**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Penyakit Ayam Hewan unggas adalah hewan yang memiliki sayap, berkaki dua, bertelur serta tergolong keluarga burung (aves). Hewan unggas yang paling banyak dikembangkan untuk bisnis peternakan adalah jenis ayam. Ternak ayam memiliki keuntungan dan potensi yang tinggi. Selain daging dan telur yang selalu meningkat konsumsinya, ayam sangatlah cepat perputaran bisnisnya. Namun, tidak sedikit kerugian dialami oleh peternak karena ayam rentan terkena penyakit. Penyakit pada ayam dapat disebabkan oleh virus, bakteri, parasit dalam, parasit luar, dan jamur. Macam-macam penyakit yang disebabkan oleh bakteri dan virus pada ayam

Antisipasi untuk mencegah dan mengenali gejala penyakit yang berbahaya sangatlah penting. Proses untuk mengenali dengan cepat dan tepat dari serangan jenis penyakit sangatlah sulit karena gejala yang ditimbulkan umumnya mirip dan sama. Akan tetapi, biasanya ada beberapa gejala yang khas untuk setiap jenis penyakit pada ternak unggas, misalnya pada ayam. Penyakit tersebut dapat disebabkan oleh serangan virus ataupun bakteri. Berikut akan diuraikan penerapan aplikasi sistem pakar yang dapat melakukan diagnosis terhadap penyakit unggas (ayam).

Penelitian ini diharapkan dapat mencegah timbulnya kematian pada unggas dengan memberikan saran pencegahan yang tepat dan rasional.

* 1. **Rumusan Masalah**

Untuk dapat mendiagnosa penyakit pada ayam melalui sistem pakar maka rumusan masalahnya adalah “Bagaimana merancang dan membangun sebuah Sistem Pakar Diagnosis Penyakit Unggas Ayam dan Penanganannya. Dengan Menggunakan Program Visual Prolog”*.*

* 1. **Batasan Masalah**

Agar pembahasan penelitian ini tidak menyimpang dari apa yang telah dirumuskan, maka diperlukan batasan-batasan. Batasan-batasan dalam penelitian ini adalah :

1. Program digunakan untuk mendiagnosis jenis penyakit dari ayam
2. Tidak semua penyakit diberikan solusi berupa vaksin.
3. Gejala yang dibahas adalah gejala yang dialami oleh ayam.
4. Memasukkan inputan sesuai dengan apa yg tertera di akhir contoh : (y/t)
5. Perancangan  program  aplikasi  system  pakar  menggunakan  bahasa pemrograman *Visual Prolog*
   1. **Tujuan dan Manfaat**

Tujuan yang ingin dicapai adalah :

1. Membangun suatu sistem pakar deteksi jenis narkoba yang digunakan pecandu, yang dapat membantu dan mempermudah proses mendeteksi pecandu narkoba tanpa harus pergi ke dokter langsung.
2. Memberikan informasi deteksi candu narkoba.
3. Menerapkan metode system pakar visual prolog.

**BAB II**

**LANDASAN TEORI**

**2.1 Fungsi-Fungsi Visual Prolog yang Digunakan**

Terdapat beberapa fungsi yang digunakan dalam program sistem pakar diagnosis

**2.1.1 Lingkungan Visual Prolog**

**1. Visual Prolog Environment (VPE)**

VPE didesain agar seorang programmer dapat dengan mudah, nyaman dan cepat dalam membangun, menguji dan memodifikasi suatu aplikasi atau program yang ditulis dalam Visual Prolog. VPE memiliki beberapa varian sehingga mendukung untuk digunakan di beberapa platform sistem operasi seperti MS-DOS, MS Windows 3.1, MS Windows 95, MS Windows NT, Win-OS/2, atau OS/2 PM, yang digunakan pada platform prosesor 16 bit ataupun prosesor 32 bit dari keluarga prosesor Intel 80x86 ataupun kompatibelnya seperti prosesor AMD

Pengguna VPE diasumsikan mempunyai pengalaman dan pengetahuan dalam menggunakan sistem GUI (*Graphical User Interface*), seperti menggunakan menu, menutup, meminimize, memaximize ataupun meresize suatu jendela (*window*), meloading file dari jendela File Open dialog, mengklik toolbar dan lainlain. Jika praktikan belum punya pengetahuan ini, praktikan dapat mempelajari dari literature yang berkaitan dengan sistem operasi terkait, seperti buku cara menggunakan MS Windows 95 atau yang lainnya.

**2. Menjalankan VPE**

Sebelum menjalankan VPE, tentu saja, diasumsikan program VPE sudah terinstall di komputer yang digunakan. Program instalasi akan membuat sebuah program group yang di dalamnya terdapat icon yang digunakan untuk menjalankan VPE, yaitu dengan cara mengklik icon tersebut. Namun ada banyak cara untuk menjalankan VPE, seperti, dengan menggunakan Windows Explorer, men-double klik file VIP.EXE di direktori BIN\WIN\16 untuk platform 16 bit atau direktori BIN\WIN\32 untuk platform 32 bit yang direktori tersebut terletak di bawah direktori utama VIP. Jika Visual Prolog telah pernah membuka suatu project (dengan ekstensi .VPR) terakhir kali dan VPE ditutup, maka secara otomatis akan membuka project tersebut ketika VPE dijalankan kembali.

**3. Membuka jendela editor (*editor window*)**

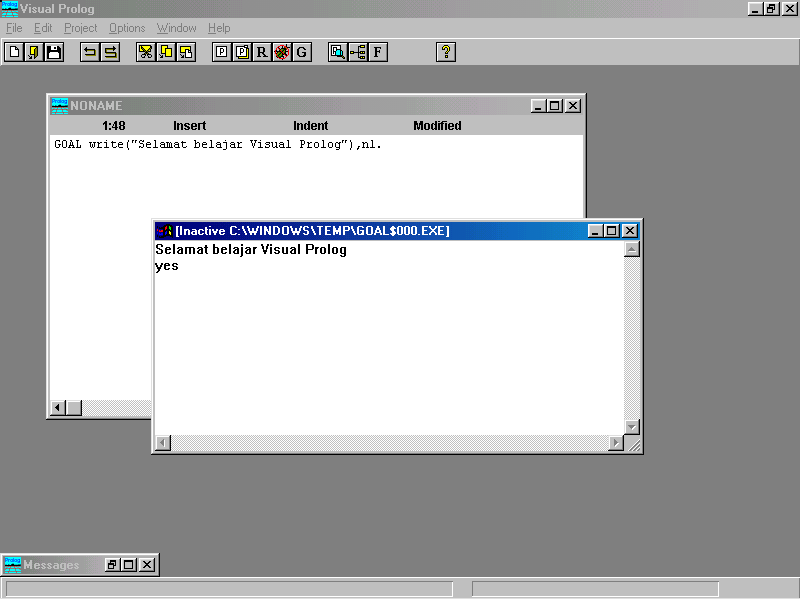
Untuk menciptakan jendela editor yang baru, praktikan dapat menggunakan menu perintah **File | New**. Setelah itu akan muncul jendela editor baru dengan judul “NONAME”. Editor ini layaknya seperti editor teks standar lainnya, seperti NOTEPAD yang dimiliki oleh MS Windows. Praktikan dapat menggunakan tombol kursor (tanda panah atas, bawah, kiri dan kanan) dan mouse untuk menggerakkan kursor seperti layaknya editor lain. Juga editor ini mendukung operasi *cut, copy and* *paste, undo/redo* yang diaktifkan dari menu **Edit** atau melalui penekanan tombol akselerator yang dapat dilihat pada menu **Edit**. Untuk lebih mendalami editor ini lebih jauh praktikan dapat membaca dari manual Visual Development Environment (VDE) pada file VDE.DOC di direktori DOC yang terletak di bawah direktori utama VIP.

**4. Menjalankan dan menguji suatu program**

Untuk mengecek bahwa sistem diset dengan baik, praktikan dapat mencoba mengetikkan teks berikut pada jendela editor:

GOAL write(“Selamat belajar Visual Prolog”),nl.

Baris kode di atas pada Prolog dinamakan GOAL dan baris tersebut telah cukup syarat untuk menjadi program yang bisa dieksekusi. Untuk mengeksekusi GOAL, aktifkan item menu **Project | Test Goal**, atau cukup dengan menekan tombol akselerator **Ctrl+G**. Jika sistem terinstall dengan baik, maka di layar akan tampak seperti gambar berikut:



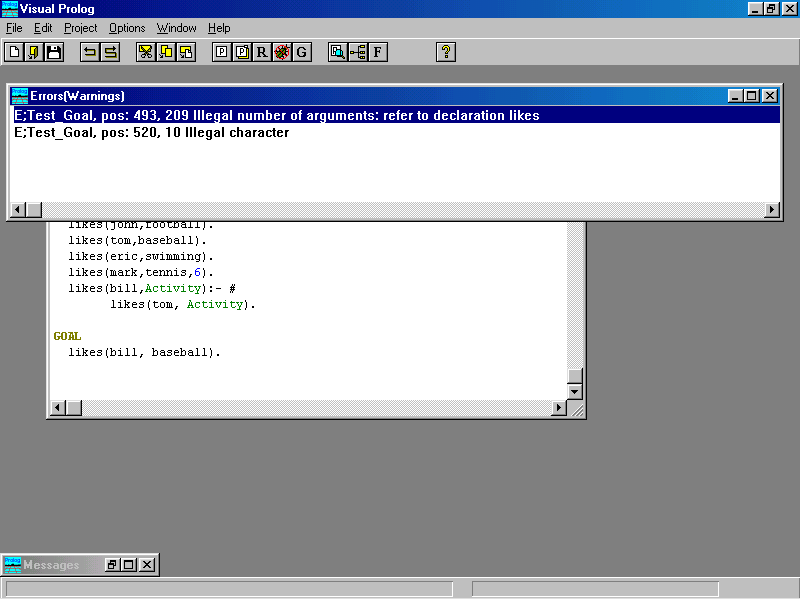
Hasil eksekusi akan ditampilkan pada jendela yang berbeda yang harus ditutup sebelum menguji GOAL lainnya. Visual Prolog memperlakukan GOAL sebagai sebuah program yang telah ter-*compile*, kemudian di-*links* dan dibangkitkan menjadi suatu bentuk jendela yang dapat dieksekusi (*executable*).

**5. Membuka file dari disk**

Contoh-contoh program untuk mendukung manual Language Tutorial Visual Prolog disediakan di direktori /DOC/EXAMPLES pada direktori utama VIP (dengan syarat manual beserta contohnya terinstall pada waktu menginstall Visual Prolog pertama kali). Salah satu file contoh tersebut dapat dibuka dengan menggunakan item menu **File | Open** atau menekan tombol akselerator **F8**. Pilih salah satu file (berekstensi .PRO) dan uji GOAL program tersebut dengan menekan tombol **Ctrl+G**.

**6. Melihat dan memperbaiki kesalahan**

Jika programmer membuat kesalahan dalam menuliskan kode Visual Prolog, maka VPE akan menampilkan jendela kesalahan (*error window*) di mana pada jendela tersebut terdapat daftar kesalahan yang dibuat. Programmer dapat mendouble klik kesalahan tersebut agar kursor beralih ke posisi di mana kesalahan tersebut dibuat pada baris kode di jendela editor, sehingga dengan lebih cepat kesalahan tersebut dapat diperbaiki. Tampilan jendela kesalahan dapat dilihat pada gambar berikut:



**2.1.2 Dasar-dasar Prolog**

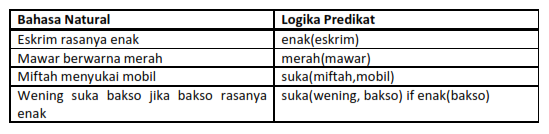
**Pemrograman Logika**

Prolog dikenal sebagai bahasa deskriptif (*descriptive language*), yang berarti dengan diberikan serangkaian fakta-fakta dan aturan-aturan, Prolog, dengan menggunakan cara berpikir deduktif (*deductive reasoning*), akan dapat menyelesaikan permasalahan suatu program. Ini dikontraskan dengan bahasa komputer tradisional seperti C, BASIC, Pascal yang dikenal sebagai bahasa prosedural (*procedural language*). Dalam bahasa prosedural, programmer harus memberikan instruksi tahap demi tahap agar komputer dapat dengan pasti bagaimana menyelesaikan permasalahan yang diberikan. Dengan kata lain, programmer harus tahu lebih dahulu bagaimana cara menyelesaikan permasalahan sebelum diinstuksikan ke komputer. Lain jika dibandingkan dengan programmer Prolog. Programmer Prolog hanya membutuhkan deskripsi/gambaran permasalahan, lalu menerjemahkannya ke bahasa Prolog. Selanjutnya tinggal sistem Prolog yang menentukan bagaimana mencari solusinya.

Prolog didasarkan pada klausa-klausa Horn (Horn *clauses*), yang merupakan himpunan bagian dari sistem formal yang dinamakan logika predikat (*predicate* *logic*). Logika predikat menyederhanakan cara agar jelas bagaimana berpikir akan dilakukan. Prolog menggunakan variasi sintak logika predikat yang telah disederhanakan dengan demikian sintaknya mudah dimengerti dan sangat mirip dengan bahasa natural.

Prolog mempunyai mesin inferensi (*infrence engine*) yang merupakan suatu proses berpikir logis mengenai informasi. Mesin inferensi mempunyai pencocok pola (*pattern matcher*) yang akan mengambil informasi yang telah disimpan (diketahui) dan kemudian mencocokkan jawaban atas pertanyaan. Satu *feature* penting dari Prolog adalah bahwa, sehubungan mencari jawaban logis atas pertanyaan yang diajukan, ia dapat berhubungan dengan banyak alternatif dan mencari semua kemungkinan dari pada hanya satu solusi.

Logika predikat dibangun agar mudah menerjemahkan ide-ide berbasis logika menjadi bentuk tertulis. Prolog mengambil keunggulan dari sintak ini untuk membangun suatu bahasa pemrograman yang berbasis logika. Dalam logika predikat, pertama kali harus membuang semua kata-kata yang tidak dibutuhkan dari suatu kalimat. Kemudian mentransformasi kalimat tersebut dengan mencari relasi terlebih dahulu, kemudian setelah itu melakukan pengelompokkan *object*. *Object* kemudian menjadi argumen dari relasi atas *object* tersebut. Berikut ini merupakan contoh kalimat bahasa natural ditransformasikan menjadi sintak logika predikat:



**a. Kalimat : Fakta dan Aturan**

Programmer Prolog mendefinisikan *object-object* dan relasi-relasi, kemudian mendefinisikan aturan mengenai kapan relasi-relasi ini dikatakan benar (*true*). Contoh kalimat :

Wening suka bakso.

Memperlihatkan relasi antara *object* Wening dan bakso, relasi ini adalah suka. Berikutnya membuat aturan kapan Wening suka bakso adalah kalimat yang benar:

Wening suka bakso jika bakso rasanya enak.

**Fakta : Apa yang diketahui**

Dalam Prolog, relasi antara *object-object* dinamakan **predikat**. Dalam bahasa natural relasi disimbolisasikan oleh suatu kalimat. Dalam logika predikat yang Prolog gunakan, suatu relasi adalah kesimpulan dalam suatu frase sederhana. Suatu fakta memiliki nama relasi diikuti *object* atau *object-object* di dalam tanda kurung. Sebagaimana kalimat, fakta tersebut diakhiri dengan tanda titik (.). Berikut ini beberapa fakta yang mengekspresikan relasi *suka* dalam bahasa natural:

Toni suka Inung.

Inung suka Toni.

Toni suka kucing.

Dan dalam sintak Prolog, fakta di atas ditulis:

suka(toni,inung).

suka(inung,toni).

suka(toni,kucing).

Fakta juga bisa mengekspresikan sifat (*property*) dari suatu *object* sebagaimana suatu relasi. Dalam bahasa natural kalimat “Kodok berwarna hijau” dan “Komeng adalah laki-laki” diubah ke dalam sintak Prolog menjadi:

hijau(kodok).

lakilaki(komeng).

**Aturan: Apa yang dapat disimpulkan dari fakta yang ada**

Aturan membuat kita dapat mengambil suatu fakta dari fakta yang lain. Dengan bahasa lain, aturan adalah suatu konklusi diketahui benar jika satu atau lebih konklusi atau fakta lain ditemukan benar. Berikut ini beberapa aturan yang berhubungan dengan relasi suka:

Inung suka apapun yang Toni suka.

Komeng suka apapun yang berwarna hijau.

Dengan aturan di atas, dapat disimpulkan dari beberapa fakta sebelumnya mengenai sesuatu yang Inung dan Komeng suka:

Inung suka Inung.

Komeng suka Kodok.

Untuk menuliskan aturan tersebut dalam sintak Prolog hanya dibutuhkan sedikit perubahan seperti ini:

suka(inung, Sesuatu):- suka(toni, Sesuatu).

suka(komeng, Sesuatu):- hijau(Sesuatu).

Simbol :- diucapkan dengan “jika” dan memisahkan dua bagian dari aturan yaitu *head* dan *body*. Aturan di atas dalam makna prosedur, bisa berarti “untuk membuktikan Inung suka sesuatu, buktikanlah bahwa Toni suka hal yang sama” dan “untuk membuktikan Komeng suka sesuatu, buktikanlah bahwa sesuatu itu berwarna hijau”.

**b. Query**

Kalau kita sudah memberikan Prolog sekumpulan fakta, selanjutkan dapat diajukan pertanyaan sehubungan dengan fakta-fakta. Ini dikenal dengan nama memberikan query ke sistem Prolog (*querying the Prolog system*). Pertanyaan yang diajukan ke Prolog sama tipenya seperti dalam bahasa natural. Berdasarkan pada fakta dan aturan yang diberikan sebelumnya.

Sintak Prolog tidak berubah ketika mengajukan query dan tampak bahwa sintak query sangat mirip dengan sintak fakta. Yang penting untuk diingat adalah bahwa *object* kedua Apa diawali dengan huruf besar sedang *object* pertama toni tidak. Ini karena toni sudah *fix* sebagai konstanta *object* sedangkan Apa adalah sebuah variabel. Variabel selalu diawali dengan sebuah huruf besar atau sebuah garis bawah (*underscore, \_* ).

Prolog selalu mencari jawaban dari suatu query dengan memulai dari bagian paling atas (*top*) dari fakta-fakta dan melihat setiap fakta sampai ke bagian paling bawah (*bottom*) dimana tidak ada satupun fakta lagi.

**c. Variabel**

Seperti yang telah disebutkan sebelumnya, untuk memberikan nama variabel dalam Visual Prolog harus diawali dengan huruf besar (*capital letter*) atau garis bawah (*underscore*), berikutnya dapat berupa huruf (besar atau kecil), angka (“0-9”) dan garis bawah (“\_”). Berikut ini merupakan contoh penamaan

**d. Komentar**

Programmer yang baik selalu memberikan catatan atau komentar untuk menjelaskan sesuatu yang mungkin tidak jelas bagi orang lain (atau bahkan bagiprogrammer sendiri dalam setahun ke depan misalnya). Komentar akan membuat program menjadi lebih mudah untuk dimengerti. Komentar dengan baris jamak harus dimulai dengan karakter /\* (*slash,* *asterik*) dan diakhiri dengan karakter \*/ (*asterik, slash*). Untuk memberikan komentar dengan baris tunggal dapat menggunakan karakter yang sama atau dapat dimulai dengan tanda persen (%). Contoh:

/\* Ini contoh sebuah komentar \*/

% Ini juga contoh komentar

/\* Ini komentar

dengan

tiga baris \*/

/\* Ini contoh komentar /\* dalam komentar \*/ di Visual Prolog \*/

2.1.3 Program Visual Prolog

**1. Section dasar Visual Prolog**

Secara umum, program Visual Prolog terdiri dari empat section dasar, yaitu section **clauses**,section **predicates**, section **domains**, dan terakhir section **goal**. Berikut akan dijelaskan secara singkat masing-masing section tersebut.

* **Section Clauses**

Section clauses merupakan section yang paling penting pada program Visual Prolog. Pada section inilah kita meletakkan fakta dan aturan. Ketika mencari jawaban, Visual Prolog akan mencari dari bagian paling atas dari section clauses, melihat setiap fakta dan aturan untuk mendapat jawaban benar, hingga ke bagian paling bawah dari section ini.

* **Section Predicates**

Sebelum mendefinsikan predikat di section clauses, maka predikat tersebut harus dideklarasikan terlebih dahulu di section predicates. Kalau tidak, Visual Prolog tidak akan mengenal predikat yang kita tuliskan tersebut. Ketika mendeklarasikan suatu predikat, kita memberitahu Visual Prolog domain dari argumen yang dimiliki predikat tersebut.

Visual Prolog mempunyai perpustakaan predikat yang kalau dipakai tidak perlu dideklarasikan, karena sudah *built-in*. Untuk melihat predikat apa saja serta manfaatnya yang ada di perpustakaan Visual Prolog dapat melihat *help* dari Visual Prolog (**Help | Contents**).

* **Section Domains**

Section domains mempunyai 2 manfaat utama, yaitu pertama, kita dapat memberikan nama yang berarti untuk domain, walaupun secara internal domaintersebut sama tipenya dengan domain yang telah ada; yang kedua, kita dapatmendeklarasi domain khusus yang digunakan untuk mendeklarasikan struktur datayang tidak didefinisikan oleh standar domain. Dengan mendeklarasikan domain jugadapat mencegah kesalahan logika pada program.

* **Section Goal**

Secara esensial, section **goal** sama dengan *body* dari sebuah aturan (*rule*), yaitu sederetan sub-sub goal. Perbedaan antara section **goal** dengan suatu aturan adalah setelah kata kunci **goal** tidak diikuti tanda :- dan Visual Prolog secara otomatis mengeksekusi goal ketika program dijalankan.

**2. Section Program Lainnya.**

Ada beberapa section lainnya yang digunakan di Visual Prolog yaitu section **facts**, section **constants**, dan section **global**. Kali ini akan dibahas secara singkat ketiga section tersebut sebagai perkenalan. Section **facts** akan dibahas lebih mendalam pada modul yang lain.

* **Section Facts**

Program Visual Prolog merupakan suatu koleksi dari sekumpulan fakta dan aturan. Kadang ketika program sedang berjalan, kita ingin meng-*update* (merubah, menambah, atau menghapus) beberapa fakta dari program. Pada kasus ini fakta menjadi suatu database yang dinamis atau database internal, dan fakta tersebut dapat berubah ketika program sedang berjalan. Visual Prolog menyediakan section khusus untuk mendeklarasikan fakta di program yang menjadi bagian dari database dinamis, yaitu section **facts**.

Kata kunci **facts** untuk memulai section **facts**. Visual Prolog menyediakan sejumlah predikat *built-in* yang mempermudah penggunaan section fakta dinamis ini. Kata kunci **facts** dapat digantikan dengan kata kunci **database**, untuk maksud yang sama.

* **Section Constants**

Konstanta simbolis dapat digunakan di program Visual Prolog. Untuk itu sebelumnya harus dideklarasikan terlebih dahulu. Deklarasi konstanta diakukan pada section **constants**, diikuti dengan deklarasi menggunakan sintak :

<id> = <Macro\_definition>

<id> adalah nama dari konstanta simbolis dan <Macro\_definition> adalah apa yang akan diisikan (assign) kedalam konstanta. Setiap <Macro\_definition> diakhiri dengan baris baru. Dengan demikian hanya ada satu deklarasi konstanta pada tiap barisnya.

* **Section Global**

Visual Prolog memperbolehkan untuk mendeklarasikan beberapa domain, predikat dan klausa menjadi global (daripada hanya lokal). Caranya dengan menset secara terpisah section **global domains**, **global predicates**, dan **global facts** pada bagian paling atas dari program. Modul ini bukan tempatnya untuk membahas secara detail mengenai section **global**.

**3*. Compiler Drectives***

Visual Prolog mendukung *compiler directives* yang dapat ditambahkan ke badan program untuk memberitahukan ke komputer bagaimana secara spesifik memperlakukan kode-kode waktu di-*compile.* Untuk menset *compiler* *directives* sebagian besar dilakukan melalui menu **Options | Project | Compiler Options**. Modul ini tidak akan membahas secara detil mengenai compiler directives, namun akan memperkenalkan salah satu diantaranya yaitu include directive.

Kalau sudah terbiasa membuat program menggunakan Visual Prolog, seringkali kita memakai suatu prosedur tertentu berulang kali, sehingga setiap kali membuat program baru dan menggunakan prosedur tersebut, prosedur tersebut harus diketikkan kembali. Untuk menghemat waktu, maka dapat digunakan include.

**4. Aritas jamak (*multiple arity*)**

Aritas (*arity*) suatu predikat adalah jumlah argumen yang ada pada predikat tersebut. Visual Prolog memperbolehkan kita mempunyai 2 atau lebih predikat dengan nama yang sama namun dengan aritas yang berbeda. Aritas yang berbeda dari nama predikat yang sama harus dikelompokkan bersama baik pada section **predicates** maupun pada section **clauses**. Perbedaan aritas oleh Visual Prolog akan diperlakukan secara berbeda pula.

**5. Sintak Aturan (*Rule Syntax*)**

*Rule* pada Prolog adalah ketika kebenaran sebuah fakta tergantung pada kesukesan (kebenaran) dari satu atau lebih fakta yang lain. Seperti yang telah dijelaskan pada modul sebelumnya aturan terdiri dari 2 bagian yaitu *head* dan *body*. Berikut ini merupakan aturan generik penulisan sintak *rule* pada Visual Prolog:

HEAD:- <subgoal>, <subgoal>, ..., <subgoal>.

Bagian *body* dari *rule* terdiri dari satu atau lebih subgoal. Setiap subgoal dipisahkan oleh koma, menspesifikasikan konjungsi, dan subgoal terakhir diakhiri dengan tanda titik.

Untuk membuat suatu *Rule* dikatakan sukses (benar), Prolog harus memeriksa kebenaran dari setiap subgoal yang ada pada aturan tersebut. Jika ada subgoal yang gagal (salah), Prolog akan kembali ke atas dan mencari alternatif bagi subgoal yang paling awal, kemudian kembali memproses dengan nilai variabel yang berbeda. Ini dinamakan lacakbalik (*backtracking*). Penjelasan yang lebih rinci mengenai lacakbalik akan diberikan pada modul yang lain.

**6. Konversi Tipe Otomatis (*Automatic Type Conversions*)**

Ketika Visual Prolog mencocokkan 2 variabel, keduanya tidak selalu berasal dari domain yang sama. Juga kadang variabel diikat (*bound*) menjadi konstan dari domain lain. Percampuran domain ini diperbolehkan karena Visual Prolog melakukan konversi tipe otomatis dengan syarat konversi bisa terjadi bila:

* Antara **strings** dan **symbols.**
* Antara semua domain **integer** dan juga **real**. Ketika suatu karakter (**char**) dikonversikan ke nilai numeris, angka nilai ASCII dari karakter tersebut yang digunakan.

**2.1.4 Unifikasi Dan Lacakbalik (Unification And Backtracking)**

**1. Unifikasi (*Unification*)**

Pada waktu Visual Prolog mencoba untuk mencocokkan suatu panggilan (dari sebuah subgoal) ke klausa (pada section **clauses**), maka proses tersebut melibatkan suatu prosedur yang dikenal dengan unifikasi (*unification*), yang mana berusaha untuk mencocokkan antara struktur data yang ada di panggilan (subgoal) dengan klausa yang diberikan. Unifikasi pada Prolog mengimplementasikan beberapa prosedur yang juga dilakukan oleh beberapa bahasa tradisional seperti melewatkan parameter, menyeleksi tipe data, membangun struktur, mengakses struktur dan pemberian nilai (*assignment*). Pada intinya unifikasi adalah proses untuk mencocokkan dua predikat dan memberikan nilai pada variabel yang bebas untuk membuat kedua predikat tersebut identik. Mekanisme ini diperlukan agar Prolog dapat mengidentifikasi klausa-klausa mana yang dipanggil dan mengikat (*bind*) nilai klausa tersebut ke variabel.

**2. Lacakbalik (*Backtracking*)**

Pada waktu menyelesaikan masalah, seringkali, seseorang harus menelusuri suatu jalur untuk mendapatkan konklusi yang logis. Jika konklusi ini tidak memberikan jawaban yang dicari, orang tersebut harus memilih jalur yang lain. Perhatikan permainan *maze* berikut. Untuk mencari jalan keluar dari *maze*, seseorang harus selalu mencoba jalur sebelah kiri terlebih dahulu pada setiap percabangan hingga menemukan jalan buntu. Ketika menemukan jalan buntu maka orang tersebut harus kembali ke percabangan terakhir (*back-*up) untuk mencoba lagi (*try again*) ke jalur kanan dan jika menemukan percabangan lagi maka tetap harus mencoba jalur kiri terlebih dahulu. Jalur kanan hanya akan sekali-sekali dipilih. Dengan begitu orang tersebut akan bisa keluar dari *maze*, dan memenangkan permainan. Metode balik-ke-atas-dan-coba-lagi (*backing-up-and-trying-again*) ini pada Visual Prolog disebut lacakbalik (*backtracking*). Visual Prolog menggunakan metode ini untuk menemukan suatu solusi dari permasalahan yang diberikan.Visual Prolog dalam memulai mencari solusi suatu permasalahan (atau goal) harus membuat keputusan diantara kemungkinan-kemungkinan yang ada. Ia menandai di setiap percabangan (dikenal dengan titik lacak balik) dan memilih subgoal pertama untuk telusuri. Jika subgoal tersebut gagal (ekivalen dengan menemukan jalan buntu), Visual Prolog akan lacakbalik ke titik lacakbalik (*back-tracking point*) terakhir dan mencoba alternative subgoal yang lain.

**3. Pengendalian proses lacakbalik**

Mekanisme lacak balik dapat menghasilkan pencarian yang tidak perlu, akibatnya program menjadi tidak efisien. Misalnya adanya beberapa jawaban yang muncul ketika kita hanya membutuhkan solusi tunggal dari masalah yang diberikan. Pada kasus lain, suatu kebutuhan untuk memaksa Visual Prolog untuk melanjutkan mencari jawaban tambahan walaupun goal tersebut sudah terpenuhi. Untuk kasuskasus tersebut, kita harus mengontrol proses lacakbalik.

Visual Prolog menyediakan 2 alat yang memperbolehkan kita untuk mengendalikan mekanisme lacak balik yaitu predikat ***fail*** yang digunakan untuk memaksa lacakbalik dan predikat ***cut*** (ditandai dengan **!**) yang digunakan untuk mencegah lacakbalik.

**Predikat *fail***

Visual Prolog akan memulai lacakbalik jika ada panggilan yang gagal. Pada situasi tertentu, ada kebutuhan untuk memaksa lacakbalik dalam rangka mencari alternative solusi. Visual Prolog menyediakan predikat khusus ***fail*** untuk memaksa kegagalan sehingga memicu terjadinya lacakbalik. Efek dari ***fail*** sama dengan efek memberikan perbandingan 2=3 atau subgoal yang tidak mungkin (*impossible*) lainnya.

**Predikat *Cut* (!)**

Visual Prolog memiliki *cut* yang digunakan untuk mencegah lacakbalik, ditulis berupa sebuah tanda seru (!). Efek dari *cut* adalah sederhana, yaitu tidak akan memungkinkan terjadinya lacakbalik melewati sebuah *cut*. Kita menempatkan *cut* dalam program sama persis seperti menempatkan sebuah subgoal pada *body* dari suatu *rule*. Ketika proses melewati *cut*, pemanggil ke *cut* dinyatakan sukses dan subgoal berikutnya (jika ada) dipanggil. Sekali sebuah *cut* dilewati, adalah menjadi tidak mungkin untuk melakukan lacakbalik pada subgoal yang berada pada sebelum *cut* pada klausa yang sedang diproses dan adalah menjadi tidak mungkin untuk melakukan lacakbalik ke predikat lain yang mendefinisikan predikat yang sekarang diproses (predikat yang mengandung *cut*).

**4. Predikat *Not***

Program berikut memperlihatkan bagaimana penggunaan predikat ***not*** untuk mengidentifikasikan seorang mahasiswa teladan yaitu mahasiswa yang mempunyai Indeks Prestasi Kumulatif (IPK) minimal 3,5 dan tidak sedang dalam masa percobaan (sedang menjalani masa hukuman karena melakukan tindak kejahatan).

Ada satu catatan ketika menggunakan ***not*** yaitu predikat ***not*** akan sukses ketika subgoal tidak bisa dibuktikan kebenarannya. Hal ini untuk mencegah suatu situasi variabel yang belum diikat akan diikat menggunakan ***not***. Jika subgoal dengan variable bebas dipanggil melalui ***not***, maka Visual Prolog akan mengeluarakan pesan kesalahan Free variables not allowed in ‘not’ or ‘retractall’.

**2.1.5 Data Object Sederhana Dan Jamak (Simple Dan Compound Data Object)**

**1. Data *Object* sederhana**

Data *object* sederhana terdiri dari 2 yaitu variabel atau konstanta. Konstanta yang dimaksud tidak sama dengan konstanta simbolis yang ditulis di *section* **constants** pada bagian program. Yang dimaksud dengan konstanta di sini adalah apapun yang diidentifikasikan sebagai sebuah *object* bukan *subject* yang nilainya bisa bervariasi, seperti sebuah karakter (**char**), angka (integer atau real) atau sebuah **atom** (symbol atau string).

1. **Variabel**

Variable harus dimulai dengan sebuah huruf kapital (A ..Z) atau sebuah *underscore* ( \_ ). Sebuah *underscore* tunggal merepresentasikan sebuah variable anonim. Variabel dalam prolog bersifat lokal bukan global, oleh karena itu jika ada dua klausa yang mengandung sebuah variabel X maka X pada kedua klausa tersebut adalah variabel yang berbeda.

1. **Konstanta**

Konstanta meliputi karakter, angka, dan atom. Suatu nilai konstanta juga merupakan nama dari konstanta tersebut. Konstanta 2 merepresentasikan angka 2 dan konstanta halo merepresentasikan simbol halo.

* **Karakter**

Karakter bertipe **char**, yaitukarakter-karakter yang bisa tercetak (ASCII 32 – 127), karakter angka (0 – 9), huruf kecil (a – z), huruf kapital (A – Z) dan tanda baca. Konstanta karakter ditulis dengan diapit oleh tanda petik tunggal (‘). Contoh : ‘a’, ‘\*’, ‘{‘, ‘W’. Jika kita menginginkan sebuah backslash atau petik tunggal menjadi karakter menuliskannya harus didahului dengan sebuah tanda backslash. Contoh : ‘\\’ (backslash), ‘\’’ (tanda petik tunggal). Beberapa karakter mempunyai fungsi khusus ketika didahului oleh karakter escape ( \ ). Contoh : ‘\n’ (ganti baris), ‘\t’ (tabulasi). Konstanta karakter dapat juga ditulis berdasarkan kode ASCII-nya, dengan didahului backslash. Contoh : ‘\64’ (@), ‘\90’ (Z)

* **Angka**

Angka merupakan salah satu dari integer atau real.

* **Atom**

Sebuah atom terdiri dari sebuah simbol atau sebuah string. Perbedaan antara simbol dan string berkaitan dengan representasi dan implementasi mesin. Visual Prolog melakukan konversi tipe otomatis diantara domain string dan domain simbol. Jadi simbol dapat digunakan untuk domain string dan string dapat digunakan untuk domain simbol. Namun string dinyatakan sebagai sesuatu yang berada diantara tanda petik ganda sedang simbol tidak membutuhkan tanda petik ganda. Simbol dimulai dengan sebuah huruf kecil dan hanya boleh berisikan huruf, angka, dan *underscore.* String adalah sesuatu yang diapit tanda petik ganda dan berisikan kombinasi dari karakter, kecuali ASCII NULL (0), yang dipakai untuk menandai akhir dari string.

**2. Data object jamak**

Data object jamak memperlakukan beberapa informasi sebagai sebuah item tunggal. Contohnya : tanggal 1 desember 1999. Tanggal tersebut terdiri dari 3 jenis informasi yaitu hari, bulan dan tahun.

Penulisan ini mirip dengan penulisan suatu fakta tetapi ini bukan fakta. Ini adalah sebuah data object. Data object dimulai dengan sebuah nama yang biasa disebut functor (dalam contoh adalah tanggal) yang diikuti oleh 3 argumen. Sebuah functor dalam Visual Prolog tidak seperti sebuah fungsi pada bahasa pemrograman lain. Functor tidak melakukan apa-apa, hanya sebuah nama yang mengidentifikasi sebuah jenis data object jamak yang didalamnya terdapat argumen. Argumen dari sebuah data object jamak bisa dalam bentuk data object jamak pula.

**3. Deklarasi Domain-Campuran Jamak (*Compound Mix-Domain*)**

Deklarasi domain-campuran jamak bermaksud :

* memiliki sebuah argumen dengan kemungkinan lebih dari satu tipe argumen;
* memiliki beberapa macam argumen, masing-masing dengan tipe yang berbeda;
* memiliki beberapa macam argumen, beberapa diantaranya dengan kemungkinan lebih dari satu tipe argumen.

Agar suatu predikat Visual Prolog dapat menerima suatu argumen yang memberikan informasi dengan tipe yang berbeda maka functor tersebut harus dideklarasikan.

**2.1.6 Perulangan Dan Rekursi (Repetition And Recursion)**

**A. Perulangan dan Rekursi**

Komputer memiliki bermacam kemampuan yang berguna salah satunya adalah kemampuan melakukan sesuatu berulang-ulang. Prolog dapat melakukan perulangan dalam dua hal yaitu berupa prosedur dan struktur data. Ide dari struktur data repetitif (rekursif) adalah bagaimana menciptakan struktur data yang ukuran (*size*) akhirnya belum diketahui ketika struktur tersebut pertama kali dibuat (*create*).

**1. Proses Perulangan**

Prolog menyediakan dua jenis perulangan yaitu lacakbalik (mencari jawaban jamak dari satu pertanyaan) dan rekursi (prosedur pemanggilan dirinya sendiri).

* **Lacakbalik**

Ketika suatu prosedur melakukan lacakbalik, prosedur akan mencari alternatif jawaban dari sebuah goal ayang sudah terpenuhi. Lacakbalik merupakan salah satu cara untuk melakukan proses perulangan.

* **Implementasi Lacakbalik dengan Loop**

Lacakbalik merupakan cara yang baik untuk mencari alternatif jawaban dari sebuah goal. Namun jika suatu goal tidak memiliki alternatif jawaban, lacakbalik masih dapat digunakan untuk melakukan perualangan. Berikut ini didefiniskan predikat duaklausa.

ulang.

ulang:-ulang.

Ini untuk mengakali struktur kendali Prolog agar berpikir bahwa terdapat sejumlah jawaban berbeda yang tak terbatas (cara kerjanya akan dibahas pada bagian mengenai rekursi ekor / *tail recursion*). Kegunaan ulang adalah agar lacakbalik terjadinya tak terhingga).

* **Rekursi**

Cara lain untuk melakukan perulangan adalah melalui rekursi. Prosedur rekursi adalah prosedur yang di dalamnya ada pemanggilan terhadap dirinya sendiri. Prosedur rekursi dapat merekam perkembangannya karena ia melewatkan (*passing*) pencacah, total, dan hasil sementara sebagai argumen dari satu iterasi ke iterasi berikutnya.

* **Rekursi Ekor (*Tail Recursion*)**

Rekursi mempunyai kelemahan yaitu memakan memori. Ketika suatu prosedur memanggil dirinya, keadaan pemanggil prosedur dari eksekusi harus disimpan sehingga prosedur pemanggil dapat meresume keadaan tersebut setelah prosedur pemanggil selesai. Ini berarti jika ada suatu prosedur memanggil dirinya 100 kali, maka ada 100 keadaan dari eksekusi yang harus disimpan. Keadaan (*state*) yang disimpan tersebut dikenal dengan nama *stack frame*. Ukuran *stack* maksimum pada platform 16 bit, seperti IBM-PC dengan sistem operasi DOS, adalah 64KByte yang bisa mengandung sekitar 3000 atau 4000 *stack frame*. Pada paltform 32 bit, secara teoritis bisa sampai ukuran Giga Byte.

Untuk mengatasi kelemahan tersebut, maka digunakan optimasi rekursi ekor (*tail recursion optimization*). Diumpamakan, selain memanggil prosedur C, prosedur B memanggil dirinya sendiri pada langkah terakhir. Ketika prosedur B memanggil B, *stack* *frame* dari pemanggilan B akan ditimpa nilainya oleh sebuah *stack frame* dari pemanggil B, jadi tidak menambah *stack frame* baru. Hanya argumen yang perlu di-*update* nilainya dan kemudian proses akan melompat ke awal prosedur. Dalam perspektif procedural adalah sama seperti memperbaharui variabel pengendali perulangan. Syarat dari rekursi ekor adalah :

- Pemanggil merupakan subgoal terakhir dari klausa tersebut.

- Tidak ada titik lacak balik sebelumnya pada klausa.

**2. Struktur Data Rekursif**

Tidak hanya *rule* yang bisa rekursif tapi juga struktur data. Prolog merupakan satu-satunya bahasa pemrograman yang digunakan secara luas yang memperbolehkan mendefinisikan tipe struktur data rekursif. Salah satu tipe struktu data rekursif yaitu struktur data pohon (*tree*).

**2.1.7 List**

Pada Prolog, yang dimaksud dengan *list* adalah sebuah *object* yang didalamnya mengandung sejumlah *object* yang lain (jumlahnya dapat berubah-ubah). *List* dalam bahasa pemrograman lain bisa disamakan dengan tipe data *pointer* (C dan Pascal). Berikut ini cara penulisan *list* pada Prolog.

[ 1, 2, 3 ] /\* list yang mengandung integer 1, 2 dan 3 \*/

[ kucing, anjing, tikus ] /\* list yang terdiri dari 3 buah symbol \*/

[ “Syarif Musadek”, “Yusida Andriani”, “Diana Putri” ]

/\* list yang terdiri dari 3 buah string \*/

Tanda asterik (\*) berarti domain tersebut merupakan sebuah *list*. Tanda asterik juga dipakai di bahasa C untuk pendeklarasian tipe data *pointer*. Pada Pascal pendeklarasian *pointer* menggunakan tanda ^. Elemen *list* bisa berupa apapun, termasuk suatu *list* yang lain, namun semua elemen dari suatu *list* harus berasal dari domain yang sama.

***Head* dan *Tail* (Kepala dan Ekor)**

*List* adalah suatu data *object* jamak rekursif (*recursive compound object*). *List* terdiri dari 2 bagian yaitu *head*, yang merupakan elemen pertama dari *list* dan *tail*, elemen sisanya. *Tail* dari *list* adalah juga merupakan sebuah *list*, sedangkan *head* dari *list* merupakan sebuah elemen.

* + 1. **Section Facts**

*Section facts* terdiri dari fakta-fakta yang mana fakta-fakta tersebut dapat ditambah dan dihapus secara langsung dari sebuah program pada saat program sedang berjalan (*at run* *time*). Kita dapat mendeklarasikan sebuah predikat pada *section facts* dan predikat tersebut dapat digunakan sama halnya seperti kalau dideklarasikan pada *section predicates*.Visual Prolog menyediakan beberapa predikat *built-in* untuk menangani hal yang berkaitan dengan penggunaan *section facts*, antara lain :

* assert, asserta dan assertz untuk menambah fakta baru pada *section facts*.
* retract dan retractall untuk menghapus fakta yang ada.
* consult untuk membaca fakta dari sebuah file dan menyertakan fakta tersebut ke dalam fakta internal.
* save menyimpan isi fakta internal ke dalam sebuah file.

* **Deklarasi Section Facts**

Kata kunci facts atau bisa juga database menandai permulaan sederetan deklarasi dari predikat yang ada pada section facts. Kita dapat menambahkan fakta-fakta (bukan rule) pada suatu section facts dari keyboard pada saat run time dengan menggunakan asserta dan assertz atau memanggil predikat consult untuk mengambil fakta tambahan dari sebuah file.

* **Menambah fakta pada saat *run time***

Pada saat *run time*, fakta-fakta dapat ditambah ke *section* *facts* dengan menggunakan predikat assert, asserta dan assertz atau me-*load* sebuah file yang berisikan fakta menggunakan predikat consult. Cara penulisannya adalah sebagai berikut:

* asserta(<fakta>[, nama\_section\_facts])
* assertz(<fakta>[, nama\_section\_facts])
* assert(<fakta>[, nama\_section\_facts])
* consult(namafile[, nama\_section\_facts])

Perbedaan assert, asserta dan assertz adalah asserta menyertakan sebuah fakta baru pada *section facts* sebelum fakta-fakta yang telah ada untuk predikat tersebut, sedangkan assertz menyertakan sebuah fakta baru setelahnya, sedangkan assert berfungsi sama seperti assertz. Sedangkan consult membaca dari sebuah file dan menyertakan fakta-fakta yang ada di file tersebut sesudah fakta-fakta yang telah ada.

Tidak seperti assertz, jika consult dipanggil hanya dengan satu argumen (tidak ada nama *section facts*) maka hanya akan menyertakan fakta-fakta yang predikatnya telah dideklarasikan di *section facts default* yaitu **dbasedom**. Jika memanggil consult dengan dua argumen (nama file dan nama *section facts*) maka hanya akan menyertakan fakta-fakta yang predikatnya dideklarasikan pada *section facts* dengan nama yang sesuai. Jika file tersebut mengandung fakta-fakta yang bukan milik dari *section facts* tersebut, maka akan terjadi *error* pada saat membaca bagian tersebut. Perlu diperhatikan bahwa consult membaca fakta satu demi satu, jika pada file ada 10 fakta dan pada fakta ke-7 terjadi *error*, maka consult akan menyertakan 6 fakta pertama pada *section facts* kemudian menampilkan pesan kesalahan.

Sebagai catatan, consult hanya bisa membaca sebuah file dengan syarat format file tersebut sama persis dengan format file yang disimpan menggunakan predikat save, yaitu :

- tidak ada karakter kapital kecuali dalam tanda petik dua (penulisan string).

- tidak spasi kecuali dalam tanda petik dua.

- tidak ada komentar.

- tidak ada baris kosong.

- tidak ada symbol tanpa di dalam tanda petik dua.

* **Menghapus fakta pada saat *run time***

Predikat retract mengunifikasi suatu fakta dan menghapus fakta tersebut dari *sectio facts*. Cara penulisannya adalah sebagai berikut :

retract(<fakta>[, nama\_section\_facts])

retract akan menghapus fakta pertama yang cocok dengan argumen <fakta> yang diberikan. Karena retract merupakan predikat nondeterministik maka selama lacakbalik, retract akan menghapus fakta-fakta yang cocok dengan argumen <fakta>, kecuali jika fakta yang akan dihapus, predikatnya dideklarasikan deterministik. Ketika semua fakta yang cocok sudah terhapus, pemangilan retract berikutnya akan gagal. Predikat retractall akan menghapus semua fakta yang cocok dengan argumen <fakta> dan penulisannya sebagai berikut :

retractall(<fakta>[, nama\_section\_facts])

retractall berperilaku sama seperti kalau didefinsiskan sebagai berikut :

retractall(X):- retract(X), fail. %fail untuk memaksa lacak balik retractall(\_).

* **Menyimpan database fakta-fakta pada saat *run time***

Predikat save berfungsi untuk menyimpan fakta-fakta yang ada pada *section facts* ke dalam sebuah file. Cara penulisannya sebagai berikut :

save(nama\_file[, nama\_section\_facts])

Jika memanggil save hanya dengan satu argumen (tidak ada nama *section facts*), maka akan menyimpan fakta-fakta dari *section facts default* **dbasedom** ke file dengan nama yang sesuai dengan argumen. Jika memanggil save dengan dua argumen (nama file dan nama *section* *facts*), maka akan menyimpan semua fakta yang ada pada *section facts* yang sesuai ke dalam file dengan nama yang sesuai pula.

* **Kata kunci pada *section facts***

Fakta-fakta pada *section facts* dapat dideklarasikan dengan beberapa kata kunci opsional berikut:

nondeterm menentukan bahwa kemungkinan ada sejumlah fakta dari suatu

**2.2 Teori Keilmuan yang Diimplementasikan**

Terdapat beberapa keilmuan yang di implementasikan dalam mendeteksi pecandu narkotika. Teori keilmuwan yang di implemntasikan pada program kali ini yaitu :

**2.2.1 Flu Burung**

Flu Burung adalah infeksi yang disebabkan oleh sejenis virus influenza avian. Meskipun ada banyak jenis flu burung, jenis yang kini dikhawatirkan oleh masyarakat ialah virus flu burung H5N1. Virus ini terdapat pada unggas-unggas liar.

Sering kali, unggas liar tidak terkena efek dari virus. Akan tetapi, unggas liar dapat dengan mudah menularkan virusnya kepada unggas-unggas yang diternakkan sebagai bahan makanan, seperti ayam, bebek, dan kalkun. Virus flu burung dapat menyebabkan unggas-unggas ini menjadi sangat sakit.

Ayam atau unggas yang terdeteksi menggalami Flu Burung mengalami gangguan, antara lain yaitu :

- Gangguan Pernapasan,

- Jengger dan Pial bewarna kemerahan sampe kebiruan,

- Penurunan Produksi Telur,

- Diare,

- Kelumpuhan,

- Bercak Darah Pada Kaki,

- Menyebabkan Kematian,

- Angka Kematian Tinggi.

**2.2.2 Gumboro**

Penyakit Gumboro atau Penyakit Infectious Bursal Disease(IBD) merupakan penyakit pada ayam yang pertama kali dilaporkan oleh Cosgrove pada tahun 1962 berdasarkan kasus yang terjadi pada tahun 1957 di desa Gumboro-Delaware, negara bagian Amerika Serikat. Sesuai dengan nama asal daerah ditemukannya, penyakit ini dikenal juga sebagai Gumboro. Penyebab penyakit gumboro (IBD) adalah virus yang berbentuk*icosahedral*yang terdiri dari 2 segmen untaian ganda RNA, yang termasuk dalam famili *Birnaviridae*(Lukert dan Saif, 2003) Virus *very virulent*IBD (vvIBDv) bersifat sangat menular dan akut, menyebabkan angka morbiditas dan mortalitas yang tinggi. Penyakit ini berdampak ekonomis karena menyerang organ pertahanan ayam yaitu bursa *Fabricius*sehingga merugikan peternak.

Ayam yang terdeteksi menggunakan jenis narkoba Heroin mengalami gangguan, antara lain yaitu :

- Kehilangan Nafsu Makan,

- Tampak lesu,

- Membungkuk.

**2.2.3 Mareks**

Penyakit Marek adalah suatu penyakit neoplastik dan neuropathic pada unggas, terutama ayam, disebabkan oleh virus sangat infeksius dari herpesvirus cell-associated (VENUGOPAL, 2000). Penyakit ini pertama kali dilaporkan oleh seorang ahli patologi dari Hungaria bernama Jozsef Marek pada tahun 1907 yang menemukan ayam menderita paralisis berhubungan dengan polineuritis. Penyakit ini kemudian diketahui tersebar luas di berbagai negara dan mengakibatkan kerugian yang sangat besar, seperti tercatat di Amerika kerugian akibat penyakit per tahunnya sebelum adanya program vaksinasi mencapai 150 juta dollar (FENNER et al., 1987). Penyakit Marek disebabkan oleh herpesvirus serotipe 1 yang bersifat onkogenik (dapat menimbulkan tumor), sangat menular dan limfoproliferatif (CALNEK dan WITTER, 1997). Virus ini bersifat cell-associated karena sulit bertahan di luar sel induk semangnya, kecuali pada sel epitel folikel bulu dimana dapat ditemukan virus terbebas dari sel.

Ayam yang terdeteksi menggalami Mareks mengalami gangguan, antara lain yaitu :

- Gangguan Pernapasan,

- Sayap menggantung,

- Kaki Pincang,

- Sempoyongan.

**2.2.4 Cacar Unggas**

Cacar Unggas adalah penyakit menular pada unggas yang disebabkan oleh virus familia poxviridae dan genus Avipoxvirus. Penyakit ini dapat menyerang ayam, kalkun, burung kenari, burung puyuh, dan burung merpati. Penyebaran virus ini lambat dengan pertumbuhan plotiteratif pada bagian tidak berbulu dan bercak perkejuan di selaput lender mulut. Meskipun demikian, daging ayam boleh dikonsumsi; asalkan mulut, kerongkongan, esophagus ayam yang tersebut dibuang.

Penyakit ini dapat ditularkan secara langsung melalui kontak dengan luka di kulit, dan secara tidak langsung melalui udara, melalui pakan tercemar virus, atau vector gigitan serangga seperti nyamuk.

Pecandu yang terdeteksi menggunakan jenis narkoba Kokain mengalami gangguan, antara lain yaitu :

- Gangguan Pernapasan,

- Muncul lesi di daerah yang tidak di tumbuhi bulu,

- Pembengkakan dari sinus dan mata.

**2.2.5 Penyakit Produksi Telur**

 Penurunan produksi telur sering kali merupakan salah satu gejala awal adanya serangan penyakit. Gejala lainnya dapat berupa lesu dan bulu kusam, mata berair, keluar ingus dari hidung, batuk, rontok bulu, pincang, sampai kematian. Jika peternak rnelihat seekor ayam sakit, lakukan isolasi atau pengafkiran dan amati keseluruhan populasi secara teliti. Jika curiga ada serangan penyakit, segera hubungi dokter hewan setempat agar dapat membantu memeriksa sehingga diperoleh diagnosa dan pengobatan yang akurat.  
Pada umumnya, saat ayam terkena penyakit apapun, maka produksi telur akan terganggu. Penyakit yang secara langsung dapat menyebabkan penurunan produksi telur. di antaranya adalah: EDS, ND, IB, CRD dan colibacillosis. Penyakit ND dan IB menurunkan kualitas kerabang dan bagian dalam telur. EDS menyebabkan kerabang telur sangat tipis sehingga telur mudah pecah, sedangkan ND dan IB dapat merusak saluran produksi.

Pecandu yang terdeteksi menggunakan jenis narkoba Sabu-Sabu mengalami gangguan, antara lain yaitu :

- Gangguan Pernapasan,

- Penurunan produksi telur,

- Kualitas telur jeluk .

**2.2.6 Penyakit Saluran Pernapasan**

Penyakit tersebut selain menimbulkan kematian dan gangguan produksi daging atau telur, juga dapat meningkatkan angka afkir ayam, *feed conversion rasio* (FCR) tinggi serta daya tanggap kekebalan menurun.

Fungsi utama saluran pernapasan ayam adalah menyediakan oksigen, mengeluarkan karbondioksida (CO2), membantu proses kekebalan primer dan memperlancar mekanisme pengaturan suhu tubuh. Syarat utama agar sistem pernapasan berfungsi baik adalah ketersediaan udara bersih dan saluran pernapasan yang sehat.

Ayam yang terdiagnosis mengalami penyakit saluran pernapasa mengalami gejala, antara lain yaitu :

- Gangguan pernapasan,

- Penurunan Produksi Telur,

- Menyebabkan Kematian,

- Angka Kematian Tinggi,

- Diare Berwarna Hijau,

- Kehilangan Nafsu Makan,

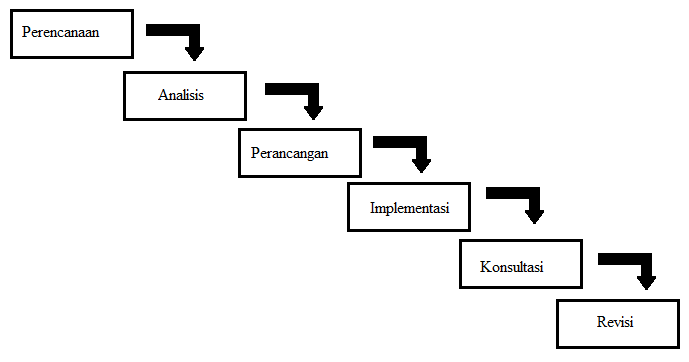
- Kehilangan nafsu minum,

- Kelemahan / malas bergerak.

**BAB III**

**METODOLOGI**

**3.1 Alur Pembuatan Sistem**



Gambar 3.1 Alur Pembuatan Sistem.

1. Perencanaan

Perencanaan pembuatan sistem pakar ini yaitu untuk mengetahui gejala-gejala yang ada dan dapat mencegah timbulnya kematian pada unggas dengan memberikan saran pencegahan yang tepat dan rasional. Sistem pakar ini dapat digunakan untuk memberikan tindakan pencegahan secara umum untuk membantu peternak ayam menghindari kerugian ekonomi yang lebih besar dengan mengetahui lebih awal penyakit ayam yang menyerang.

1. Analisis

Analisa yang dijelaskan dalam sistem pakar ini yaitu :

* Prosedur Sistem Berjalan
* Basis Pengetahuan
* Table pakar dan Rule rule pada Pakar

1. Perancangan

Tahap ini terfokus kepada perancangan alur program dan kebutuhan program.

1. Implementasi

Implementasi menuliskan basic pengetahuan dan rule -rule kedalam script

Program sesuai dengan methode yang telah ada di analisis.

1. Konsultasi

Meminta saran untuk program yang sebaian jadi untuk lebih di perbaiki untuk pengujian nantinya.

1. Revisi

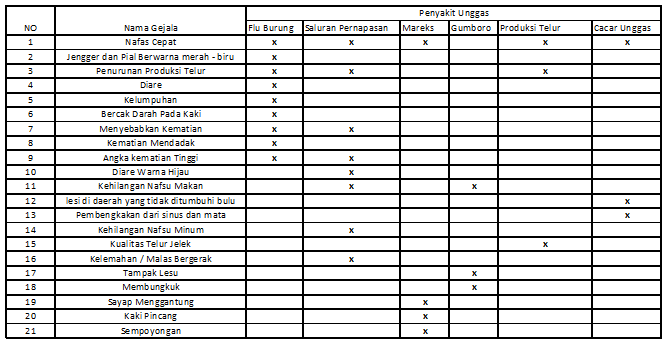
Memperbaiki program sesuai dengan saran apa yang sudah di konsultasikan sebelumnya.

**BAB IV**

**HASIL DAN PEMBAHASAN**

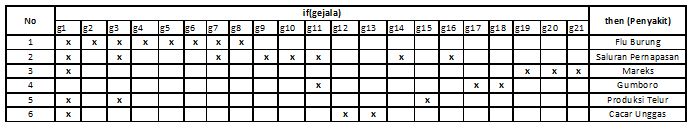
* 1. **Tabel Kebenaran dan Rule**

Tabel 4.1 Tabel Kebenaran



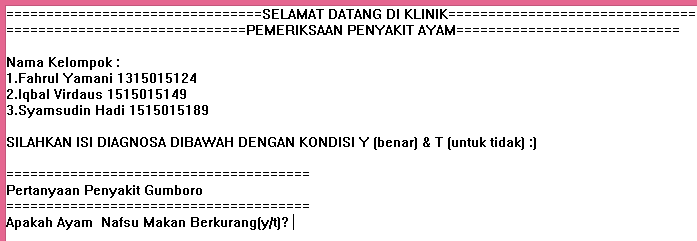
Tabel Kebenaran adalah tabel yang digunakan melihat nilai kebenaran dari suatu premis/pernyataan. Terdapat gejala-gejala yang di alami oleh pecandu Narkoba di dalam tabel tersebut.

Tabel 4.2 Tabel Rule



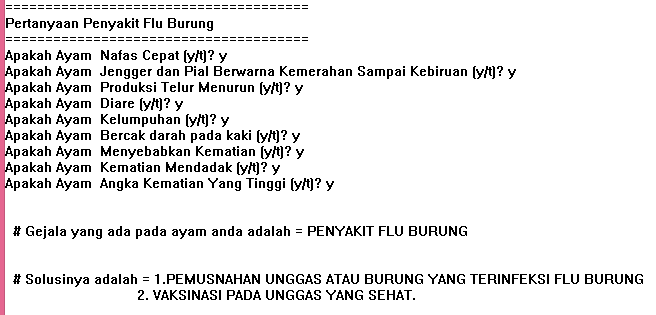
Tabel keputusan (decision table) adalah tabel yang digunakan sebagai alat bantu untuk menyelesaikan logika dalam program. Algoritma yang berisi keputusan bertingkat yang banyak sekali sangat sulit untuk digambarkan langsung dengan structured English atau pseudocode dan dapat dibuat terlebih dahulu dengan menggunakan tabel keputusan. Dengan demikian tabel keputusan efektif digunakan bilamana kondisi yang akan diseleksi didalam program jumlahnya cukup banyak dan rumit. Di dalam table tersebut gejala-gejala pada penyakit unggas ayam di inisialka menjadi g1,g2,g3,…dan seterusnya.

**4.2 Analisis Aplikasi**



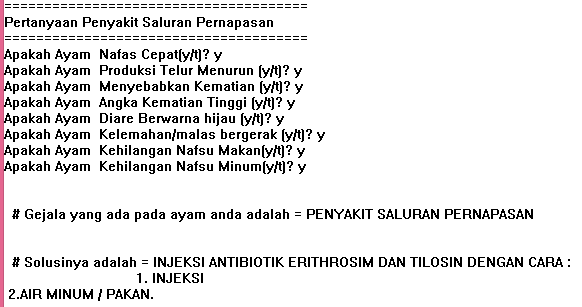
Gambar 4.2.1 : Tampilan Awal Program.

Dimana pada saat program pertmakali dijalankan maka akan muncul tampilan seperti diatas, yang berisa Nama kelompok dan pengimputan nama.



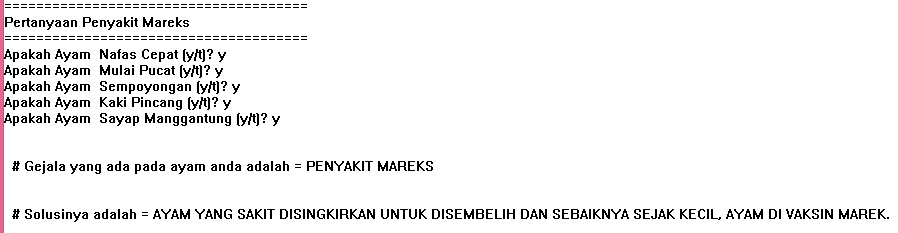
Gambar 4.2.2 : Tampilan Gejala – gejala Pada Flu Burung

Untuk memulai program tekan tombol ‘Y/y’. Setelah itu gejala – gejala yang ada akan tampil satu per satu, seperti contoh diatas gejala yang pertama muncul yaitu Apakah Nafas Cepat ? disini kami memilih ‘y’ maka program akan mengarahkan ke gejala yang selenjutnya yaitu Jengger dan Pial Berwarna Kemerahan Sampai Kebiruan ? jika ‘y’ maka akan diarahkan ke gejala Produksi telur menurun? jika ‘y’ maka akan diarahkan ke diare? jika ‘y’ maka akan diarahkan ke gejala selanjutnya yaitu apakah kelumpuhan? jika ‘y’’ maka akan diarahkan ke gejala selanjutnya yaitu apakah bercak darah pada kaki? jika ‘y’’ maka akan diarahkan ke gejala selanjutnya yaitu apakah menyebabkan kematian? jika ‘y’’ maka akan diarahkan ke gejala selanjutnya yaitu apakah kematian mendadak? jika ‘y’ ’ maka akan diarahkan ke gejala selanjutnya yaitu apakah angka kematian tinggi? jika ‘y’maka dari semua gejala yang ada bisa disimpulkan bahwa Ayam anda mengalami penyakit jenis flu burung. Dan setelah mengetahui jenis penyakit yang Ayam anda alami program ini juga memberitahukan solusi vaksinasi dan pemusnahan. Setelah itu ada menu pilihan jika anda ingin kembali ke gejala – gejala narkotika silahkan tekan tombol ‘M/m’ dan jika ingin keluar tekan tombol ‘X/x’.

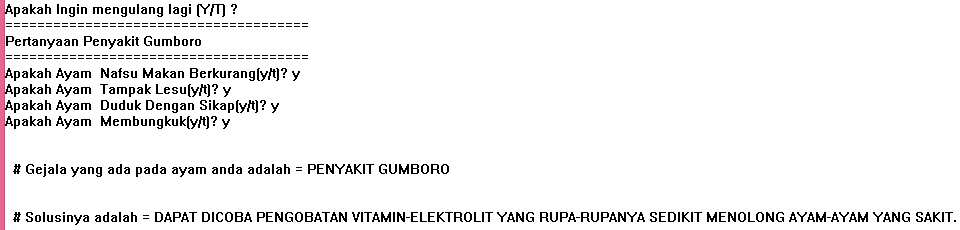


Gambar 4.2.3 : Tampilan Gejala – gejala Pada Penyakit saluran pernapasan.

Untuk memulai program tekan tombol ‘Y/y’. Setelah itu gejala – gejala yang ada akan tampil satu per satu, seperti contoh diatas gejala yang pertama muncul yaitu Apakah Nafas Cepat ? disini kami memilih ‘y’ maka akan diarahkan ke gejala Produksi telur menurun? jika ‘y’ maka akan diarahkan menyebabkan kematian? jika ‘y’ maka akan diarahkan ke gejala selanjutnya yaitu apakah kematian tinggi? jika ‘y’’ maka akan diarahkan ke gejala selanjutnya yaitu apakah diare berwarna hijau? jika ‘y’’ maka akan diarahkan ke gejala selanjutnya yaitu apakah menyebabkan kelemahan/malas bergerak? jika ‘y’’ maka akan diarahkan ke gejala selanjutnya yaitu apakah kehilangan nafsu makan? jika ‘y’ ’ maka akan diarahkan ke gejala selanjutnya yaitu apakah angka kehilangan nafsu minum? jika ‘y’ maka dari semua gejala yang ada bisa disimpulkan bahwa Ayam anda mengalami penyakit saluran pernapasan . Dan setelah mengetahui jenis penyakit yang program ini juga memberitahukan solusi yaitu injeksi anti biotik.

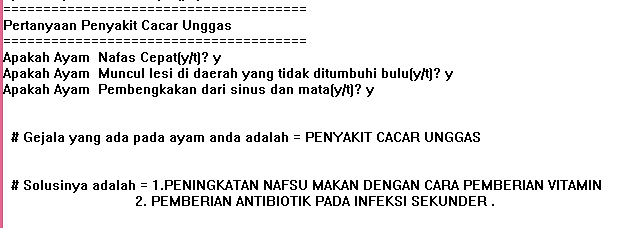
Gambar 4.2.4 : Tampilan Gejala – gejala Pada Penyakit Mareks.

Setelah itu gejala – gejala yang ada akan tampil satu per satu, seperti contoh diatas gejala yang pertama muncul yaitu Apakah Ayam Nafas Cepat ? disini kami memilih ‘y’ maka program akan mengarahkan ke gejala yang selenjutnya yaitu mulai pucat? jika ‘y’ maka akan diarahkan ke gejala Apakah Ayam Sempoyongan? jika ‘y’ maka akan diarahkan ke gejala apakah ayam kiki pincang? jika ‘y’ maka akan diarahkan ke gejala selanjutnya yaitu apakah Ayam sayap menggantung? maka dari semua gejala yang ada bisa disimpulkan bahwa ayam anda mengalami penyakit mareks. Dan setelah mengetahui jenis Penyakit program ini juga memberitahukan solusi yang sesuai dengan penyakit yang di alami yaitu di sembelih sejak kecil atau ayam di vaksin mareks .



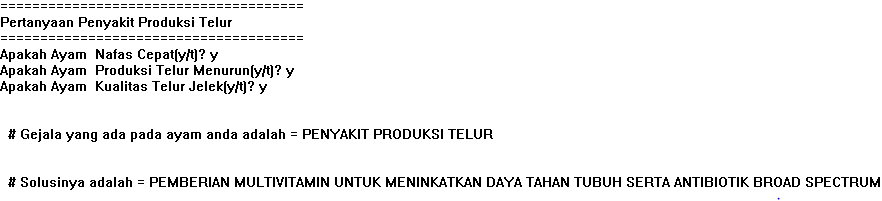
Gambar 4.2.5 : Tampilan Gejala – gejala Pada Gumboro.

Setelah itu gejala – gejala yang ada akan tampil satu per satu, seperti contoh diatas gejala yang pertama muncul yaitu Apakah ayam nafsu makan bekurang ? disini kami memilih ‘y’ maka program akan mengarahkan ke gejala yang selanjutnya yaitu Tampak lesu? jika ‘y’ maka akan diarahkan ke gejala Apakah Ayam duduk dengan sikap? jika ‘y’ maka akan diarahkan ke gejala ayam membungkuk? jika ‘y’ maka dari semua gejala yang ada bisa disimpulkan bahwa Ayam anda mengalami gejala penyakit Gumboro. Dan setelah mengetahui jenis penyakit lalu program ini juga memberitahukan solusi Yang sesuai dengan penyakit gumboro disini diberitahukan untuk memberikan vitamin elektrolit .



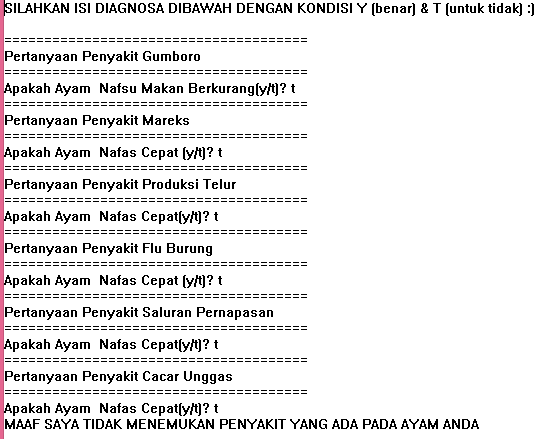
Gambar 4.2.6 : Tampilan Gejala – gejala Pada Cacar Unggas.

Setelah itu gejala – gejala yang ada akan tampil satu per satu, seperti contoh diatas gejala yang pertama muncul yaitu Apakah Ayam Nafas Cepat ? disini kami memilih ‘y’ maka program akan mengarahkan ke gejala yang selenjutnya yaitu muncul lesi di daerah yang tidak di tumbuhi bulu? jika ‘y’ maka akan diarahkan ke gejala apakah pembengkakan sinus dan mata? jika ‘y’ maka dari semua gejala yang ada bisa disimpulkan bahwa ayam anda mengalami penyakit jenis cacar unggas. Dan setelah mengetahui jenis penyakit yang ada program ini juga memberitahukan solusi yaitu pemberian vitamin dan anti biotik.



Gambar 4.2.7 : Tampilan Gejala – gejala Pada Penyakit Produksi telur.

Setelah itu gejala – gejala yang ada akan tampil satu per satu, seperti contoh diatas gejala yang pertama muncul yaitu Apakah Nafas Cepat ? disini kami memilih ‘y’ maka program akan mengarahkan ke gejala yang selenjutnya yaitu produksi telur menurun ? jika ‘y’ maka akan diarahkan ke gejala Apakah Ayam Kualitas Telur Jelek? jika ‘y’ maka dari semua gejala yang ada bisa disimpulkan bahwa ayam anda mengalami penyakit produksi telur. Dan setelah mengetahui jenis penyakitnya program ini juga memberitahukan solusi yang tepat yakni pemberian multivitamin untuk .



Gambar 4.2.8 : Tampilan jika gejala – gejala tak sesuai.

Jadi apabila gejala – gejala yang ada pada program tidak sesuai dengan yang anda rasakan maka program akan berikan notif yaitu “maaf Saya Tidak Menemukan Penyakit yang ada pada ayam anda. Setelah itu ada menu pilihan jika anda ingin kembali ke gejala – gejala narkotika silahkan tekan tombol ‘M/m’ dan jika ingin keluar dari program tekan tombol ‘X/x’.

**BAB V**

**PENUTUP**

**5.1 Kesimpulan**

Setelah melalui tahap pengujian pada sistem pakar untuk mendiagnosis penyakit unggas dan penanganannya, maka dapat diambil beberapa kesimpulan sebagai berikut :

System pakar adalah system yang berusaha mengadopsi pengetahuan manusia kekomputer, agar komputer dapat menyelesaikan masalah yang seperti biasa dilakukan oleh para ahli.

Ada beberapap hal penting yang perlu diperhatian bahwa secangih apapun suatu system atau sebesar apapun basis pengetahuan yang dimiliki, tentu saja ada kelemahannya sebagai konsekuensi logis kelemahan manusia sebagai penyusun element-elementnya. Bahwa system tidak memilik inisiatif untuk melakukan suatu tindakan diluar dari apa yang telah diprogamkan untuknya.

Berdasarkan hasil pengujian, program system pakar ini berguna untuk membantu dan mempermudah pengguna dalam memperoleh informasi mengenai pecandu narkoba serta mendapatkan hasil diagnosa jenis narkoba yang dikonsumsi pecandu melalui gejala-gejalanya.

**5.2 Saran**

Beberapa hal yang dapat diungkap sebagai saran untuk pengembangan sistem pakar mendiagnosis penyakit unggas dan penanganannya adalah sebagai berikut :

Dilakukan pengembangan program sejenis dengan permasalahan domain yang lebih luas.

Data mengenai gejala dan semua mengenai penyakit unggas kiranya dapat lebih dimaksimalkan sampai mendapatkan informasi atau perhitungan yang lebih akurat serta dicari alternative lain yang memungkinkan penyelesaian yang jauh lebih baik.

**DAFTAR PUSTAKA**

https://id.wikipedia.org/wiki/

Yusti Farlina1, Jamal Maulana Hudin2, “*APLIKASI SISTEM PAKAR DIAGNOSA PENYAKIT AYAM STUDI KASUS:PADA CV. INTAN JAYA ABADI SUKABUMI*”, STMIK Nusa Mandiri Sukabumi,2014

Lampiran

1. Source Code

/\*====================================================================================

SISTEM PAKAR TENTANG DIAGNOSA PENYAKIT UNGGAS AYAM

KELOMPOK 6

KECERDASAN BUATAN

=====================================================================================\*/

domains

kerusakan = symbol

gejala = symbol

tanya = string

jawab = char

kondisi = list\*

list = string

FACTS

xcari(gejala)

xgagal(gejala)

nama(kondisi)

false(list)

predicates

nondeterm mulai

nondeterm lihat(kondisi)

cari(tanya,gejala)

gagal(tanya,gejala)

nondeterm gejala(gejala)

clear\_fakta2

simpan(gejala,jawab)

tanya(tanya,gejala,jawab)

gojek

nondeterm diagnosa(kerusakan)

solusi(kerusakan)

penanganan(kerusakan)

nondeterm ini(char)

nondeterm awal

lacak(char,char)

clauses

lacak(Jawab,Jawab):-!.

lacak(\_,\_):-fail.

nama(["1.Fahrul Yamani 1315015124", "2.Iqbal Virdaus 1515015149", "3.Syamsudin Hadi 1515015189"]).

awal:-

write("================================SELAMAT DATANG DI KLINIK==============================="),nl,

write("==============================PEMERIKSAAN PENYAKIT AYAM============================"),nl,nl,

write("Nama Kelompok : "),nl,

nama(Kelompok),

lihat(Kelompok),nl,

write("SILAHKAN ISI DIAGNOSA DIBAWAH DENGAN KONDISI Y (benar) & T (untuk tidak) :)"),nl,nl.

mulai:-

gojek,nl,nl,nl,nl,nl,

write("Apakah Ingin mengulang lagi (Y/T) ?"),nl,

readchar(Jawab),ini(Jawab).

ini(Jawab):-lacak(Jawab,'y'),mulai.

lihat ([H|T]):-

not(false(H)),

write(H),nl,

lihat(T).

lihat([H|\_]):-

assertz(false(H)).

gojek:-

diagnosa(\_),!,write(""),

save("test.dat"),

clear\_fakta2.

gojek:-

write("MAAF SAYA TIDAK MENEMUKAN PENYAKIT YANG ADA PADA AYAM ANDA"),nl,

clear\_fakta2.

cari(\_,Gejala):-write("Apakah "),

xcari(Gejala),!.

cari(Tanya,Gejala):-write("Ayam "),

not(xgagal(Gejala)),

tanya(Tanya,Gejala,Jawab),

Jawab='y'.

gagal(\_,Gejala):-

xgagal(Gejala),!.

gagal(Tanya,Gejala):-

not(xcari(Gejala)),

tanya(Tanya,Gejala,Jawab),

Jawab='t'.

tanya(Tanya,Gejala,Jawab):-

write(Tanya),

readchar(Jawab),

write(Jawab),nl,

simpan(Gejala,Jawab).

simpan(Gejala,'y'):-

asserta(xcari(Gejala)).

simpan(Gejala,'t'):-

asserta(xgagal(Gejala)).

clear\_fakta2:-

retract(xcari(\_)),fail.

clear\_fakta2:-

retract(xgagal(\_)),fail.

clear\_fakta2.

/\*```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````\*/

/\*===================================================================G E J A L A=====================================================================\*/

/\*```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````\*/

gejala(Gejala):-

xcari(Gejala),!.

gejala(Gejala):-

xgagal(Gejala),!,fail.

gejala(gumboro):-

write("======================================"),nl,

write("Pertanyaan Penyakit Gumboro"),nl,

write("======================================"),nl,

cari(" Nafsu Makan Berkurang(y/t)? ",gumboro),

cari(" Tampak Lesu(y/t)? ",gumboro1),

cari(" Duduk Dengan Sikap(y/t)? ",gumboro2),

cari(" Membungkuk(y/t)? ",gumboro3).

gejala(mareks):-

write("======================================"),nl,

write("Pertanyaan Penyakit Mareks"),nl,

write("======================================"),nl,

cari(" Nafas Cepat (y/t)? ",mareks),

cari(" Mulai Pucat (y/t)? ",mareks1),

cari(" Sempoyongan (y/t)? ",mareks2),

cari(" Kaki Pincang (y/t)? ",mareks3),

cari(" Sayap Manggantung (y/t)? ",mareks4).

gejala(produksi\_telur):-

write("======================================"),nl,

write("Pertanyaan Penyakit Produksi Telur"),nl,

write("======================================"),nl,

cari(" Nafas Cepat(y/t)? ",produksi\_telur),

cari(" Produksi Telur Menurun(y/t)? ",produksi\_telur1),

cari(" Kualitas Telur Jelek(y/t)? ",produksi\_telur2).

gejala(flu\_burung):-

write("======================================"),nl,

write("Pertanyaan Penyakit Flu Burung"),nl,

write("======================================"),nl,

cari(" Nafas Cepat (y/t)? ",flu\_burung),

cari(" Jengger dan Pial Berwarna Kemerahan Sampai Kebiruan (y/t)? ",flu\_burung1),

cari(" Produksi Telur Menurun (y/t)? ",flu\_burung2),

cari(" Diare (y/t)? ",flu\_burung3),

cari(" Kelumpuhan (y/t)? ",flu\_burung4),

cari(" Bercak darah pada kaki (y/t)? ",flu\_burung5),

cari(" Menyebabkan Kematian (y/t)? ",flu\_burung6),

cari(" Kematian Mendadak (y/t)? ",flu\_burung7),

cari(" Angka Kematian Yang Tinggi (y/t)? ",flu\_burung8).

gejala(saluran\_pernapasan):-

write("======================================"),nl,

write("Pertanyaan Penyakit Saluran Pernapasan"),nl,

write("======================================"),nl,

cari(" Nafas Cepat(y/t)? ",saluran\_pernapasan),

cari(" Produksi Telur Menurun (y/t)? ",saluran\_pernapasan1),

cari(" Menyebabkan Kematian (y/t)? ",saluran\_pernapasan2),

cari(" Angka Kematian Tinggi (y/t)? ",saluran\_pernapasan3),

cari(" Diare Berwarna hijau (y/t)? ",saluran\_pernapasan4),

cari(" Kelemahan/malas bergerak (y/t)? ",saluran\_pernapasan5),

cari(" Kehilangan Nafsu Makan(y/t)? ",saluran\_pernapasan6),

cari(" Kehilangan Nafsu Minum(y/t)? ",saluran\_pernapasan7).

gejala(cacar\_unggas):-

write("======================================"),nl,

write("Pertanyaan Penyakit Cacar Unggas"),nl,

write("======================================"),nl,

cari(" Nafas Cepat(y/t)? ",cacar\_unggas),

cari(" Muncul lesi di daerah yang tidak ditumbuhi bulu(y/t)? ",cacar\_unggas1),

cari(" Pembengkakan dari sinus dan mata(y/t)? ",cacar\_unggas2).

/\* -----------------------------------------------------------D I A G N O S A-------------------------------------------------------- \*/

diagnosa("Penyakit\_Gumboro"):-

gejala(gumboro),

gejala(gumboro1),

gejala(gumboro2),

gejala(gumboro3),

solusi("Penyakit Gumboro"),

penanganan("Dapat dicoba pengobatan vitamin-elektrolit yang rupa-rupanya sedikit menolong ayam-ayam yang sakit.").

diagnosa("Penyakit\_Mareks"):-

gejala(mareks),

gejala(mareks1),

gejala(mareks2),

gejala(mareks3),

gejala(mareks4),

solusi("Penyakit Mareks"),

penanganan("Ayam yang sakit disingkirkan untuk disembelih dan Sebaiknya sejak kecil, ayam di vaksin Marek.").

diagnosa("Produksi\_Telur"):-

gejala(produksi\_telur),

gejala(produksi\_telur1),

gejala(produksi\_telur2),

solusi("Penyakit Produksi Telur"),

penanganan("pemberian multivitamin untuk meninkatkan daya tahan tubuh serta antibiotik broad spectrum untuk mencegah infeksi sekunder.").

diagnosa("Penyakit flu Burung"):-

gejala(flu\_burung),

gejala(flu\_burung1),

gejala(flu\_burung2),

gejala(flu\_burung3),

gejala(flu\_burung4),

gejala(flu\_burung5),

gejala(flu\_burung6),

gejala(flu\_burung7),

gejala(flu\_burung8),

solusi("Penyakit Flu Burung"),

penanganan("1.Pemusnahan unggas atau burung yang terinfeksi flu burung \n\t\t 2. Vaksinasi Pada Unggas Yang Sehat.").

diagnosa("Penyakit Saluran Pernapasan"):-

gejala(saluran\_pernapasan),

gejala(saluran\_pernapasan1),

gejala(saluran\_pernapasan2),

gejala(saluran\_pernapasan3),

gejala(saluran\_pernapasan4),

gejala(saluran\_pernapasan5),

gejala(saluran\_pernapasan6),

gejala(saluran\_pernapasan7),

solusi("Penyakit Saluran Pernapasan"),

penanganan("Injeksi ANTIBIOTIK ERITHROSIM DAN TILOSIN dengan cara :\n\t\t 1. injeksi \n 2.air minum / pakan.").

diagnosa("Penyakit Cacar Unggas"):-

gejala(cacar\_unggas),

gejala(cacar\_unggas1),

gejala(cacar\_unggas2),

solusi("Penyakit cacar unggas"),

penanganan("1.Peningkatan nafsu makan dengan cara pemberian vitamin \n\t\t 2. pemberian antibiotik pada infeksi sekunder .").

/\*==================================================================================================================================\*/

solusi(Kerusakan):-

upper\_lower(XKerusakan,Kerusakan),nl,nl,

write(" # Gejala yang ada pada ayam anda adalah = ",XKerusakan),nl.

penanganan(Kerusakan):-

upper\_lower(BKerusakan,Kerusakan),nl,nl,

write(" # Solusinya adalah = ",BKerusakan),nl.

goal

awal,

mulai.

Kartu Konsul

