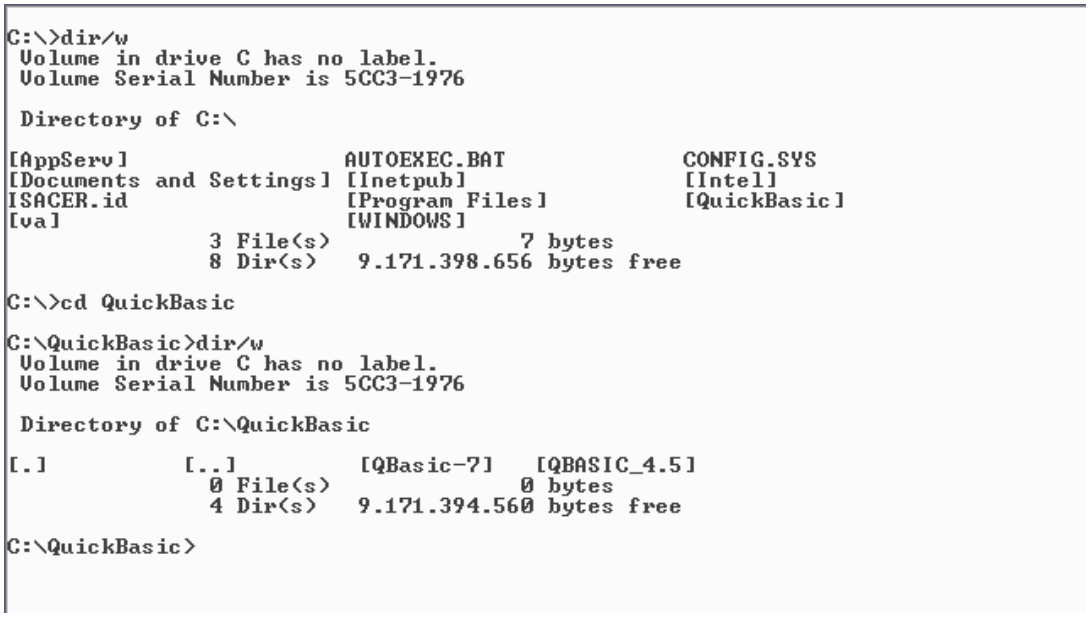
Sistem operasi telah berkembang melalui jalan yang panjang. Dari yang paling sederhana sampai yang paling modern dewasa ini. Masing-masing memiliki kelebihan dan kekurangan terutama sehubungan dengan fungsi-fungsi yang dimilikinya. Pada bagian berikut ini akan dibahas beberapa sistem operasi yang banyak digunakan dan familiar bagi pengguna komputer.

4.2.1. DOS

DOS adalah singkatan dari *Disk Operating System*. DOS merujuk pada perangkat sistem operasi yang digunakan di banyak komputer yang menyediakan abstraksi dan pengelolaan perangkat penyimpan sekunder dan informasinya. Misalnya penggunaan sistem file yang mengelola file-file yang ada pada perangkat penyimpan. DOS biasanya dijalankan dari satu atau dua disc. Hal ini karena pada masa DOS digunakan media penyimpan masih sangat terbatas kemampuannya (paling besar mungkin hanya 1,4 *Megabyte*).

Ada banyak jenis DOS diantaranya Apple DOS, Commodore DOS, Atari DOS dan lain-lain. Jenis ini sangat bergantung dengan jenis perangkat komputernya. Jenis DOS yang paling terkenal adalah jenis DOS yang berjalan pada mesin-mesin yang compatible dengan IBM *Personal Computer*.

Untuk menjalankan perintah-perintah sistem operasi, DOS menggunakan perintah berbasis teks atau CLI. Setiap kali selesai mengetikkan suatu perintah, kita harus menekan tombol ENTER untuk mengeksekusi perintah tersebut. Contoh operasi dengan menggunakan DOS dapat dilihat pada Gambar 4.8.

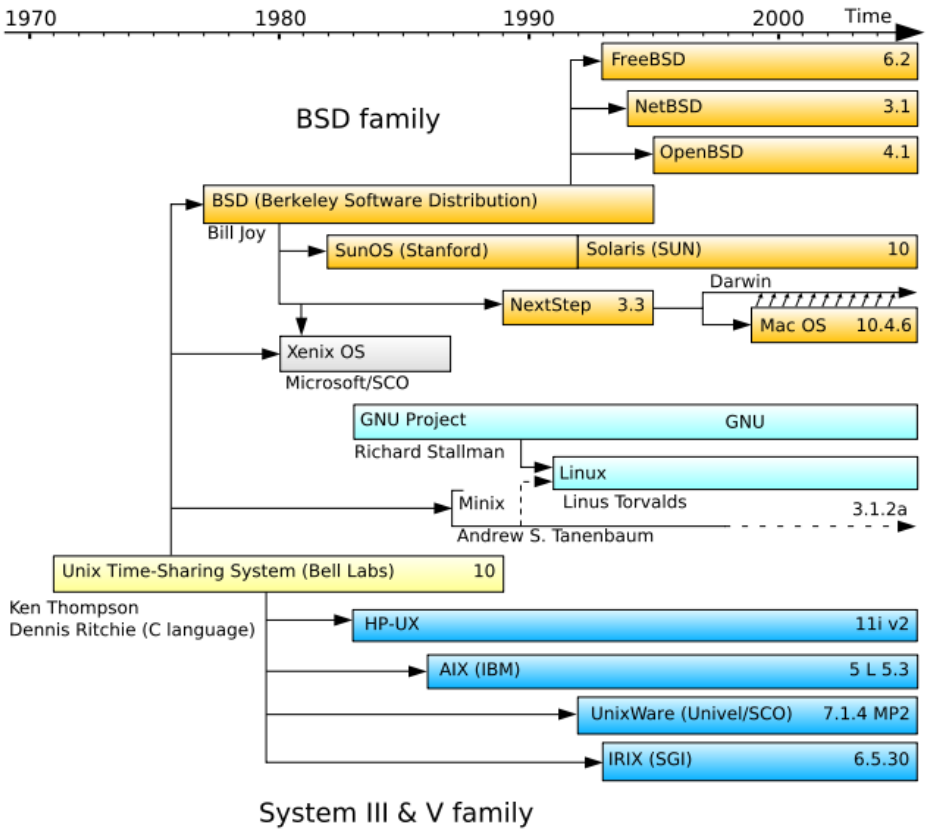


Gambar 4.8. Contoh penggunaan DOS.

4.2.2. UNIX

UNIX adalah sistem operasi yang mula-mula dikembangkan oleh suatu kelompok di AT & T pada laboatorium Bell. Unix banyak digunakan baik untuk server maupun workstation. Linkungan Unix dan model program client-server menunjukkan bahwa Unix lebih dikembangkan sebaga sistem operasi yang kuat di jaringan komputer dari pada sistem operasi untuk computer personal.

UNIX dirancang untuk *portable*, *multi-tasking*, dan *multi-user*. Konsep utama Unix antara lain banyak menggunakan file teks biasa untuk menyimpan data, menggunakan sistem file berjenjang, memperlakukan perangkat sebagai suatu file, dan menggunakan banyak program kecil yang eksekusinya pada CLI dapat digabung dengan tanda pipeline (|). Pada Gambar 5.2 di atas, tampak beberapa perintah UNIX yang digabung dengan pipeline. Konsep yang sangat solid dan stabil membuat Unix banyak dijadikan dasar sistem operasi modern. Gambar 4.9. menunjukkan bagaimana Unix merupakan dasar dari banyak sistem operasi yang ada sekarang.



Gambar 4.9. Unix dan sistem operasi turunannya.

Sistem UNIX terdiri dari beberapa komponen yang biasanya dipaket bersama. Umumnya paket-paket tersebut adalah sebagai berikut:

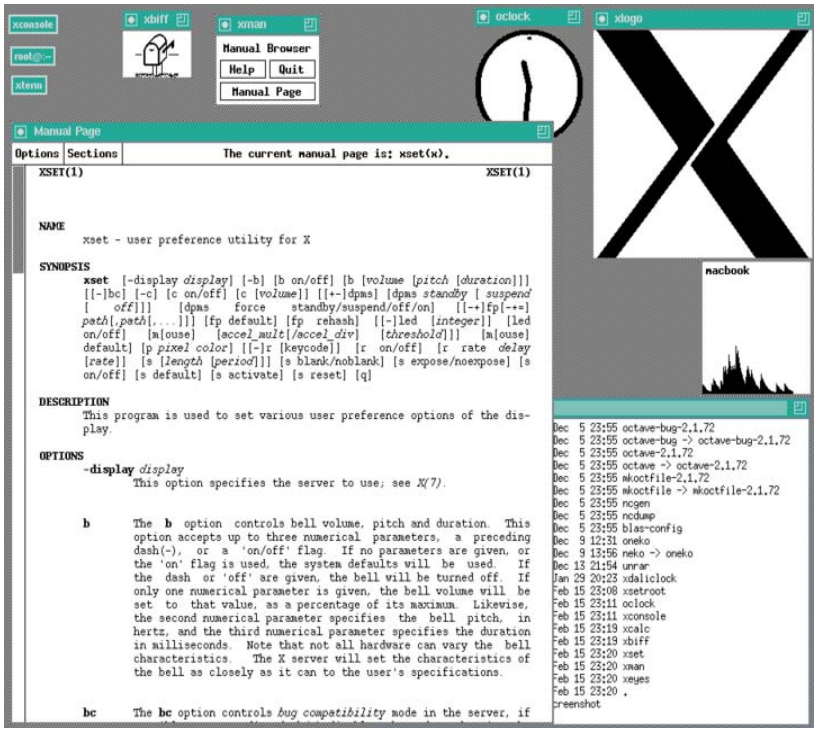
- *Kernel* dengan sub komponen seperti :
    - *conf* — file konfigurasi.
    - *dev* — driver perangkat keras
    - *sys* — kernel sistem operasi, manajemen memori, penjadwalan proses, sistem calls dan lain-lain.
    - *h* — header files, mendefinisikan struktur kunci di dalam sistem.

```
16087 ?? IW      0:00.04 bash /home/users/t/ta/tanders/src/net-snmp-main/dist/ns
16200 ?? DW      0:00.00 grep -v ^ *+ conftest.erl
16610 ?? IW      0:04.82 /bin/bash /home/users/t/ta/tanders/src/net-snmp-V5-1-pa
16929 ?? I       0:00.16 sh Compile cvs/RELEASE wxGTK
17066 ?? I       0:00.00 sh Compile cvs/RELEASE wxGTK
17686 ?? IW      0:00.03 bash /home/users/t/ta/tanders/src/net-snmp-main/dist/ns
17774 ?? IW      0:06.28 /bin/bash /home/users/t/ta/tanders/src/net-snmp-V5-3-pa
18239 ?? I       0:00.00 sh Compile cvs/RELEASE wxGTK
19497 ?? IWs     0:00.12 sshd: tanders@notty
23894 ?? S       0:00.07 gmake
24011 ?? S       2:09.33 snmpd -d -r -U -p /tmp/snmp-test-31-11960/snmpd.pid.num
24544 ?? S       0:00.00 /bin/sh ../../../../../bk-deps g++ -c -o ogledit_view.o
26353 ?? IW      0:00.04 bash /home/users/t/ta/tanders/src/net-snmp-main/dist/ns
27935 ?? DE      0:00.01 (bash)
28366 ?? IWs     0:00.13 sshd: tanders@notty
29812 ?? IWs     0:00.03 bash /home/users/t/ta/tanders/src/net-snmp-main/dist/ns
 2336 p0 R+      0:00.00 ps ax
21906 p0 Is      0:00.01 -bash
24304 p0 S       0:00.03 sh
  697 00 IWs+    0:00.01 /usr/libexec/getty Pc console
  639 E1 IWs+    0:00.01 /usr/libexec/getty Pc ttyE1
  734 E2 IWs+    0:00.01 /usr/libexec/getty Pc ttyE2
  762 E3 IWs+    0:00.01 /usr/libexec/getty Pc ttyE3
$
```

Gambar 4.10. Manajemen memori dan penjadwalan proses pada Unix.

- **Development Environment**:
  - *cc* —compiler untuk bahasa C
  - *as* — machine-language assembler
  - *ld* — linker, untuk menggabung file-file object
  - *lib* — object-code libraries (diinstall di folder /lib atau /usr/lib) *libc*, kumpulan pustaka untuk bahasa C
  - *make* – program untuk mengkompilasi kode program
  - *include* — file-file *header* untuk pengembangan perangkat lunak dan menentukan standar *interface*
  - *Other languages* — bahasa-bahasa pemrograman lain seperti Fortran-77, Free Pascal, dan lain-lain.
- **Commands**:
  - *sh* —"Shell" untuk melakukan pemrograman berbasis CLI atau mengeksekusi perintah-perintah tertentu.
  - *Utilities* — Sekumpulan perintah CLI yang berguna untuk fungsi-fungsi yang bermacam-macam, meliputi:
    - *System utilities* — Program-program untuk pengelolaan sistem seperti *mkfs*, *fsck*, dan lain-lain.
    - *User utilities* — Program-program untuk pengelolan lingkungan kerja, seperti *passwd*, *kill*, dan lain-lain.
  - *Document formatting* — Program untuk penyiapan dokumen seperti *nroff*, *troff*, *tbl*, *eqn*, *refer*, dan *pic*. Beberapa sistem Unix modern juga memasukkan aplikasi seperti TeX dan Ghostscript.

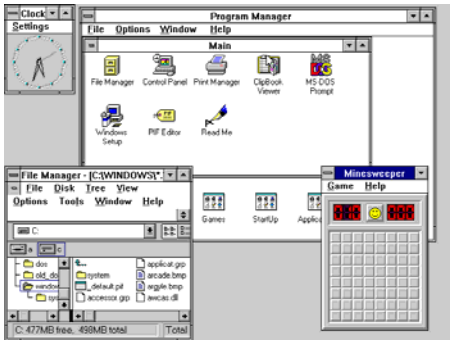
- *Graphics* — Sistem Unix modern menyediakan X11 sebagai sistem standard windowing dan GUI.



Gambar 4.11. X windows system di UNIX.

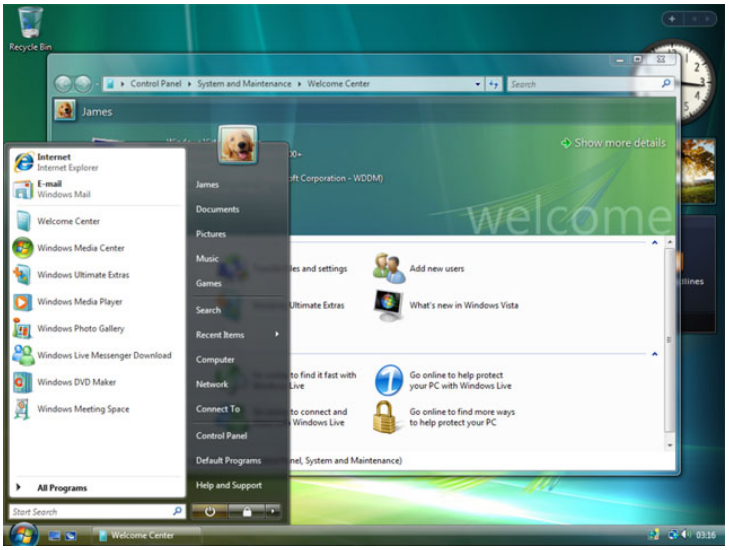
4.2.3. Microsoft Windows

Micosoft Windows atau orang lebih sering menyebut Windows saja pada awalnya hanyalah add-on dari MS-DOS karena tingginya tuntutan pada sistem operasi yang berbasis GUI. Versi awal Windows berjalan di atas MS-DOS. Meski demikian Windows versi awal telah menunjukkan beberapa fungsi-fungsi yang umum dijumpai dalam sistem operasi, antara lain: memiliki tipe file executable tersendiri, memiliki driver perangkat keras sendiri, dan lain-lain.



Gambar 4.12. Windows versi 3.11.

Secara konsep sebenarnya Windows lebih banyak ditujukan bagi komputer personal. Pada awalnya Windows juga tidak mendukung konsep multi-tasking dan multi-user. Akomodasi terhadap jaringan atau fungsi-fungsi *client-server* juga tidak sekuat pada UNIX dan turunannya. Sehingga masalah yang sering muncul di sistem operasi Windows adalah masalah keamanan yang berhubungan dengan jaringan. Namun Windows memiliki kelebihan dari sisi kemudahan pemakaian. Pada versi yang terbaru (Windows Vista) konsep *multi-user* dan *multi-tasking* telah semakin matang. Selain itu tampilan GUI telah dirubah dengan banyak menggunakan efek tiga dimensi.

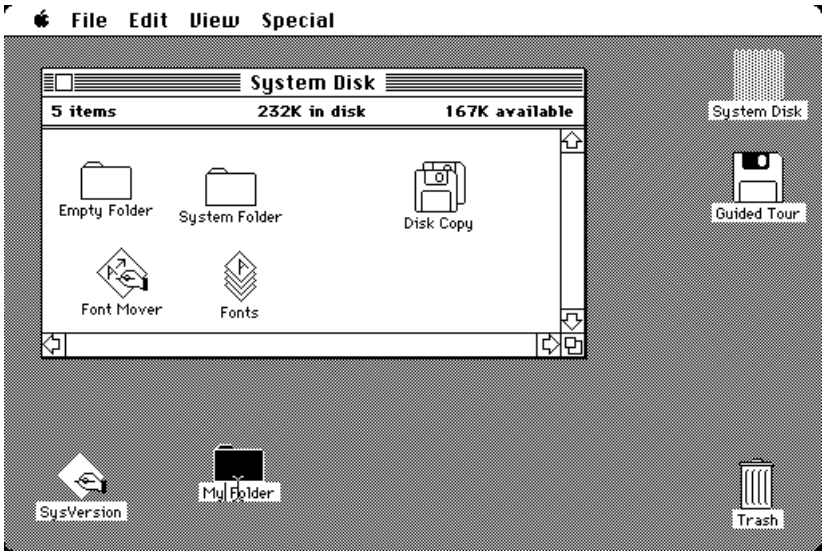


Gambar 4.13. Windows Vista.

4.2.4. Apple Mac OS

Seperti terlihat pada Gambar 5.10, Apple Mac OS merupakan turunan dari UNIX melalui jalur BSD (*Berkeley Software Distribution*). Oleh karena itu kekuatan dalam *multi-tasking, multi-user, networking* yang ada pada UNIX juga dimiliki oleh Mac OS. Mac OS adalah sistem operasi berbasis GUI. Apple merupakan pelopor dalam penggunaan GUI pada sistem operasi. Penggunaan *icon, mouse* dan beberapa komponen GUI merupakan sumbangan yang luar biasa bagi perkembangan sistem operasi berbasis GUI.

Versi awal dari Mac OS hampir secara penuh mengandalkan pada kemampuan GUI-nya dan sangat membatasi penggunaan CLI (Gambar 5.15). Meskipun sangat memudahkan namun ada beberapa kelemahan, antar lain: multi-tasking yang tidak berjalan sempurna, pengelolaan memori yang terbatas, dan konflik pada beberapa program yang ditanamkan. Memperbaiki sistem Mac OS kadang-kadang menjadi suatu pekerjaan yang sangat melelahkan.



Gambar 4.14. Mac OS versi awal.

Pada Mac OS X (versi terbaru), semua kelemahan pada versi lama telah coba dihilangkan. *Multi-tasking* telah berjalan dengan baik dan manajemen memori yang jauh lebih baik. Selain itu tampilan GUI-nya disebut-sebut sebagai yang terbaik di antara sistem operasi yang ada..

Gambar 4.15. Mac OS X.

4.2.5. Linux

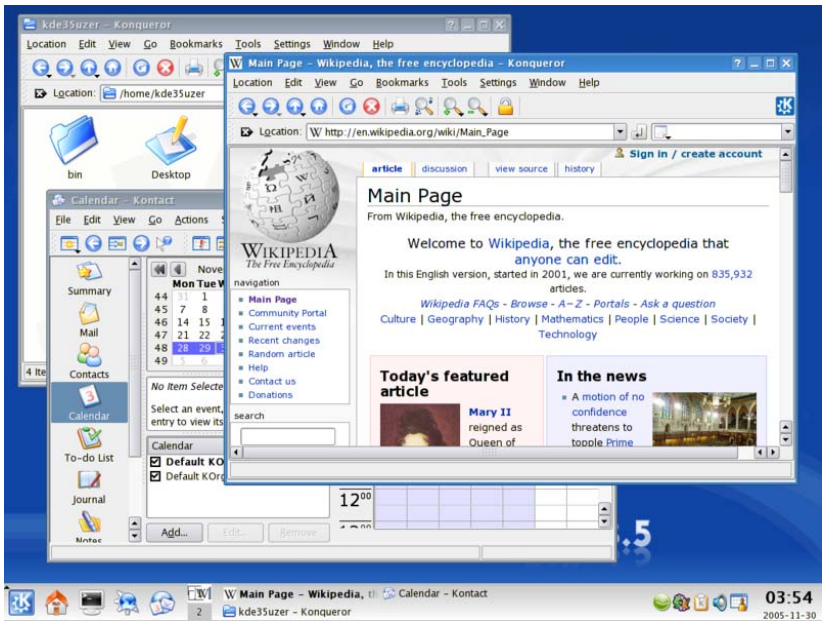
Linux sangat mirip dengan sistem-sistem UNIX, hal ini dikarenakan kompatibilitas dengan UNIX merupakan tujuan utama desain dari proyek Linux. Perkembangan Linux dimulai pada tahun 1991, ketika mahasiswa Finlandia bernama Linus Torvalds menulis Linux, sebuah *kernel* untuk prosesor 80386, prosesor 32-bit pertama dalam kumpulan CPU intel yang cocok untuk PC.

Dalam banyak hal, kernel Linux merupakan inti dari proyek Linux, tetapi komponen lainlah yang membentuk secara komplit sistem operasi Linux. Dimana kernel Linux terdiri dari kode-kode yang dibuat khusus untuk proyek Linux, kebanyakan perangkat lunak pendukungnya tidak eksklusif terhadap Linux, melainkan biasa dipakai dalam beberapa sistem operasi yang mirip UNIX. Contohnya, sistem operasi BSD dari Berkeley, *X Window System* dari MIT, dan proyek GNU dari *Free Software Foundation*.

Pembagian (sharing) alat-alat telah bekerja dalam dua arah. Sistem perpustakaan utama Linux awalnya dimulai oleh proyek GNU, tetapi perkembangan perpustakaannya diperbaiki melalui kerjasama dari komunitas Linux terutama pada pengalamatan, ketidak efisienan, dan bugs. Komponen lain seperti GNU C Compiler, gcc, kualitasnya sudah cukup tinggi untuk dipakai langsung dalam Linux. Alat-alat administrasi network dibawah Linux berasal dari kode yang dikembangkan untuk 4.3BSD, tetapi BSD yang lebih baru , salah satunya FreeBSD, sebaliknya meminjam kode dari Linux, contohnya adalah perpustakaan matematika Intel *floating-point-emulation*.

Saat ini, Linux merupakan salah satu sistem operasi yang perkembangannya paling cepat. Kehadiran sejumlah kelompok pengembang, tersebar di seluruh dunia, yang selalu memperbaiki segala fiturnya, ikut membantu kemajuan sistem operasi Linux. Bersamaan dengan itu, banyak pengembang yang sedang bekerja untuk memindahkan berbagai aplikasi ke Linux (dapat berjalan di Linux).

Masalah utama yang dihadapi Linux dahulu adalah *interface* yang berupa teks (*text based interface*). Ini membuat orang awam tidak tertarik menggunakan Linux karena harus dipelajari terlebih dahulu dengan seksama untuk dapat dimengerti cara penggunaannya (tidak *user-friendly*). Tetapi keadaan ini sudah mulai berubah dengan kehadiran KDE dan GNOME. Keduanya memiliki tampilan desktop yang menarik sehingga mengubah persepsi dunia tentang Linux.



Gambar 4.16. Linux dengan desktop KDE.

4.3. MENYIAPKAN DAN MENJALANKAN SISTEM OPERASI

Mengenal sistem informasi saja tidak cukup. Bagi seorang yang bergerak dalam pemrograman perlu mengetahui secara lebih mendalam tentang bagaimana instalasi, booting dan menjalankan sistem operasi, dari pada pengguna biasa.

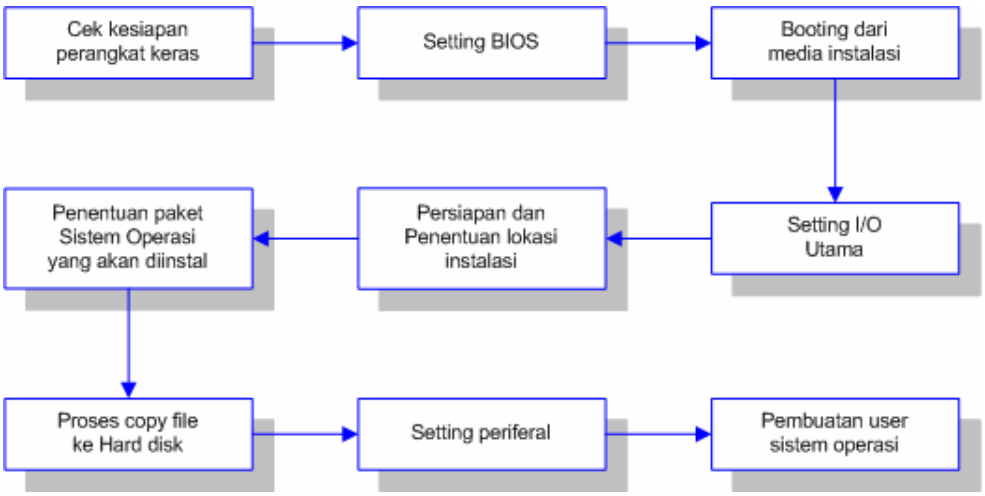
4.3.1. Instalasi

Instalasi adalah pemasangan perangkat lunak pada system computer. Sedangkan **Instalasi Sistem Operasi** adalah pemasangan system operasi pada sistem computer. Sistem operasi akan dipasang terlebih dahulu dibanding perangkat lunak yang lain. Perangkat lunak yang lain baru bisa dijalankan setelah sistem operasi terinstal dengan benar.

Seperti telah dijelaskan, masing-masing sistem operasi memiliki ciri tersendiri. Demikian juga dengan proses instalasi sistem operasi. Proses instalasi sangat bergantung pada jenis sistem operasinya. Berdasarkan tampilan anta mukanya kita dapat membagi menjadi dua, yaitu yang berbasis GUI dan berbasis CLI. Proses instalasi berbasis GUI ada pada sistem operasi Microsoft Windows (GUI penuh pada versi Vista), Apple Mac OS ver X dan yang di atasnya, beberapa versi Linux seperti, Ubuntu dan turunannya (Xubuntu, Kubuntu, Edubuntu, dan lain-lain), Mandriva dan turunannya (PC Linux OS), dan Fedora versi terbaru. Sedangkan versi CLI ada pada Linux versi Slackware, Gentoo dan lain-lain.

Proses instalasi juga dapat dibagi berdasarkan sumber instalasinya, yaitu bersumber dari media baik itu CD, DVD atau hard-disc dan yang bersumber dari network (jaringan). Proses instalasi dengan menggunakan media CD atau DVD merupakan metode yang paling umum digunakan. Pada bagian ini hanya akan dijelaskan tentang proses instalasi dengan sumber dari CD/DVD

Tahapan-tahapan dalam instalasi biasanya seperti terlihat pada Gambar 4.16. Tahapan-tahapan instalasi ini mungkin bervariasi antar sistem operasi. Namun secara umum tahapan dalam sistem operasi apapun tidak akan berbeda jauh.

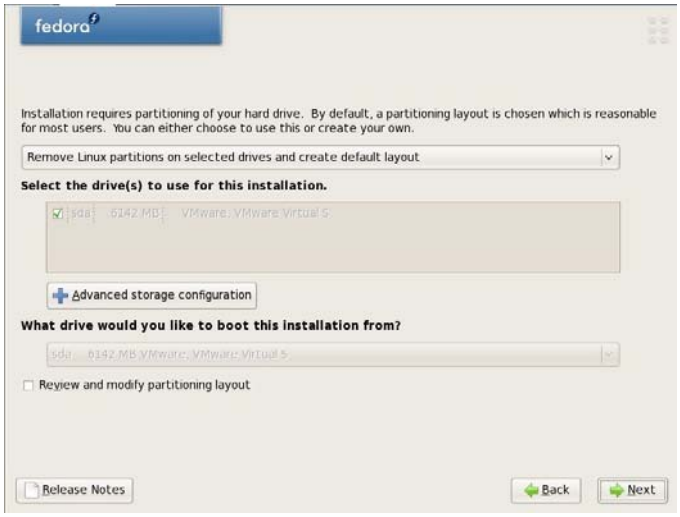


Gambar 4.16. Tahapan-tahapan instalasi.

- Cek kesiapan perangkat keras. Tahap ini bertujuan untuk memastikan bahwa semua perangkat perangkat keras dan periferalnya terpasang dengan benar. Selain itu juga untuk melihat apakah spesifikasi perangkat keras komputer didukung oleh sistem operasi tersebut.
- *Setting* BIOS. Pada dasarnya tahapan ini adalah untuk mengkonfigurasi BIOS agar meletakkan media instalasi dalam urutan paling atas dalam prioritas booting.
- Booting dari media instalasi. Apabila setting BIOS berhasil dengan baik, maka komputer akan *boot* dari media instalasi. Gambar 4.17 merupakan *screen-shot* dari proses *booting* di bagian awal.

Gambar 4.17. Testing media instalasi.

- Setting I/O utama. Tahapan ini bertujuan untuk mengatur agar perangkat input / output utama (*mouse, keyboard* dan *video*) dapat berjalan dengan baik ketika proses instalasi dilakukan.
- Persiapan dan penentuan lokasi instalasi. Media yang paling umum digunakan sebagai target instalasi adalah hard disk yang tertanam di komputer. Kita perlu mempersiapkan hard disk tersebut agar siap ditulis. Persiapan ini meliputi partisi hard disk (termasuk besarnya volume untuk masing-masing partisi) dan format partisi sesuai dengan sistem file yang disyaratkan oleh sistem operasi. Untuk Microsoft Windows, dapat menggunakan sistem file NTFS atau FAT32. Untuk linux dapat digunakan sistem file ext2, ext3, ReiserFS, dan XFS. Untuk Apple Mac OS X biasanya digunakan HFS+. Gambar 4.18 menunjukkan proses penentuan lokasi instalai pada proses instalasi Fedora Core 8.

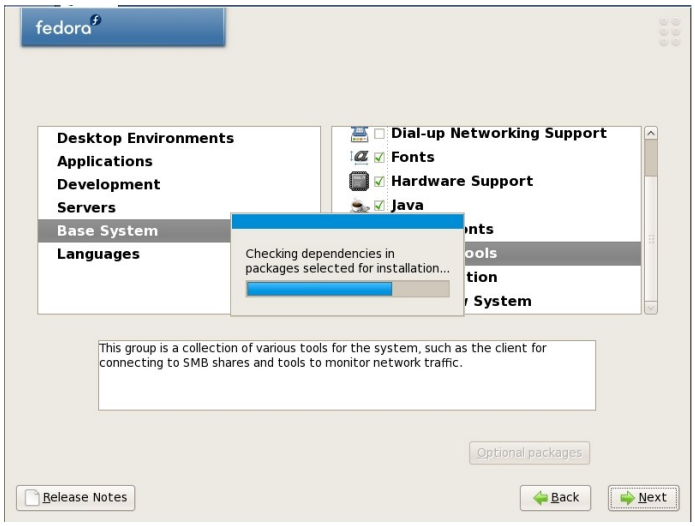


Gambar 4.18. Proses penentuan target instalasi.

- Penentuan paket Sistem Operasi yang akan diinstal. Tahap ini kadang tidak diperlukan jika kita memilih instalasi secara *default.* Namun bila kita ingin menginstal sistem operasi agar sesuai dengan keinginan kita (*custom*

installation) maka tahapan ini harus dilakukan. CD atau DVD instalasi, biasanya mempunyai paket-paket aplikasi yang dapat kita pilih ketika instalasi sistem operasi berjalan atau ketika proses instalasi telah selesai.

- Proses *copy* ke hard disk. Setelah penentuan paket aplikasi dilakukan, maka proses *copy* file instalasi ke hard disk dapat segera dilakukan. Gambar 4.19. merupakan contoh proses *copy* file sistem operasi.

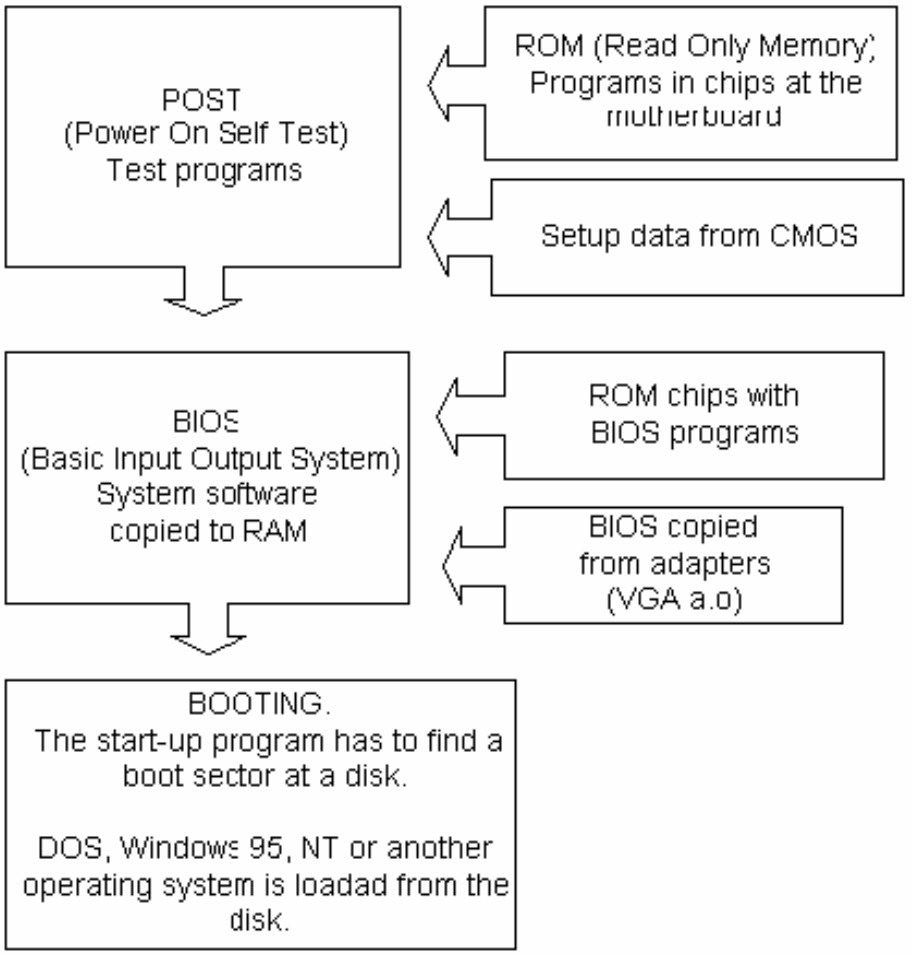


Gambar 4.19. Proses copy file pada Fedora.

- *Setting peripheral* lain. Tahapan ini bertujuan untuk menginstal *driver* bagi peripheral (kartu VGA, kartu suara, *chipset motherboard* dan lain-lain) pada suatu komputer agar dapat bekerja dengan optimal.
- Penentuan *user*. *User* adalah pengguna dari sistem operasi yang telah diinstal. Data dari user yang biasanya ditanyakan adalah user name dan password. Secara umum ada dua level pengguna, yaitu administrator dan user biasa. Administrator mempunyai hak pada semua bagian dari sistem operasi sedangkan user biasa mempunyai hak yang ditentukan oleh administrator.

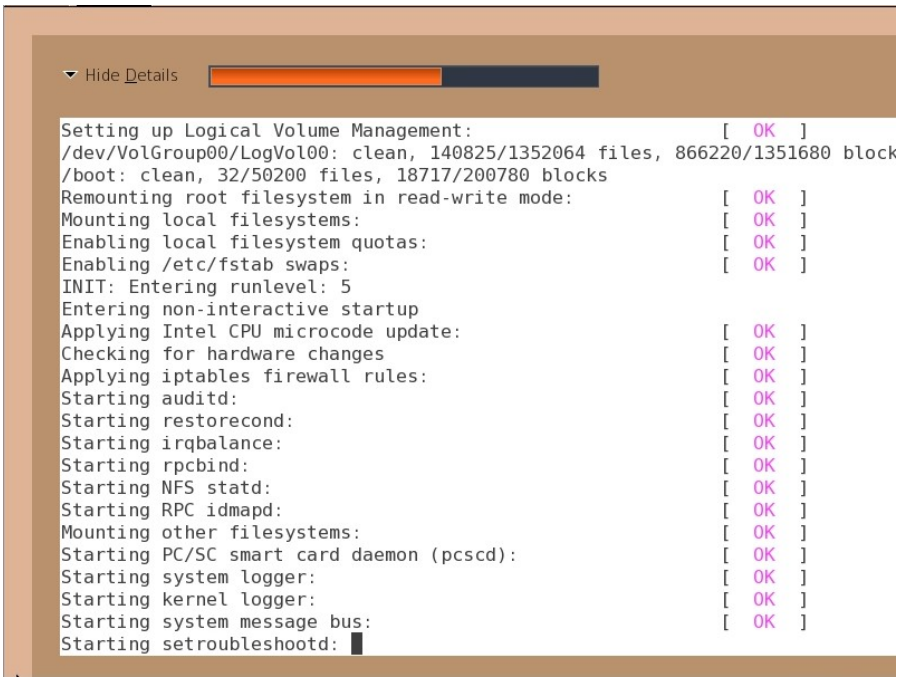
4.3.2. Booting

Booting adalah proses awal saat komputer dihidupkan. Proses awal *booting* dapat dijelaskan dengan menggunakan skema pada Gambar 4.20. Proses awal booting dimulai dari pembacaan dan eksekusi program yang tersimpan di ROM komputer dan data setup yang tersimpan dalam CMOS. Bagian ini disebut POST (*Power On Self Test*) apabila berhasil, maka perangkat lunak sistem BIOS yang berisi program BIOS dari ROM dan BIOS dari adapter (misalnya dari VGA) akan dimuat ke memori utama (RAM) dan dilanjutkan dengan pembacaan program start-up yang tersimpan di dalam boot sector hard disk. Dari sini barulah sistem operasi dimuat dari hard disk.



Gambar 4.20. Proses awal booting.

Pada sistem operasi seperti Microsoft Windows, kita tidak dapat melihat apa yang terjadi ketika sistem operasi dimuat (mulai dijalankan). Kita hanya disuguhi tampilan (biasanya logo) yang disebut sebagai boot-splash. Tetapi pada keluarga Linux, kita dapat memilih apakah proses jalannya sistem operasi ditampilkan atau tidak dengan mengkonfigurasi file boot-loader (biasanya menggunakan LILO atau Grub). Gambar 4.21 menunjukkan proses booting pada Linux Fedora.



Gambar 4.21 Proses booting pada Linux Fedora

4.3.3. Perintah berbasis teks

Bagi banyak orang bekerja dengan perintah berbasis teks (CLI) ketika berhadapan dengan sistem operasi mungkin sangat menyulitkan karena harus menghapal perintah dan mengetikkan perintah tersebut serta tampilan yang tidak menarik. Namun sesungguhnya bekerja dengan memiliki keuntungan tersendiri, antara lain:

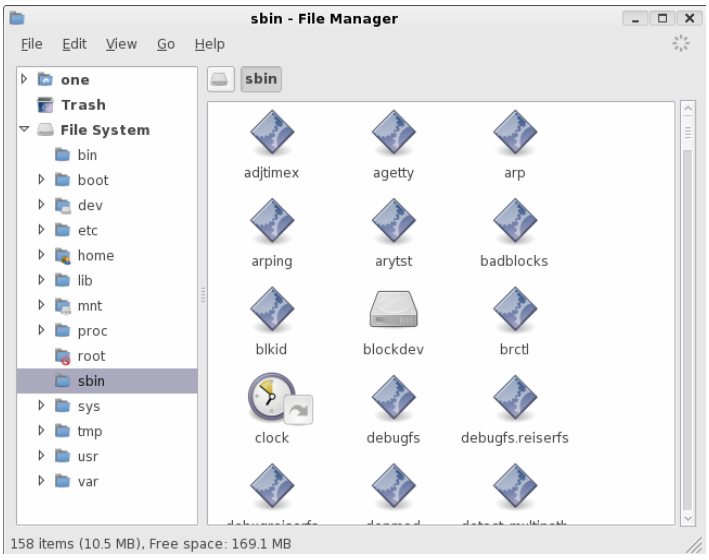
- eksekusi perintah relative lebih cepat.
- hemat dalam penggunaan sumberdaya (terutama CPU dan memori utama).
- tidak bergantung pada perangkat keras dengan spesifikasi tinggi (terutama pada VGA dan monitor).

Pada sistem operasi Microsoft Windows dan Apple Mac OS X, mode CLI mungkin jarang digunakan, bahkan mungkin tidak pernah. Tetapi pada keluarga Linux dan Unix, mode CLI ini tetap merupakan bagian penting, terutama untuk administrasi sistem dan jaringan. Pada bagian ini kita akan membahas beberapa perintah yang sering digunakan pada mode CLI di sistem operasi Linux. Untuk menjalankan mode CLI ini dapat digunakan *console* atau terminal emulator yang tersedia di Linux, seperti *Konsole, xterm, aterm* dan lain-lain (Gambar 4.22).

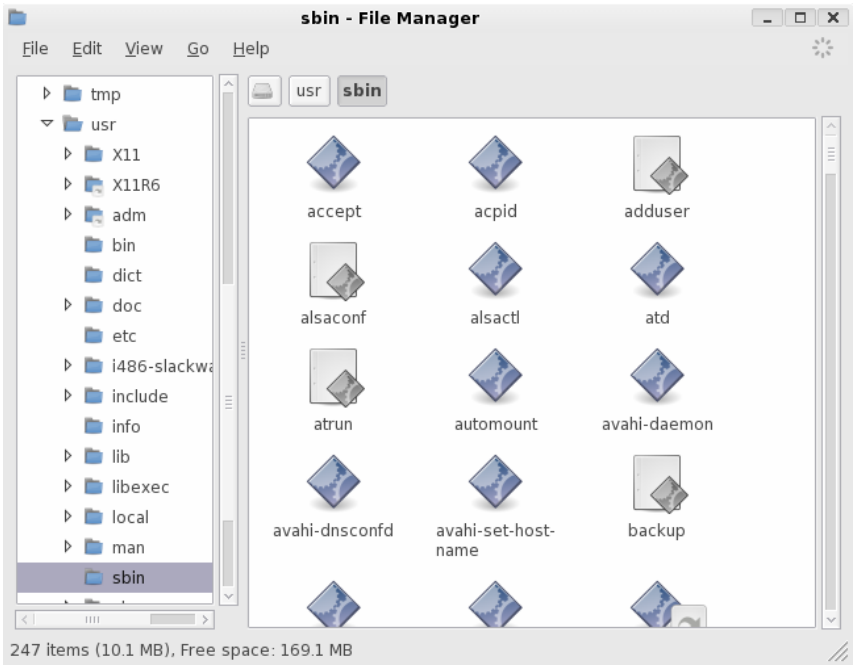
Gambar 4.22. Terminal sedang menjalankan mode CLI.

Ada dua kelompok utama dalam perintah-perintah mode CLI:

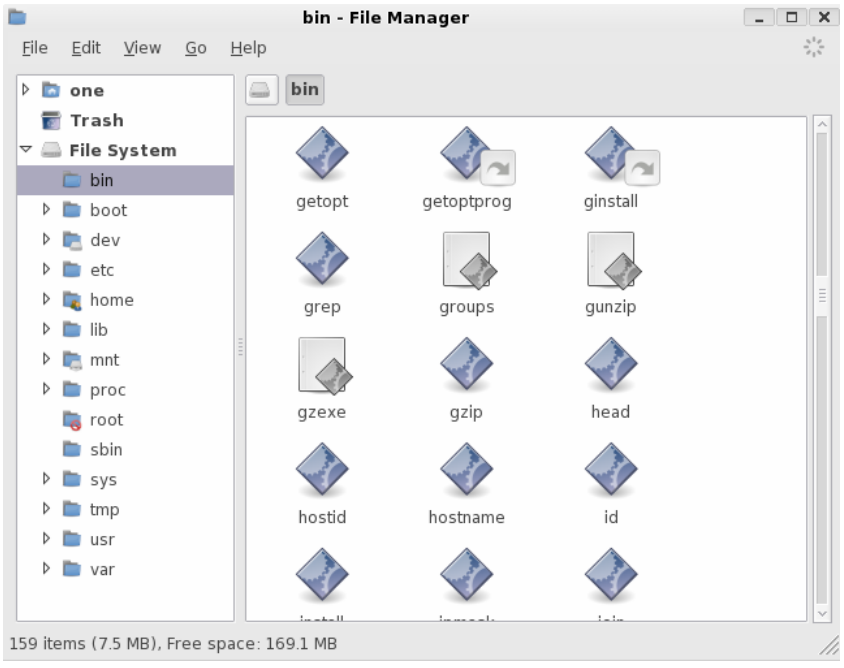
- Perintah yang berhubungan dengan administrasi sistem. Perintah-perintah yang termasuk dalam kelompok ini biasanya hanya dapat dilaksanakan oleh pengguna yang mempunyai hak sebagai administrator (*root*). Perintah-perintah yang termasuk kelompok ini biasanya tersimpan di direktori /sbin (Gambar 4.23) dan /usr/sbin (Gambar 4.24).
- Perintah untuk penggunaan biasa. Perintah ini dapat diakses oleh pengguna biasa. Perintah-perintah yang termasuk kelompok ini biasanya tersimpan di direktori /bin (Gambar 4.25) dan /usr/bin (Gambar 4.26).



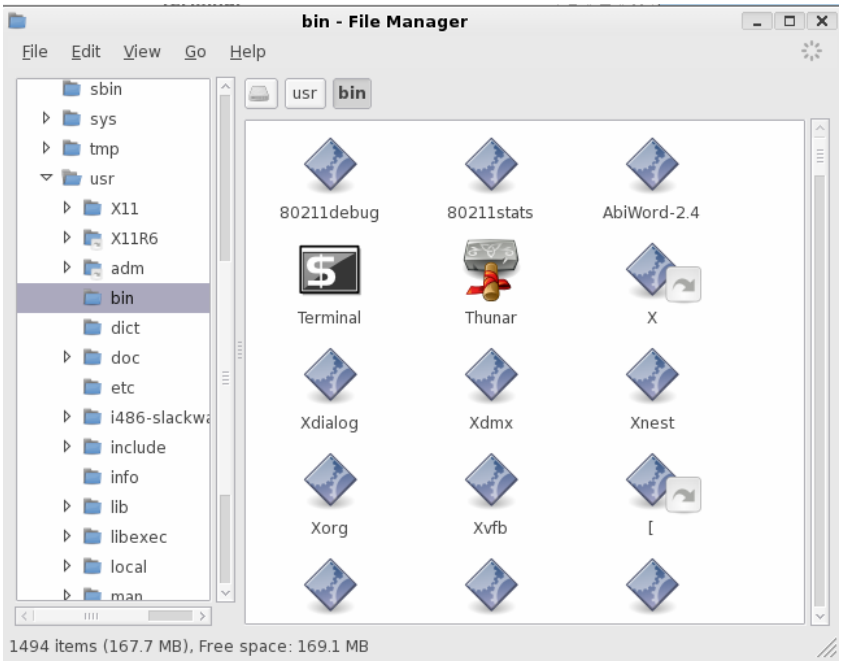
Gambar 4.23. Perintah-perintah pada direktori sbin.



Gambar 4.24. Perintah-perintah pada direktori /usr/sbin.



Gambar 4.25. Perintah-perintah pada direktori bin.

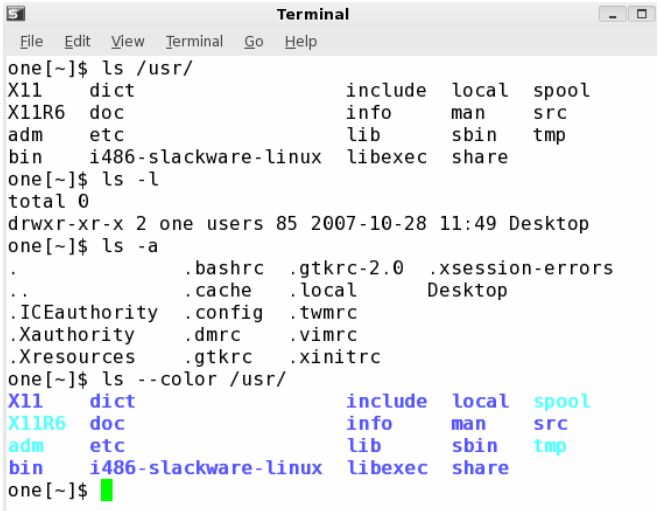


Gambar 4.26. Perintah-perintah pada direktori /usr/bin.

Berikut ini beberapa perintah-perintah penting dalam mode CLI.

- Menampilkan isi direktori

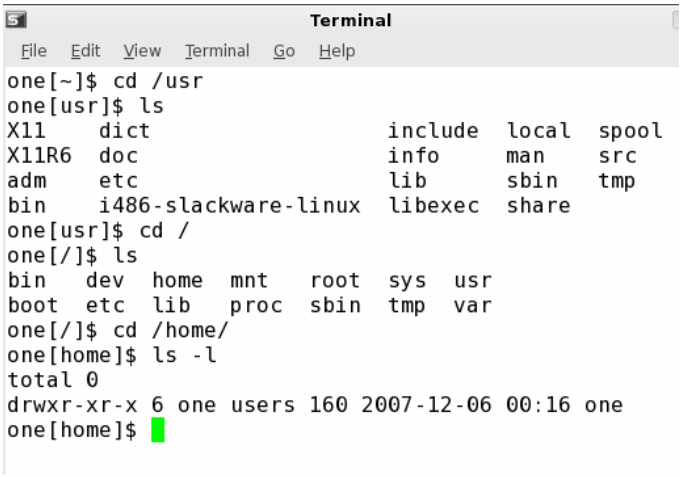
Untuk menampilkan isi direktori dapat digunakan perintah ls diikuti dengan argument lain. Beberapa contoh penggunaan dapat dilihat pada gambar 4.27.



Gambar 4.27. Contoh penggunaan perintah ls.

- Pindah direktori

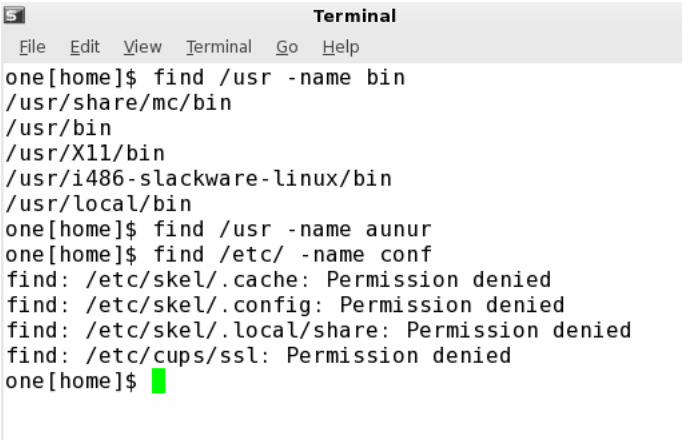
Berpindah direktori dapat dilakukan dengan perintah cd diikuti lokasi dimana kita mau berpindah. Beberapa contoh penggunaan dapat dilihat pada gambar 4.28.



Gambar 4.28. Contoh penggunaan perintah cd.

- Mencari file

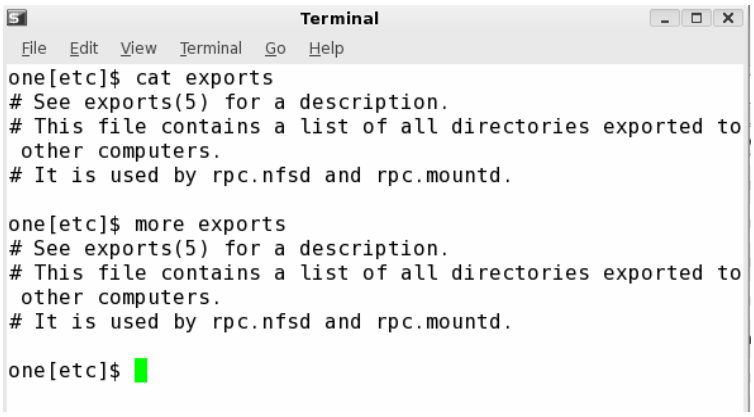
Perintah find dapat digunakan untuk mencari file tertentu di lokasi yang ditentukan. Beberapa contoh penggunaan dapat dilihat pada gambar 4.29.



Gambar 4.29. Contoh penggunaan perintah find.

- Menampilkan isi file

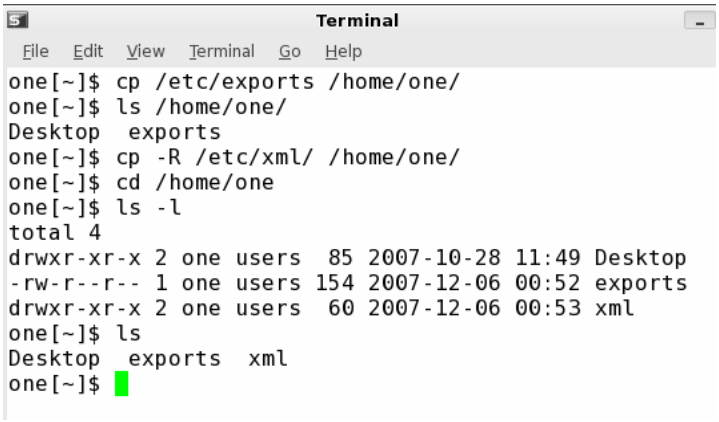
Untuk menampilkan isi file dapat digunakan perintah more, less atau cat diikuti dengan nama filenya. Beberapa contoh penggunaan dapat dilihat pada gambar 4.30.



Gambar 4.30. Contoh penggunaan perintah cat dan more.

- Menyalin file dan directory

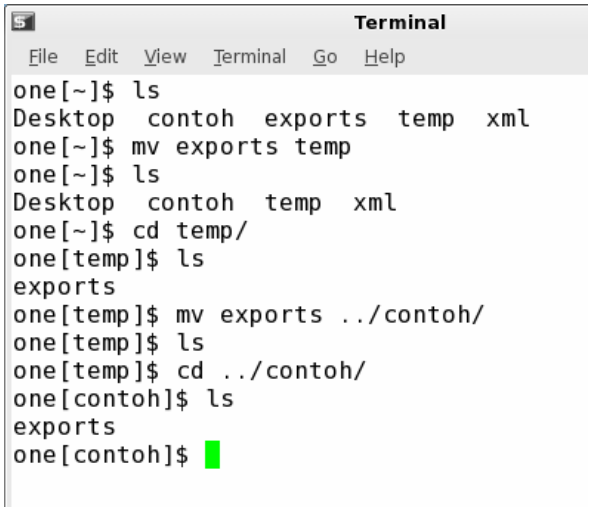
Perintah cp bertujuan untuk menyalin file atau directory. Beberapa contoh penggunaan dapat dilihat pada gambar 4.31.



Gambar 4.31. Contoh penggunaan perintah cp.

- Memindahkan file

Untuk memindahkan file dapat digunakan perintah mv. Beberapa contoh penggunaan dapat dilihat pada gambar 4.32.



Gambar 4.32. Contoh penggunaan perintah mv untuk memindahkan file.

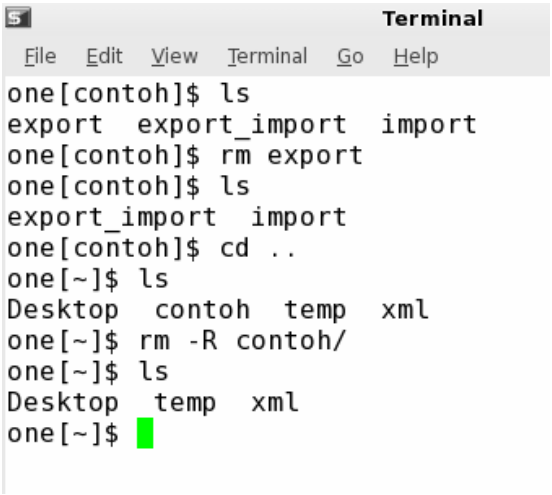
- Mengganti nama file

Perintah mv dapat juga digunakan untuk mengganti nama file. Beberapa contoh penggunaan dapat dilihat pada gambar 4.33.

Gambar 4.33. Contoh penggunaan perintah mv untuk mengganti nama file.

- Menghapus file dan direktori

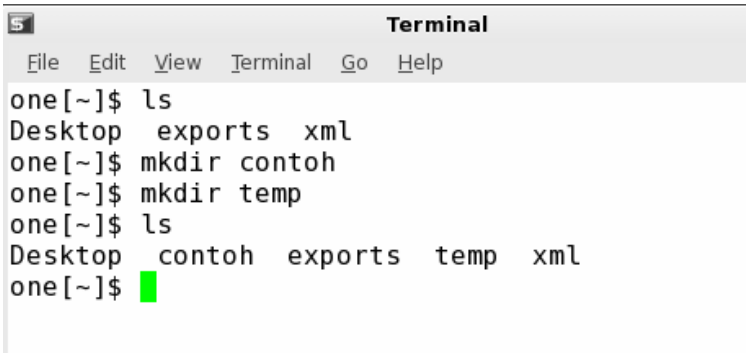
Perintah untuk menghapus file dan directory adalah rm. Beberapa contoh penggunaan dapat dilihat pada gambar 4.34.



Gambar 4.34. Contoh penggunaan perintah rm untuk menghapus file atau direktori.

- Membuat direktori

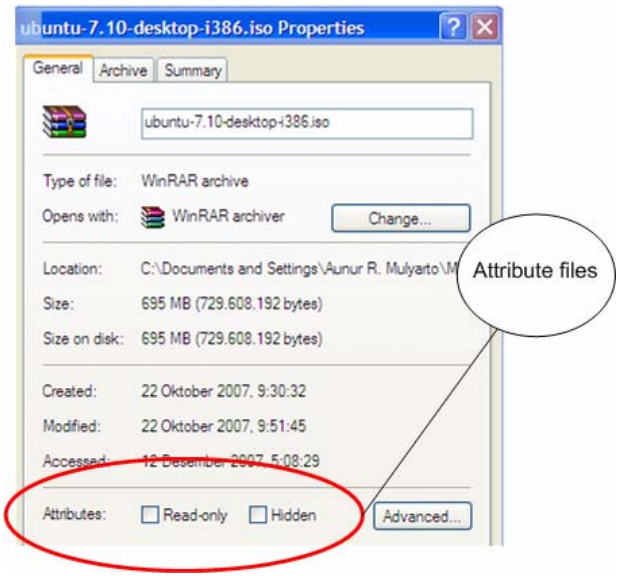
Perintah mkdir merupakan perintah untuk membuat directory baru. Beberapa contoh penggunaan dapat dilihat pada gambar 4.35.



Gambar 4.35. Contoh penggunaan perintah mkdir.

- Memahami hak akses file dan direktori

Pada sistem operasi windows, file dan direktori tidak memiliki file proteksi yang cukup karena file dan direktori hanya mempunyai attribute yang terbatas (Gambar 4.36).



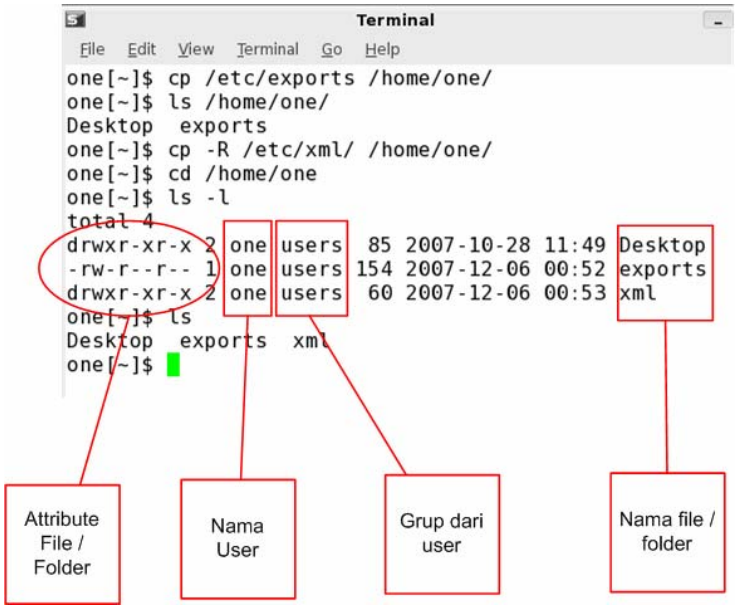
Gambar 4.36. Attribute file / folder pada Microsoft WIndows

Pada gambar 4.36, tampak bahwa attribute file/direktori hanya memiliki attribute Read-only dan Hiden. Apabila kotak pilihan Read-only dipilih, maka file hanya akan dapat dibaca saja dan sebaliknya. Apabila kotak pilihan Hiden dipilih maka file/direktori akan disembunyikan sehingga tidak tampak ketika dicari dengan Windows Explorer.

Pada Unix dan keluarganya, termasuk linux, masalah attribute suatu file/direktori diatur dengan sangat ketat. Hal ini untuk meningkatkan keamanan dan memberi keleluasaan pada user untuk mengelola file dan direktori sesuai kebutuhannya.

Ada 4 bagian penting dalam suatu file / direktori, yaitu attribute, user atau (owner) pemilik dari file tersebut, grup dimana user sebagai anggota dan nama file/direktori.

Pada bagian attribute, ada penanda apakah itu direktori atau file biasa (ditandai dengan huruf d untuk direktori atau tanda – untuk file biasa). Selanjutnya ada sembilan kolom (karakter) yang menunjukkan hak akses terhadap file/direktori tersebut. Tiga kolom pertama menunjukkan hak akses owner, tiga kolom berikutnya hak akses grup dan tiga kolom terakhir adalah hak akser untuk other (user lain diluar owner dan anggota grup). Huruf **r** menunjukkan file/direktori bisa dibaca, **w** menunjukkan file/direktori bisa ditulis dan **x** menunjukkan file/direktori bisa dieksekusi. Perhatikan Gambar 4.37 berikut ini.



Gambar 4.37. Attribute file / direktori pada keluarga Unix

Gambar 4.37 menunjukkan hal sebagai berikut:

- **Desktop** dan **xml** adalah direktori karena mempunyai tanda d, sedangkan **exports** adalah file biasa karena bertanda -.
- **Desktop** dan **xml** mempunyai attribute **drwxr-xr-x** yang berarti owner (yaitu **one**) mempunyai hak untuk membaca, menulis dan mengeksekusi direktori ini. Sedangkan grup (yaitu **users**) mempunyai hak untuk membaca dan mengeksekusi saja. Other (user lain) juga mempunyai hak membaca dan mengeksekusi pada direktori ini.
- **exports** mempunyai attribute -rw-r--r—yang berarti owner (yaitu **one**) mempunyai hak untuk membaca dan menulis. Sedangkan grup dan other hanya mempunyai hak untuk membaca saja.

Untuk merubah attribute file/direktori dapat digunakan perintah seperti pada table berikut ini.

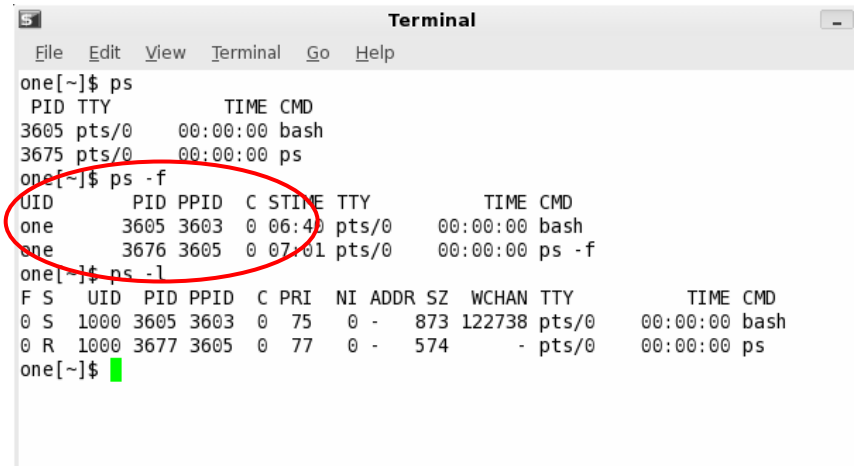
Tabel 4.1. Perintah yang berhubungan dengan pengelolaan file/direktori.

Perintah	**Fungsi**
chgrp [options] group file	Mengubah kepemilikan grup suatu file/direktori
chmod [options] owner file	Mengubah hak akses suatu file/direktori
chown [options] owner file	Mengubah kepemilikan owner suatu file/direktori

- Mengontrol proses

Proses merupakan bagian yang sangat penting dalam Linux sehingga perintah-perintah yang berhubungan dengan proses menjadi penting untuk diketahui.

Untuk melihat proses yang sedang berjalan dapat digunakan perintah ps. Perhatikan gambar 4.38 berikut ini.



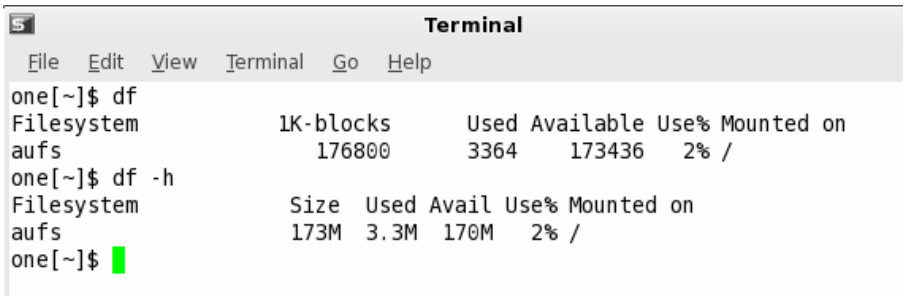
Gambar 4.38. Eksekusi perintah ps.

Seperti terlihat pada gambar 4.38, perintah ps memiliki beberapa opsi (opsi selengkapnya dapat dilihat dengan mengetikkan perintah **man ps** pada terminal). Pada gambar tersebut ada dua proses yang sedang dijalankan oleh user **one** (lihat bagian UID) yaitu **bash dengan nomor proses (PID) 3605** dan **ps –f dengan PID 3676**.

Untuk menghentikan proses kita dapat menggunakan perintah **kill** diikuti nomor prosesnya (PID). Misalnya : **kill 3605** untuk menghentikan proses bash.

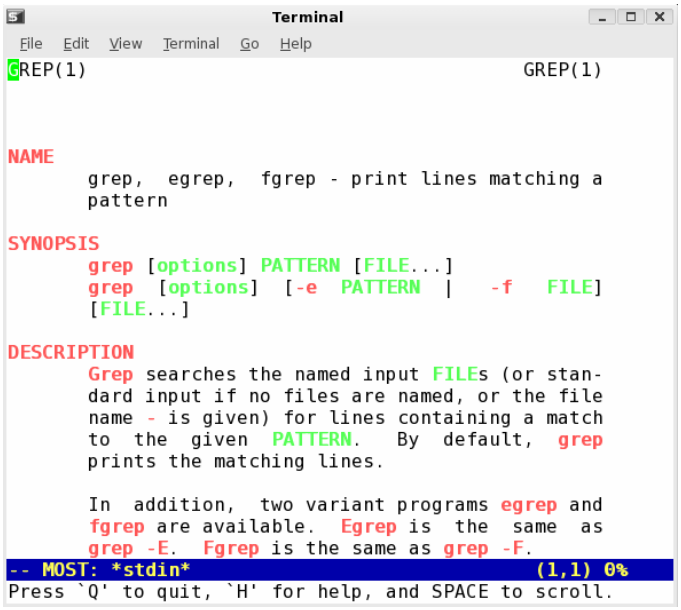
- Mengetahui ruang kosong pada disk

Kadang-kadang kita ingin mengetahui seberapa banyak sisa disk kita yang masih ada. Untuk mengetahui hal ini dapat digunakan perintah df seperti terlihat pada Gambar 4.39.



Gambar 4.39. Penggunaan perintah df.

Masih banyak sekali perintah yang digunakan dalam CLI di Linux. Jika kalian ingin mengetahui arti suatu perintah coba ketikkan perintah **man** diikuti nama perintah (Gambar 4.40).



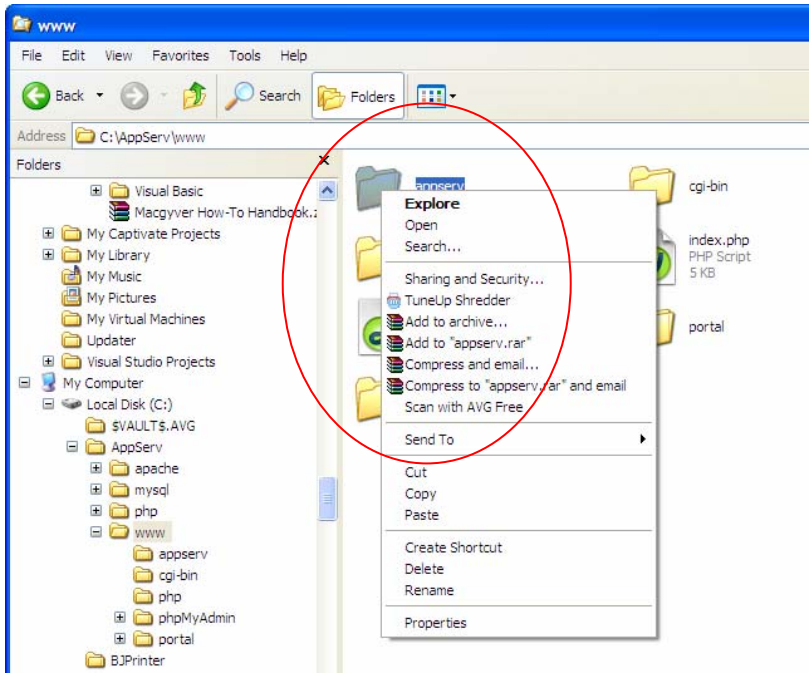
Gambar 4.40. Contoh hasil eksekusi perintah man untuk melihat manual suatu perintah.

4.3.4. Bekerja dengan GUI

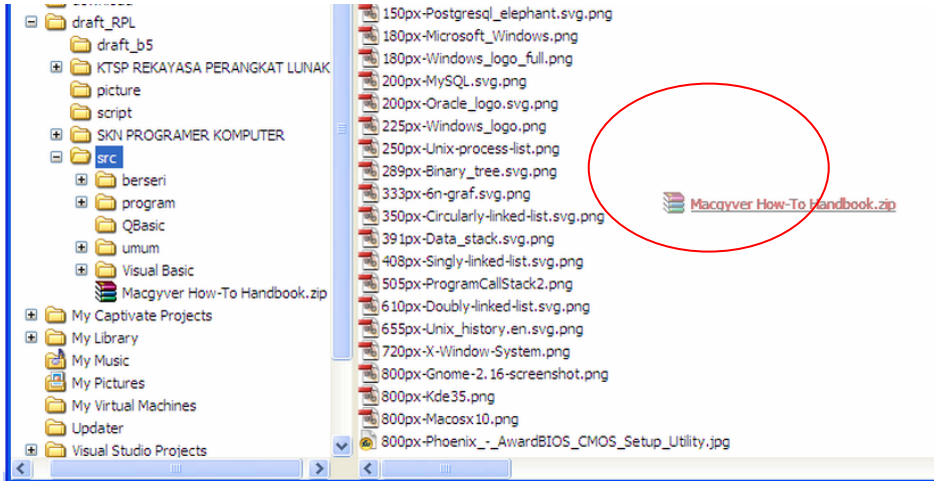
Secara umum bekerja dengan GUI pada sistem operasi sangat memudahkan pengguna karena pengguna hanya membutuhkan kerja mouse untuk melakukan sejumlah perintah. Mouse memiliki beberapa penggunaan, antara lain :

- klik satu kali digunakan untuk menunjuk satu file sebelum dilakukan operasi lain.

- Klik ganda (double-click) untuk mengeksekusi suatu perintah, misalnya membuka folder dan menjalankan file yang bisa dieksekusi.
- Klik kanan untuk membuka konteks menu (Gambar 4.41)
- Drag and drop untuk memindahkan file dari satu tempat ke tempat lain (Gambar 4.42).



Gambar 4.41. Membuka konteks menu dengan klik kanan.



Gambar 4.42. Drag and drop.

4.4. BEKERJA DALAM KOMPUTER JARINGAN

Bekerja dalam komputer yang terhubung ke jaringan, saat ini bukanlah sesuatu yang aneh. Hampir semua tempat yang memiliki banyak komputer, selalu menggunakan jaringan sebagai sarana berkomunikasi. Oleh karena itu pengetahuan dasar bagaimana dapat bekerja dalam komputer yang terhubung ke jaringan menjadi sangat penting.

4.4.1. Persiapan

Ada tiga hal penting yang harus dipersiapkan dalam koneksi ke jaringan komputer, yaitu perangkat keras, perangkat lunak dan akses ke jaringan.

o Perangkat keras

Gambar 4.43. Network Interface Card

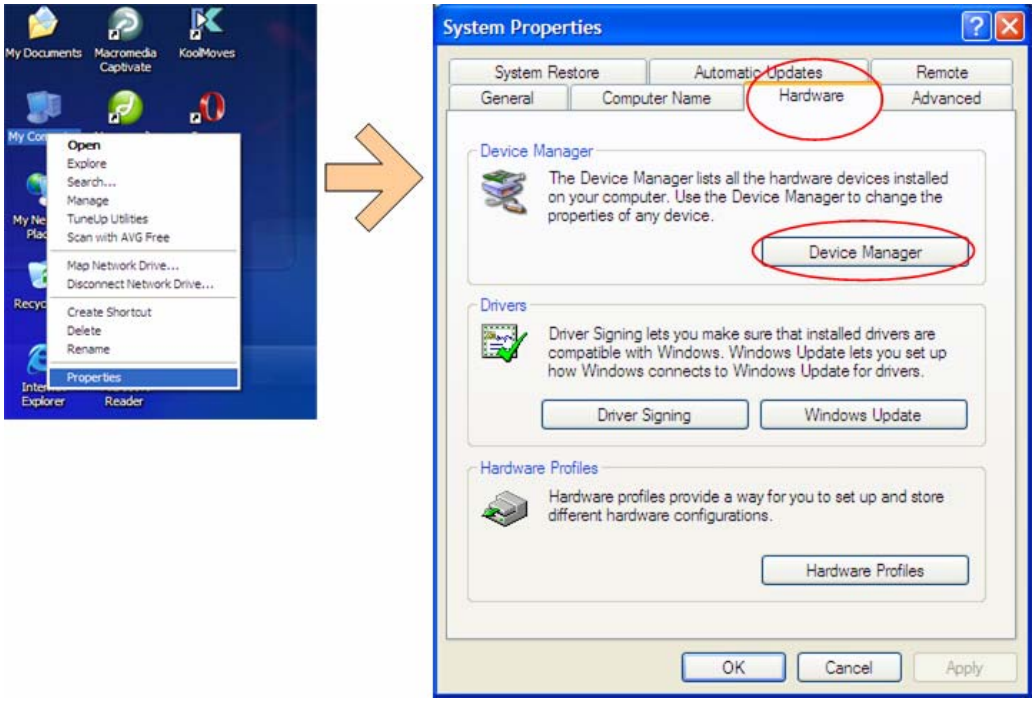
Kebutuhan perangkat keras sangat bergantung pada tipe koneksi jaringan yang akan digunakan. Untuk koneksi ke jaringan LAN maka kebutuhan utama adalah NIC (*Network Interface Card*) yang telah terpasang dengan baik dan telah terinstal driver yang sesuai dan kabel jaringan. Untuk koneksi ke jaringan dengan cara dial-up, dibutuhkan modem dan kabel telepon analog.

Kita dapat melihat apakah perangkat keras jaringan (NIC, modem atau yang lainnya) sudah terinstall dengan benar dengan memeriksa pada daftar perangkat keras yang dikenali oleh komputer.

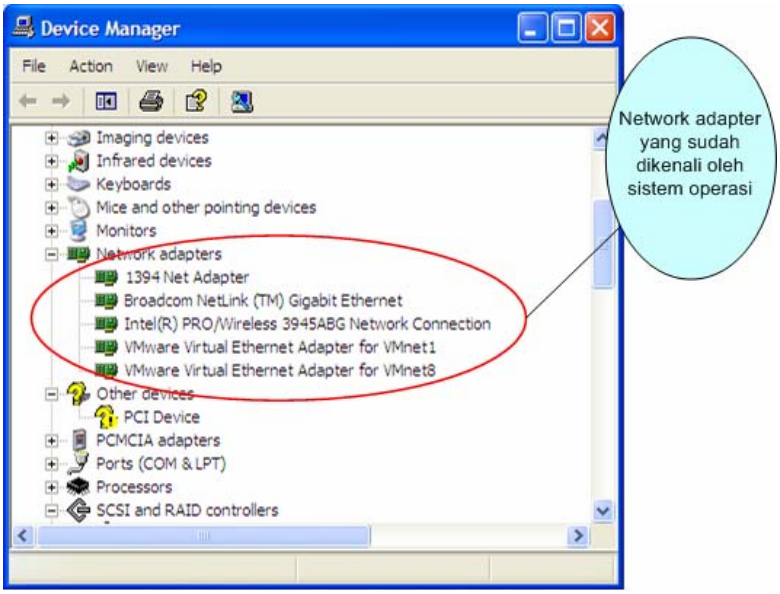
Pada sistem operasi Windows kita dapat melihat dengan cara klik kanan pada icon My Computer di desktop kemudian pilih Properties pada menu yang muncul (Gambar 4.44). Pada jendela System Properties pilih tab Hardware kemudian klik pada Device Manager (Gambar 4.44), sehingga akan muncul jendela Device Manager (Gambar 4.45).

Pada Gambar 4.45 terlihat bahwa network adapter yang digunakan oleh komputer sudah dikenali dengan baik. Apabila kita menemukan gambar tanda tanya pada suatu perangkat lunak berarti perangkat keras tersebut belum dikenali dengan baik (lihat Gambar 4.45).

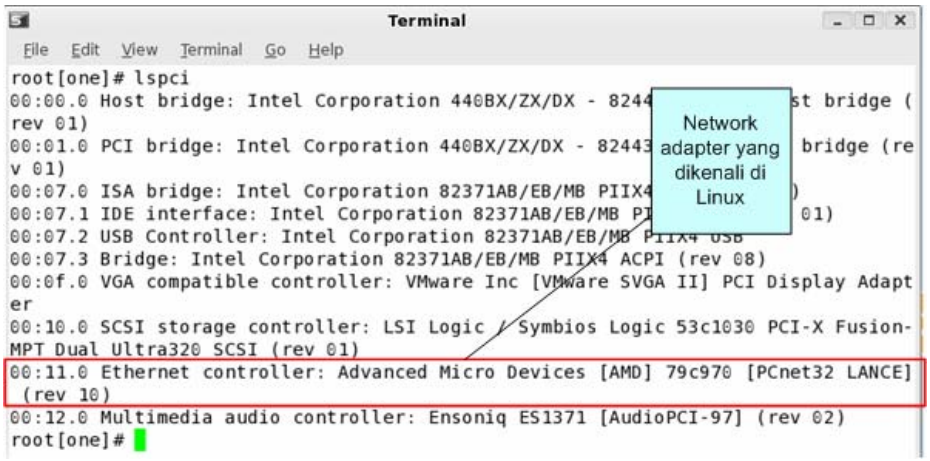
Pada sistem operasi Linux dan keluarganya kita dapat memeriksa apakah perangkat keras sudah dikenali atau tidak dengan cara mengetikkan perintah **lspci** (Gambar 4.46) dan ifconfig.



Gambar 4.44. Membuka system properties.



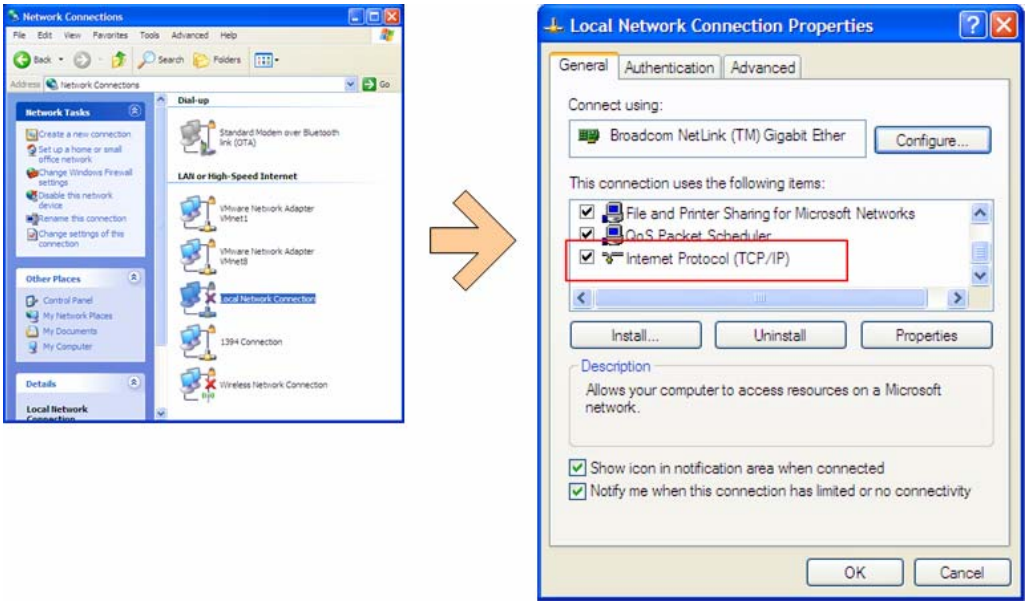
Gambar 4.45. Device manager.



Gambar 4.46. Output perintah lspci untuk memeriksa network adapter..

o Perangkat lunak

Perangkat lunak utama, selain sistem operasi adalah apakah paket TCP/IP sudah terinstall dengan benar pada komputer. Pada sistem operasi windows dapat dilakukan dengan double klik pada tipe koneksi, kemudian setelah jendela properties muncul cek apakah sudah ada TCP/IP yang sudah terinstal (Gambar 4.47).



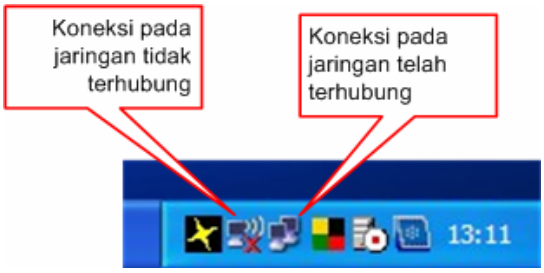
Gambar 4.47. Memeriksa protocol TCP/IP.

o Akses ke jaringan

Akses jaringan ini berhubungan dengan hak atau kewenangan kita dalam jaringan komputer yang akan kita masuki, terutama pemberian alamat IP (IP Address) dan password untuk masuk ke jaringan. Pada jaringan yang menerapkan DHCP maka kita tidak perlu khawatir karena no IP akan diberikan langsung ketika komputer berhubungan ke jaringan. Apabila tidak menggunakan DHCP maka kita harus memberikan no IP static yang diberikan oleh administrator jaringan pada komputer.

4.4.2. Konfigurasi koneksi jaringan

Pada sistem operasi modern sekarang ini koneksi ke jaringan bukan pekerjaan yang menyulitkan karena hampir semua koneksi telah dijalankan otomatis oleh sistem operasi. Pada jaringan LAN yang menggunakan DHCP, komputer yang menggunakan sistem operasi Windows (versi 2000 dan yang lebih baru) maupun Linux akan secara otomatis terkoneksi ke jaringan dan memperoleh no IP *dynamic*. Pada sistem operasi Windows, untuk memeriksa apakah komputer sudah tersambung ke jaringan kita bisa melihat pada *systray (notification area)* yang terletak di bagian kanan bawah desktop (Gambar 4.48).



Gambar 4.48. Kondisi koneksi jaringan.

4.4.3. Berbagi file, printer, dan sumber daya lain

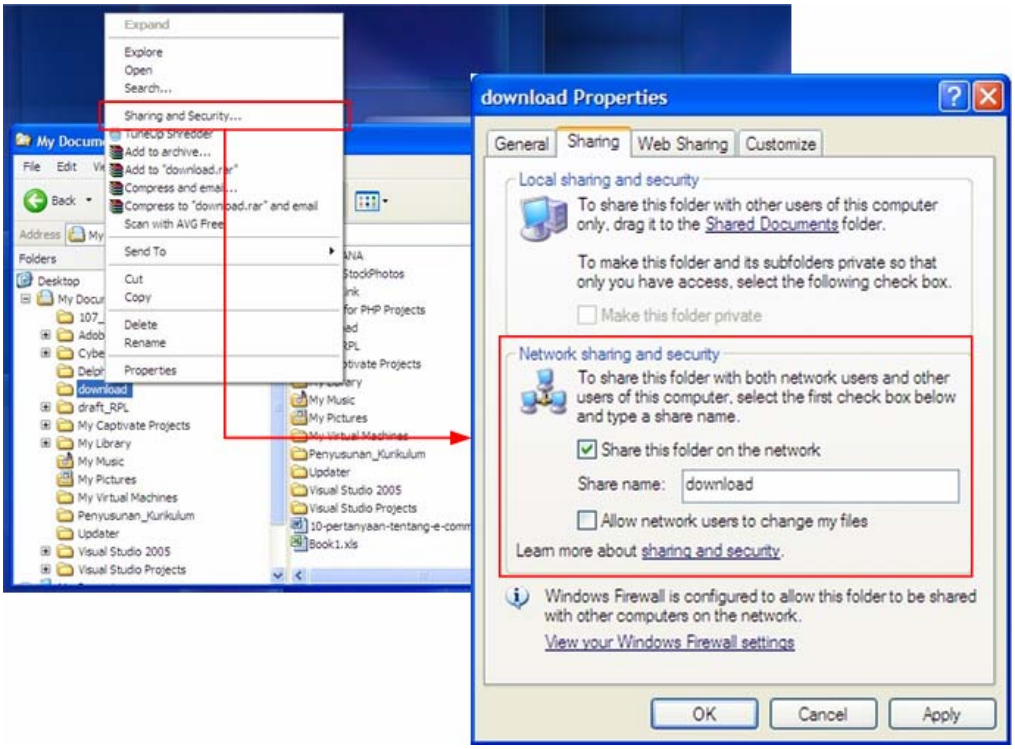
Kelebihan utama dari jaringan komputer adalah penggunaan secara bersama berbagai macam sumber daya, seperti: file, printer, media perekam (CD-RW atau DVD-RW), scanner dan lain-lain. Pada bagian berikut ini akan dijelaskan bagaimana berbagi file dan printer. Sumber daya yang lain dapat digunakan secara bersama-sama dengan cara yang tidak jauh berbeda dengan file dan printer.

- Berbagi file

Direktori atau file yang ada pada komputer kita dapat diatur agar dapat digunakan oleh komputer lain di dalam jaringan. Demikian pula sebaliknya kita dapat menggunakan direktori atau file pada komputer lain di jaringan.

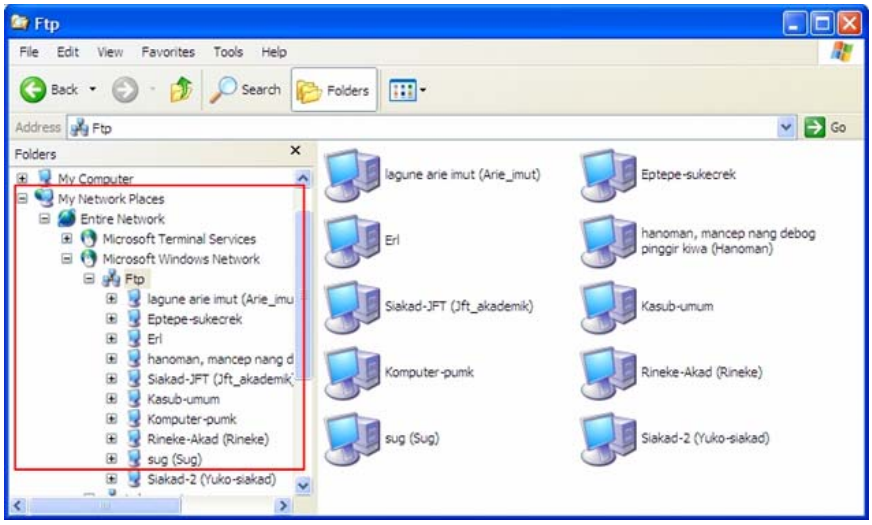
Untuk membagi (share) direktori atau file kita dapat menempuh cara berikut. Buka **Windows Explorer**, kemudian klik kanan pada direktori atau file yang akan kita share dan pilih **Sharing and Security**. Setelah

muncul jendela **properties**, pilih tab **Sharing** dan pada bagian **Network sharing and security** cek pada **Share this folder on the netw**ork dan beri nama untuk direktori yang di-share (Gambar 4.49).



Gambar 4.49. Mengatur file sharing.

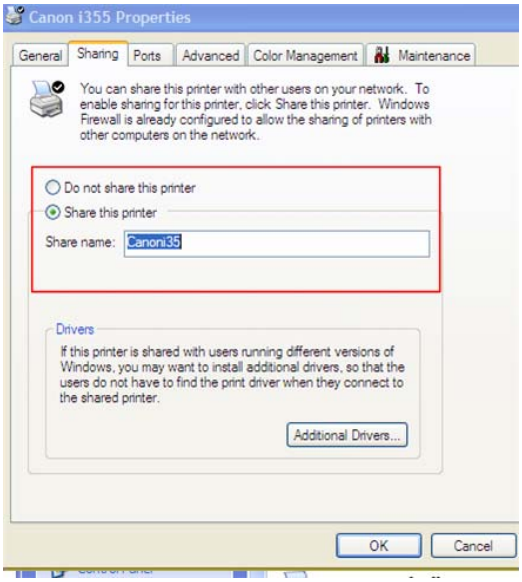
Untuk bisa mengakses direktori atau file di komputer lain, kita bisa membuka Windows Eksplorer kemudian klik pada My Network Places -> Entire Network -> Microsoft Windows Network. Kita akan mendapati tampilan seperti pada Gambar 4.50. Dari sini kita bisa melihat sumber daya apa yang dibagi pada masing-masing komputer yang terhubung ke jaringan dengan cara double klik pada nama komputer.



Gambar 4.50. Menjelajah komputer yang ada di jaringan.

o Berbagi printer

Untuk dapat berbagi printer yang ada di komputer, cara yang hampir sama dapat kita lakukan. Pertama kita buka jendela Printers and Faxes dengan cara Start -> Settings -> Printers and Faxes. Setelah jendela terbuka, klik kanan pada printer yang akan di share dan pilih Sharing …. Setelah jendela properties printer terbuka, pilih tab Sharing dan klik pada Share this printer serta beri nama (Gambar 4.51).



Gambar 4.51. Printer sharing.

4.5. RINGKASAN

- Sistem Operasi adalah perangkat lunak yang bertugas mengelola penggunaan sumberdaya dalam komputer dan menyediakan antarmuka bagi pengguna untuk mengakses sumberdaya tersebut.
- Fungsi-fungsi system operasi adalah sebagai antar muka pengguna, manajemen memori, manajemen file, manajemen proses dan manajemen input/output
- Fungsi utama BIOS (Basic Input/Output System) adalah untuk mengidentifikasi dan mengenali perangkat keras komputer.
- Ada beberapa sistem operasi yang dikenal yaitu DOS, Windows, Mac OS, UNIX dan Linux.
- Setiap sistem operasi yang akan dijalankan harus diinstal terlebih dahulu.
- Sistem operasi dapat menggunakan perintah berbasis teks atau GUI tergantung pada konfigurasi dan fasilitas yang dipunyai oleh system operasi tersebut.

4.6. SOAL-SOAL LATIHAN

1. Sebutkan pengertian sistem operasi.
2. Jelaskan fungsi-fungsi sistem operasi.
3. Bagaimanakah tahapan-tahapan proses booting suatu computer?
4. Cobalah instalasi satu distro sistem operasi linux pada sebuah computer dan cermati jalannya instalasi. Kemudian bandingkan dengan proses instalasi pada sistem Windows. Menurut kalian apakah ada perbedaan penting dalam proses instalasi kedua sistem operasi tersebut?
5. Cobalah booting pada system operasi Linux, kemudian cermati jalannya proses booting dan bandingkan dengan proses booting pada Windows. Bagaimanakah menurut kalian perbedaannya?.
6. Jalankan system operasi Linux, kemudian bukalah jendela terminal terminal. Lakukan serangkaian perintah dengan menggunakan perintah ls, cd, find, cat, cp, mv, dan mkdir. Catatlah apa yang kalian temui ketika menjalankan perintah-perintah tersebut.
7. Sebuah file mempunyai atribut **–rw-r--r--** dan dimiliki oleh user bernama rony. Apakah arti dari atribut tersebut. Bagaimanakah caranya jika ada user lain supaya bias mempunyai hak akses membaca dan menulis pada file tersebut?

BAB 5 ALGORITMA PEMROGRAMAN DASAR

Perangko dari Rusia pada Gambar 5.1. di samping ini bergambar seorang pria dengan nama Muhammad ibn Mūsā al-Khwārizmī. Bagi kalian yang sedang berkecimpung dalam dunia komputer maka seharusnya mengetahui siapa orang di samping ini. Dia adalah seorang ilmuwan Islam yang karya-karyanya dalam bidang matematika, astronomi, astrologi dan geografi banyak menjadi dasar perkembangan ilmu modern. Dan dari namanya istilah yang akan kita pelajari dalam bab ini muncul. Dari Al-Khawarizmi kemudian berubah menjadi ***algorithm*** dalam Bahasa Inggris dan diterjemahkan menjadi ***algoritma*** dalam Bahasa Indonesia.

(Sumber: www.wikipedia.org)

Gambar 5.1. Perangko bergambar Muhammad ibn Mūsā al-Khwārizmī.

Standar kompetensi algoritma pemrograman dasar terdiri atas empat kompetensi dasar. Dalam penyajian pada buku ini, setiap kompetensi dasar memuat uraian materi, dan latihan. Ringkasan diletakkan pada setiap akhir bab. Kompetensi dasar pada bab ini adalah menjelaskan variabel, konstanta dan tipe data, membuat algoritma/logika alur pemrograman, menerapkan pengelolaan array, dan mengoperasikan file. Sebelum mempelajari kompetensi ini ingatlah kembali sistem operasi, prinsip pemecahan masalah, dan materi-materi pendukung dari mata pelajaran matematika.

Pada akhir bab, tercantum soal-soal latihan yang disusun dari soal-soal yang mudah hingga soal-soal yang sulit. Latihan soal ini digunakan untuk mengukur kemampuan terhadap kompetensi dasar ini. Artinya setelah mempelajari kompetensi dasar ini secara mandiri dengan bimbingan guru sebagai fasilitator, ukurlah sendiri kemampuan dengan mengerjakan soal-soal latihan tersebut.

TUJUAN

Setelah mempelajari bab ini diharapkan pembaca akan mampu :

- Menjelaskan variabel, konstanta dan tipe data
- Membuat algoritma/logika alur pemrograman
- Menerapkan pengelolaan array
- Mengoperasikan file

5.1. VARIABEL, KONSTANTA DAN TIPE DATA

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

5.1.1. Variabel

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

Contoh 5.1. Nama variabel dan nilai.

```
username = "joni"
Nama = "Al-Khawarizmi"
Harga = 2500
HargaTotal = 34000
```

Pada contoh 5.1. di atas, **username, Nama, harga** dan **HargaTotal** adalah nama dari variabel sedangkan **"joni", "Al-Khawarizmi", 2500** dan **34000** adalah nilai dari masing-masing variabel. Nilai-nilai ini akan tersimpan di dalam nama variabel masing-masing sepanjang tidak kita rubah.

Pada sebagian besar bahasa pemrograman, variabel harus dideklarasikan lebih dulu untuk mempermudah *compiler* bekerja. Apabila variabel tidak dideklarasikan maka setiap kali *compiler* bertemu dengan variabel baru pada kode program akan terjadi waktu tunda karena *compiler* harus membuat variabel baru. Hal ini memperlambat proses kerja compiler. Bahkan pada beberapa bahasa pemrograman, *compiler* akan menolak untuk melanjutkan proses kompilasi.

Pemberian nama variabel harus mengikuti aturan yang ditetapkan oleh bahasa pemrograman yang kita gunakan. Namun secara umum ada aturan yang berlaku untuk hampir semua bahasa pemrograman. Aturan-aturan tersebut yaitu:

- Nama variabel harus diawali dengan huruf.
- Tidak boleh menggunakan spasi pada satu nama variabel. Spasi bisa diganti dengan karakter underscore (_).

- Nama variabel tidak boleh mengandung karakter-karakter khusus, seperti : .,+, -, *, /, <, >, &, (,) dan lain-lain.
- Nama variabel tidak boleh menggunakan kata-kata kunci d bahasa pemrograman

Contoh 5.2. Contoh penamaan variabel.

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

5.1.2. Konstanta

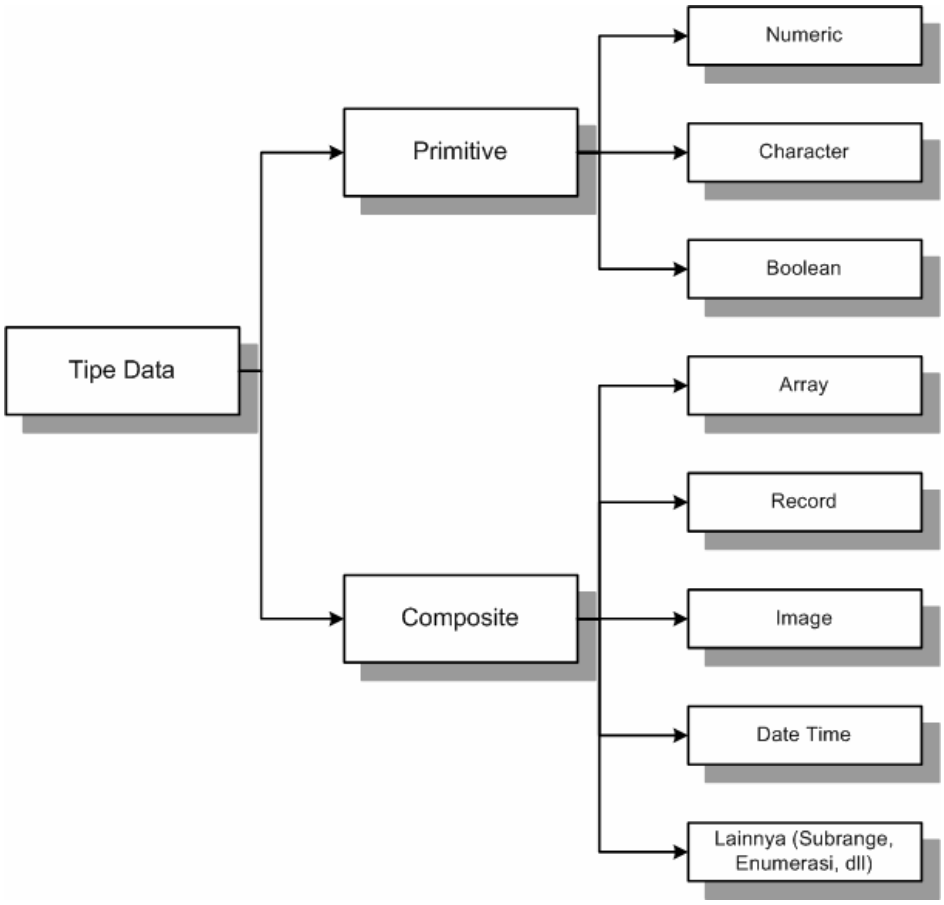
Konstanta adalah variabel yang nilai datanya bersifat tetap dan tidak bisa diubah. Jadi konstanta adalah juga variabel bedanya adalah pada nilai yang disimpannya. Jika nilai datanya sepanjang program berjalan tidak berubah-ubah, maka sebuah varibel lebih baik diperlakukan sebagai konstanta. Pada sebuah kode program, biasanya nilai data dari konstanta diberikan langsung di bagian deklarasi konstanta. Sedangkan untuk variabel biasanya hanya ditentukan nama variabel dan tipe datanya tanpa isian nilai data. Aturan penamaan variabel juga berlaku untuk penamaan konstanta. Demikian juga aturan penetapan tipe data.

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

5.1.3. Tipe Data

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

Ada banyak tipe data yang tersedia tergantung jenis bahasa pemrograman yang dipakai. Namun secara umum dapat dikelompokkan seperti pada Gambar 5.2.



Gambar 5.2. Pengelompokkan tipe data.

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

Tipe data *numeric*

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

Jenis yang termasuk dalam tipe data *numeric* antara lain *integer* (bilangan bulat), dan *float* (bilangan pecahan). Selain jenis, dalam bahasa pemrograman juga diterapkan presisi angka yang digunakan, misalnya tipe data *Single* adalah tipe data untuk bilangan pecahan dengan presisi yang terbatas, sedangkan tipe data *Double* adalah tipe data untuk bilangan pecahan dengan presisi yang lebih akurat. Pada bab-bab berikutnya yang membahas aplikasi bahasa pemrograman bagian ini akan diuraikan lebih lanjut.

Penentuan tipe data numeric untuk suatu variabel/konstanta harus sangat berhati-hati. Manual dan petunjuk pada masing-masing bahasa pemrograman pada bagian tipe data harus diperhatikan dengan seksama. Perhatikan contoh berikut.

Contoh 5.3. Penggunaan tipe data *numeric*.

Kode Program A

```
#include <iostream>
using namespace std;
int main() {
      int x, z;
      float y;
      x = 12;
      y = 2.15;
      z = x * y;
      cout << "X =" << x << endl;
      cout << "Y =" << y << endl;
      cout << "Z =" << z << endl;
      return 0; }
```

Hasil eksekusi Program A

```
X =12
Y =2.15
Z =25
```

Kode Program B

```
#include <iostream>
using namespace std;
int main() {
      int x;
      float y, z;
      x = 12.8;
      y = 2.15;
      z = x * y;
      cout << "X =" << x << endl;
      cout << "Y =" << y << endl;
      cout << "Z =" << z << endl;
      return 0; }
```

Hasil eksekusi Program B

```
X =12
Y =2.15
Z =25.8
```

Kode Program C

```
#include <iostream>
using namespace std;
int main() {
      int x;
      float y, z;
      x = 12;
      y = 2.15;
      z = x * y;
      cout << "X =" << x << endl;
      cout << "Y =" << y << endl;
      cout << "Z =" << z << endl;
      return 0; }
```

Hasil eksekusi Program C

```
X =12
Y =2.15
Z =25.8
```

Ketiga kode program di atas (A, B dan C) ditulis dengan bahasa C++. Sekilas sama namun berbeda pada penggunaan tipe data dan pengisian nilai.

Pada kode program A, variabel *x* dan *z* kita deklarasikan bertipe data *int* (Integer = bilangan bulat) dan *y* bertipe data *float* (pecahan). Hasil eksekusi program A menunjukkan hasil yang tidak kita inginkan. Nilai z yang merupakan perkalian *x* dengan *y* harusnya bernilai 25.8 (hasil dari 12 x 2.15). Namun karena *z* dideklarasikan bertipe data *int* maka hasilnya menjadi 25. Dari ketiga kode program di atas yang paling benar adalah kode program C. Mengapa kode program B salah? Cobalah cermati bagian kode yang dicetak tebal kemudian tentukan dimana terjadi kesalahan.

Character

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

Contoh 5.4. Penggunaan tipe data character.

Kode program

```
#include <iostream>
using namespace std;

int main() {
    int x;
    x = 5;
    char huruf = 'A';
    char* kata = "Java";

    cout << "X = " << x << endl;
    cout << "Isi variabel huruf = "
    << huruf << endl;
    cout << "Isi variabel kata = " <<
    kata << endl;
    return 0; }
```

Hasil eksekusi program

```
X = 5
Isi variabel huruf = A
Isi variabel kata = Java
```

Pada contoh ini kita mendeklarasikan variabel *x* sebagai int (Integer), sedangkan variabel *huruf* dan *kata* bertipe data char (character). Perhatikan hasil eksekusi kode program di atas.

Boolean

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

Array

Array atau sering disebut sebagai larik adalah tipe data yang sudah terstruktur dengan baik, meskipun masih sederhana. Array mampu menyimpan sejumlah data dengan tipe yang sama (homogen) dalam sebuah variabel. Setiap lokasi data array diberi nomor indeks yang berfungsi sebagai alamat dari data tersebut. Penjelasan tentang array akan disampaikan lebih detil pada bagian lain dari bab ini.

Record atau Struct

Seperti halnya Array, Record atau Struct adalah termasuk tipe data komposit. Record dikenal dalam bahasa Pascal/Delphi sedangkan Struct dikenal dalam bahasa C++. Berbeda dengan array, tipe data record mampu menampung banyak data dengan tipe data berbeda-beda (heterogen). . Sebagai ilustrasi array mampu menampung banyak data namun dengan satu tipe data yang sama, misalnya integer saja. Sedangkan dalam record, kita bisa menggunakan untuk menampung banyak data dengan tipe data yang berbeda, satu bagian integer, satu bagian lagi character, dan bagian lainnya Boolean. Biasanya record digunakan untuk menampung data suatu obyek. Misalnya, siswa memiliki nama, alamat, usia, tempat lahir, dan tanggal lahir. Nama akan akan menggunakan tipe data string, alamat bertipe data string, usia bertipe data single (numeric), tempat lahir bertipe data string dan tanggal lahir bertipe data date. Berikut ini contoh pendeklarasian record dalam Delphi.

Contoh 5.5. Deklarasi tipe data record pada Delphi.

```
Type TRecord_Siswa = Record
                Nama_Siswa          : String[30]
                Alamat              : String[50]
                Usia                : Real
           EndRecord
```

Image

Image atau gambar atau citra merupakan tipe data grafik. Misalnya grafik perkembangan jumlah siswa SMK, foto keluarga kita, video perjalanan dan lain-lain. Pada bahasa-bahasa pemrograman modern terutama yang berbasis visual tipe data ini telah didukung dengan sangat baik.

Date Time

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

Contoh 5.6. Penggunaan tipe data date time pada Visual Basic.

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

Tipe data lain

Subrange

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

Contoh 5.7. Deklarasi tipe data subrange pada Delphi.

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

Enumerasi

Tipe data ini merupakan tipe data yang mempunyai elemen-elemen yang harus disebut satu persatu dan bernilai konstanta integer sesuai dengan urutannya. Nilai konstanta integer elemen ini diwakili oleh suatu nama variable yang ditulis di dalam kurung. Tipe data ini juga dijumpai pada Delphi dan bahasa pemrograman deklaratif seperti SQL. Berikut ini contoh deklarasi tipe data enumerasi dalam Delphi.

Contoh 5.8. Penggunaan tipe data enumerasi.

```
Type
     Hari_dlm_Minggu = (Nol, Senin, Selasa, Rabu,
                        Kamis, Jumat, Sabtu,
                        Minggu)
     Nama_Bulan = (Nol, Januari, Pebruari, Maret,
                   April, Mei, Juni, Juli,
                   Agustus,
                   September, Oktober, Nopember,
                   Desember)
Var
     No_Hari         : Hari_dlm_Minggu
     No_Bulan        : Nama_Bulan
```

Pada contoh di atas tipe data Hari_dlm_Minggu termasuk enumerasi dengan rentang nilai Nol, Senin sampai dengan Minggu dan nilai data dari 0, 1, sampai dengan 7. Sedangkan tipe data Nama_Bulan termasuk enumerasi dengan rentang nilai Nol, Januari sampai dengan Desember dan nilai data dari 0, 1, sampai dengan 12.

Object

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

Contoh 5.9. Penggunaan tipe data *object.*

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

Pada contoh ini variabel A dideklarasikan bertipe data Object yaitu CommandButton. Kemudian kita set variabel A dengan control Command button yang ada pada form (Command1). Dengan cara ini kita dapat mengakses seluruh *property, method* dan *event* obyek Command1 dengan menggunakan variabel A.

Variant

Tipe data hanya ada di Visual Basic. Tipe ini adalah tipe data yang paling fleksibel di antara tipe data yang lain, karena dapat mengakomodasi semua tipe data yang lain seperti telah dijelaskan.

5.2. STRUKTUR ALGORITMA PEMROGRAMAN

5.2.1. Pengertian Algoritma

Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis. Masalah dapat berupa apa saja, dengan catatan untuk setiap masalah, ada syarat kondisi awal yang harus dipenuhi sebelum menjalankan algoritma. Konsep algoritma sering kali disetarakan dengan sebuah resep. Sebuah resep biasanya memiliki daftar bahan atau bumbu yang akan digunakan, urutan pengerjaan dan bagaimana hasil dari urutan pengerjaan tersebut. Apabila bahan yang digunakan tidak tertera (tidak tersedia) maka resep tersebut tidak akan dapat dikerjakan. Demikian juga jika urutan pengerjaannya tidak beraturan, maka hasil yang diharapkan tidak akan dapat diperoleh.

Algoritma yang berbeda dapat diterapkan pada suatu masalah dengan syarat yang sama. Tingkat kerumitan dari suatu algoritma merupakan ukuran seberapa banyak komputasi yang dibutuhkan algoritma tersebut untuk menyelesaikan masalah. Umumnya, algoritma yang dapat menyelesaikan suatu permasalahan dalam waktu yang singkat memiliki tingkat kerumitan yang rendah, sementara algoritma yang membutuhkan waktu lama untuk menyelesaikan suatu masalah membutuhkan tingkat kerumitan yang tinggi.

Perhatikan algoritma sederhana berikut.

Contoh 5.10. Algoritma menghitung luas segitiga.

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

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

Sekilas algoritma di atas benar, namun apabila dicermati maka algoritma ini mengandung kesalahan yang mendasar, yaitu tidak ada pembatasan pada nilai data untuk alas dan tinggi. Bagaimana jika nilai data alas atau tinggi adalah bilangan 0 atau bilangan negatif ? Tentunya hasil yang keluar menjadi tidak sesuai dengan yang diharapkan. Dalam kasus seperti ini kita perlu menambahkan langkah untuk memastikan nilai alas dan tinggi memenuhi syarat, misalnya dengan melakukan pengecekan pada input yang masuk. Apabila input nilai alas dan tinggi kurang dari 0 maka program tidak akan dijalankan. Sehingga algoritma di atas dapat dirubah menjadi seperti contoh berikut.

Contoh 5.11. Hasil perbaikan algoritma perhitungan luas segitiga.

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

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

5.2.2. Cara Penulisan Algoritma

Ada tiga cara penulisan algoritma, yaitu :

- ***Structured English (SE)***

SE merupakan alat yang cukup baik untuk menggambarkan suatu algoritma. Dasar dari SE adalah Bahasa Inggris, namun kita dapat memodifikasi dengan Bahasa Indonesia sehingga kita boleh menyebutnya sebagai Structured Indonesian (SI). Algoritma seperti pada Contoh 5.10 dan 5.11 merupakan algoritma yang ditulis menggunakan SI. Karena dasarnya adalah bahasa sehari-hari, maka SE atau SI lebih tepat untuk menggambarkan suatu algoritma yang akan dikomunikasikan kepada pemakai perangkat lunak.

- ***Pseudocode***

Pseudocode mirip dengan SE. Karena kemiripan ini kadang-kadang SE dan Pseudocode dianggap sama. *Pseudo* berarti imitasi atau tiruan atau menyerupai, sedangkan *code* menunjuk pada kode program. Sehingga *pseudocode* adalah kode yang mirip dengan instruksi kode program sebenarnya. *Pseudocode* didasarkan pada bahasa pemrograman yang sesungguhnya seperti BASIC, FORTRAN atau PASCAL. *Pseudocode* yang berbasis bahasa PASCAL merupakan *pseudocode* yang sering digunakan. Kadang-kadang orang menyebut *pseudocode* sebagai *PASCAL-LIKE* algoritma. Apabila Contoh 5.10 ditulis dalam pseudocode berbasis bahasa BASIC akan tampak seperti pada contoh 5.12.

Contoh 5.12. *Pseudocode.*

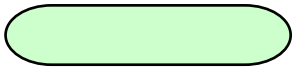
```
1. Start
2. READ alas, tinggi
3. Luas = 0.5 * alas * tinggi
4. PRINT Luas
5. Stop
```

Pada Contoh 5.12 tampak bahwa algoritma sudah sangat mirip dengan bahasa BASIC. Pernyataan seperti READ dan PRINT merupakan *keyword* yang ada pada bahasa BASIC yang masing-masing menggantikan kata "baca data" dan "tampilkan". Dengan menggunakan *pseudocode* seperti di atas maka proses penterjemahan dari algoritma ke kode program menjadi lebih mudah.

- ***Flowchart***

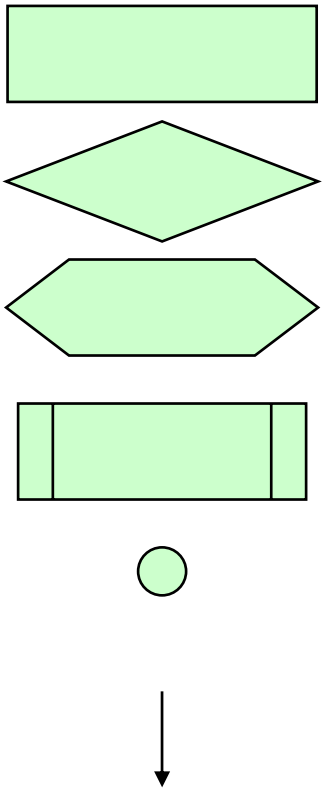
Flowchart atau bagan alir adalah skema/bagan (*chart*) yang menunjukkan aliran (*flow*) di dalam suatu program secara logika. *Flowchart* merupakan alat yang banyak digunakan untuk menggambarkan algoritma dalam bentu notasi-notasi tertentu. Secara lebih detil bagian ini akan dibahas pada bagian berikutnya.

Pada *flowchart* ada beberapa simbol penting yang digunakan untuk membuat algoritma sebagaimana tercantum pada Gambar 5.3.



Notasi ini disebut *Terminator* yang berarti digunakan untuk menunjukkan awal dan akhir suatu algoritma

Notasi ini disebut *Data* yang digunakan untuk mewakili data input atau output atau menyatakan operasi pemasukan data dan pencetakan hasil.



Notasi ini disebut *Process* yang digunakan untuk mewakili suatu proses.

Notasi ini disebut *Decision* yang digunakan untuk suatu pemilihan, penyeleksian kondisi di dalam suatu program

Notasi ini disebut *Preparation* yang digunakan untuk memberi nilai awal, nilai akhir, penambahan/pengurangan bagi suatu variable *counter*.

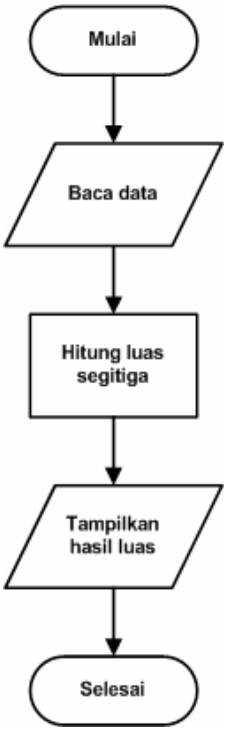
Notasi ini disebut *Predefined Process* yang digunakan untuk menunjukkan suatu operasi yang rinciannya ditunjukkan ditempat lain (prosedur, sub-prosedur, fungsi)

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

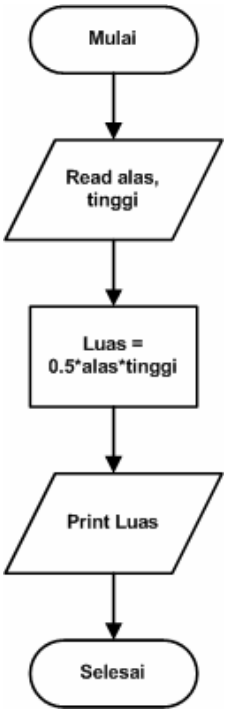
Notasi ini disebut *Arrow* yang digunakan untuk menunjukkan arus data atau aliran data dari proses satu ke proses lainnya.

Gambar 5.3. Simbol-simbol yang digunakan dalam *flowchart.*

Program *Flowchart* dapat terdiri dari dua macam, yaitu bagan alir logika program (*program logic flowchart*) dan bagan alir program komputer terinci (*detailed computer program flowchart*). Bagan alir logika program digunakan untuk menggambarkan tiap-tiap langkah di dalam program komputer secara logika dan biasanya dipersiapkan oleh seorang analis system. Sedangkan bagan alir program komputer terinci digunakan untuk menggambarkan instruksi-instruksi program komputer secara terinci dan biasanya dipersiapkan oleh seorang programmer. Apabila Contoh 5.10 dibuat program *flowchart*nya maka akan tampak pada gambar 5.4.



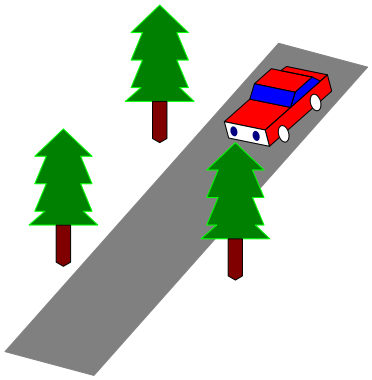
Bagan alir logika program



Bagan alir program komputer terinci

Gambar 5.4. *Program flowchart*.

5.2.3. Struktur Algoritma Berurutan



Gambar 5.5. Mobil sedang berjalan pada jalur lurus.

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

Pada bagian ini kita akan bahas lebih dulu struktur algoritma berurutan. Struktur berurutan dapat kita samakan dengan mobil yang sedang berjalan pada jalur lurus yang tidak terdapat persimpangan seperti tampak pada Gambar 5.5. Mobil tersebut akan melewati kilometer demi kilometer jalan sampai tujuan tercapai.

Struktur berurutan terdiri satu atau lebih instruksi. Tiap instruksi dikerjakan secara berurutan sesuai dengan urutan penulisannya, yaitu sebuah instruksi dieksekusi setelah instruksi sebelumnya selesai dieksekusi. Urutan instruksi menentukan keadaan akhir dari algoritma. Bila urutannya diubah, maka hasil akhirnya mungkin juga berubah. Menurut Goldshlager dan Lister (1988) struktur berurutan mengikuti ketentuan-ketentuan sebagai berikut:

- tiap instruksi dikerjakan satu persatu
- tiap instruksi dilaksanakan tepat sekali, tidak ada yang diulang
- urutan instruksi yang dilaksanakan pemroses sama dengan urutan aksi sebagaimana yang tertulis di dalam algoritmanya
- akhir dari instruksi terakhir merupakan akhir algoritma.

Contoh 5.13. *Flowchart* untuk menghitung luas bangun.

Buatlah *flowchart* untuk menghitung:

a. volume balok

b. luas lingkaran

Penyelesaian:

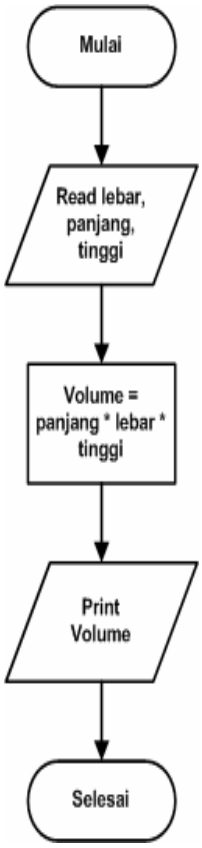
Soal ini merupakan permasalahan dengan algoritma struktur berurutan karena tidak ada proses pemilihan atau pengulangan. Untuk volume balok, kita harus menentukan variabel input dan output yang dibutuhkan. Untuk menghitung volume balok dibutuhkan variabel input panjang, lebar dan tinggi. Sedangkan variabel outputnya adalah volume. Pada luas lingkaran dibutuhkan variabel input radius dan variabel output luas. Untuk menghitung luas lingkaran ini kita juga membutuhkan konstanta phi. *Flowchart* untuk dua masalah ini dapat dilihat pada Gambar 5.6.

Contoh 5.14. *Flowchart* untuk konversi suhu.

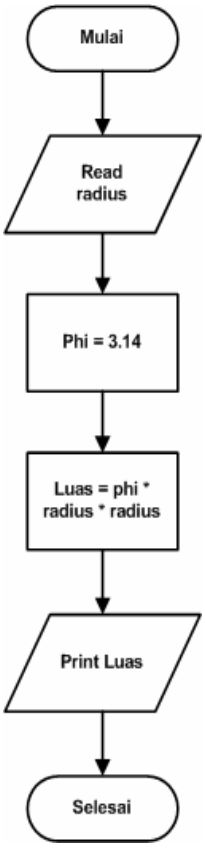
Buat *flowchart* untuk mengubah temperatur dalam Fahrenheit menjadi temperatur dalam Celcius dengan rumus $^{o}C = 5/9 \times (^{o}F - 32)$.

Penyelesaian:

Soal ini juga masih menggunakan algoritma dengan struktur berurutan. Variabel input yang dibutuhkan adalah F dan variabel outputnya adalah C. *Flowchart* untuk dua masalah ini dapat dilihat pada Gambar 5.7.

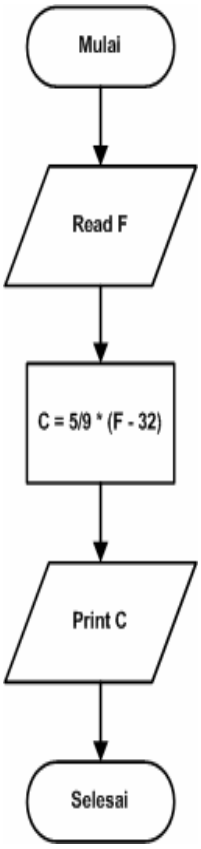


a. *flowchart* menghitung volume balok



b. *flowchart* menghitung luas lingkaran

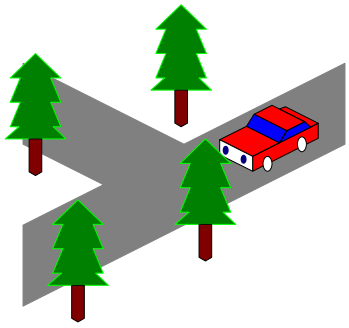
Gambar 5.6. *Flowchart* menghitung volume balok dan luas lingkaran.



Gambar 5.7. *Flowchart* untuk konversi suhu.

5.2.4. Struktur Algoritma Percabangan

Sebuah program tidak selamanya akan berjalan dengan mengikuti struktur berurutan, kadang-kadang kita perlu merubah urutan pelaksanaan program dan menghendaki agar pelaksanaan program meloncat ke baris tertentu. Peristiwa ini kadang disebut sebagai percabangan/pemilihan atau keputusan. Hal ini seperti halnya ketika mobil berada dalam persimpangan seperti pada Gambar 5.7. Pengemudi harus memutuskan apakah harus menempuh jalur yang kanan atau yang kiri.



Gambar 5.8. Mobil sedang dalam persimpangan.

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

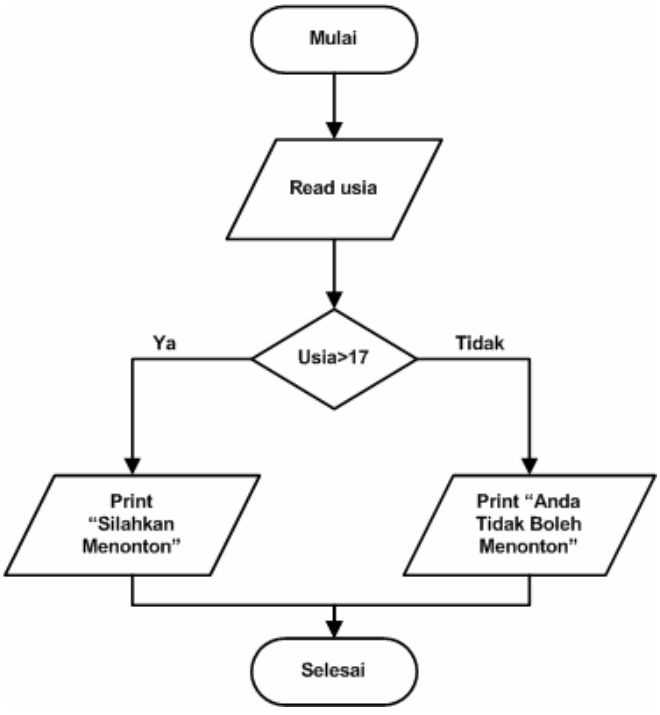
Contoh 5.15. Struktur percabangan untuk masalah batasan umur.

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

Penyelesaian:

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

Flowchart penyelesaian masalah tampak pada Gambar 5.9. Pada gambar tersebut, tampak penggunaan simbol Decision. Pada simbol ini terjadi pemeriksaan kondisi, yaitu apakah usia lebih dari 17 tahun atau tidak. Jika jawaban ya maka program akan menghasilkan keluaran teks "Silahkan Menonton", sedangkan jika input usia kurang dari 17 tahun maka program akan menghasilkan keluaran teks "Anda Tidak Boleh Menonton".



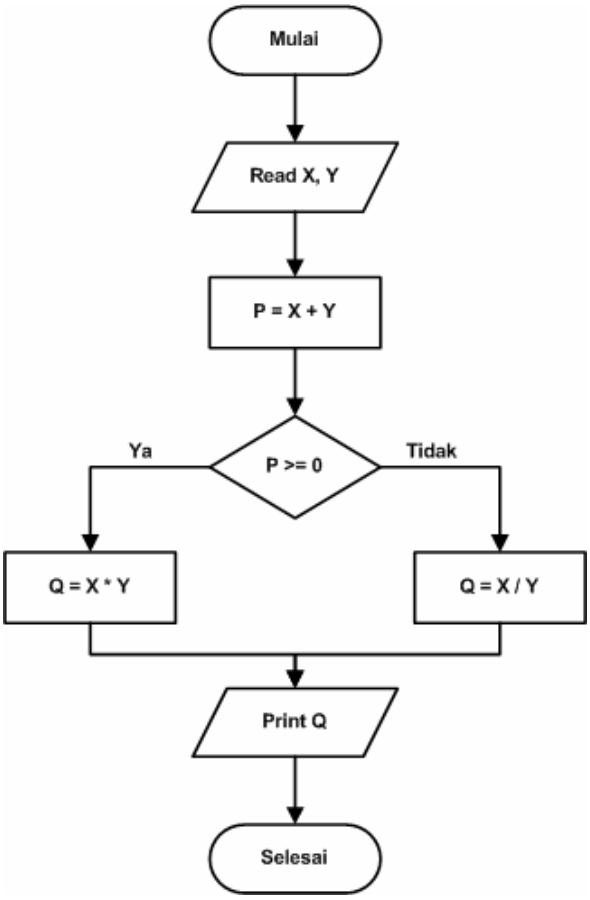
Gambar 5.9. *Flowchart* penyelesaian masalah nonton film.

Contoh 5.16. Struktur percabangan untuk perhitungan dua buah bilangan.

Dalam suatu perhitungan nilai P = X + Y. Jika P positif, maka Q = X * Y, sedangkan jika negative maka nilai Q = X/Y. Buatlah *flowchart* untuk mencari nilai P dan Q

Penyelesaian:

Pada contoh ini input yang dibutuhkan adalah nilai X dan Y, sedangkan proses pemeriksaan kondisi dilakukan pada nilai P apakah positif (termasuk 0) ataukah negative. Perhatikan *flowchart* penyelesaian masalah pada Gambar 5.10.



Gambar 5.10. *Flowchart* penyelesaian untuk perhitungan dua buah bilangan.

Kedua contoh di atas (5.15 dan 5.16) merupakan contoh struktur percabangan sederhana yang melibatkan hanya satu percabangan. Pada masalah-masalah yang lebih rumit, kita akan menjumpai lebih banyak percabangan. Kita juga akan menjumpai suatu struktur percabangan berada di

dalam struktur percabangan yang lain, atau yang biasa disebut nested (bersarang). Perhatikan contoh-contoh berikut.

Contoh 5.17. Struktur percabangan bersarang untuk masalah fotokopi.

Sebuah usaha fotokopi mempunyai aturan sebagai berikut :

- jika yang fotokopi statusnya adalah langganan, maka berapa lembar pun dia fotokopi, harga perlembarnya Rp. 75,-
- jika yang fotokopi bukan langganan, maka jika dia fotokopi kurang dari 100 lembar harga perlembarnya Rp. 100,-. Sedangkan jika lebih atau sama dengan 100 lembar maka harga perlembarnya Rp. 85,-.

Buat *flowchart* untuk menghitung total harga yang harus dibayar jika seseorang memfotokopi sejumlah X lembar.

Penyelesaian:

Pada contoh ini, masalah terlihat lebih rumit. Ada dua percabangan yang terjadi. Yang pertama adalah pemeriksaan apakah status seseorang pelanggan atau bukan. Kedua, apabila status seseorang bukan pelanggan, maka dilakukan pemeriksaan berapa jumlah lembar fotokopi, apakah lebih dari 100 lembar atau tidak.

Pada soal ini kita juga menjumpai apa yang disebut sebagai nested. Perhatikan pernyataan pada syarat kedua dari persoalan di atas.

> ***jika*** *yang fotokopi bukan langganan, maka* ***jika*** *dia fotokopi kurang dari 100 lembar harga perlembarnya Rp. 100*

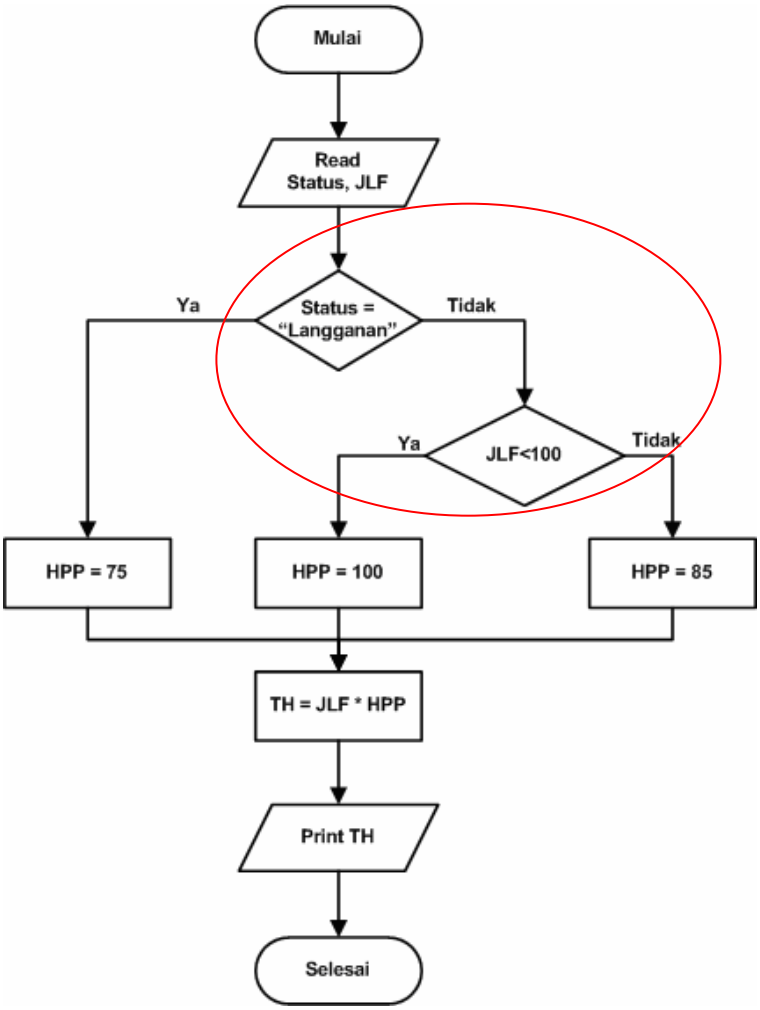
pernyataan jika yang kedua berada di dalam jika yang pertama.

Input yang dibutuhkan untuk permasalahan ini adalah status orang yang fotokopi dan jumlah lembar yang difotokopi. Sehingga variable input yang digunakan adalah:

- **Status** untuk status orang yang fotokopi
- **JLF** untuk jumlah lembar yang difotokopi

Selain itu terdapat variable dengan nama HPP yang digunakan untuk menyimpan harga per lembar dan TH untuk menyimpan nilai total harga. Perhatikan, variable Status bertipe data char, sehingga penulisannya harus menggunakan tanda " ".

Flowchart penyelesaian masalah ini dapat dilihat pada Gambar 5.11.



Gambar 5.11. *Flowchart* penyelesaian untuk masalah fotokopi.

Contoh 5.18. Struktur percabangan bersarang untuk masalah kelulusan siswa.

Aturan kelulusan siswa pada mata pelajaran Pemrograman Web diterapkan sebagai berikut :

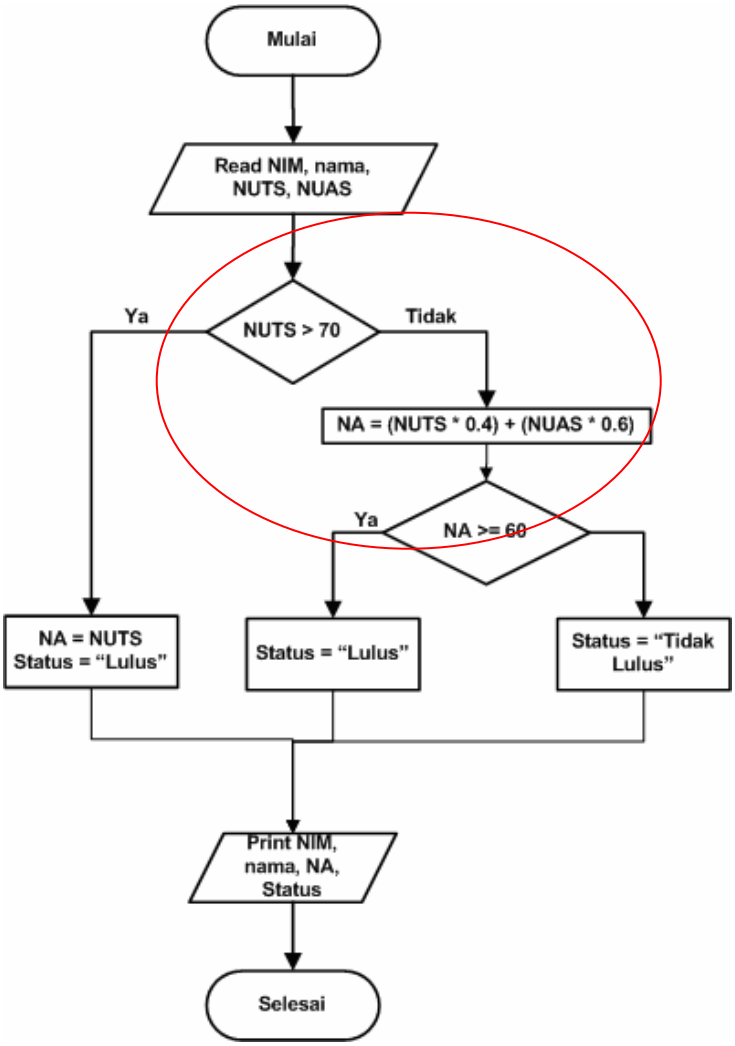
- Jika nilai ujian tengah semester (UTS) lebih besar dari 70 maka siswa dinyatakan lulus dan Nilai Akhir sama dengan nilai UTS.
- Jika nilai UTS kurang atau sama dengan 70 maka siswa dinyatakan lulus jika Nilai Akhir lebih besar atau sama dengan 60 dimana Nilai Akhir = (nilai UTS x 40%) + (nilai UAS x 60%).

Buatlah *flowchart* penyelesaian masalah tersebut apabila output yang diinginkan adalah NIM, Nama Siswa, Nilai Akhir dan Status Kelulusan.

Penyelesaian:

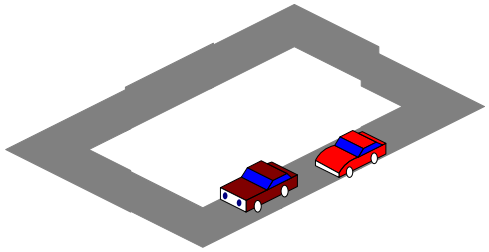
Pada contoh ini, ada dua percabangan. Yang pertama adalah pemeriksaan apakah nilai UTS siswa lebih dari 70. Kedua, apabila nilai UTS tidak lebih dari 70, maka dilakukan pemeriksaan apakah nilai akhir lebih dari 60.

Input yang dibutuhkan untuk permasalahan ini adalah NIM, nama siswa, nilai UTS, dan nilai UAS. Sehingga variable input yang digunakan adalah: NIM untuk Nomor induk siswa, nama untuk nama siswa, NUTS untuk nilai ujian tengah semester, dan NUAS untuk nilai ujian akhir semester. Sedangkan variabel ouput terdiri dari NA yang digunakan untuk menyimpan nilai akhir dan Status untuk menyimpan status kelulusan.



Gambar 5.12. *Flowchart* penyelesaian untuk kelulusan siswa.

5.2.5. Struktur Algoritma Pengulangan



Gambar 5.13. Lomba balap mobil di sirkuit.

Dalam banyak kasus seringkali kita dihadapkan pada sejumlah pekerjaan yang harus diulang berkali. Salah satu contoh yang gampang kita jumpai adalah balapan mobil seperti tampak pada gambar 5.13. Mobil-mobil peserta harus mengelilingi lintasan sirkuit berkali-kali sesuai yang ditetapkan dalam aturan lomba. Siapa yang mencapai garis akhir paling cepat, dialah yang menang.

Pada pembuatan program komputer, kita juga kadang-kadang harus mengulang satu atau sekelompok perintah berkali-kali agar memperoleh hasil yang diinginkan. Dengan menggunakan komputer, eksekusi pengulangan mudah dilakukan. Hal ini karena salah satu kelebihan komputer dibandingkan dengan manusia adalah kemampuannya untuk mengerjakan tugas atau suatu instruksi berulangkali tanpa merasa lelah, bosan, atau malas. Bandingkan dengan pengendara mobil balap, suatu ketika pasti dia merasa lelah dan bosan untuk berputar-putar mengendarai mobil balapnya.

Struktur pengulangan terdiri dari dua bagian :

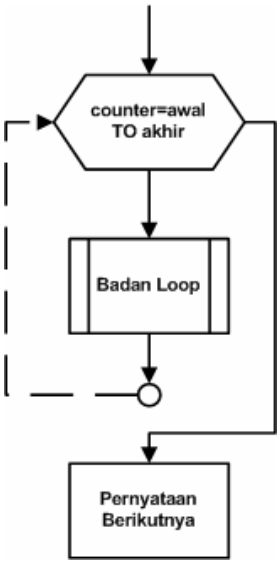
1. **Kondisi pengulangan**, yaitu syarat yang harus dipenuhi untuk melaksanakan pengulangan. Syarat ini biasanya dinyatakan dalam ekspresi Boolean yang harus diuji apakah bernilai benar (true) atau salah (false)
2. **Badan pengulangan (loop body)**, yaitu satu atau lebih instruksi yang akan diulang

Pada struktur pengulangan, biasanya juga disertai bagian inisialisasi dan bagian terminasi. **Inisialisasi** adalah instruksi yang dilakukan sebelum pengulangan dilakukan pertama kali. Bagian insialisasi umumnya digunakan untuk memberi nilai awal sebuah variable. Sedangkan **terminasi** adalah instruksi yang dilakukan setelah pengulangan selesai dilaksanakan.

Ada beberapa bentuk pengulangan yang dapat digunakan, masing-masing dengan syarat dan karakteristik tersendiri. Beberapa bentuk dapat dipakai untuk kasus yang sama, namun ada bentuk yang hanya cocok untuk kasus tertentu saja. Pemilihan bentuk pengulangan untuk masalah tertentu dapat mempengaruhi kebenaran algoritma. Pemilihan bentuk pengulangan yang tepat bergantung pada masalah yang akan diprogram.

- **Struktur pengulangan dengan For**

Pengulangan dengan menggunakan *For*, merupakan salah teknik pengulangan yang paling tua dalam bahasa pemrograman. Hampir semua bahasa pemrograman menyediakan metode ini, meskipun sintaksnya mungkin berbeda. Pada struktur *For* kita harus tahu terlebih dahulu seberapa banyak badan loop akan diulang. Struktur ini menggunakan sebuah variable yang biasa disebut sebagai *loop's counter*, yang nilainya akan naik atau turun selama proses pengulangan. *Flowchart* umum untuk struktur *For* tampak pada Gambar 5.14. Perhatikan penggunaan simbol *preparation* pada *flowchart* tersebut.



Gambar 5.14. Struktur algoritma pengulangan dengan For.

Dalam mengeksekusi sebuah pengulangan dengan For, urutan langkah-langkah adalah sebagai berikut :

1. Menetapkan nilai *counter* sama dengan awal.
2. Memeriksa apakah nilai *counter* lebih besar daripada nilai akhir. Jika benar maka keluar dari proses pengulangan. Apabila kenaikan bernilai negatif, maka proses akan memeriksa apakah nilai *counter* lebih kecil daripada nilai akhir. Jika benar maka keluar dari proses pengulangan.
3. Mengeksekusi pernyataan yang ada di badan loop
4. Menaikkan/menurunkan nilai *counter* sesuai dengan jumlah yang ditentukan pada argument *increment*. Apabila argument *increment* tidak ditetapkan maka secara default nilai *counter* akan dinaikkan 1.
5. Ulang kembali mulai langkah no 2.

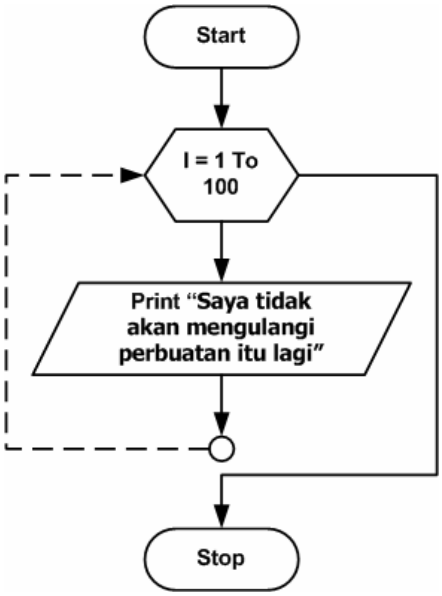
Satu hal yang penting yang harus kita perhatikan adalah nilai *counter* selalu ditetapkan diawal dari pengulangan. Apabila kita mencoba merubah nilai akhir pada badan *loop*, maka tidak akan berdampak pada berapa banyak pengulangan akan dilakukan.

Contoh 5.19. Algoritma untuk mencetak pernyataan sebanyak 100 kali

Mungkin kalian pernah ketika masih di sekolah dasar melakukan perbuatan nakal yang membuat kalian disuruh menuliskan pernyataan tertentu sebanyak 100 kali sebagai hukuman atas kenakalan tersebut. Misalkan pernyataan yang harus ditulis adalah "Saya tidak akan mengulangi perbuatan itu lagi". Bagaimanakah caranya algoritma untuk kasus ini?

Penyelesaian:

Pada contoh ini, kita memerlukan variabel counter, misalkan kita beri nama I. Nilai awalnya adalah 1 dan nilai akhirnya adalah 100. Sedangkan increment atau kenaikan tiap kali pengulangan dari I adalah satu. Perintah untuk mencetak pernyataan akan diulang satu persatu sampai nilai akhir dari *counter* terpenuhi (100). *Flowchart* penyelesaian untuk contoh ini dapat dilihat pada Gambar 5.15.



Gambar 5.15. *Flowchart* menulis pernyataan 100 kali.

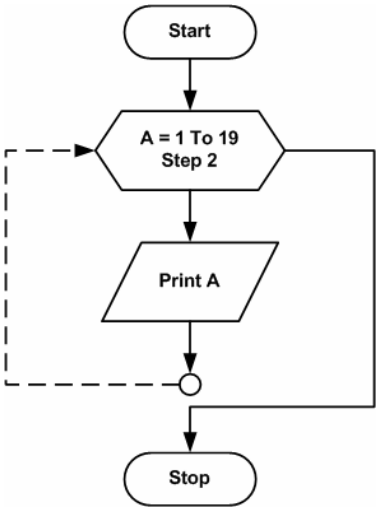
Perhatikan bagaimana mudahnya kita melakukan pengulangan. Pada Gambar 5.15 tersebut *increment* tidak dicantumkan, karena sesuai langkah-langkah yang dijelaskan sebelumnya, jika tidak dicantumkan maka otomatis nilai *increment* adalah satu.

Contoh 5.20. *Flowchart* untuk mencetak anggota suatu himpunan.

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

Penyelesaian:

Pada contoh ini, kita memerlukan variabel *counter*, misalkan kita beri nama A (sesuai dengan nama himpunan). Nilai awalnya adalah 1 dan nilai akhirnya adalah 19. Dari pola himpunan kita tahu bahwa kenaikan bilangan adalah 2 (1 ke 3, 3 ke 5, dan seterusnya). Sehingga bisa kita nyatakan *increment* atau kenaikan tiap kali pengulangan dari A adalah 2. *Flowchart* penyelesaian untuk contoh ini dapat dilihat pada Gambar 5.16.



Gambar 5.16. *Flowchart* mencetak anggota himpunan.

Pada Gambar 5.16 tersebut, perhatikan pada simbol *preparation*. Terdapat tambahan pernyataan **step 2**. Inilah yang disebut sebagai *increment*. Setiap kali pengulangan, maka nilai *counter* yaitu A akan bertambah 2 sehingga yang akan tercetak adalah 1, 3, 5, .., 19.

Contoh 5.21. Menentukan hasil dari suatu *flowchart* pengulangan.

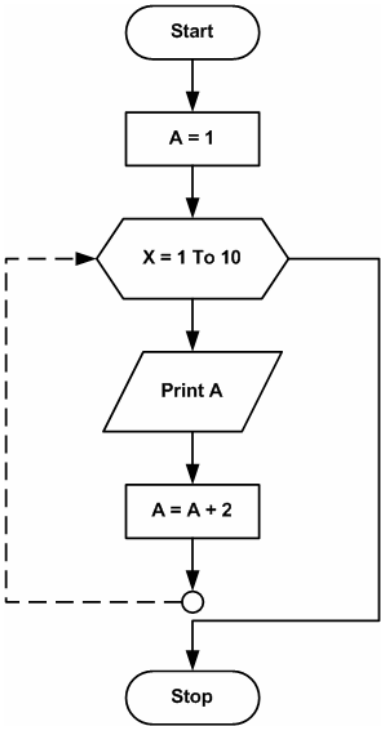
Perhatikan *flowchart* pada Gambar 5.17. Tentukan hasil dari *flowchart* tersebut.

Penyelesaian:

Pada contoh ini, kita mencoba menentukan hasil dari sebuah *flowchart*. Bagaimana menurut kalian jawabannya? Marilah kita uraikan jalannya fowchart tersebut.

Pada *flowchart*, setelah Start, kita meletakkan satu proses yang berisi pernyataan A = 1. Bagian inilah yang disebut **inisialisasi**. Kita memberi nilai awal untuk A = 1. Variabel *counter*-nya adalah X dengan nilai awal 1 dan nilai akhir 10, tanpa *increment* (atau secara *default increment*-nya

adalah 1). Ketika masuk ke badan *loop* untuk pertama kali maka akan dicetak langsung nilai variabel A. Nilai variabel A masih sama dengan 1. Kemudian proses berikutnya adalah pernyataan A = A + 2. Pernyataan ini mungkin agak aneh, tapi ini adalah sesuatu yang pemrograman. Arti dari pernyataan ini adalah gantilah nilai A yang lama dengan hasil penjumlah nilai A lama ditambah 2. Sehingga A akan bernilai 3. Kemudian dilakukan pengulangan yang ke-dua. Pada kondisi ini nilai A adalah 3, sehingga yang tercetak oleh perintah print adalah 3. Baru kemudian nilai A kita ganti dengan penjumlahan A + 2. Nilai A baru adalah 5. Demikian seterusnya. Sehingga output dari *flowchart* ini adalah 1, 3, 5, 7, .., 19.



Gambar 5.17. *Flowchart* mencetak bilangan tertentu.

Kita dapat melihat sekarang bahwa Gambar 5.16 dan 5.17 memberikan output yang sama. Dari kedua *flowchart* tersebut, Gambar 5.17 merupakan *flowchart* yang disarankan. Meskipun lebih panjang tetapi lebih terstruktur. Selain itu tidak semua bahasa pemrograman memberi fasilitas pengaturan *increment.*

Flowchart pada Gambar 5.17 harus kita perhatikan benar-benar, terutama posisi pernyataan Print A. Cobalah membalik posisinya sehingga letak pernyataan A = A + 2 berada di atas pernyataan Print A. Bagaimanakah hasilnya?

Seperti halnya struktur percabangan, kita juga akan menjumpai bentuk struktur pengulangan bersarang (*nested*). Artinya, ada suatu pengulangan yang berada di dalam pengulangan yang lain. Perhatikan Contoh 5.22 berikut ini.

Contoh 5.21. Menentukan hasil dari suatu *flowchart* pengulangan.

Perhatikan *flowchart* pada Gambar 5.18. Tentukan hasil dari *flowchart* tersebut.

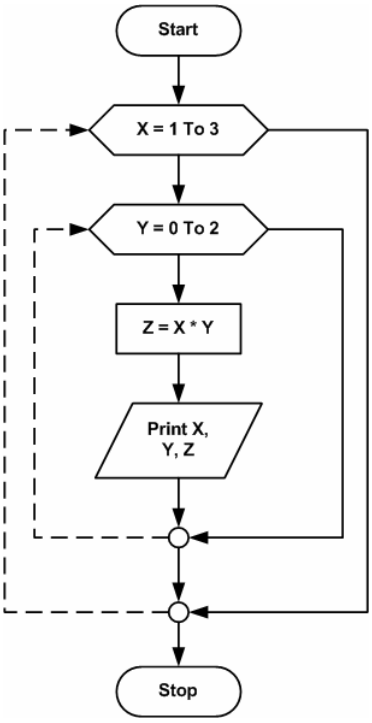
Penyelesaian:

Pada contoh ini, kita mencoba menentukan hasil dari sebuah *flowchart* dengan pengulangan bersarang. Bagaimana menurut kalian jawabannya? Marilah kita uraikan jalannya *flowchart* tersebut.

Pada Gambar 5.18 tersebut, terlihat ada dua simbol *preparation*. Yang pertama dengan variabel *counter* X dan yang kedua dengan variabel *counter* Y. Dalam posisi, variabel *counter* Y terletak setelah variabel *counter* X. Hal ini berarti pengulangan dengan variabel *counter* Y terletak di dalam variabel *counter* X. Inilah yang disebut sebagai pengulangan bersarang.

Pada bentuk pengulangan seperti ini, alur eksekusi program akan berjalan sebagai berikut:

- Variabel X akan diisi dengan nilai awal *counter*-nya yaitu 1.
- Variabel Y akan diisi dengan nilai awal *counter*-nya yaitu 0
- Nilai Z dihitung dengan mengalikan X dengan Y. Nilai X = 1 dan nilai Y = 0 jadi nilai Z = 0
- Nilai X, Y dan Z dicetak di layar.
- Alur berputar, dan nilai Y dinaikkan menjadi 1 sedangkan nilai X masih satu (karena bagian X belum berputar).
- Nilai Z = 1 hasil perkalian X = 1 dan Y = 1.
- Nilai X, Y, dan Z akan dicetak lagi di layar.
- Alur berputar kembali sehingga nilai Y menjadi 2, sehingga nilai Z akan menjadi 2.



Gambar 5.18. *Flowchart* dengan pengulangan bersarang.

- Setelah ini perputaran akan keluar dari Y karena nilai akhir *counter* yaitu sudah tercapai.
- Nilai X akan dinaikkan 1 menjadi 2, dan proses kembali melakukan pengulangan pada bagian Y seperti di atas.
- Proses pengulangan diulang terus sampai nilai akhir dari *counter* X yaitu 3 tercapai. Sehingga hasil akhir dari *flowchart* tersebut adalah sebagai berikut:

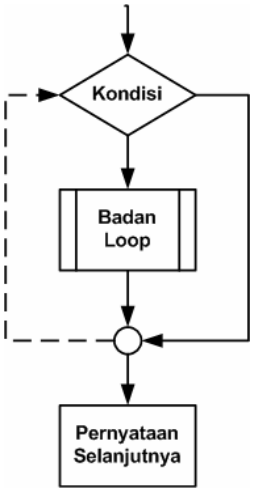
```
X       Y       Z
1       0       0
1       1       1
1       2       2
2       0       0
2       1       2
2       2       4
3       0       0
3       1       3
3       2       6
```

Dari Contoh 5.21. di atas kita bisa melihat ada aturan-aturan yang harus dipenuhi dalam pengulangan bersarang, yaitu:

- Masing-masing pengulangan (badan loop) mempunyai variabel *counter* sendiri-sendiri.
- Pengulangan-pengulangan tersebut tidak boleh tumpang tindih.

- **Struktur pengulangan dengan While**

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



Gambar 5.19. *Flowchart* umum *While*.

Struktur *While* akan mengulang pernyataan pada badan loop sepanjang kodisi pada *While* bernilai benar. Dalam artian kita tidak perlu tahu pasti berapa kali diulang. Yang penting sepanjang kondisi pada *While* dipenuhi maka pernyataan pada badan loop akan diulang. *Flowchart* umum untuk struktur *While* dapat dilihat pada Gambar 5.19.

Pada Gambar 5.19., tampak bahwa simbol preparasi untuk pengulangan seperti pada For tidak digunakan lagi. Namun kita menggunakan simbol decision untuk mengendalikan pengulangan. Selain kondisi, biasanya pada

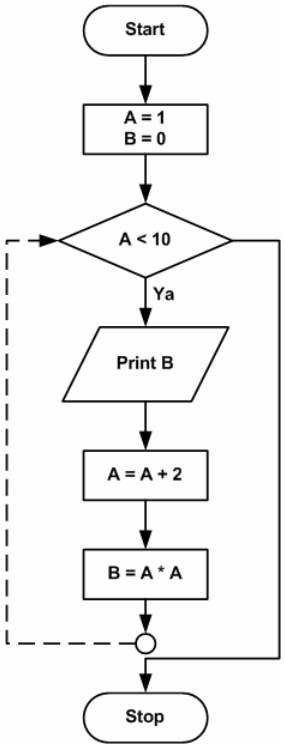
pengulangan *While* harus dilakukan inisialisasi variabel terlebih dahulu.

Contoh 5.22. Pengulangan dengan *While* untuk mencetak nilai tertentu.

Perhatikan *flowchart* pada Gambar 5.20. Bagaimanakah *output* dari *flowchart* tersebut?

Penyelesaian:

Perhatikan Gambar 5.20. bisakah kalian menentukan hasil dari *flowchart* tersebut? Perhatikan tahapan eksekusi *flowchart* berikut ini.



Gambar 5.20. *Flowchart* pengulangan dengan while untuk mencetak nilai tertentu.

1. Pada *flowchart* ini ada dua variabel yang kita gunakan yaitu A dan B. Kedua variabel tersebut kita inisialisasi nilai awalnya (A = 1 dan B = 0) sebelum proses loop terjadi. Variabel A adalah variabel *counter*.
2. Pada simbol decision, nilai A akan diperiksa apakah memenuhi kondisi (< 10). Jika Ya maka perintah berikutnya dieksekusi, jika tidak maka program akan berhenti. Pada awal eksekusi ini kondisi akan terpenuhi karena nilai A = 1.
3. Jalankan perintah Print B.
4. Nilai variabel A kemudian diganti dengan nilai A lama (1) ditambah 2. Sehingga nilai variabel A baru adalah 3. Sedangkan nilai variabel B = 9 (hasil perkalian A = 3).
5. Program akan berputar kembali untuk memeriksa apakah nilai variabel A masih lebih kecil dari 10. Pada kondisi ini nilai A = 3, sehingga kondisi masih terpenuhi. Kemudian langkah berulang ke langkah ke 3. Begitu seterusnya sampai nilai variabel A tidak lagi memenuhi syarat kurang dari 10. Sehingga output dari *flowchart* ini adalah : 0, 9, 25, 49, 81.

Seperti halnya pengulangan dengan *For*, pengulangan dengan *While* juga memungkinkan terjadinya pengulangan bersarang. Aturan dan cara yang dilakukan sama dengan pengulangan dengan *For*.

Pada beberapa bahasa pemrograman juga disediakan pengulangan dengan cara *Do ... Loop* dan *Repeat .. Until*. Kedua cara ini mirip dengan *While*, perbedaannya adalah letak dari kondisi. Pada *While* pemeriksaan kondisi diletakkan sebelum badan *loop*. Sedangkan *Do ... Loop* dan *Repeat ... Until*, pemeriksaan kondisi dilakukan setelah badan loop.

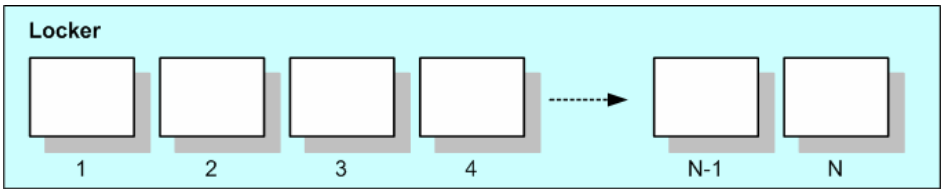
5.3. PENGELOLAAN ARRAY

Variabel array telah kita singgung di bagian depan, namun masih sangat terbatas. Pada bagian ini kita akan pelajari lebih detil tentang array.

5.3.1. Pengertian Array

Variabel-variabel yang kita gunakan selama ini adalah variable biasa yang memiliki sifat bahwa sebuah nama variable hanya dapat menyatakan sebuah nilai *numeric* atau *string* pada suatu saat. Apabila kita ingin memberi nilai yang baru pada variable tersebut maka nilai lama akan hilang tergantikan oleh nilai yang baru. Bagaimana apabila kita ingin menyimpan beberapa nilai/data dalam sebuah variable dengan nama yang sama, tetapi semua nilai tetap tersimpan? Solusi yang dapat dilakukan adalah dengan menggunakan indeks pada nama variable tersebut. Cara ini biasa disebut dengan array.

Array adalah struktur data yang menyimpan sekumpulan elemen yang bertipe sama, setiap elemen diakses langsung melalui indeksnya. Indeks array haruslah tipe data yang menyatakan keterurutan, misalnya integer atau string. Array dapat dianalogikan sebagai sebuah lemari atau locker yang memiliki sederetan kotak penyimpanan yang diberi nomor berurutan (lihat Gambar 5.21). Untuk menyimpan atau mengambil sesuatu dari kotak tertentu kita hanya cukup mengetahui nomor kotaknya saja.



Gambar 5.21. Lemari dengan banyak kotak laci di dalamnya

Pada variabel array, kita tidak hanya menentukan tipe datanya saja, tetapi juga jumlah elemen dari array tersebut atau dalam hal ini adalah batas atas indeksnya. Pada banyak bahasa pemrograman seperti C++, Visual Basic, dan beberapa yang lainnya, nilai indeks awal adalah 0 bukan 1. Cara menuliskan variabel array berbeda-beda tergantung bahasa pemrograman apa yang dipakai. Tetapi yang pasti tipe data harus disebutkan dan batas atas indeks harus

ditentukan. Untuk mengisi data pada array kita dapat langsung menentukan pada indeks berapa kita akan isikan demikian juga untuk memanggil atau menampilkan data dari array.

Contoh deklarasi, pengisian dan pemanggilan array adalah sebagai berikut.

Contoh 5.23. Penulisan array pada C++ dan Visual Basic.

Array pada C++

```
#include <iostream>
using namespace std;

int main() {

// Mendeklarasikan array A dengan 3 buah elemen bertipe int
  int A[3];

// Mengisikan nilai elemen array
A[0] = 5;
A[1] = 10;
A[2] = 20;

  // Menampilkan nilai elemen array
cout<<"Nilai elemen ke-1 = "<<A[0];
cout<<"Nilai elemen ke-2 = "<<A[1];
cout<<"Nilai elemen ke-3 = "<<A[2];

  return 0;
}
```

Array pada Visual Basic

```
'Mendeklarasikan array A dengan 3 buah elemen bertipe integer

Dim A (2) as Integer

'Mengisikan nilai elemen array
A(0) = 5;
A(1) = 10;
A(2) = 20;

'Menampilkan nilai elemen array
Print A(0);
Print A(1);
Print A(2);
```

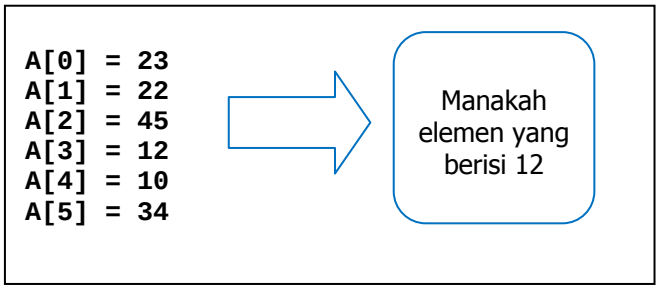
Perhatikan pada kedua kode di atas. Pada pendeklarasian variabel array nilai maksimal indeks adalah 2 tetapi jumlah elemennya ada 3 karena indeks dimulai dari 0 bukan dari 1.

5.3.2. Pencarian Data dalam Array

Salah satu permasalahan yang sering dijumpai dalam array adalah bagaimana mencari elemen tertentu dari array. Misalnya pada kasus loker pada Gambar 5.21 di atas tersedia 100 kotak. Kemudian kita diminta untuk mencari nomor kotak keberapa yang dimiliki oleh seorang siswa bernama "Rudi". Contoh yang lain, misalkan ada banyak siswa dalam satu sekolah dan kita diminta

mencari data seorang siswa dengan nama tertentu atau alamat tertentu. Perhatikan contoh berikut.

Contoh 5.24. Pencarian pada array.



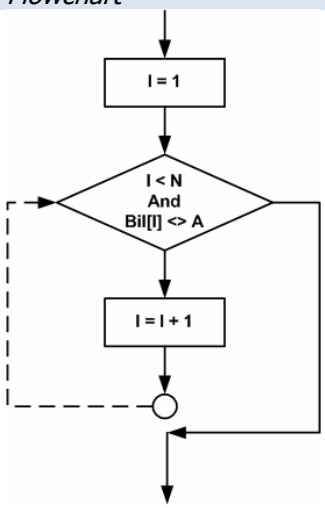
Pada contoh ini kita diminta mencari elemen yang berisi angka 12 dari sekumpulan elemen dalam array. Ada 6 elemen pada array tersebut. Menurut kalian bagaimanakah algoritma penyelesaiannya?

Cara yang paling umum dan paling mudah dilakukan adalah dengan cara pencarian berurutan (*linear search*). Pada masa lalu cara ini dianggap tidak efisien karena membutuhkan waktu lama. Namun dengan perkembangan komputer yang sangat cepat, waktu eksekusi algoritma ini tidak terlalu dipermasalahkan. Cara ini dilakukan dengan cara membandingkan isi dari elemen dengan apa yang kita cari. Satu per satu dimulai dari elemen yang paling awal.

Apabila kita terapkan pada Contoh 5.24, maka eksekusi program akan berlangsung berurutan sebagai berikut:

- Tetapkan bilangan yang ingin kita cari (yaitu 12)
- Ambil elemen paling awal (yaitu A[0]), bandingkan isi elemen tersebut (yaitu 23) dengan bilangan yang kita cari. Jika sama maka stop.
- Jika tidak maka lanjutkan dengan elemen berikutnya (yaitu A[1]), bandingkan isi elemen tersebut dengan bilangan yang kita cari. Jika sama maka stop.
- Jika tidak maka lanjutkan dengan elemen berikutnya. Dan seterusnya sampai dijumpai elemen yang berisi sama dengan bilangan yang kita cari.

Contoh 5.24 akan memberikan hasil elemen A[3] yang memiliki isi 12. Apabila digambarkan dalam bentuk *flowchart* maka akan tampak seperti pada Gambar 5.22.

Flowchart	Keterangan
	- I adalah variabel counter sekaligus indeks - N adalah batas atas dari indeks - Bil[I] adalah nama variabel array yang berisi bilangan - A adalah bilangan yang kita cari

Gambar 5.22. *Flowchart* untuk pencarian bilangan.

Pada *flowchart* di Gambar 5.22, kita menggunakan pengulangan model *While*. Kondisi yang harus dipenuhi disini ada dua, yaitu I < N dan Bil[I] <> A. Arti dari kondisi ini adalah jika nilai indeks I kurang dari batas atas indeks dan isi dari Bil[I] tidak sama dengan bilangan yang kita cari, maka pencarian akan diteruskan pada indeks yang lebih tinggi. Selama kondisi ini dipenuhi maka pencarian akan terus dilakukan. Perhatikan bahwa di sini kita menggunakan "dan" yang artinya kedua kondisi harus dipenuhi agar dianggap benar. Pencarian akan hanya akan berhenti jika salah satu kondisi atau kedua kondisi tidak dipenuhi lagi. Sehingga misalnya Bil[I] mempunyai isi yang sama dengan A maka pencarian akan dihentikan karena kondisi pada *While* sudah tidak dipenuhi lagi.

5.3.3. Pengurutan Data pada Array

Permasalahan lain dalam array yang juga banyak digunakan adalah bagaimana mengurutkan elemen-elemen dari variabel array tersebut. Perhatikan kembali Contoh 5.24. Pada contoh tersebut terlihat bahwa isi elemen-elemen dari array tidak dalam posisi berurutan. Bagaimanakah caranya agar isi elemen-elemen tersebut terurut dari besar ke kecil atau sebaliknya?

Ada beberapa algoritma yang dapat digunakan untuk mengurutkan sekumpulan bilangan, antara lain *bubble sort, selection sort, shell sort, quick sort*, dan lain-lain. Pada buku ini kita akan membahas satu algoritma yaitu *bubble sort*. Meskipun kinerjanya tidak sebaik algoritma yang lain, algoritma ini mudah dimengerti dan banyak digunakan. Perhatikan contoh berikut.

Contoh 5.24. Pengurutan dengan *bubble sort*.

Misalkan sebuah variabel array dengan nama Bil yang terdiri dari 5 elemen yang masing-masing berisi bilangan "5 1 4 2 8". Urutkan dari mulai nilai terkecil sampai ke yang paling besar.

Penyelesaian:

Kita akan menggunakan metode bubble sort untuk mengurutkan array ini. Bubble sort dilakukan dengan cara membandingkan dua bilangan yang berurutan letaknya. Jika urutan letaknya benar maka dilanjutkan dengan membandingkan dua bilangan berikutnya. Jika tidak maka tukar letak dari dua bilangan tersebut.

Marilah kita terapkan algoritma ini. Perhatikan tabel array berikut. Kondisi awal adalah pada posisi J = 0. Pertama kita bandingkan antara Bil[0] dengan Bil[1]. Bil[0] = 5 sedangkan Bil[1] = 1. Berdasarkan aturan bubble sort, isi dari Bil[0] tidak sesuai letaknya karena lebih besar dari isi Bil[1]. Sehingga kita perlu menukar isi dari dua elemen array ini. Sehingga Bil[0] = 1 dan Bil[1] = 5 (perhatikan baris pada J = 1). Langkah berikutnya kita membandingkan Bil[1] dengan Bil[2]. Bil[1] = 5 dan Bil[2] = 4, sehingga kembali kita harus menukar isi dari elemen ini (perhatikan baris J = 2). Hal ini terus dilakukan sampai pada perbandingan Bil[3] dengan Bil[4].

J	Bil[0]	Bil[1]	Bil[2]	Bil[3]	Bil[4]
0	5	1	4	2	8
1	1	5	4	2	8
2	1	4	5	2	8
3	1	4	2	5	8
4	1	4	2	5	8

Pada posisi akhir dari tabel di atas, kita lihat bahwa bilangan belum terurut sepenuhnya. Karena kita baru menggunakan satu kali putaran dengan Bil[0] sebagai patokan. Kita akan lakukan perbandingan lagi, namun dengan Bil[1] sebagai patokan. Hal ini karena Bil[0] pasti sudah berisi bilangan paling kecil. Sehingga tabel baru kita buat seperti berikut.

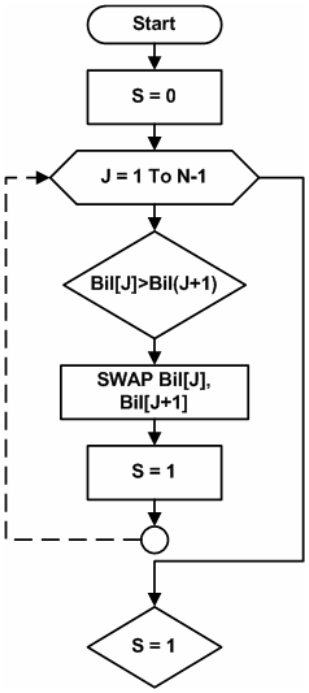
J	Bil[0]	Bil[1]	Bil[2]	Bil[3]	Bil[4]
1	1	4	2	5	8
2	1	2	4	5	8
3	1	2	4	5	8
4	1	2	4	5	8

Pada posisi tabel di atas, sebenarnya urutan bilangan sudah benar, tapi algoritma belum berhenti karena algoritma belum memeriksa putaran yang berikutnya. Sehingga diperlukan dua putaran lagi dengan dengan dasar masing-masing pembanding adalah Bil[2] dan Bil[3]. Kedua tabel tersebut adalah sebagai berikut.

J	Bil[0]	Bil[1]	Bil[2]	Bil[3]	Bil[4]
2	1	2	4	5	8
3	1	2	4	5	8
4	1	2	4	5	8

J	Bil[0]	Bil[1]	Bil[2]	Bil[3]	Bil[4]
3	1	2	4	5	8
4	1	2	4	5	8

Kalau digambarkan dalam bentuk *flowchart* akan tampak seperti pada Gambar 5.23.



Gambar 5.23. *Flowchart* untuk pengurutan bilangan.

5.4. OPERASI FILE

File seringkali digunakan untuk menyimpan data agar data tidak hilang. Data atau yang ada dan dihasilkan pada program akan hilang ketika program diakhiri, sehingga file digunakan untuk menyimpan data tersebut. Ada dua jenis file yaitu file program dan file data. File program berisi kode-kode program sedangkan file data hanya berisi data. File data terdiri dari dua jenis yaitu file data berurutan (*sequential data file*) dan file data acak (*random-access data file*). Perbedaan utama dari kedua jenis file data ini adalah dapat dilihat pada tabel berikut.

File data berurutan	File data acak
- *Record* atau baris data harus dibaca berurutan mulai dari yang pertama - Panjang *field* untuk setiap *record* tidak perlu sama - Pengubahan serta penambahan *record* tertentu sukar dilakukan	- *Record* tidak perlu dibaca berurutan - Panjang *field* untuk setiap *record* harus sama - Pengubahan serta penambahan *record* lebih mudah dilakukan

5.4.1. Algoritma Penulisan Data pada File

Algoritma yang digunakan untuk penulisan data untuk file data berurutan maupun acak secara prinsip sama, hanya modusnya yang berbeda. Berikut ini adalah algoritma penulisan data dalam SE.

```
Open "modus", <buffer number>, "nama file data"
Write <record number>, field 1, field 2, .. field n
Close buffer number
```

Modus O menunjukkan file ini dibuka untuk ditulisi.

Contoh 5.25. Contoh penerapan algoritma penulisan data.

Misalkan kita punya file data dengan nama "siswa.dat" yang *field*-nya adalah nama siswa, alamat, nomor telepon. Maka untuk menuliskan data adalah sebagai berikut.

```
Open "O", #1, "siswa.dat"
Write #1, <nama>, <alamat>, <no.telepon>
Close #1
```

Notasi #1 menunjukkan siswa.dat akan ditempatkan dalam *buffer* no 1. Notasi ini harus sama digunakan di seluruh progam di atas. Artinya kalau kita menempatkan suatu file dengan nomor *buffer* #1 maka ketika membuka, menulis, membaca dan menutup harus menggunakan notasi tersebut. Demikian juga bila kita menempatkan pada *buffer* no #2.

5.4.2. Algoritma Pembacaan Data pada File

Algoritma membaca data algoritmanya hampir sama dengan menuliskan data, tetapi modus yang digunakan tidak O tetapi I. I adalah input yang berarti file data dibuka untuk dibaca datanya sebagai input. Berikut ini algoritmanya dalam SE.

```
Open "modus", <buffer number>, "nama file data"
While not EOF:
       Input <record number>, field 1, field 2, ..
      field n
       Print field 1, field 2, .. field n
End while
Close buffer number
```

Pernyataan *While Not EOF* digunakan untuk memeriksa apakah sudah ada pada baris terakhir dari data. Jika belum maka baris-baris data akan dibaca dan dicetak sampai baris terakhir. Pernyataan input digunakan untuk mengambil data dari file untuk dimuat ke dalam program. Sedangkan pernyataan *print* digunakan untuk mencetak data ke layar komputer.

Contoh 5.26. Contoh penerapan algoritma penulisan data.

File data dengan nama "siswa.dat" seperti pada contoh 5.25 yang *field*-nya adalah nama siswa, alamat, nomor telepon. Maka untuk membaca data adalah sebagai berikut.

```
Open "I", #2, "siswa.dat"
While not EOF:
       Input #2, <nama>, <alamat>, <no.telepon>
       Print <nama>, <alamat>, <no.telepon>
End while
Close buffer number
```

5.5. Ringkasan

- Variabel adalah tempat dimana kita dapat mengisi atau mengosongkan nilainya dan memanggil kembali apabila dibutuhkan. Setiap variabel akan mempunyai nama (*identifier*) dan nilai.
- Konstanta adalah variabel yang nilai datanya bersifat tetap dan tidak bisa diubah.
- Tipe data adalah jenis data yang dapat diolah oleh komputer untuk memenuhi kebutuhan dalam pemrograman komputer.
- Tipe data dapat dikelompokkan menjadi tipe data primitive dan tipe data composite. Tipe data primitive terdiri dari *numeric, character*, dan *bolean*. Sedangkan tipe data composite terdiri dari *array, record/struct, image, date time, subrange*, enumerasi, obyek dan *variant*.
- Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis. Algoritma harus benar dan harus berhenti. Setelah berhenti, algoritma memberikan hasil yang benar.
- Algoritma dapat ditulis dengan cara *Structured English, Pseudocode* dan *Flowchart*.
- Struktur berurutan terdiri satu atau lebih instruksi. Tiap instruksi dikerjakan secara berurutan sesuai dengan urutan penulisannya.
- Pada struktur percabangan, program akan berpindah urutan pelaksanaan jika suatu kondisi yang disyaratkan dipenuhi.
- Struktur pengulangan terdiri dari dari kondisi pengulangan dan badan pengulangan dan dapat dilakukan dengan *For* dan *While*.

- Array adalah struktur data yang menyimpan sekumpulan elemen yang bertipe sama, setiap elemen diakses langsung melalui indeksnya. Operasi pencarian pada array dapat dilakukan dengan cara linear search sedangkan pengurutan dengan metode *bubble sort*.
- File data ada yang bersifat urut dan ada yang acak. Metode pembacaan dan penulisan dibedakan dari modusnya.

5.6. Soal-Soal Latihan

1. Tentukan salah atau benar pada nama-nama variabel berikut ini. Jika salah cobalah berikan alasan.

   a. nama.guru
   b. NamaGuru
   c. 2x
   d. harga/buku
   e. hargaPerBuku

2. Tentukan tipe data yang cocok untuk hal-hal berikut ini (perhatikan ini bukan nama variabel) dan jelaskan alasannya.

   a. Jumlah murid
   b. Berat badan
   c. Tinggi badan
   d. Nama siswa
   e. Tempat lahir
   f. Tanggal lahir

3. Lihat kembali Contoh 5.3.

   a. Jika pada kode program A semua variabel (x, y, z) dideklarasikan bertipe data int, berapakah nilai z?

   b. Jika pada kode program A semua variabel (x, y, z) dideklarasikan bertipe data float, berapakah nilai z?

   c. Jika pada kode program B variabel x juga dideklarasikan bertipe data float, berapakah nilai z?

4. Ada dua gelas A dan B, gelas A berisi larutan berwarna merah, gelas B berisi larutan berwarna kuning. Pertukarkan isi dari kedua gelas itu sedemikian rupa sehingga gelas A berisi larutan kuning dan gelas B berisi larutan merah. Buatlah algoritma penyelesaiannya dengan menggunakan SI (*Structured Indonesia*).

5. Variabel A = 6, Variabel B = 10. Buatlah *flowchart* untuk menukar nilai variabel A dan B sehingga variabel A = 10 dan variabel B = 6.

6. PT. Sandang Nyaman bermaksud menggunakan komputer untuk menghitung upah mingguan pegawainya. Data yang diperlukan adalah nama pegawai dan jumlah jam kerja selama seminggu. Upah per jam ditetapkan Rp. 4500,-. Buatlah *flowchart* untuk masalah ini jika output

yang diinginkan adalah nama pegawai, jam kerja dan upah yang diterima.

7. Sama seperti soal no 3, tetapi jika jam kerja melebihi 25 jam per minggu maka kelebihannya dianggap lembur. Upah perjam lembur adalah satu setengah kali dari upah per jam pada kondisi biasa. Bagaimanakah *flowchart*nya?

8. Seorang penjual buku pelajaran SD sedang berusaha menarik pembeli buku dengan ketentuan-ketentuan sebagai berikut :

   - Jika jumlah buku yang dibeli lebih kecil atau sama dengan 100 eksemplar, maka pembeli tidak mendapat potongan,
   - Jika jumlah buku yang dibeli lebih besar dari 100 tetapi kurang atau sama dengan 200 eksemplar, maka untuk 100 eksemplar yang pertama mendapat diskon 5%, sedangkan sisanya mendapat diskon 15%,
   - Jika jumlah buku yang dibeli lebih besar dari 200 eksemplar, maka untuk 100 eksemplar yang pertama didiskon 7%, untuk 100 eksemplar yang kedua didiskon 17% dan sisanya didiskon 27%.

   Apabila harga satu eksemplar buku Rp. 5000, buatlah *flowchart* untuk menyelesaikan aturan tersebut apabila output yang diinginkan adalah Jumlah eksemplar buku yang dibeli dan total harga yang harus dibayar. (Tentukan terlebih dahulu, semua variable, konstanta dan tipe data masing-masing).

9. Lihat kembali Contoh 5.16. Buatlah *flowchart* untuk menghasilkan output yang sama seperti Contoh 5.16, namun dengan menggunakan pengulangan *While*.

10. Diketahui 2 buah himpunan, A = {1, 3, 5, ..., 19) dan B = {2, 5, 8, ..., 29}. Buatlah *flowchart* untuk mencari irisan dari himpunan A dan B. (Petunjuk: irisan himpunan terjadi jika nilai anggota himpunan A = nilai anggota himpunan B).

11. Buatlah *flowchart* untuk mencetak tabel perkalian seperti berikut:

```
1 x 1 = 1   2 x 1 = 2   3 x 1 = 3   4 x 1 = 4
1 x 2 = 2   2 x 2 = 4   3 x 2 = 6   4 x 2 = 8
1 x 3 = 3   2 x 3 = 6   3 x 3 = 9   4 x 3 = 12
1 x 4 = 4   2 x 4 = 8   3 x 4 = 12  4 x 4 = 16
1 x 5 = 5   2 x 5 = 10  3 x 5 = 15  4 x 5 = 20
```

12. Urutkan bilangan berikut ini dengan algoritma *bubble sort*:

    a. 5 - 3 - 12 – 2 - 52

    b. 4 - 33 – 4 – 3 -12 -2

    c. 5 - 3 - 12 - 2 – 50 – 52 – 10

Daftar Pustaka

Anonymous. 2004. Guide to the Software Engineering Body of Knowledge (SWEBOK). The Institute of Electrical and Electronics Engineers, Inc.

Balter, A. 2006. Sams Teach Yourself Microsoft® SQL Server™ 2005 Express in 24 Hours. Sams.

Bass, L., P. Clements, and R. Kazman. 2003. Software Architecture in Practice. 2nd Edition. Addison-Wesley.

Cormen, T.H. 2001. Introduction to Algorithm: Second Edition. The MIT Press.

Deek, FP., J.A.M. McHugh, and O.M. Eljabiri. 2005. Strategic software engineering : An Interdisciplinary Approach. Auerbach Publications.

den Haan, P., L. Lavandowska, S.N. Panduranga, and K. Perrumal. 2004. Beginning JSP 2: From Novice to Professional. Apress.

Dobson, R. 1999. Programming Microsoft Access 2000: The Developer's Guide to Harnessing the Power of Access. Microsoft Press.

Felleisen, M, R.B. Findler, M. Flatt, and S. Krishnamurthi. 2001. How to Design Programs; An Introduction to Computing and Programming. The MIT Press.

Kak, A.C. 2003. Programming With Objects: A Comparative Presentation of Object Oriented Programming with C++ and Java. John Wiley & Sons, Inc.

Kaisler, S.H. 2005. Software Paradigm. John Wiley & Sons, Inc.

Kennedy, B. and C. Musciano. 2006. HTML & XHTML: The Definitive Guide, 6th Edition. O'Reilly.

Lafore, R. 1998. Data Structures & Algorithm in Java. Waite Group Press.

Laurie, B and P. Laurie. 2001. Apache: The Definition Guide. 2nd Edition. O'Reilly and Associates, Inc.

Leffingwell, D. and D. Widrig. 2003. Managing Software Requirements: A Use Case Approach. 2nd Edition. Addison-Wesley.

Lischner, R. 2000. Delphi in a Nutshell. O'Reilly and Associates, Inc.

Luckey, T. and J. Phillips. 2006. Software Project Management for Dummies. Wiley Publishing, Inc.

McConnel, S. 2003. Professional Software Development: Shorter Schedules, Higher Quality Products, More Successful Projects, Enhanced Careers. Addison-Wesley.

Meyer, B. 2000. Object Oriented Software Construction. 2nd Edition. ISE, Inc.

Musciano, C. and B. Kennedy. 2002. HTML and XHTML: The Definition Guide. 4th Edition. O'Reilly and Associates, Inc.

Navarro, A. 2001. Effective Web Design. 2nd Edition. SYBEX, Inc.

Powell, G. 2006. Beginning Database Design. Wiley Publishing, Inc.

Riordan, R.M. 2005. Designing Effective Database Systems. Addison Wesley Professional.

Robbins, J. N. 2006. Web Design in a Nutshell, 3rd Edition. O'Reilly.

Suehring, S. 2002. MySQL Bible. Wiley Publishing, Inc.

Taylor, D.A. 1998. Object Technology: A Manager's Guide. Addison-Wesley.

Van Roy, P and S. Haridi. 2004. Concepts, Techniques, and Models of Computer Programming. The MIT Press.

Lampiran 1
Daftar Istilah / Glosari

Abstraction
Merupakan prinsip penyederhanaan dari sesuatu yang kompleks dengan cara memodelkan kelas sesuai dengan masalahnya

Algoritma
Urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis

Array
Struktur data yang menyimpan sekumpulan elemen yang bertipe sama

Atribut
Karakteristik atau ciri yang membedakan antara entitas satu dengan entitas yang lainnya

Authentication
Proses memeriksa keabsahan seseorang sebagai user (pengguna) pada suatu system (misalnya pada DBMS)

Basic Input/Output System (BIOS)
Kode-kode program yang pertama kali dijalankan ketika komputer dinyalakan (booting)

Basis data (database)
Kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan dalam perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya

Command Line Interface (CLI)
Antar muka pengguna dengan model perintah-perintah teks

Compiler
Penerjemah bahasa pemrograman tingkat tinggi ke bahasa mesin dengan cara sekaligus seluruh kode program. Prosesnya disebut kompilasi.

Component Object Model (COM)
Infrastruktur yang disediakan oleh Visual Basic untuk mengakses obyek-obyek atau kontrol-kontrol lain sepanjang punya antar muka yang dapat diakses oleh Visual Basic.

Constraint
Batasan-batasan dari masalah

Control
Aktivitas monitoring dan evaluasi terhadap feedback untuk menentukan apakah system telah bekerja dengan baik atau tidak

Counter
Variable pencacah yang digunakan dalam struktur algoritma pengulangan

Database Management System (DBMS)
Perangkat Lunak yang khusus / spesifik ditujukan untuk pengelolaan basis data

Disk Operating System (DOS)
Salah satu sistem operasi lama berbasis CLI

Elektronika
Ilmu yang mempelajari alat listrik **arus lemah** yang dioperasikan dengan cara mengontrol aliran elektron atau partikel bermuatan listrik dalam suatu alat

Entitas
Individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain

Extensible Hypertext Markup Language (XHTML)
HTML versi terakhir (4.01) yang ditulis ulang dengan dengan aturan-aturan yang lebih ketat mengacu pada XML

Extensible Markup Language (XML)
Sekumpulan aturan untuk menyusun bahasa *markup*

Feedback
Data tentang kinerja sistem

Flowchart
Skema/bagan (*chart*) yang menunjukkan aliran (*flow*) di dalam suatu program secara logika

Gejala
Signal atau tanda terjadinya suatu masalah

Gerbang logika
blok-blok penyusun dari perangkat keras elektronik

Graphical User Interface (GUI)
Antar muka pengguna dengan model grafis

Identifier

Nama dari suatu variable atau konstanta

Ilmu komputer
Suatu studi sistematis pada proses-proses algoritma yang menjelaskan dan mentransfor-masikan informasi

Inheritance atau pewarisan
Prinsip pewarisan sifat dari orang tua ke anak atau turunannya yang diterapkan pada kelas

Inisialisasi
Instruksi yang dilakukan pertama kali pada suatu variabel atau ekpresi pemrograman

Input
Elemen-elemen yang masuk ke dalam system

Integrated Developement Environment (IDE)
Lingkungan pengembangan aplikasi terintegrasi. Perangkat lunak untuk membantu mempermudah pembuatan aplikasi komputer

Interpreter
Penerjemah bahasa pemrograman tingkat tinggi ke bahasa mesin dengan cara satu per satu baris dibaca dan langsung diterjemahkan

Kardinalitas
Jumlah maksimum entitas pada suatu himpunan entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain

Konstanta
Variabel yang nilai datanya bersifat tetap dan tidak bisa diubah.

Loop
Proses pengulangan suatu perintah

Masalah (problem)
Perbedaan antara situasi aktual dan situasi yang diharapkan atau perbedaan antara kondisi sekarang dengan target atau tujuan yang diinginkan

Model
Penyederhanaan dari suatu system atau Tiruan dari suatu sistem dengan sedikit atau banyak penyederhanaan

Multi-tasking

Kemampuan sistem operasi untuk menjalankan beberapa tugas / aplikasi secara bersamaan

Multi-user
Kemampuan system operasi untuk dijalankan oleh pengguna yang berbeda pada waktu bersamaan

Output
Perpindahan elemen-elemen yang dihasilkan dari proses perubahan ke tujuan yang diinginkan

Pemecahan masalah
Sebuah proses dimana suatu situasi dianalisa kemudian solusi-solusi dibuat bila ditemukan ada masalah dengan cara pendefinisian, pengurangan atau penghilangan, atau pencegahan masalah

Pemrograman Berorientasi Obyek (*Object Oriented Programming – OOP*)
Paradigma pemrograman yang menggunakan obyek dan interaksinya untuk merancang aplikasi dan program komputer

Pemrograman web
Usaha untuk membuat halaman web dengan menggunakan bahasa pemrograman web (*script*)

Perangkat lunak
Seluruh instruksi yang digunakan untuk memproses informasi

Permissions
Proses untuk menentukan apa yang bisa dilakukan seorang pengguna pada suatu sistem

Pointer
Variabel yang menyimpan alamat pada memori komputer

Polymorphism
Kemampuan dari suatu obyek untuk mempunyai lebih dari satu bentuk

Programmer
Seseorang yang bekerja membuat program komputer

Prosedur
- Instruksi yang dibutuhkan oleh pengguna dalam memproses informasi
- Sekumpulan perintah yang merupakan bagian dari program yang lebih besar yang berfungsi mengerjakan suatu tugas tertentu

Proses
Perubahan atau transformasi input menjadi output

Prototyping
Salah satu pendekatan dalam pengembangan perangkat lunak yang secara langsung mendemonstrasikan bagaimana sebuah perangkat lunak atau komponen-komponen perangkat lunak akan bekerja dalam lingkungannya sebelum tahapan konstruksi aktual dilakukan

Pseudocode
Cara penulisan algoritma dengan menggunakan kode-kode yang mirip dengan bahasa pemrograman

Query
Permintaan atau pencarian pada data-data tertentu pada suatu basis data

Record
Baris data dari suatu tabel

Rekayasa Perangkat Lunak
suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, disain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan

Relationship atau relasi
Hubungan yang terjadi antara sejumlah entitas

Sistem
Kumpulan dari elemen-elemen yang saling berinteraksi untuk mencapai tujuan tertentu

Sistem basis data
Kumpulan elemen-elemen seperti basis data, perangkat lunak, perangkat keras, dan manusia yang saling berinteraksi untuk mencapai tujuan yaitu pengorganisasian data.

Software
Lihat Perangkat Lunak

Software Engineering
Lihat Rekayasa Perangkat Lunak

Solusi
Bagian akhir atau output dari proses pemecahan masalah.

Stored procedure

Potongan kode program yang dapat menerima parameter input dan menghasilkan satu atau lebih parameter output dan digunakan untuk operasi-operasi basis data

Structured Query Language (SQL)
Bahasa query terstruktur untuk mengelola basis data

Strategi pemecahan masalah
Metode atau pendekatan yang digunakan seseorang ketika menghadapi masalah

Struktur algoritma
Cara atau urutan untuk membuat suatu algoritma

Tipe data
Jenis data yang dapat diolah oleh komputer untuk memenuhi kebutuhan dalam pemrograman komputer

Trigger
Tipe khusus dari stored procedure yang akan dieksekusi ketika suatu kejadian muncul

Variabel
Tempat dimana kita dapat mengisi atau mengosongkan nilainya dan memanggil kembali apabila dibutuhkan pada suatu program

View
Tabel virtual yang isinya berdasarkan pada query yang dilakukan pada basis data.

Web browser
Perangkat lunak yang berfungsi menerjemahkan kode-kode HTML menjadi tampilan yang kita kehendaki

Web dinamis
Halaman-halaman web yang isi dan informasinya berubah-ubah sesuai dengan permintaan pengguna

Web server
Perangkat lunak yang bertindak melayani permintaan-permintaan *client* terhadap halaman-halaman *web* tertentu

Web statis
Halaman-halaman web yang isi dan informasinya tidak berubah-ubah

Lampiran 2
Daftar Alamat Situs

Berikut ini daftar alamat situs-situs internet yang penting dan digunakan sebagai rujukan dalam buku ini.

Alamat	Keterangan
http://www.apache.org	Situs resmi web server Apache. Situs ini menyediakan kode sumber Apache dan file-file binary Apache yang siap diinstall di berbagai platform sistem operasi. Selain itu juga menyediakan dokumentasi Apache yang lengkap.
http://www.borland.com	Situs resmi Borland. Borland merupakan perusahaan perangkat lunak yang memproduksi Borland Delphi, Borland JBuilder, Turbo Pascal, Turbo Delphi, Borland C++ dan lain-lain.
http://www.debian.org	Situs resmi distribusi linux Debian.
http://www.eclipse.org	Situs resmi proyek eclipse, perangkat pengembang terpadu yang mendukung banyak bahasa pemrograman.
http://www.google.com	Situs resmi search engine Google.
http://www.ilmukomputer.com	Situs berbahasa Indonesia yang menyediakan dokumen-dokumen untuk belajar berbagai sub bidang dalam ilmu computer.
http://www.javasoft.com	Situs resmi yang diluncurkan Sun Microsystem dan berisi dokumentasi dan informasi online tentang bahasa pemrograman Java.
http://www.kambing.vlsm.org	Situs dengan server local di Indonesia. Situs ini menyediakan file-file iso dari berbagai jenis distribusi linux dan dapat didownload secara bebas. Selain itu situs ini juga sebagai mirror dari berbagai distribusi linux dan aplikasi yang berjalan di linux.
http://www.linuxdoc.org	Situs yang berisi dokumentasi bebas tentang linux. Sumber informasi online yang sangat bagus untuk mempelajari linux

http://www.microsoft.com	Situs resmi Microsoft. Microsoft merupakan perusahaan perangkat lunak yang memproduksi system operasi keluarga Windows, IDE Microsoft Visual Studio, Microsoft Office, Microsoft SQL Server, dan lain-lain.
http://www.mysql.com	Situs resmi MySQL Database Software. Situs ini menyediakan file-file instalasi MySQL untuk berbagai platform sistem operasi. Selain itu juga menyediakan dokumentasi MySQL yang lengkap.
http://www.netbeans.org	Situs resmi IDE Netbeans, perangkat lunak pengembang aplikasi Java
http://www.php.net	Situs resmi bahasa pemrograman dan interpreter PHP. Situs ini menyediakan kode sumber dan file-file instalasi PHP untuk berbagai platform sistem operasi. Selain itu juga menyediakan dokumentasi PHP yang lengkap.
http://www.w3.org	Situs resmi The World Wide Web Consortium (W3C). W3C adalah konsorsium yang menentapkan standar dalam teknologi internet, terutama tentang HTML, XML, CSS, XHTML dan teknologi lain. Dokumentasi tentang teknologi tersebut dapat dijumpai di situs ini.

Lampiran 3
Fungsi-fungsi Built-in Pada Visual Basic

IsNumeric(ekspresi)

Fungsi ini digunakan untuk menguji apakah suatu ekspresi menghasilkan nilai numeric atau bukan. Nilai yang dikembalikan adalah Boolean.

IsEmpty(ekspresi)

Fungsi untuk memeriksa apakah suatu ekspresi telah berisi nilai atau tidak. Nilai yang dikembalikan adalah Boolean..

IsNull(ekspresi)

Fungsi untuk memeriksa apakah suatu ekspresi mengandung data yang tidak valid, biasanya digunakan untuk memeriksa isi field recordset.

IsArray(varname)

Fungsi untuk memeriksa apakah suatu variabel adalah suatu array.

IsDate(ekspresi)

Fungsi untuk memeriksa apakah suatu ekspresi dapat dikonversi ke date.

IsError(ekspresi)

Fungsi untuk memeriksa apakah suatu ekspresi adalah nilai error

IsObject(ekspresi)

Fungsi untuk memeriksa apakah suatu ekspresi mengacu pada suatu OLE Automation object.

IsMissing(argname)

Fungsi untuk memeriksa apakah suatu argumen optional pada procedure ada dilewatkan atau tidak

CBool(ekspresi)

Konversi suatu ekspresi ke Boolean

CByte(ekspresi)

Konversi ekspresi ke Byte

CCur(ekspresi)

Konversi suatu ekspresi ke Currency

CDate(date)

Konversi suatu ekspresi ke date

CDbl(ekspresi)

Konversi suatu ekspresi ke Double

CInt(ekspresi)

Konversi suatu ekspresi ke Integer

CLng(ekspresi)

Konversi suatu ekspresi ke Long

CSng(ekspresi)

Konversi suatu ekspresi ke single

CStr(ekspresi)

Konversi suatu ekspresi ke string

CVar(ekspresi)

Konversi suatu ekspresi ke Variant

Asc(string)

Fungsi untuk menampilkan kode character dari huruf pertama di suatu string.

Chr(charcode)

Fungsi untuk menampilkan karakter dari suatu kode karakter

Format(ekspresi[, format[, hariPertamaDariMinggu[, mingguPertamaDariTahun]]])

Memformat suatu ekspresi berdasarkan ekspresi format

Hex(number) dan Oct(number)

Menampilkan string yang mewakili Octal atau Hexa dari suatu bilangan

Str(number)

Menampilkan string yang mewakili suatu angka.

Val(string)

Menampilkan angka yang terkandung dalam suatu string.

Now

Mengembalikan suatu Variant (Date) yang menunjukkan tanggal dan waktu berdasarkan sistem komputer.

Time

Mengembalikan waktu sistem sekarang

Timer

Mengembalikan suatu bilangan yang menunjukan jumlah detik sejak tengah malam

Date

Mengembalikan tanggal sistem sekarang

Time = Time dan Date = Date

Mengatur waktu atau tanggal sistem

Untuk sistem yang menjalankan Microsoft Windows 95, tanggal yang dibutuhkan harus berupa tanggal dari 1 Jan 1998 sampai 31 Des 2099. Untuk sistem yang menjalankan Microsoft Windows NT, tanggal yang dibutuhkan harus berupa tanggal dari 1 Jan 1980 sampai 31 Desember 2079.

Hour(time), Minute(time) dan Second(time)

Mengembalikan suatu Variant (Integer) berupa bilangan 0 s/d 23 untuk jam, 0 s/d 59 untuk menit, dan 0 s/d 59 untuk detik.

Day(date), Month(date), dan Year(date)

Mengembalikan suatu Variant (Integer) berupa bilangan 1 s/d 31 untuk bulan, 1 s/d 12 untuk bulan, dan tahun.