BAB II

TEORI PENUNJANG

2.1. ALGORITMA DAN PEMROGRAMAN

2.1.1. ALGORITMA

Difinisi :

Algoritma adalah suatu prosedur yang memerinci sejumlah langkah operasional yang perlu dilaksanakan untuk menyelesaikan masalah dengan komputer.

Algoritma merupakan urutan langkah penyelesaiaan yang terarah dan sistematis yang bertujuan memberikan effisiensi pemrograman. Algoritma adalah struktur dasar dari program yang effisien dan efektif. Hal ini didasari dari kenyataan bahwa baik tidaknya suatu algoritma sangat berpengaruh terhadap penggunaan ruang dan waktu eksekusi program.

Algoritma yang baik memenuhi kriteria sebagai berikut :

1. Algoritma tersusun dari sejumlah langkah operasional yang berurutan.
2. Susunan langkah operasional algoritma mempunyai awalan dan akhiran.
3. Urutan langkahnya jelas atau tidak berbelit-belit dan terarah.
4. Algoritma mempunyai tujuan yang jelas, tepat guna, effisien dan efektif dalam memecahkan suatu masalah.
5. Kompatibel terhadap semua bahasa pemrograman.

Untuk mengilustrasikan suatu algoritma, ada 3 bentuk penulisan yang dipakai yaitu :

1. Bahasa Alami (Natural Language).

Dengan menggunakan bahasa alami (bahasa sehari-hari), maka algoritma mudah dibaca. Namun kerugiannya yaitu bentuk penulisan algoritma akan menjadi panjang dan dapat mengakibatkan tersamarnya alur atau langkah prosedural algoritma.

2. Sandi Semu (Pseudocode)

Sandi Semu adalah salah satu cara mengilustrasikan algoritma dengan menggunakan kata-kata sandi dan tanda-tanda operasi yang mempunyai arti tertentu. Penulisan dengan sandi semu lebih mendekati ke bahasa pemrograman sehingga bagi yang belum terbiasa melakukan pemrograman akan terasa sulit. Keuntungannya yaitu penulisan menjadi singkat dan mudah untuk dikodekan.

3. Diagram Alir (Flow Chart)

Bentuk penulisan algoritma yang umum adalah diagram alir. Diagram alir adalah bentuk penulisan algoritma yang mengunakan gambar/simbol untuk mewakilkan langkah operasional algoritma. Gambar/simbol yang sering digunakan antara lain :

Teminal : gambar /simbol yang mewakili awalan atau akhiran dari langkah prosedural algoritma.

Persiapan (Preparation): gambar/simbol yang mewakili langkah pendahuluan seperti pendeklarasian variabel dan pendifinisian nilai awal variabel.

Proses (Process) : gambar/simbol yang mewakili langkah pemrosesan data. Hasil proses dapat berupa data atau perintah ke suatu peripheral.

Masukkan/keluaran (Input/Output) : gambar/simbol yang mewakili langkah pemasukkan data atau pengiriman data ke suatu output device.

Keputusan (Decision) : gambar/simbol yang mewakili langkah percabangan dengan pemilihan keputusan "Yes" atau "No", "True" atau "False" dari suatu kalimat logika (logical statement).

Penghubung (Connector in page) : gambar/simbol yang menerangkan bahwa aliran diputus dan dilanjutkan pada gambar/simbol serupa di halaman yang sama. Pemutusan dimaksudkan agar diagram aliran algoritma tidak terlihat rumit sebagai akibat adanya lompatan dan loop yang panjang.

Penghubung halaman (Off page connector) : gambar/simbol yang menerangkan bahwa aliran prosedural algoritma terputus oleh halaman dan dilanjutkan pada gambar/simbol serupa di halaman selanjutnya.

Aliran (Flow) : gambar/simbol yang menujukkan arah aliran algoritma menuju ke gambar/simbol berikutnya sesuai arah anak panah.

Penulisan algoritma mengunakan flow chart lebih mudah untuk dipahami pemikiran/ide dasarnya. Untuk pemrograman yang besar dan kompleks, penulisan algoritmanya dibagi ke dalam 2 jenis diagram alir yaitu sketsa flow chart dan detail flow chart. Sketsa flow chart adalah diagram alir yang menggambarkan secara garis besar langkah-langkah algoritma. Sketsa flow chart terdiri dari modul-modul, fungsi dan prosedur yang digunakan pada algoritma. Detail flow chart adalah diagram alir yang menggambarkan secara terperinci setiap komponen dari sketsa flow chart. Detail flow chart terdiri dari statement-statement dasar operasi yang harus dilakukan sesuai algoritma.

Dengan memahami permasalahan dan memecah permasalahan menjadi bagian-bagian sederhana yang dapat diselesaikan, maka algoritma penyelesaiannya dapat

ditemukan. Satu hal yang perlu ditekankan bahwa algoritma yang diperoleh belum tentu merupakan algoritma yang effisien terhadap kompleksitas ruang dan waktu.

2.1.2. PEMROGRAMAN

Difinisi :

Program adalah suatu daftar instruksi komputer yang terperinci dan terdokumentasi.

Sebuah program merupakan sederetan instruksi yang memerintahkan komputer untuk melakukan langkah-langkah operasional seperti perhitungan, penyimpanan, penghapusan, pengiriman data dan lain sebagainya. Program adalah komponen terpenting yang menjadi referensi selain perangkat keras komputer, sebab tanpa program komputer hanyalah sebuah barang elektronik yang tidak berguna. Pengertian terdokumentasi yaitu bahwa setiap program tersimpan dalam bermacam-macam bentuk seperti executable file, script file, batch file, memory chip program (tersimpan berupa data bit yang tersupply/tidak tersupply oleh baterry), kartu lubang , dan lain-lainnya. Jika suatu instruksi masih berupa imaginasi atau kata-kata yang tak berujud sehingga komputer tidak dapat mengerjakannya maka instruksi tersebut bukan suatu program.

Program komputer sangat sulit diciptakan tanpa bantuan compiler atau interpreter. Pada dasarnya program komputer terdiri dari instruksi-instruksi mesin yang hanya berupa kode bit (kode biner 1 dan 0 yang menyatakan keadaan bit-bit registers). Kode bit merupakan pulsa digital rangkaian transistor yang berbentuk gerbang logika sehingga instruksi mesin sangat sulit untuk dibaca oleh manusia. Agar komputer dapat dioperasikan sesuai keinginan maka diperlukan interpreter/compiler yang secara langsung dapat menghubungkan bahasa yang dimengerti manusia ke instruksi mesin. Program komputer

yang dapat dibuat manusia adalah program berbahasa tingkat tinggi. Program berbahasa tingkat tinggi lebih mendekati bahasa manusia sehingga mudah dipelajari.

Selain Algoritma, teknik pemrograman juga menjadi syarat suatu progam yang berkualitas tinggi. Program yang dimaksud memiliki kriteria sebagai berikut :

1. Minim terhadap kesalahan (Minimum Bug).
2. Ramah dan mudah dimengerti (User Friendly).
3. Kompatibel terhadap semua komputer (Compatible for all PC).
4. Tampilan menarik (Nice).

Untuk menghasilkan program sesuai kriteria diatas diperlukan perancangan program yang benar.

Program tidak dapat dibuat ketika ada permasalahan, namun pembuatannya melalui beberapa tahapan. Tahapan-tahapan yang terjadi dalam proses pembuatan program adalah sebagai berikut :

1. Pemahaman permasalahan.

   Tahapan ini memerlukan penalaran yang baik agar permasalahan dapat dipahami dengan benar. Untuk memahami permasalahan biasanya permasalahan diuraikan dan dipisahkan menjadi komponen-komponen permasalahan.

2. Perencanaan pemecahan masalah.

   Pencarian solusi permasalahan merupakan hal terpenting dalam mewujudkan program yang berkualitas. Pemilihan teknik atau metode yang dipakai dapat berpedoman kepada hasil analisis algoritmanya. Untuk permasalahan yang kompleks maka perlu dibedakan permasalahan yang dapat diselesaikan dengan komputasi dan

permasalahan yang tidak bisa dipecahkan dengan komputasi. Hasil dari tahapan ini adalah suatu algoritma pemecahan masalah yang terperinci bagian demi bagian.

3. Penulisan program / pengkodean.

   Tahapan penulisan program dapat dilakukan setelah adanya konsep pemecahan masalah yang berupa algoritma. Pengkodean tidak harus selesai menjadi program yang besar dalam satu kesempatan, tetapi dilakukan langkah demi langkah yang diselingi dengan proses pelacakkan kesalahan (debugging).

4. Pengujian dan proses pembetulan kesalahan.

   Proses pengujian dapat dilakukan berulang-ulang bergantian dengan tahap pengkodean. Hal ini disebabkan proses pembetulan kesalahan akan menjadi sangat sulit jika tubuh program telah besar. Pengujian langkah per langkah dilakukan pada proses kompilasi dan ditujukan untuk melacak terjadinya kesalahan pengkodean. Kesalahan yang sering terjadi adalah kesalahan berkaitan dengan penulisan statement bahasa pemrograman. Pengujian program juga dilakukan saat proses integrasi modul-modul program (linking process). Kesalahan yang umum terjadi dalam proses integrasi adalah tidak tersedianya komponen yang dibutuhkan.

   Proses pengujian terakhir dilakukan setelah program selesai. Tujuan pengujian ini adalah mengetahui ada tidaknya kesalahan algoritma. Kesalahan algoritma merupakan tipe kesalahan yang sulit untuk dilacak. Kesalahan algoritma tidak dapat dilacak oleh compiler/interpreter. Ciri-ciri kesalahan algoritma adalah program dapat berjalan, namun memberikan hasil yang salah.

5. Dokumentasi program.

Program yang baik mempunyai dokumentasi perograman yang terpelihara. Tujuan dokumentasi program adalah memberikan informasi penting yang berkaitan dengan pemrograman. Hal yang biasa ada dalam dokumentasi program adalah spesifikasi peralatan yang disarankan, referensi teknis, dan bantuan mengatasi kesalahan.

2.2. PEMROGRAMAN C++

2.2.1. Pemrograman Berorientasi Objek (OOP)

Difinisi :

Objek dalam pengertian paling sederhana adalah benda atau entitas dalam dunia nyata yang menjadi sasaran, permasalahan, atau fokus suatu konteks pengertian.

Ketika pemrogram menciptakan suatu program, instruksi yang ditulis bekerja dengan hal-hal berbeda, seperti variabel atau file. Suatu hal yang berbeda menyebabkan perbedaan dalam operasi program. Kenyataan inilah yang mendasari pemikiran konsep OOP. Suatu hal yang spesifik dan berbeda dalam pemrograman dapat dipandang sebagai suatu objek. Sebagai Contoh file adalah sesuatu yang spesifik dan mempunyai banyak sekali fungsi yang berkenaan dengannya yaitu penghapusan, penulisan, pembacaan, dan pencetakkan. Data dan fungsi dibangun oleh class untuk membentuk kelas objek (object class). Fungsi yang memanipulasi data-data kelas dikenal sebagai metode. Dalam pemrograman, kelas objek merupakan koleksi data dengan sekumpulan operasi yang memanipulasi data.

Difinisi :

Pemrograman berorientasi objek (OOP) adalah cara untuk melihat program dalam bentuk objek-objek permasalahan yang membentuk sebuah sistem.

Sebenarnya pemrograman OOP dapat menggunakan bahasa apapun, tetapi teknik pemrogramannya akan lebih sulit dibandingkan dengan bahasa yang sudah dilengkapi sarana pemrograman OOP. Bahasa yang "berorientasi-objek" biasanya menyediakan struktur data kelas yang memungkinkan program mengelompokkan data dan metode ke dalam sebuah variabel.

Class dalam C++ dipandang sebagai perluasan dari struktur. Sebuah kelas dapat dipandang sebagai struktur data yang telah diperlengkapi dengan metode. Seperti halnya struktur data, kelas hanya mendeklarasikan type variabel dan tidak mengalokasikan memori untuk suatu variabel. Kelas mempunyai nama (tag) dan anggota (field) serta label yang memberikan ruang lingkup kerja anggota kelas yaitu public, private, dan protected. Cara pendeklarasian kelas seperti mendeklarasikan struktur. Contoh berikut mendeklarasikan kelas segitiga dengan prototipe.

```
.....
class segitiga
{ private:
   int tinggi.lebar,sisi_A,sisi_B;    // anggota data privat
   public:
      segitiga ();      //fungsi konstruktor
      ~segitiga();    //fungsi destruktor
      int luas(int t, int l);              //fungsi anggota public
      int luas_max(int A, int B);   // fungsi anggota public
   };
.....
```

Dalam contoh diatas ditemukan 3 jenis fungsi antara lain fungsi konstruktor, destruktor, dan fungsi anggota. Fungsi konstruktor adalah fungsi yang berguna untuk memulai kelas objek atau instance. Fungsi destruktor adalah fungsi yang berguna untuk menghapus kelas objek. Kedua fungsi diatas secara default diciptakan oleh program pada saat kelas object dimulai dan dihapus pada saat program berakhir. pendeklarasian fungsi konstruktor mempunyai tujuan untuk menginisialisasi variabel dan parameter yang diperlukan oleh fungsi anggota. Pendeklarasian Fungsi destruktor bertujuan untuk mengakhiri kelas objek sebelum program berakhir. Fungsi konstruktor dan fungsi destruktor tidak mengembalikan nilai tetapi tidak perlu mendeklarasikannya sebagai void function.

Dalam konsep pemrograman OOP dikenal tiga istilah penting pemrograman antara lain :

1. **Pengkapsulan (Encaptulation).**

   Difinisi:

   Encaptulation atau pengkapsulan kode mempunyai pengertian menyembunyikan rincian fungsi anggota kelas objek.

   Pemrogram tidak harus tahu bagaimana suatu fungsi melakukan kerjanya, melainkan mengetahui parameter-parameter yang dibutuhkan metode sebagai antarmukanya. Dalam pemrograman C++, untuk menyembunyikan rincian fungsi dasar dapat dilakukan dengan membagi difinisi kelas menjadi bagian privat dan public. Data dan metode public dapat diakses langsung oleh program, sedangkan data dan metode private hanya digunakan oleh metode anggota kelas bersangkutan.

2. **Pewarisan (Inheritance).**

Difinisi:

Pengertian dari inheritence atau pewariasan adalah suatu mekanisme penciptaan kelas baru yang mewarisi sifat atau karakteristik kelas lain yang lebih sederhana.

Difinisi:

Kelas dasar (base class) adalah kelas yang berdiri sendiri dan tidak berasal dari penurunan kelas lain.Kelas dasar biasanya terdiri dari fungsi-fungsi sederhana dan mendasar dan bersifat umum.

Difinisi :

Kelas turunan (derived class) adalah kelas yang mewarisi kelas dasar (base class).

Pewarisan kelas objek bertujuan memperluas dan meningkatkan daya guna program. Konsep pemrograman seperti ini memungkinkan pemrogram mengembangkan suatu kelas objek baru dengan menggabungkan kelas-kelas lain. Pembuatan kelas objek dari dua buah kelas objek atau lebih disebut dengan pewarisan berganda (multiple inheritence).

3. **Multi bentuk (Polymorphism).**

Difinisi :

Polymorphism berasal dari bahasa yunani yang berarti memiliki banyak bentuk. Pengertian polymorphism yaitu bahwa suatu kegiatan/proses memiliki tingkah laku yang berbeda pada objek yang berbeda.

Polymorphism membantu dalam menyederhanakan sintaksis untuk menjalankan kegiatan yang sama pada sekumpulan objek atau hirarki kelas. Contoh sederhana berikut mengilustrasikan pemahaman konsep polymorphism.

//Anggap "bentuk" merupakan kumpulan bentuk-bentuk geometris

```
// dan "luas" adalah fungsi penghitung luas (apapun bentuknya)
for (I=0; i<jumlah_bentuk; I++)
        luas_bentuk = bentuk[i].luas();
```

Terlihat bahwa apapun bentuk geometris objek, masing-masing kelas objek mendukung fungsi luas dan menghitung dalam cara yang sesuai dengan bentuk tersebut.

2.2.2. Bahasa Pemrograman C++

Bahasa C++ dikembangkan oleh Bjarne Stroustrup sebagai kelanjutan bahasa C untuk pemrograman berkonsep object oriented. Kemudian di tahun 1980-an, perusahaan terkenal AT&T mulai mempopulerkan bahasa C++ ke pemrograman berbasis Personal Computer (PC). Pemrograman menggunakan bahasa C++ mulai muncul pada PC di tahun 1988. Dengan semakin meluasnya pemakaian konsep pemrograman object oriented, bahasa C++ menjadi lebih banyak digunakan untuk membuat program-program aplikasi yang profesional. Karena dipandang cukup effisien maka banyak muncul produk-produk compiler seperti Turbo C++ dan Borland C++ (Borland Corp.), Microsoft C++ dan Visual C++ (Microsoft Inc.), Symantech C++, dan Watcom C++.

Bahasa C++ masih membawa sifat pendahulunya yaitu sebagai bahasa untuk pemrograman teknis sehingga bahasa C++ dianggap sebagai bahasa serbaguna. Yang perlu diperhatikan pada bahasa C++ yaitu fungsi-fungsi pemeriksaan program tidak diikutsertakan pada saat jalannya program. Pada bahasa PASCAL, BASIC, dan FORTRAN menyediakan fungsi pemeriksaan yang secara otomatis dituliskan ke dalam executable program sehingga akan memperlambat jalannya program. Pemahaman tentang

perilaku perangkat keras sangat diharapkan agar pemrograman C++ dapat memberikan keunggulannya dibanding bahasa-bahasa lain.

Pernyataan-pernyataan Input/Output dari beberapa bahasa permrograman seperti PASCAL, BASIC, dan FORTRAN, menjerat bahasa-bahasa bersangkutan menjadi bersifat spesifik terhadap perangkat keras. C++ tidak memiliki penyataan masukkan dan keluaran seperti pada bahasa lain. Hal ini merupakan suatu alasan yang penting mengapa C++ lebih fleksibel dalam menangani perangkat keras. Setiap kompiler menyertakan pustaka fungsi-fungsi I/O standart yang bersifat tidak bergantung pada perangkat keras. Masukan dan keluaran dibentuk melalui pemakaian operator yang melimpah dan pemanggilan fungsi-fungsi I/O standart.

Bahasa pemrograman C++ memiliki banyak keistimewaan sebagai bahasa beraras tinggi, begitu pula dalam menangani secara rinci pemrograman yang sama dengan bahasa assembly. Kemampuan inilah yang membuat bahasa C++ mampu mengembangkan diri mengikuti kemajuan tekhnologi perangkat keras komputer.

Mulai standart AT&T 3.0, bahasa C++ memiliki kemampuan tambahan yang disebut template. Kemampuan ini tidak didapatkan pada bahasa pemrograman beroreantasi struktur seperti PASCAL. Untuk memakai template maka source program harus dicompile dengan compiler C++ berstandart AT&T 3.0. Template disebut juga dengan generik atau tipe yang diparameterkan. Ada dua macam template yaitu template fungsi dan template class. Template C++ memberikan kemudahan dalam mereduksi duplikasi fungsi-fungsi yang berbeda dalam parameter dan tipe pengembalian.

2.2.3. Konsep Dasar Pemrograman C++

2.2.3. Konsep Dasar Pemrograman C++

Bahasa pemrograman C++ adalah hasil penyempurnaan dari bahasa C. Pembaharuan arsitektur pemrograman C oleh C++ adalah ditambahkanya konsep permrograman beroreantasi pada objek (Object Oriented Program). Selain itu, kata kunci (Keyword) tambahan juga didifinisikan dalam kompiler C++, antara lain:

asin, catch, class, delete, friend, inline, new, operator, private, protected, public, template, this, dan virtual.

Konsep Permrograman Beroreantasi Objek mulai banyak dipergunakan sekitar tahun 1988. Konsep ini sebenarnya sudah digunakan oleh bahasa pemrograman SIMULA67 dan SMALLTALK. Bahasa Pemrograman C++ dapat dianggap sebagai bahasa yang mengadopsi konsep tersebut sekaligus menyempurnakannya. Istilah Object Oriented Programming sudah cukup dikenal, namun para pakar komputer belum memiliki kesepakatan mengenai difinisinya.

Pemrograman seperti dalam PASCAL dan C merupakan pemrograman dengan pendekatan prosedural (Procedure-Oriented Approach atau POA). Dalam pendekatan prosedural, problem pemrograman dipandang sebagai suatu urutan kegiatan yang perlu dilakukan. Aliran program yang dibuat dengan pendekatan prosedural berawal dari puncak fungsi main dan berakhir pada dasar fungsi main sehingga disebut Top-Down Approach. Kelemahan yang terdapat pada POA adalah perawatan program tidak mudah. Untuk mengadakan penambahan atau penghapusan bagian program penelusuran langkah harus dilakukan lagi pada seluruh tubuh program.

Object Oriented Programming berbeda dengan POA. Dalam OOP, program merupakan sekumpulan objek yang dijadikan fokus permasalahan dan dimanipulasikan

dengan parameter atau argumen. Class adalah pondasi dasar dari OOP, karena berfungsi membangun objek. Keuntungan yang didapat dengan OOP adalah perawatan program menjadi lebih mudah. Dengan OOP penambahan dan penghapusan hanya berkaitan dengan objek dari program sehingga lebih mudah dibandingkan dengan menghapus atau merubah fungsi. Konsep pemrograman OOP dapat menyederhanakan pola pikir programmer dalam perancangan program, karena pemakaian objek membawa permasalahan dapat diuraikan bagian demi bagian.

2.3. ARTIFICIAL INTELLIGENCE (EXPERT SYSTEM)

2.3.1. PENGERTIAN DAN DIFINISI ARTIFICIAL INTELLIGENCE

Difinisi :

Artificial Intelligence (AI) atau kadang disebut sebagai mechine intelligence atau heuristic programming adalah tekhnologi yang digunakan manusia untuk membuat mesin komputer menjadi pintar mampu memberikan tanggapan dan jawaban atas permasalahan non numerik.

Penelitian di dalam AI difokuskan pada pendekatan secara komputasional pola pikir kecerdasan manusia. AI berusaha mengadaptasikan dan mengimplementasikan cara kerja kecerdasan manusia pada mesin komputer untuk menyelesaikan masalah yang rumit dan kompleks, tak menentu, dan membias (ambiguity).

Program-program komputer yang berdasarkan pada AI, mengkonsentrasikan pemecahan masalahnya dengan proses simbolik. Solusi-solusi secara algoritmik tidak dimungkinkan pada AI sehingga proses pencarian lebih ditekankan. Tipe-tipe solusi problem dan pembuatan keputusan lebih berguna di dalam AI, seperti manusia cara

memahami permasalahan dan menentukan keputusan. Konsep yang berbeda terlihat jelas pada pemecahan masalah menggunakan perhitungan di bidang keilmuan (scientific dan engineering yang selalu menggunakan perhitungan numerik) yaitu jawaban dapat dijamin kebenarannya. Sedangkan pada penyelesaian masalah menggunakan AI (lebih cenderung menyelesaikan permasalahan pada pendekatan konsep dan simbolik), jawaban tidak bisa dipastikan selalu benar dengan kesalahan yang ditoleransi seperti manusia memecahkan masalahnya.

Difinisi :

Heuristic adalah metode analisa yang didasarkan pada proses pencarian berdasarkan kode pengenal sekumpulan data.

Cara bekerja program berbasis AI didasarkan pada pencarian heuristic (heuristic search). Jika tidak ada yang terbaik untuk dilakukan komputer, maka program akan mencoba dengan beberapa pendekatan pemecahan masalah. Pada permasalahan yang kompleks, kemungkinan bilangan yang digunakan untuk menunjukkan path dari solusi permasalahan dapat menjadi tak berhingga. Dengan demikian pemecahan masalah pada AI selalu berpedoman pada aturan-aturan empirik dari pola penyelesaian yang dikenal dengan heuristic.

Pemrograman berbasis AI mempunyai perbedaan dengan pemrograman konvensional pada teknik yang digunakan, struktur program, proses dan solusi yang akan dihasilkan. Perbedaan pemrograman berbasis AI dengan pemrograman konvensional dapat dilihat sebagai berikut :

Pemrograman Artificial Intellegence

- Mengutamakan pemrosesan secara simbolik.

- Solusi didapatkan secara implisit dengan pencarian heuristic (Heuristic search).
- Struktur control biasanya dipisahkan dari daerah penggetahuan (domain knowledge).
- Biasanya mudah untuk dimodifikasi, di-update, dan diperbanyak/diperbesar.
- Beberapa kesalahan jawaban dapat ditoleransi.
- Jawaban yang memuaskan selalu dapat diterima.

Pemrograman komputer konvensional

- Selalu mengutamakan penyelesaian secara numerik.
- Solusi didapatkan secara explisit dengan algorithma (Algorithmic).
- Struktur kontrol dan informasi digabungkan.
- Sukar untuk dimodifikasi.
- Jawaban yang benar diharapkan.
- Kemungkinan Solusi terbaik yang selalu dicari.

Program berbasis AI terdiri dari beberapa jenis dan masing-masing mempunyai fungsi dan tujuan yang berbeda, antara lain sebagai berikut :

1. Pemrosesan bahasa alami (Natural Language Processing).

   Program yang dapat melakukan proses penerjemahan bahasa, menjelaskan arti suatu kalimat, dan aplikasi lain yang berhubungan.

2. Pengenalan pola (Computer vision).

   Program yang memungkinkan komputer dapat mengenali bentuk suatu objek/benda.

3. Sistim pakar (Expert system).

   Program yang membuat komputer dapat bertindak seperti pakar dalam suatu domain permasalahan.

4. Pemecahan masalah dan perencanaan (Problem solving and planning).

   Program yang memiliki kegunaan umum sebagai pencari pendekatan terbaik bagi permasalahan yang tidak berpakar. Di dalam program terdapat dasar perencanaan sistem yang dikonsetrasikan pada teknik penyelesaian dengan pengetahuan.

2.3.2. EXPERT SYSTEM

Difinisi :

Expert system adalah suatu program komputer yang mengemulasikan perilaku atau pola pikir para ahli untuk digunakan sebagai referensi dalam menyelesaikan permasalahan.

Pengertian yang terkandung dalam konsep expert system akan kelihatan mudah dan jelas bila pembuatan expert system jauh dari kekompleksitasan permasalahan. Referensi yang diberikan para ahli digunakan bahan dasar untuk mendapatkan pendekatan solusi yang terbaik.

Pembuatan expert system untuk menangani masalah yang kompleks tidak mudah dan sesederhana yang dibayangkan namun dapat disimpulkan sebagai berikut :

1. Penguraian metode dan pengetahuan para pakar untuk menyelesaikan permasalahan.
2. Penyusunan kembali metode/pengetahuan ke dalam bentuk yang terorganisasi untuk kemudian dipakai dalam struktur expert system.

Proses extrasi dan reformasi metode/pengetahuan di atas disebut *knowledge acquisition and representation*. Gabungan dari kedua proses tersebut dikenal sebagai perancangan pengetahuan (knowledge engineering). Biasanya proses ini dilakukan secara terus menerus, berulang-ulang sepanjang terjadi penambahan pengetahuan baru pada basis pengetahuan.

Karakteristik dari expert system yaitu adanya penggunaan heuristic atau aturan sidik jari dalam proses pemanipulasian pengetahuan dan proses inferensial atau proses penarikan kesimpulan. Heuristic sering digunakan oleh manusia untuk memecahkan permasalahan atau membuat suatu keputusan. Heuristic selalu didasarkan pada hasil pengalaman atau dari pengetahuan. Contoh jika suatu permasalahan terjadi pada suatu mobil, dimana mobil tersebut tidak bisa dinyalakan. Untuk mendapatkan jawaban dari permasalahan, maka heuristic akan mencari dari daftar pengalaman/pengetahuan yang ada. Dari pengalaman yang diperoleh bahwa kerusakan mungkin terjadi pada sistim pengapian. Setelah dilakukan pengetesan ternyata sistem pengapian berjalan normal. Kemudian Heuristic akan melanjutkan prosesnya pada kenyataan dan aturan berikutnya hingga didapatkan jawaban yang sesuai. Proses inferensial ditujukan untuk memberikan tanggapan dari hasil pencarian heuristics dan kenyataan yang terjadi. Biasanya pada proses inferensial digunakan suatu strategi respon baik secara deduksi, induksi, maupun abduksi dari permasalahan.

Hasil/jawaban dari proses inferensial tidak selamanya benar sebagai solusi permasalahan. Expert system didesain untuk bekerja pada permasalahan yang penyelesaiannya non-algorithmic. Dengan kata lain, macam dari problem yang menghendaki keahlian manusia.

Expert system terdiri dari empat karakteristik dasar yaitu expertise, depth, Symbolic reasoning, self-knowledge.

Difinisi:

Kepakaran (Expertise) adalah kemampuan yang harus dimiliki expert system sekurang-kurangnya pada level yang sama dengan keahlian manusia/pakar, termasuk juga kemampuan memecahkan permasalahan dalam rentang waktu yang

pendek jika mungkin. Expertise juga berati bahwa pengetahuan tentang domain permasalahan dimiliki oleh expert system harus luas dan mendalam.

Difinisi :

Kedalaman (Depth) mempunyai arti bahwa pengetahuan dalam expert system mempunyai scope pembahasan masalah dan kemampuan memperluas existensi pengetahuan serta menduga pengetahuan baru.

Expert system harus mempunyai pengetahuan yang lebih dalam dan lebih luas dari pada pemakai.

Difinisi :

Pemikiran simbolik (Simbolic reasoning) adalah teknik pemecahan masalah dengan memanipulasi simbol. Simbol dapat berarti kata, tersusun dari huruf, yang mempresentasikan beberapa konsep permasalahan nyata.

Expert system memecahkan permasalahan dengan memanipulasi simbol dan tidak bertujuan melakukan perhitungan matematis atau menyelesaikan dengan suatu persamaan. Kenyataan menunjukkan bahwa memanipulasi simbol bukan berarti sama sekali tidak menggunakan perhitungan matematis, tetapi perhitungan dilakukan jika dibutuhkan. Dengan kata lain perhitungan bukan merupakan konsep utama dalam teknik pemecahan masalah.

Difinisi :

Pengertian pengetahuan diri (Self-knowledge) yaitu bahwa expert system memiliki pengetahuan tentang segala sesuatu yang telah diketahuinya.

Sebagai contoh, sebagian besar expert system mempunyai fasilitas yang dapat menerangkan kepada pemakai bagaimana dan/atau mengapa jawaban/kesimpulan dicapai.

Fasilitas semacam ini merupakan sebagian kecil dari karakteristik self-knowledge. Nantinya expert system akan mempunyai kemampuan merumuskan strategi penarikan kesimpulan agar dioperasikan lebih effisien dan menjadi pertimbangan bagi pengetahuan baru yang didapat dari pengalaman. Karakteristik self-knowledge akan membawa expert system melangkah semakin dekat seperti manusia/pakar.

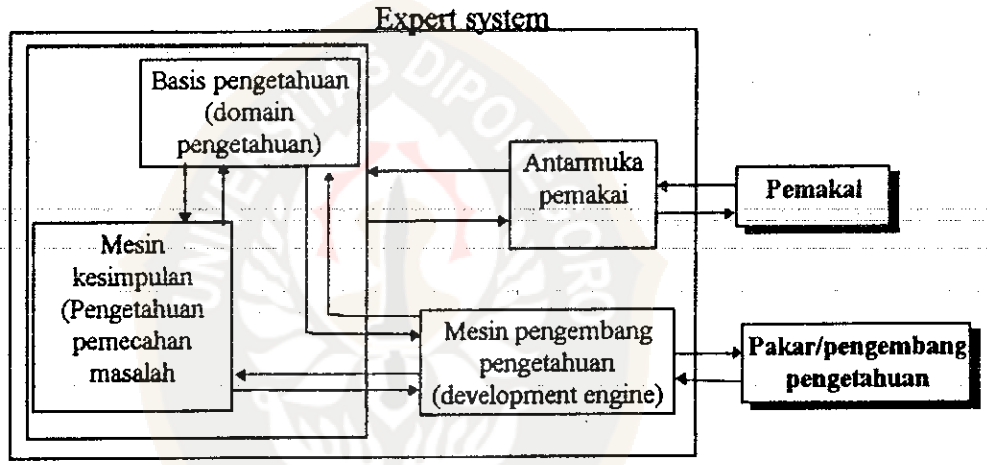
2.3.3. KOMPONEN EXPERT SYSTEM

Expert system pada dasarnya terdiri dari empat komponen penting diantaranya yaitu basis pengetahuan, mesin kesimpulan, antarmuka pemakai, dan mesin pengembang. Basis pengetahuan di dalam expert system terdiri dari fakta dan aturan atau pengambaran lain yang dimiliki sistem tentang domain kenyataan permasalahan. Fakta dan kenyataan baru dapat ditambahkan secara external di suatu waktu dan ditambahkan secara internal selama proses berlangsung bagi fakta dan aturan turunan.

Mesin kesimpulan di dalam expert system melakukan tugas memutuskan bagaimana dan kapan fakta dan aturan digunakan untuk membuat keputusan atau solusi permasalahan. Struktur dari mesin kesimpulan tidak tergantung kepada basis pengetahuan tetapi mungkin berlaku hal yang sama untuk struktur permasalahan.

Antarmuka pemakai di dalam expert system memiliki arti sebagai kommunikator antara pemakai yang menginginkan solusi dengan sistem pemberi solusi (expert system). Penanganan antarmuka ini merupakan hal yang vital yang dapat membuat komunikasi lebih bermakna dan bersahabat. Antarmuka dapat terdiri dari pengatur perintah, menu pengendali atau diorientasikan pada icon (dalam graphical objects).

Kemudian untuk memenuhi karakteristik self-knowledge biasanya expert system dilengkapi dengan kemampuan menambah pengetahuan. Mesin pengembang (Development engine) adalah suatu program yang digunakan oleh para pakar untuk mengembangkan atau menyempurnakan aturan-aturan dan referensi yang ada di dalam basis pengetahuan. Gambar 2.1 memperlihatkan hubungan dari ke-empat komponen tersebut.



Gambar 2.1.