BAB II

TINJAUAN PUSTAKA

A. Penelitian Terdahulu

Pada penelitian sebelumnya ini membahas tentang contoh-contoh sistem pakar yang telah dibangun sebelumnya. Contoh sistem pakar yang telah banyak dikembangkan untuk membantu pengguna dalam menyelesaikan masalah misalnya dalam bidang pertanian yaitu aplikasi sistem pakar untuk simulasi diagnosa hama dan penyakit tanaman bawang merah dan cabai menggunakan *forward chaining* dan pendekatan berbasis aturan (Sasmito, 2010). Sistem pakar yang dibuat digunakan untuk simulasi diagnosa hama dan penyakit tanaman hortikultura yang mencakup bawang merah dan cabai dengan menggunakan teknik inferensi *forward chaining* dan pendekatan berbasis aturan, serta memberikan solusi terhadap kesimpulan dari suatu hama dan penyakit yang telah didiagnosa berdasarkan gejala-gejalanya dan dilengkapi dengan keterangan tanaman yang terserang hama dan penyakit beserta gambar.

Sistem pakar berikutnya yang telah dikembangkan yaitu sistem pakar untuk menentukan jenis gangguan perkembangan pada anak (Rohman, dkk 2008). Penelitian ini bertujuan menghasilkan suatu sistem yang dapat digunakan oleh orang awam dalam menyelesaikan masalah yang sedikit rumit maupun sangat rumit sekalipun tanpa bantuan para ahli dalam bidangnya dan sistem ini dapat digunakan oleh seorang pakar menjadi asisten yang berpengalaman. Sistem pakar ini dibangun untuk melakukan

diagnosis gangguan perkembangan anak di bawah umur 10 tahun dengan hanya memperhatikan gejala-gejala yang dialami. Metode *Certainty Factor* ini digunakan agar didapat nilai kemungkinan gangguan yang dialami pasien melalui penginputan gejala-gejalanya.

Dan sistem pakar dalam bidang kesehatan berikutnya yaitu sistem pakar diagnosa penyakit diabetes *nefropathy* dengan metode *certainty factor* berbasis *website* dan *Mobile* (Puspitasari, 2010). Sistem ini dibangun dengan tujuan membantu dokter dan paramedis dalam mengambil keputusan tentang penyakit apa yang diderita oleh pasien berdasarkan inputan yang diberikan pada sistem. Sehingga paramedis dapat memberikan solusi-solusi apa yang harus dilakukan oleh pasien dalam mengatasi penyakit yang dideritanya secara tepat dan sedini mungkin. Kemudahan dalam mengakses perangkat lunak melalui komputer atau *handphone* diharapkan dapat mempercepat proses diagnosa secara tepat.

B. Kecerdasan Buatan

Kecerdasan buatan (*Artificial Intelligence*) adalah ide-ide untuk membuat suatu perangkat lunak komputer yang memiliki kecerdasan sehingga perangkat lunak komputer tersebut dapat melakukan suatu pekerjaan yang dilakukan oleh manusia (Artanti, 2004), dengan kata lain membuat sebuah komputer dapat berpikir dan bernalar seperti manusia. Tujuan dari kecerdsan buatan adalah membuat komputer lebih cerdas, mengerti tentang kecerdasan dan membuat mesin lebih berguna bagi manusia. Kecerdasan

buatan dapat membantu meringankan beban kerja manusia misalnya dalam membuat keputusan, mencari informasi secara lebih akurat atau membuat komputer lebih mudah digunakan dengan tampilan yang lebih mudah dipahami. Cara kerja kecerdasan buatan adalah menerima input, untuk kemudian diproses dan kemudian mengeluarkan output yang berupa keputusan.

Menurut John Mc Carthy 1956, sistem pakar adalah proses untuk mengetahui dan memodelkan proses-proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia. Kecerdasan atau kepandaian ini di dapat berdasarkan pengetahuan dan pengalaman, untuk itu agar perangkat lunak yang dikembangkan dapat mempunyai kecerdasan maka perangkat lunak tersebut harus diberi suatu pengetahuan dan kemampuan untuk menalar dari pengetahuan yang telah didapat dalam menemukan solusi atau kesimpuan layaknya seorang pakar dalam bidang tertentu yang spesifik.

Kecerdasan buatan menawarkan media dan uji teori kecedasan. Teori ini dapat dinyatakan dalam bahasa program komputer dan dibutikan melalui eksekusinya pada komputer nyata. Implementasi dari kecerdasan buatan saat ini dapat ditemui adalam-daam bidang antara lain:

1. *Fuzzy logic* merupakan suatu metode kecerdasan buatan yan banyak terdapat pada alat elektronik dan robot dimana ala-alat elektronik dan robot terebut mampu berpikir dan bertingkah laku seperti manusia.
2. *Komputer vision* merupakan suatu metode kecerdasan buatan yang memungkinkan sebuah sistem komputer mengenali gamar sebagai

inputnya. Contohnya adalah mengenali dan membaca tulisan yang ada gambarnya.

3. *Artificial intelligence* dalam game merupakan metode kecerdasan buatan yang berguna untuk meniru cara berpikir manusia dalam bermain game.
4. *Speech recognition* merupakan suatu metode kecerdasan buatan yang berguna untuk mengenali suara manusia dengan cara dicocokkan dengan acuhan atau *pattern* yang telah diprogram sebelumnya. Contohnya adalah suara dari *user* dapat diterjemahkan menjadi sebuah perintah bagi komputer.
5. *Expert system* merupakan metode kecerdasan buatan yang berguna untuk meniru cara berpikir dan penalaran seorang ahli dalam mengambil keputusan berdasarkan situasi yang ada.

C. Sistem Pakar

Sistem pakar (*expert system*) adalah salah satu teknik kecerdasan buatan yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli. Dengan sistem pakar ini, orang awampun dapat menyelesaikan masalah yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Bagi para ahli, sistem pakar ini juga akan membantu aktivitasnya sebagai asisten yang sangat berpengalaman. Ada beberapa definisi tentang sistem pakar, antara lain : (Sri Kusumadewi, 2003:109).

1. Menurut Durkin

   Sistem pakar adalah suatu program komputer yang dirancang untuk memodelkan kemampuan penyelesaian masalah yang dilakukan seorang pakar.

2. Menurut Ignizio

   Sistem pakar adalah suatu model dan prosedur yang berkaitan, dalam suatu domain tertentu, yang mana tingkat keahliannya dapat dibandingkan dengan keahlian seorang pakar.

3. Menurut Giarratano dan Riley

   Sistem pakar adalah suatu sistem komputer yang bisa menyamai atau meniru kemampuan seorang pakar

D. Konsep Dasar Sistem Bakar

Menurut Efraim Turban, konsep dasar sistem pakar mengandung: keahlian, ahli, pengalihan keahlian, inferensi, aturan dan kemampuan menjelaskan keahlian adalah suatu kelebihan penguasaan pengetahuan di bidang tertentu yang diperoleh dari pelatihan, membaca atau pengalaman. Contoh bentuk pengetahuan yang termasuk keahlian adalah :

1. Fakta-fakta pada lingkup permasalahan tertentu.
2. Teori-teori pada lingkup permasalahan tertentu.
3. Prosedur-prosedur dan aturan-aturan berkenaan dengan lingkup permasalahan tertentu.
4. Strategi-strategi global untuk menyelesaikan masalah.
5. *Meta-knowledge* (pengetahuan tentang pengetahuan).

Bentuk-bentuk ini memungkinkan para ahli untuk dapat mengambil keputusan lebih cepat dan lebih baik daripada seseorang yang bukan ahli. Seorang ahli adalah seseorang yang mampu menjelaskan suatu tanggapan, mempelajari hal-hal baru seputar topik permasalahan (domain), menyusun kembali pengetahuan jika dipandang perlu, memecah aturan-aturan jika dibutuhkan, dan menentukan relevan tidaknya keahlian mereka.

Pengalihan keahlian dari para ahli ke komputer untuk kemudian dialihkan lagi ke orang lain yang bukan ahli, merupakan tujuan utama dari sistem pakar. Proses ini membutuhkan 4 aktivitas yaitu :

1. Tambahan pengetahuan (dari para ahli atau sumber-sumber lainnyan. b. Representasi pengetahuan (ke komputer).
2. Inferensi pengetahuan.
3. Pengalihan pengetahuan ke *user*.

Pengetahuan yang disimpan di komputer disebut dengan nama basis pengetahuan. Ada dua tipe pengetahuan, yaitu: fakta dan prosedur (biasanya berupa aturan).

Salah satu fitur yang harus dimiliki oleh sistem pakar adalah kemampuan untuk menalar. Jika keahlian-keahlian sudah tersimpan sebagai basis pengetahuan dan sudah tersedia program yang mampu mengakses basisdata, maka komputer harus dapat diprogram untuk membuat inferensi. Proses inferensi ini dikemas dalam bentuk motor inferensi (inference engine)..

Fitur lainnya dari sistem pakar adalah kemampuan untuk merekomendasi. Kemampuan inilah yang membedakan sistem pakar dengan sistem konvensional.

E. Struktur Sistem Pakar

Sistem pakar terdiri-dari 2 bagian pokok, yaitu: lingkungan pengembangan (*development environment*) dan lingkungan konsultasi (*consultation environment*). Lingkungan pengembangan digunakan sebagai pembangunan sistem pakar baik dari segi pembangunan komponen maupun basis pengetahuan. Lingkungan konsultasi digunakan oleh seorang yang bukan ahli untuk berkonsultasi. (Sri Kusumadewi, 2003 : 113)

Menurut Turban (1995), terdapat 3 orang yang terlibat dalam lingkungan *system* pakar, yaitu :

1. Pakar

   Pakar adalah orang yang memiliki penegtahuan khusus, pendapat, pengalaman dan metode, serta kemampuan untuk mengaplikasikan keahliannya tersebut guna menyelasaikan masalah.

2. *Knowledge engineer* (perekayasa sistem)

   Knowledge engineer adalah orang yang membantu pakar dalam menyusun area permasalahan dengan menginterpretasikan dan mengitegrasikan jawaban-jawaban pakar atas pertanyaan yang diajukan, menggambarkan analogi, mengajukan counter example dan menerangkan kessulitan kesulitan konseptual.

3. Pemakai

   Sistem pakar memiliki beberpa pemakai ,yaitu : pemakai bukan pakar,, pembangun system pakar yang ingin meningkatkan dan menambah basis pengetahuan, dan pakar (Muhamad Arhami, 2005:12-13).

F. Penyakit Tanaman Tembakau

Menurut Tim Bina Karya Tani (2008), penyakit tanaman adalah gangguan pada tanaman yang disebabkan oleh *mikroorganisme*. *Mikroorganisme* tersebut adalah virus, bakteri, *protozoa*, jamur dan cacing *nematoda*.

Mikroorganisme ini dapat menyerang organ tumbuhan seperti akar, batang, daun dan buah. Penyebaran penyakit pada tanaman dapat terjadi melalui angin, air atau serangga. Serangga dapat menularkan virus, bakteri, *protozoa* dan jamur dari satu tanaman ke tanaman lain. Selain itu, factor lingkungan, misalnya kelembaban dan suhu juga mempengaruhi penyakit pada tanaman. Tanaman yang terserang suatu penyakit akan terhambat pertumbuhannya, dan akhirnya dapat mati.

G. *PHP*

PHP merupakan bahasa pemrograman yang berjalan dalam sebuah *web server*. *PHP* diciptakan oleh seorang *programmer unix* dan *perl* yang bernama Rasmush Lerdoft pada bulan Agustus-September tahun 1994. Pada awalnya Rasmush mencoba menciptakan sebuah *script* dalam *website* pribadinya dengan tujuan untuk memonitor siapa saja yang pernah mengunjungi *website*nya.

Pada awal tahun 1995 *PHP* mulai dikenalkan Rasmush kepada *programmer* pemula, dengan alasan bahwa bahasa yang digunakan dalam *PHP* cukup sederhana dan mudah dipahami. Selanjutnya Rasmush menulis ulang *PHP* dengan bahasa C untuk meningkatkan kecepatan aksesnya. Pada

bulan September-Oktober 1995 kode *PHP* ditulis ulang dan digabungkan menjadi *PHP*/FI. Baru kemudian di akhir tahun 1995 dirilis bagi umum secara gratis (Rafiza H;2006;1).

PHP adalah bahasa *server-side scripting* yang menyatu dengan *HTML* untuk membuat halaman *web* yang dinamis. Maksud dari *server side scripting* adalah sitaksdan perintah-perintah yang digunakan sepenuhnya dijalankan di *server* tetapi disertakan pada dokumen *HTML* (Sunarfrihantono Bimo:2003:1)

H. *XAMPP*

Xampp merupakan *software* yang berisi paket pendukung seperti interpreter *PHP*, *Web Server* dan data *MySQL*. *Xampp* meruapakn paket *PHP* yang berbasis *open source*. Informasi lengkap mengenai produk ini dapat diakses di situs resmi websitenya, yaitu: *http:/www.apachefriends.com xampp* berfungsi sebagai pengembang apikasi (*project*) berbasis *PHP*. *Xampp* mengkombinasikan eberapa paket *software* berbeda kedalam satu paket. Paket-paket yang dimaksud adalah *Apache, My Sql, PHP, Perl, FileZilla FTP Sever, Php my Admin, Open SSL, Free type, Webalizer, mood_perl, Truck MMChace, mcrypt, SQL Lit, JP Grapt, Mercury Mail Transport* Sistem, *PHPB lender PHP Compiler* (Riyanto;2009;271).

I. *MySQL*

MySQL adalah seuah program databae yang mampu menerima dan mengirimkan data denga cepat, multi user serta menggunakan perintah standar *SQL* (*Structure Query Language) MySQL* merupakansebuah *database*

yang *free*, artinya kia bebas menggunakan *database* ini untuk kepentingan pribadi dan usaha tanpa harus membeli atau membayar lisensnya. *MySQL* dirilis oleh seorang *programmer database* bernama Michael Widenius.

MySQL merupakan sebuah *database server* yang juga dapat berperan sebagai *client* sehingga sering disebut *database client/server* yang *open source* dengan kemampuan dapat berjala baik baik di Sistem Operasi manapun dengan *platform Windows* maupun Linux (Nugroho:2005).

J. *ERD*

Entity Relationship Data (*ERD*) merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi. *ERD* untuk memodelkan struktur data dan hubungan antar data, untuk menggambarkannya digunakan beberapa notasi dan simbol. Ada tiga simbol yang digunakan, yaitu:

1. *Entity*

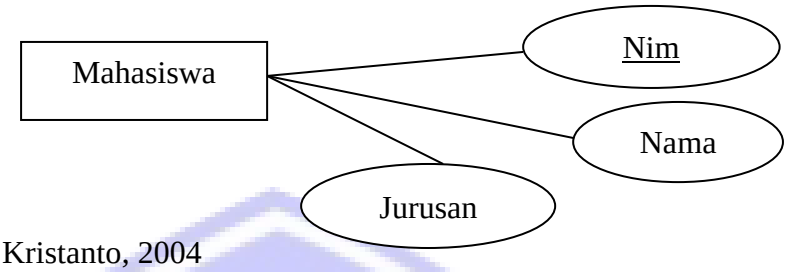
   Entity adalah orang, tempat, kejadian atau konsep yang informasinya direkam. Pada bidang administrasi siswa misalnya, *entity* adalah siswa, buku, pembayaran, nilai test. Pada bidang kesehatan, *entity* adalah pasien, obat, kamar, diet. Simbol yang digunakan untuk *entity* adalah persegi panjang.

2. *Attribute*

   Setiap *entity* mempunyai *attribute* atau sebutan untuk mewakili suatu entity. Seorang siswa dapat dilihat dari atributenya, misalnya nama, nomor siswa, alamat, nama orang tua, hobby. *Attribute* juga disebut

sebagai data elemen, data *field*, data item. (Sumber : Kristanto, 2004, 2)

Contoh:



Sumber : Kristanto, 2004

Gambar 2.1 *Atribut* dari Sebuah *Entity*.

3. *Relationship*

Hubungan yang terjadi antara satu atau lebih *entity*. *Relationship* tidak mempunyai keberadaan fisik, kecuali yang mewarisi hubungan antara *entity* tersebut. *Relationship set* adalah kumpulan *relationship* yang sejenis. Simbol yang digunakan adalah belah ketupat, *diamond* atau *rectangel*.

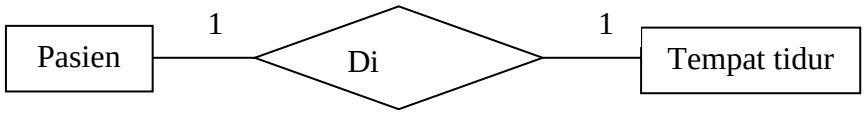
Contoh:

Sumber : Kristanto, 2004

Gambar 2.2 *Relationship*.

a. *One to one* (1:1)

Hubungan satu *entity* dengan satu *entity*



Sumber : Kristanto, 2004

Gambar 2.3 *Relationship one to one*

b. *One to many* (1:M) atau *many to one* (M:1)

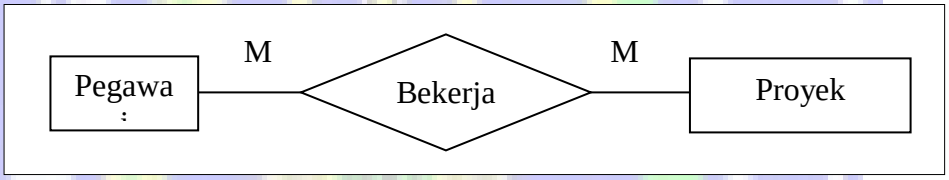
Hubungan satu *entity* dengan banyak *entity* atau banyak *entity* dengan satu entity.

Sumber : Kristanto, 2004

Gambar 2.4 *Relationship One to Many.*

c. *Many to many* (M:M)

Hubungan banyak *entity* dengan banyak *entity*.



Sumber : Kristanto, 2004.

Gambar 2.5 *Relationship Many to Many*.

K. *DFD*

Data Flow Diagram (*DFD*) adalah representasi grafik dari sebuah sistem. *DFD* menggambarkan komponen-komponen sebuah *sistem*, aliran-aliran data dimana komponen-komponen tersebut, dan asal, tujuan dan penyimpanan dari data tersebut.

Kita dapat menggunakan *DFD* untuk dua hal utama, yaitu untuk membuat dokumentasi dari sistem informasi yang ada atau untuk menyusun dokumentasi untuk sistem informasi yang baru. Empat simbol yang digunakan dalam *DFD* adalah:

1. Simbol *entitas eksternal/terminator* menggambarkan awal atau tujuan data di luar sistem.

Notasi Yourdon/DeMarco *Notasi Gane dan Sarson*

Sumber : Abdul Kadir, 2007

Gambar 2.6 Simbol *entitas eksternal/terminator*.

2. Simbol lingkaran menggambarkan entitas atau proses dimana aliran data masuk ditransformasikan ke aliran data keluar.

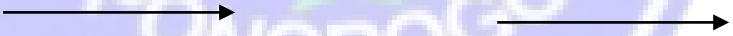
Notasi Yourdon/DeMarco *Notasi Gane dan Sarson*

Sumber : Abdul Kadir, 2007

Gambar 2.7 Simbol lingkaran.

3. Simbol aliran data menggambarkan aliran data

Notasi Yourdon/DeMarco *Notasi Gane dan Sarson*



Sumber : Abdul Kadir, 2007

Gambar 2.8 Simbol aliran data.

4. Simbol *file* menggambarkan tempat data disimpan.

Notasi Yourdon/DeMarco *Notasi Gane dan Sarson*

Sumber : Abdul Kadir, 2007

Gambar 2.9 Simbol *file*.

Abdul Kadir (2007) menjelaskan ada tiga jenis *DFD* yaitu:

1. *Context Diagram* (*CD*)

   Jenis pertama *Context Diagram*, adalah *data flow diagram* tingkat atas (*DFD Top Level*), yaitu diagram yang paling tidak detail, dari sebuah sistem informasi yang menggambarkan aliran-aliran data ke dalam dan ke luar sistem dan ke dalam dan ke luar entitas-entitas eksternal. Beberapa hal yang harus diperhatikan dalam menggambar *CD*:

   a. Batas sistem adalah batas antara "daerah kepentingan sistem."
   b. Lingkungan sistem adalah segala sesuatu yang berhubungan atau mempengaruhi sistem tersebut.
   c. *Interface* adalah aliran yang menghubungkan sebuah sistem dengan lingkungan sistem tersebut.

2. *DFD* Fisik

   DFD fisik adalah representasi grafik dari sebuah sistem yang menunjukkan *entitas-entitas internal* dan *eksternal* dari sistem tersebut, dan aliran-aliran data ke dalam dan ke luar dari entitas-entitas tersebut. *Entitas-entitas internal* adalah personel, tempat (sebuah bagian), atau mesin (misalnya, sebuah komputer) dalam sistem tersebut yang mentransformasikan data. Maka *DFD* fisik tidak menunjukkan apa yang dilakukan, tetapi menunjukkan dimana, bagaimana, dan oleh siapa proses-proses dalam sebuah sistem dilakukan.

   Perlu diperhatikan didalam memberikan keterangan di lingkaran-lingkaran (simbol proses) dan aliran-aliran data (simbol aliran data) dalam *DFD* fisik menggunakan label atau keterangan dari kata benda

untuk menunjukkan bagaimana sistem mentransmisikan data antara lingkaran-lingkaran tersebut.

Misal:

Aliran data : Kas, formulir 66W, slip setoran

Proses : Cleck penjualan, kasir, pembukuan, dll.

3. *DFD Logis*

DFD logis adalah grafik dari sebuah sistem yang menunjukkan proses-proses dalam sistem tersebut dan aliran-aliran data ke dalam dan ke luar dari proses-proses tersebut. Kita menggunakan *DFD* logis untuk membuat dokumentasi sebuah sistem informasi karena *DFD* logis dapat mewakili logika tersebut, yaitu apa yang dilakukan oleh sistem tersebut, tanpa perlu menspesifikasi dimana, bagaimana dan oleh siapa proses-proses dalam sistem dilakukan. (Sumber : Abdul Kadir, 2007, 51-53).

L. *Flowchart*

1. Pengertian *Flowchart* (Diagram Alur).

Karena komputer membutuhkan hal-hal yang rinci, maka bahasa pemrograman bukanlah alat baik untuk merancang sebuah algoritma awal. Alat yang banyak dipakai untuk membuat algoritma adalah diagram alur (*flowchart*).

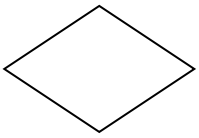
Diagram alur dapat menunjukkan secara jelas arus pengendalian suatu algoritma, yakni melaksanakan suatu rangkaian kegiatan secara logis dan sistematis. Suatu diagram alur dapat memberi gambaran dua dimensi berupa simbol-simbol grafis. Masing-masing simbol telah

ditetapkan lebih dahulu fungsi dan artinya. Simbol-simbol tersebut dipakai untuk menunjukkan berbagai kegiatan operasi dan jalur pengendalian. Arti khusus dari sebuah *flowchart* adalah simbol-simbol yang digunakan untuk menggambarkan urutan proses yang terjadi di dalam suatu program komputer secara sistematis dan logis. (Sumber : Tata Sutabri, 2004,21)

2. Simbol-simbol *Flowchart*

Sudah dikemukakan di atas bahwa diagram alur atau *flowchart* memiliki beberapa simbol yang biasa digunakan untuk menggambarkan rangkaian proses yang harus dilaksanakan. Simbol tersebut dijelaskan di bawah ini: (Sumber : Tata Sutabri, 2004, 21-22)

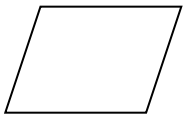
Tabel 2.1 Simbol-simbol *Flowchart*

Simbol *Flowchart*	Fungsi
1. 	*TERMINAL* Simbol ini digunakan untuk mengawali atau mengakhiri suatu proses/kegiatan.
2. 	*PREPARATION* Simbol ini digunakan untuk mempersiapkan harga awal/nilai awal suatu variabel yang akan diproses dan digunakan untuk proses *loop*.
3. 	*DECISION* Simbol ini digunakan untuk pengujian suatu kondisi yang sedang diproses.

4. 

PROSES

Simbol ini digunakan untuk menggambarkan suatu proses yang sedang dieksekusi.

5. 

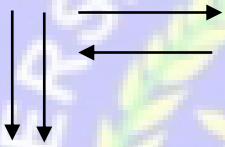
INPUT/OUTPUT

Simbol ini digunakan untuk menggambarkan proses *input* (*read*) maupun proses *output* (*print*).

6. 

SUBROUTINE

digunakan untuk menggambarkan proses pemanggilan subprogram dari mainprogram.

7. 

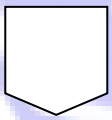
FLOW LINE

Simbol ini digunakan untuk menggambarakan arus proses dari suatu kegiatan ke kegiatan lain.

8. 

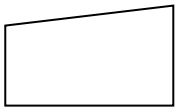
CONECTOR

Simbol ini digunakan sebagai penghubung antara suatu proses dengan proses lainnya yang ada di dalam suatu lembar halaman.

9. 

PAGE CONECTOR

Simbol ini digunakan sebagai penghubung antara suatu proses dengan proses lainnya, tetapi berpindah halaman.

10. 

MANUAL OPERATION

Simbol ini digunakan untuk menggambarkan suatu kegiatan atau proses yang bersifat manualisasi.

11. 

PRINTER

Simbol ini digunakan untuk menggambarkan suatu dokumen atau suatu kegiatan mencetak suatu informasi dengan mesin *printer*.

12. 

CONSOLE

Simbol ini digunakan untuk menggambarkan suatu kegiatan menampilkan data atau informasi melalui monitor atau CRT (*Cathode Ray Tube*).

13. 

DISK

Simbol ini digunakan untuk menggambarkan suatu kegiatan membaca atau menulis data menggunakan media *magnetic disk*.

14. 

MANUAL INPUT

Simbol ini digunakan untuk menggambarkan proses pemasukan data melalui media *keyboard*.

15. 

TAPE

Simbol ini digunakan untuk menggambarkan suatu kegiatan membaca atau menulis data menggunakan media *magnetic tape*.

---

3. Jenis-jenis *Flowchart*

Bentuk diagram alur (*flowchart*) yang sering digunakan dalam proses pembuatan suatu program komputer adalah sebagai berikut:

a. Program *Flowchart*.

Simbol-simbol yang menggambarkan proses secara rinci dan detil antara intruksi yang satu dengan intruksi yang lainnya dalam suatu program komputer yang bersifat logik.

b. Sistem *Flowchart*.

Simbol-simbol yang menggambarkan urutan prosedur secara detil dalam suatu sistem komputerisasi bersifat fisik.

c. Teknik Pembuatan *Flowchart*.

Sebelum kita membuat sebuah program komputer, yang harus kita lakukan sebelumnya adalah membuat *flowchart*. Jenis *flowchart* yang sering digunakan adalah program *flowchart*.

Teknik pembuatan program *flowchart* ini dibagi menjadi dua bagian, yaitu:

1) *General Way*

Teknik pembuatan *flowchart* dengan cara ini lazim digunakan untuk menyusun logika suatu program. Teknik ini menggunakan pengulangan proses secara tidak langsung (*Non-Direct-Loop*).

2) *Iteration Way*

Teknik pembuatan *flowchart* dengan cara ini biasanya dipakai untuk logika program yang cepat dan bentuk permasalahannya kompleks. Pengulangan proses yang terjadi bersifat langsung (*Direct-Loop*). (Sumber : Tata Sutabri, 2004; 24)

M. Faktor Kepastian *(Certainty Factor)*

Salah satu metode yang berhubungan dengan ketidakpastian adalah Faktor Kepastian (*Certainty Factor*). *Certainty Factor (CF)* menunjukan ukuran kepastian terhadap suatu fakta atau aturan. .Penentuan nilai ini diberikan oleh pakar antar 0 dan 1. Definisi menurut David McAllister adalah suatu metode untuk membuktikan apakah suatu fakta itu pasti ataukah tidak pasti yang berbentuk metric yang biasanya digunakan dalam sistem pakar. Metode ini sangat cocok untuk sistem pakar yang mendiagnosis sesuatu yang belum pasti. Metode *Certainty Factors* Aturan metode *Certainty Factors*:

1. McAllister menggambarkan aturan untuk menambahkan dua faktor *Certaint* positif adalah :

   (CFaCFb) = CFa + CFb * (1- CFa)

2. Aturan untuk menambahkan dua *Certaint* yang negatif adalah:

   (CFcCFd) = CFc + CFd + CFc*CFd

3. Aturan untuk menambahkan *Certainty Factors* positif dan *Certainty Factors* negatif lebih kompleks:

   (CFeCFf) = CFe + CFf / 1 – min{ | CFe | , | CFf | }

   Tiga aturan ini menyediakan suatu skala interval untuk Certainty Factors.

   Contoh untuk fakta yang positif: Strong suggestive (CFa): 0.8

   Suggestive (CFb) : 0.6

   CFcombine (CFa CFb) = 0.8 + 0.6 (1-0.8)

   = 0.92

Contoh untuk fakta yang negatif: Strong suggestive (CFc): -0.8

CFcombine (*CFc* CFd) = -0.8 + -0.6 + -0.8 * -0.6

= -0.92

Contoh untuk fakta yang positif dan negatif:

Certainty factor adalah 0.88 (CFe)

Certainty factor against adalah 0.90 (CFf)

CFcombine (CFeCFf) = 0.88 + (- 0,90) / 1 min { | 0.88 | , | 0,90 | }

= - 0,17

Kelebihan dan Kekurangan Metode *Certainty Factors*

1. Metode ini cocok dipakai dalam sistem pakar untuk mengukur sesuatu apakah pasti atau tidak pasti dalam mendiagnosis penyakit sebagai salah satu contohnya.
2. Perhitungan dengan menggunakan metode ini dalam sekali hitung hanya dapat mengolah 2 data saja sehingga keakuratan data dapat terjaga.

Kekurangan metode *Certainty Factors* adalah:

1. Ide umum dari pemodelan ketidakpastian manusia dengan menggunakan numerik metode *certainty factors* biasanya diperdebatkan. Sebagian orang akan membantah pendapat bahwa formula untuk metode *certainty factors* diatas memiliki sedikit kebenaran.
2. Metode ini hanya dapat mengolah ketidakpastian/kepastian hanya 2 data saja. Perlu dilakukan beberapa kali pengolahan data untuk data yang lebih dari 2 buah.