# MODUL PRAKTIKUM

# KECERDASAN BUATAN

Edisi ke – 3

September 2015

Disusun oleh :

Sri Winiarti

Andri Pranolo

Laboratorium Sistem Cerdas

Program Studi Teknik Informatika

Fakultas Teknologi Industri

Universitas Ahmad Dahlan

# DAFTAR ISI

# KATA PENGANTAR

Alhamdulillah penulis panjatkan kehadirat Allah SWT yang telah memberikan kesehatan dan kesempatan kepada penulis, sehingga Buku Petunjuk Praktikum Kecerdasan Buatan Edisi ke-3 ini bisa diselesaikan. Cukup banyak perubahan yang dilakukan dalam merevisi Buku Petunjuk Praktikum Sistem pendukung keputusan ini berdasar masukan yang telah diberikan oleh para Dosen di Program Studi teknik Informatika. Berdasarkan masukan tersebut, maka diramulah Buku Petunjuk Praktikum Kecerdasan Buatan dengan harapan sesuai dengan materi kuliah yang diberikan. Ada beberapa tools yang dipakai pada Buku praktikum ini diantaranya, Turbp prolog, MS-Access, Visio Drawing, Power Designer 6.0 dan GUI Design User Interface. Sekali lagi penulis mengucapkan terima kasih untuk rekan-rekan Dosen Teknik Informatika yang secara tidak langsung telah memberikan sumbangsih ide/pikiran guna mendukung terbentuknya Buku Petunjuk Praktikum Kecerdasan Buatan ini. Semoga Buku ini dapat bermanfaat sesuai dengan harapan Penulis.

Yogyakarta, September 2015

Penulis

Sri Winiarti
Andri Pranolo

# MATERI I
# PENGENALAN BAHASA PROLOG

**Waktu : 1.5 Jam**

**Kompetensi Dasar :** Mahasiswa mengenal salah satu bahasa pemrograman untuk membangun aplikasi Kecerasan Buatan.

**Indikator** : Setelah mengikuti praktikum ini, mahasiswa dapat memahami struktur Prolog sebagai salah satu bahasa pemrograman untuk Kecerdasan Buatan.

## Teori Pendukung

### A. Sejarah Prolog

- Prolog singkatan dari *Programming in Logic*.
- Dikembangkan oleh Alain Colmenraurer dan P.Roussel di Universitas Marseilles Perancis, tahun1972.
- Prolog populer di Eropa untuk aplikasi artificial intelligence, sedangkan di Amerika peneliti mengembangkan aplikasi yang sama, yaitu LISP.

### B. Perbedaan Prolog dengan Bahasa Lainnya

Bahasa Pemrograman yang Umum (Basic, Pascal, C, Fortran):

1. Diperlukan algoritma/prosedur untuk memecahkan masalah (procedural languange)
2. program menjalankan prosedur yang sama berulang-ulang dengan data masukan yang berbeda-beda.
3. Prosedur dan pengendalian program ditentukan oleh programmer dan perhitungan dilakukan sesuai dengan prosedur yang telah dibuat.

Bahasa Pemrograman Prolog :

1. *Object oriented languange* atau *declarative languange*.
2. Tidak terdapat prosedur, tetapi hanya kumpulan data-data objek (fakta) yang akan diolah, dan relasi antar objek tersebut membentuk aturan yang diperlukan untuk mencari suatu jawaban
3. Programmer menentukan tujuan (goal), dan komputer menentukan bagaimana cara mencapai tujuan tersebut serta mencari jawabannya.

4. Dilakukan pembuktian terhadap cocok-tidaknya tujuan dengan data-data yang telah ada dan relasinya.
5. Prolog ideal untuk memecahkan masalah yang tidak terstruktur, dan prosedur pemecahannya tidak diketahui, khususnya untuk memecahkan masalah non numerik.
6. Prolog bekerja seperti pikiran manusia, proses pemecahan masalah bergerak di dalam ruang masalah menuju suatu tujuan (jawaban tertentu). Contoh : Pembuatan program catur denganProlog.

Dalam buku petunjuk praktikum ini digunakan program Turbo Prolog untuk melengkapi pembahasan pemrograman logika dengan Prolog. Turbo prolog mirip dengan Turbo Pascal, Turbo C, dan sejenisnya. Tampilannya terbagi menjadi empat jendela yaitu *editor*, *dialog*, *message*, dan *trace*. Jendela editor sebagai tempat untuk menyunting program, sedangkan dialog tempat menuliskan goal dan menampilkan hasil query. Pesan-pesan error, compiler, running, dan lain-lain ditampilkan di dalam jendela message, sedangkan trace sebagai tempat untuk menelusuri jalannya program. Secara umum, suatu program Prolog terdiri dari beberapa kelompok, yaitu domains, predicates, clauses, goal. Masing-masing bagian akan diuraikan pada bagian berikut.

**A. DOMAINS**

Domain dalam Prolog seperti type dalam Pascal, yaitu untuk menyatakan jenis variable atau argumen, misalnya:

```
domains
kota = symbol
alamat = string
list = symbol*
```

Ada lima domain baku di dalam Prolog, yaitu:

1. **char**, karakter tunggal yang diapit oleh tanda kutip tunggal: ‘a’, ‘b’, ‘\13’.
2. **integer**, bilangan bulat antara –32768 hingga 32767. Notasi $ digunakan untuk menunjukkan bilangan heksa.
3. **real**, bilangan nyata antara 1x10 –307 hingga 1 x 10 308.
4. **string**, deretan karakter yang diapit oleh tanda kutip dobel, misalnya “ipb”.
5. **symbol**, rangkaian karakter yang diawali dengan huruf kecil da tanpa tanda apa pun.

Disamping itu terdapat domain lainnya yang tidak baku, di antaranya adalah:

1. domain **file**, yang digunakan untuk memberi nama file secara simbolik seperti contoh berikut:
   file = <nama file simbolik 1> ; <nama file simbolik 2> ; …..
2. Domain list, digunakan untuk menyatakan list (linked list) dimana elemen pertama mempunyai pointer ke elemen kedua dan seterusnya. Deklarasi list ini dapat dituliskan dengan bentuk:

<nama list> = <domain>*
list_simbol = symbol*

## B. Fakta dan Relasi

Prolog terdiri dari kumpulan data-data objek yang merupakan suatu fakta. Fakta dibedakan 2 macam :

- o Menunjukkan relasi.
- o Menunjukkan milik/sifat.
- Penulisannya diakhiri dengan tanda titik "."
- Contoh :

Tabel. 1.1 Contoh Fakta dalam Prolog

| Fakta | Prolog |
|---|---|
| Slamet adalah ayah Amin | ayah (slamet, amin). |
| Anita adalah seorang wanita | wanita (anita). |
| Angga suka renang dan tenis | suka(angga, renang). dan suka(angga,tenis). |
| Jeruk berwarna jingga | jngga(jeruk). |

## C. Aturan

Aturan adalah suatu pernyataan yang menunjukkan bagaimana fakta-fakta berinteraksi satu dengan yang lain untuk membentuk suatu kesimpulan. Sebuah aturan dinyakatakan sebagai suatu kalimat bersyarat. Kata "if" adalah kata yang dikenal Prolog untuk menyatakan kalimat bersyarat atau disimbolkan dengan ":-".

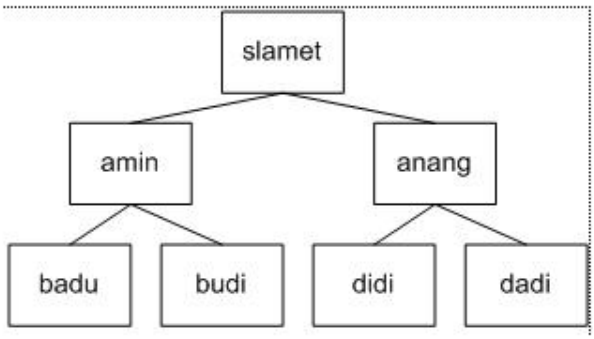
Contoh :

Tabel 1.2 Contoh Aturan dalam Prolog

| Fakta dan Aturan | Prolog |
|---|---|
| F : Tino suka apel | suka(tino, apel). |
| A : Yuli suka sesuatu yang disukai Tino | suka(yuli,Sesuatu) :- suka(tino,Sesuatu). |

Setiap aturan terdiri dari kesimpulan(kepala) dan tubuh. Tubuh dapat terdiri dari 1 atau lebih pernyataan atau aturan yang lain, Disebut subgoal dan dihubungkan dengan logika "and". Aturan memiliki sifat then/if conditional "Kepala(head) benar jika tubuh (body) benar".

Contoh : Silsilah keluarga :




Gambar 1.1 Contoh Aturan yang memiliki sifat kondisonal

**D. Pertanyaan ("Query")**

Setelah memberikan data-data berupa fakta dan aturan, selanjutnya kita dapat mengajukan pertanyaan berdasarkan fakta dan aturan yang ada. Penulisannya diawali simbol "?-" dan diakhiri tanda ".".

**E. Predikat ("Predicate")**

Predikat adalah nama simbolik untuk relasi. Contoh : ayah(slamet,amin). Predikat dari fakta tersebut ditulis : ayah(simbol,simbol). Dimana ayah adalah nama predikat, sedangkan slamet dan amin adalah menujukkan argumen.
Sebuah predikat dapat tidak memiliki atau memiliki argumen dengan jumlah bebas. Jumlah argumen suatu predikat disebut aritas (arity). ayah(nama) …… aritas-nya 1 ayah(nama1,nama2) ….. aritasnya 2. Syarat-syarat penulisan nama predikat :

- Harus diawali dengan huruf kecil dan dapat diikuti dengan huruf, bilangan atau garis bawah.
- Panjang nama predikat maksimum 250 karakter.
- Tidak diperbolehkan menggunakan spasi, tanda minus, tanda bintang dan garis miring.

**F. Variabel**

Varibel adalah besaran yang nilainya dapat berubah-ubah. Tata cara penulisan variabel :

- Nama varibel harus diawali huruf besar atau garis bawah(_)
- Nama variabel dapat terdiri dari huruf, bilangan, atau simbol dan merupakan kesatuan dengan panjang maksimum 250 karakter.
- Nama variabel hendaknya mengandung makna yang berkaitan dengan data yang dinyatakannya.

  Contoh : dari silsilah di atas :

  ?-ayah(slamet,Anak).

  Anak=budi;

Anak=badu
No

Dari query di atas akan dicari siapakah anak dari ayah yang bernama Slamet. Karena mempunyai relasi yang sama (yaitu ayah), variabel Anak akan mencari nilai dari konstanta suatu fakta/aturan yang sepadan. Tanda ";" digunakan bila terdapat kemungkinan ada lebih dari satu jawaban. "No" berarti tidak ada lagi kemungkinan jawaban.

Contoh : dari silsilah di atas :

```
?-ayah(slamet,X),ayah(X,Y).

   X=amin
   Y=budi;
   X=amin
   Y=badu;
   X=anang
   Y=didi;
   X=anang
   Y=didi
   No
```

Contoh Program : Kasus Silsilah Keluarga

```
/* Program Silsilah */

domains
      name = symbol
predicates
      ayah(nama,nama)
      setiaporang
clauses
      ayah(slamet,amin).
      ayah(slamet,anang).
      ayah(amin,badu).
      setiaporang if
              ayah(X,Y),
              write(X," is ",Y,"'s ayah\n") and
              fail.
```

Listing 1.1. Contoh Program silsilah Keluarga

**Tugas Praktikum :**

a. Coba ketik program di atas dengan bahasa prolog. Caranya :
   1. Run Program Prolog
   2. Akan muncul menu editor Prolog
   3. Untuk menulis sintaks Program tulis pada window editor
   4. Ketik semua program di atas (listing 1.1)
   5. Run Program dengan menekan tombol Alt + R
   6. Perhatikan kursor akan muncul pada window Run. Contoh : goal>
   7. Ketik : ayah(X,slamet) atau ayah(amin,X). Tekan tombol Enter .

8. Perhatikan apa yang terjadi…?

Tugas : Cobalah beberapa query(minimal 3 query). Catatlah hasilnya. **(nilai 20).**

**b.** Berdasarkan listing 1.1 cobalah kembangkan programnya, bila ditambahkan faktaya kakek,anak atau istri. **(nilai 30).**

c. Cobalah coding pada listing 1.2 berikut ini, jalankan program tersebut dengan memberikan goal sebagai berikut :
   1. Query 1 : ukuran(Z,besar).
   2. Query 2 ; warna(Z,coklat).

Setelah Anda mencoba query – query tersebut, cobalah untuk menambahkan fakta baru pada program terebut. Pada **predicates** tambahkan **jenis(symbol,symbol)** dan pada **clauses** tambahkan **jenis(beruang,karnivora**)dan pada **goal** tambahkan **jenis(Z,karnivora).(nilai 30).**

```
predicates
ukuran(symbol, symbol)
warna(symbol,symbol)
gelap(symbol)
clauses
ukuran(beruang,besar).
ukuran(gajah,besar).
ukuran(kucing,kecil).
warna(beruang,coklat).
warna(kucing,hitam).
warna(gajah,kelabu).
gelap(Z):-warna(Z, hitam).
gelap(Z):-warna(Z,coklat).
goal
clearwindow,
gelap(Z), ukuran(Z,besar), write(Z).
```

Listing 1.2 Contoh Program struktur prolog

d. Cobalah coding pada listing 1.3 berikut ini. **(nilai 20):**

```
/* Program Faktorial */
        domains
          n, f = real
       predicates
         factorial(n,f)
         clauses
        factorial(1,1).
      factorial(N,Res) if
              N > 0 and
            N1 = N-1 and
       factorial(N1,FacN1) and
           Res = N*FacN1.
```

Listing 1.3 Contoh Program Faktorial dalam Prolog

Setelah selesai coba jalankan dengan memberikan *query* sebagai berikut :

1) Query 1 : `factorial(5,fac2).`

   `Fac2=120` → `1 Solution`

2) Query 2 : `factorial(3,fac4).`

   `Fac4=6` → `1 Solution`

3) Query 3 : `factorial(3,6).`

   `true` → `1 Solution`

4) Query 4 : `factorial(5,2).`

   `false` → `1 Solution`

Coba berikan 3 buah query lain dan untuk hasilnya catatlah.

**Referensi :**

Winiarti, S, Diktat Kuliah Kecerdasan Buatan, 2010.

## MATERI 2
## PROBLEM SOLVING

**Waktu** **: 1.5 Jam**

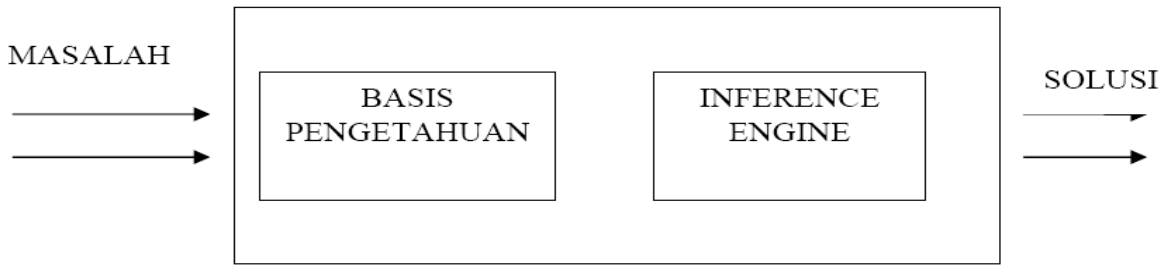
**Kompetensi Dasar** : Setelah mengikuti praktikum ini, mahasiswa dapat memahami Problem Solving dalam Kecerdasan Buatan dengan Bahasa pemrograman Prolog.

**Indikator** : mahasiswa dapat merepsentasikan masalah dengan pendekatan kecrdasan buatan dalam bahasa pemrograman Prolog.

**Teori Pendukung**

### A. Representasi Masalah

Seperti telah diketahui pada sistemyang menggunakan kecerdasan buatan akan mencoba memberikan output berupa solusi suatu masalah berdasarkan kumpulan pengetahuan yang ada ( Gambar 2.1)




Gambar 2.1 Sistem Kecerdasan Buatan

Dari gambar tersebut, input yang diberikan pada sistem yang menggunakan kecerdasan buatan berupa masalah. Pada sistem harus dilengkapai dengan sekumpulan pengetahuan yang ada pada basis pengetahuan (*knowledge base*).. Sistem harus memiliki *inference engine* agar sistem mampu mengambil kesimpulan berdasarkan fakta atau pengetahuan. Output yang diberikan berupa solusi masalah sebagai hasil dari inferensi.

Secara umum untuk membangun sistem yang mampu menyelesaikan masalah perlu mempertimbangkan 4 hal:

1. Mendefinisikan masalah dengan tepat. Pendefinisian ini mencakup spesifikasi yang tepat mengenai keadaan awal (Initial State) dan solusi yang diharapkan.
2. Menganalisis masalah serta mencari beberapa teknik penyelesaian masalah yang sesuai.
3. Merepresentasikan pengetahuan yang perlu untuk menyelesaikan masalah tersebut.
4. Memilih teknik penyelesaian masalah yang terbaik.

## B. Penyelesaian Masalah dalam AI

Dalam penyelesaian masalah dengan teknik AI menyangkut beberapa langkah yaitu:

- Analisa Masalah
- Representasi Masalah dan Pengetahuan
- Inferensi
- Penggunaan Bahasa AI

### 1. Analisa Masalah

Analisa masalah adalah langkah pertama dalam AI. Langkah ini menganalisa masalah yang dihadapi dan mengungkapkan masalah tersebut dalam satu sistem symbol. Sistem tersebut dapat merupakan *diagram, skema, graf* atau symbol-simbol yang lain. Sistem symbol ini harus diterjemahkan dalam bahasa pemrograman AI. Sistem ini harus dapat mengungkapkan dengan tepat keadaan awal (*Initial State*), keadaan akhir atau sasaran yang dituju (*Goal State*). Misal contoh pedagang mengunjungi 10 kota . Keadaan awal adalah rute perjalanan yang ada dan dapat dilukiskan sebagai berikut:

$R\ (K_1, K_2, \ldots\ldots\ldots, K_N)$

$R(K_{J1}, K_{J2}, \qquad K_{JN})$

Dengan jarak kota $K_1$ ke kota $K_{J1}$ adalah $d_{ij}$

Dimana $K_{J1}$, $K_{J2}$ adalah kota-kota dalam daftar pedagang tersebut.

Keadaan sasaran adalah salah satu rute perjalanan yang mempunyai jumlah $d_{ij}$ Minimum.

Secara umum pendefinisian masalah sebagai suatu ruang keadaan meliputi 3 hal yaitu:

- **Posisi Awal (initial State)**
- **Aturan (Rule )**
- **Tujuan (Goal)**

Contoh:

Misal permasalahan yang dihadapai adalah "permainan Catur", maka harus ditentukan:

1. **Posisi awal pada papan catur**

   Posisi awal setiap permainan catir selalu sama, yaitu semua bidak diletakkan di atas papan catur dalam 2 sisi yaitu kubu Putih dan Kubu Hitam

2. **Aturan-aturan untuk melakukan gerakan secara legal (*Rule*)**

   Aturan-aturan (rule) berguna untuk menetukan gerakan suatu bidak, yaitu melangkah dari satu keadan ke keadaan lain. Misal untuk mempermudah menunjukkan posisi bidak, setiap kotak ditunjuukan dalam huruf (a, b, c, d, e, f, g, h) pada arah horizontal dan angka (1,2,3,4,5,6,7,8) pada arah vertika. Suatu aturan untuk menggerakkan bidak dari posisi (e,2) ke (e,4) dapat ditunjukkan dengan aturan:

   IF bidak putih pada Kotak (e,2),
   AND Kotak (e,3) Kosong,
   AND Kotak(e,4) Kosong
   THEN Gerakkan bidak dari (e,2) ke (e,4)

3. **Tujuan *(Goal)***

   Tujuan yang ingin dicapai adalah posisi pada papan catur yang menunjukkan kemenagan seseorang terhadap lawannya. Kemenangan ini ditandai dengan posisi RAJA yang sudah tidak bergerak lagi.

**Tugas Praktikum :**

1. Coba ketik coding program prolog berikut ini **(Nilai : 30)** :

**Program 1 :**

```
/* Program aktivitas 1 */

domains
    person, aktivitas = symbol

predicates
    suka(person,aktivitas)

clauses
    suka(ellen,tenis).
    suka(john,football).
    suka(john,tenis).
    suka(mary,renang).
    suka(tom,tenis).
    suka(tom,basket).
    suka(eric,renang).
    suka(mary,tenis).
    suka(bill,X) if suka(tom,X).
    suka(ann,X)if suka(mary,X).
```

**Program 2 :**

```
/* Program aktivitas 1 */

domains
    person, aktivitas = symbol

predicates
    suka(person,aktivitas)

clauses
    suka(ellen,tenis).
    suka(john,football).
    suka(mary,renang).
    suka(tom,tenis).
    suka(eric,renang).
      suka(bill,X) if suka(tom,X).
      suka(ann,X)if suka(mary,X).
```

**Tugas :**

a) Cobalah membuat kedua program tersebut dengan memberikan goal di bawah ini untuk kedua program tersebut. Bagaimana hasilnya…?

```
Query 1 : suka(ellen,X),suka(tom,X)
Query 2 : suka(mary,X),suka(ann,X)
```

b) Untuk program 1 dan program 2 apakah hasilnya …?
c) Cobalah untuk memberi fakta-fakta baru untuk pengembangan.

2. Buatlah sebuah program sederhana ke dalam Prolog untuk kasus-kasus di bawah ini, catatlah hasilnya. **(nilai : 50).**

   Kalimat 1 :

   Setiap orang yang lahir di Indonesia adalah lahir di kota Jakarta.
   Anak-anak Kota Jakarta adalah anak-anak Indonesia.
   Arman adalah ayah Joko dan Dia lahir di Singapura.
   Dessy adalah ibu Joko dan Dia lahir di Indonesia.

   **Carilah solusinya** :
   Apakah Joko orang Indonesia? Siapa yang orang Jakarta?

   Kalimat 2 :

   Rima suka semua makanan.
   Apel adalah makanan. Ayam adalah makanan.
   Apapun yang dimakan orang dan ia masih hidup adalah makanan.
   Tomy makan ular dan dia masih hidup.
   Susi makan apapun yang dimakan Tomy.

   **Carilah solusinya :**
   Siapakah yang makan ular? Buktikan bahwa apel dan ayam adalah makanan !

**Referensi ;**

Winiarti, S, Diktat Kuliah Kecerdasan Buatan, 2010.

**MATERI 3**
**REPRESENTASI PENGETAHUAN**

---

**Waktu : 1.5 Jam**

**Kompetensi Dasar :** mahasiswa dapat membuat representasi pengetahuan dalam bahasa prolog.

**Indikator** : Setelah mengikuti praktikum ini, mahasiswa dapat menyajikan pengetahuan untuk model graph, dengan bahasa Prolog.

---

**Teori Pendukung :**

Pengetahuan (*Knowledge*) :

- Definisi umum : fakta atau kondisi sesuatu atau keadaan yang timbul karena suatu pengalaman.
- Cabang ilmu filsafat, yaitu *Epistemology*, berkenaan dengan sifat, struktur dan keaslian dari *knowledge*.

Teknik Representasi Pengetahuan :

1) Aturan Produksi
2) Jaringan Semantik
3) Graph
4) *Frame* dan *Scemata*
5) *Logika Proposisi*
6) *List*
7) *Tabel keputusan*

Bahasa representasi harus dapat membuat seorang programmer mampu mengekspresikan pengetahuan untuk mendapatkan solusi suatu masalah.
Secara singkat Mylopoulos dan Levesque mengklasifikasikan susunan atau pola representasi menjadi empat katagori :

*1) Representasi Logika* - Representasi ini menggunakan ekspresi-ekspresi dalam logika formal untuk merepresentasikan basis pengetahuan.
*2) Representasi Prosedural* – Menggambarkan pengetahuan sebagai sekumpulan instruksi untuk memecahkan suatu masalah. Dalam sistem yang berbasis aturan, aturan *if-then* dapat ditafsirkan sebagai sebuah prosedur untuk mencapai tujuan pemecahan masalah.

*3)Representasi Network* - Menangkap pengetahuan sebagai sebuah graf dimana simpulsimpulnya menggambarkan obyek atau konsep dalam masalah yang dihadapi, sedangkan lengkungan-lengkungannya menggambarkan hubungan atau asosiasi antar mereka. Contohnya adalah jaringan semantik.

*4) Representasi Terstruktur* - Memperluas network dengan cara membuat setiap simpulnya menjadi sebuah struktur data kompleks yang berisi tempat-tempat bernama slot dengan nilai-nilai tertentu. Nilai-nilai ini dapat merupakan data numerik atau simbolik sederhana, pointer ke bingkai (*frame*) lain, atau bahkan merupakan prosedur untuk mengerja kan tugas tertentu. Contoh : skrip (*script*), bingkai (*frame*) dan obyek (*object*).

Di dalam matematika, tidak semua kalimat berhubungan dengan logika. Hanya kalimat yang bernilai benar atau salah saja yang digunakan dalam penalaran. Kalimat tersebut dinamakan **proposisi** (*preposition*).

Proposisi adalah kalimat deklaratif yang bernilai benar (*true*) atau salah (*false*), tetapi tidak dapat sekaligus keduanya. Kebenaran atau kesalahan dari sebuah kalimat disebut nilai kebenarannya (*truth value*).

Tiga buah contoh berikut ini dapat mengilustrasikan kalimat mana yang merupakan proposisi dan mana yang bukan. Pernyataan-pernyataan berikut ini,

(a) 6 adalah bilangan genap.
(b) Soekarno adalah Presiden Indonesia yang pertama.
(c) 2 + 2 = 4.
(d) Ibukota Provinsi Jawa Barat adalah Semarang.
(e) 12 ³ 19.
(f) Kemarin hari hujan.
(g) Suhu di permukaan laut adalah 21 derajat Celcius.
(h) Pemuda itu tinggi.
(i) Kehidupan hanya ada di planet Bumi.

semuanya merupakan proposisi. Proposisi a, b, dan c bernilai benar, tetapi proposisi d salah karena ibukota Jawa Barat seharusnya adalah Bandung dan proposisi e bernilai salah karena seharusnya 12 £ 19. Proposisi f sampai i memang tidak dapat langsung ditetapkan kebenarannya, namun satu hal yang pasti, proposisi-proposisi tersebut tidak mungkin benar dan salah sekaligus. Kita bisa menetapkan nilai proposisi tersebut benar atau salah. Misalnya, proposisi f bisa kita andaikan benar (hari kemarin memang hujan) atau salah (hari kemarin tidak hujan). Demikian pula halnya untuk proposisi g dan h. Proposisi i bisa benar atau salah, karena sampai saat ini belum ada ilmuwan yang dapat
memastikan kebenarannya.

Secara simbolik, proposisi biasanya dilambangkan dengan huruf kecil seperti *p*, *q*, *r*, …. Misalnya, *p* : 6 adalah bilangan genap.

Untuk mendefinisikan *p* sebagai proposisi “6 adalah bilangan genap”. Begitu juga untuk *q* : Soekarno adalah Presiden Indonesia yang pertama.

*r* : 2 + 2 = 4.

dan sebagainya. Kita dapat membentuk proposisi baru dengan cara mengkombinasikan satu atau lebih proposisi. Operator yang digunakan untuk mengkombinasikan proposisi disebut **operator logika**. Operator logika dasar yang digunakan adalah **dan** (*and*), **atau** (*or*), dan **tidak** (*not*). Dua operator pertama dinamakan operator **biner** karena operator tersebut mengoperasikan dua buah proposisi, sedangkan operator ketiga dinamakan operator **uner** karena ia hanya membutuhkan satu buah proposisi. Proposisi baru yang diperoleh dari pengkombinasian tersebut dinamakan **proposisi majemuk** (*compound proposition*). Proposisi yang bukan merupakan kombinasi proposisi lain disebut **proposisi atomik**. Dengan kata lain, proposisi majemuk disusun dari proposisi-proposisi atomik. Metode pengkombinasian proposisi dibahas oleh matematikawan Inggris yang bernama George Boole pada tahun 1854 di dalam bukunya yang terkenal, *The Laws of Thought*. Proposisi majemuk ada tiga macam, yaitu konjungsi, disjungsi, dan ingkaran. Ketiganya didefinisikan sebagai berikut:

Misalkan *p* dan *q* adalah proposisi. **Konjungsi** (*conjunction*) *p* dan *q*, dinyatakan dengan notasi *p* Ù *q*, adalah proposisi *p* dan *q* **Disjungsi** (*disjunction*) *p* dan *q*, dinyatakan dengan notasi *p* Ú *q*, adalah proposisi *p* atau *q* **Ingkaran** atau (*negation*) dari *p*, dinyatakan dengan notasi ~*p*, adalah proposisi tidak *p*.

**Catatan:**

1. *Beberapa literatur menggunakan notasi "Øp", " p ", atau "**not** p" untuk menyatakan ingkaran.*
2. *Kata "tidak" dapat dituliskan di tengah pernyataan. Jika kata "tidak" diberikan di awal pernyataan maka ia biasanya disambungkan dengan kata "benar"menjadi "tidak benar". Kata "tidak" dapat juga diganti dengan "bukan"bergantung pada rasa bahasa yang tepat untuk pernyataan tersebut.*

Contoh :

Diketahui proposisi-proposisi berikut:

*p* : Hari ini hujan

*q* : Murid-murid diliburkan dari sekolah

maka

*p* Ù *q* : Hari ini hujan dan murid-murid diliburkan dari sekolah

*p* Ú *q* : Hari ini hujan atau murid-murid diliburkan dari sekolah

~*p* : Tidak benar hari ini hujan (atau dalam kalimat lain yang lebih lazim: Hari ini *tidak* hujan)

**Tugas Praktikum :**

Tugas a) **Nilai : 50**

1. Jalankan program **logika.pro**, caranya : pilih menu file → load →ketik logika→Enter
2. Pilih menu Run atau ketik huruf **R,** setelah muncul tanda **Goal,** berikan ekspresi logika.
3. Berikan beberapa query-query di bawah ini untuk membuktikan ekspresi logika proposisi. Catatlah hasilnya.

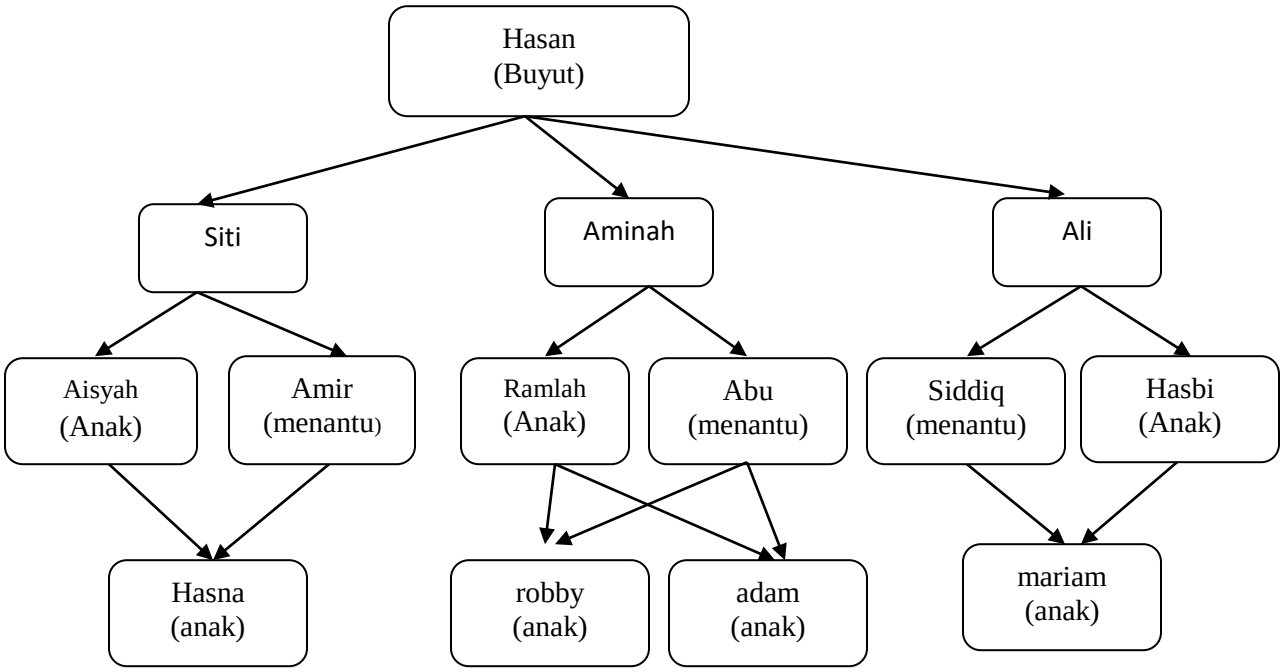
   Query 1 : not ( a and b)

Query 2 : not ( a and b) or not(a or b)
Query 3 : not(a or b) and not ( a and b)

Coding dalam Prolog sebagai berikut :

```
/***************88888888888888888*****/
/* Program sederhana Logika Proposisi  */
/*******************888888888888888888*/
predicates
 transform(string)
 start
 is_and_or(symbol)
 opposite(symbol,symbol)
goal
 start.
clauses
 start:-
 write("Masukkan ekspresi logika :"),
 readln(E),
 transform(E).
transform(S):- /* Prosedur untuk metode Assosiative */
 fronttoken(S,T,S1),
 T="(",
 fronttoken(S1,P,S2),
 fronttoken(S2,Op,S3),
 is_and_or(Op),
 fronttoken(S3,Q,S4),
 fronttoken(S4,")",S5),
 fronttoken(S5,Op2,S6),
 Op2=Op,
 fronttoken(S6,R,_),
 write(P," ",Op,"(",Q," ",Op2," ",R,")"),
 nl.
transform(S):- /* Prosedur untuk metode Demorgan */
 fronttoken(S,T,S1),
 T="not",
 fronttoken(S1,"(",S1a),
 fronttoken(S1a,P,S2),
 fronttoken(S2,Op,S3),
 is_and_or(Op),
 opposite(Op,Op2),
 fronttoken(S3,Q,S4),
 fronttoken(S4,")",_),
 write("not",P," ",Op2,"not",Q),
 nl.
 is_and_or(O):-
 O="and".
 is_and_or(O):-
 O="or".
 opposite(O,P):-
 O="and",
 P="or".
 opposite(O,P):-
 O="or",
 P="and".
```

Tugas b) **Nilai : 50** Bila diberikan sebuah pohon keluarga sebagai berikut :




Gambar 3.1. Contoh Pohon

Berdasarkan pohon keluarga tersebut cobalah representasikan ke dalam bahasa pemrograman prolog untuk mengetahui :

1. Siapakah ibu Robby dan Adam..?
2. Siapakah cucu dari Hasan…?
3. Siapakah yang disebut ayah…?

**Referensi :**

Winiarti, S, Diktat Kuliah Kecerdasan Buatan, 2010.

**MATERI 4**
**METODE SEARCHING/PELACAKAN**
***BREATH FIRST SERACH***

---

**Waktu : 1.5 jam**

**Kompetensi Dasar :** setelah mengikuti praktikum ini, mahasiswa dapat memahami dan membuat program sederhana untuk metode pelacakan dalam Kecerdasan Buatan.

**Indikator** : Membuat program sederhana dengan bahasa pemrograman prolog untuk pelacakan dengan metode Breadth First Search.

---

**Teori Pendukug**

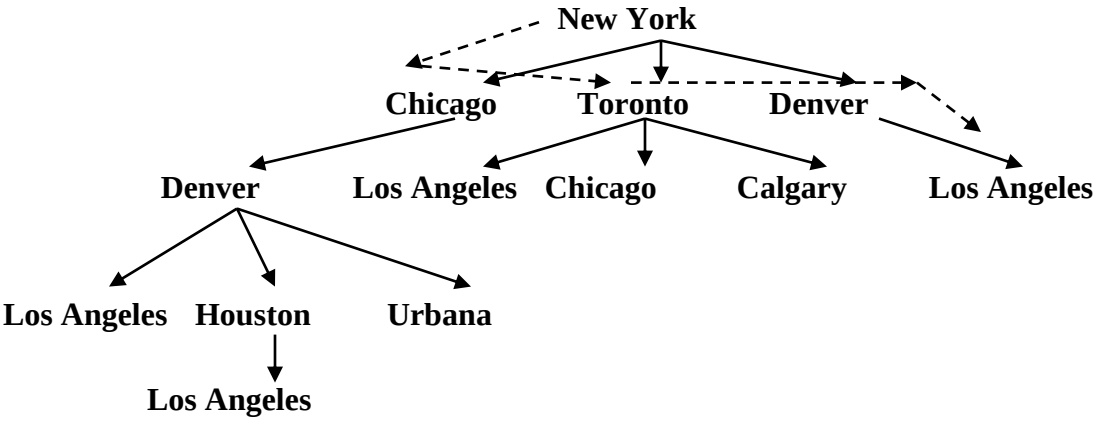
Ada 2 jenis Pelacakan dalam kecerdasan Buatan, yaitu;

1. Pelacakan Buta
   *a. Depth First Search*
   *b. Generate and test*
   *c. Breath First Search*
2. Pelacakan Heuristik
   *a. Hill Climbing*
   *b. Best First Search*

Pada praktikum ini hanya akan dibahas beberapa saja sebagai sampel penerapan dari pelacakan daam kecerdasan buatan.

**Breath First Search (BFS)**

Pencarian yang dimulai dari node akar terus ke level berikutnya dari kiri ke kanan, kemudian berpindah ke level berikutnya. Contoh pelacakan dengan metode BFS.




Gambar 4.1. contoh Pelacakan BFS

**Tugas Praktikum 1 ( Nilai 30)**

Berdasarkan gambar 5.1 dari contoh pelacakan BFS, bila diimplementasikan ke dalam bahasa prolog, codingnya sebagai berikut **(silahkan load nama file ; BFS.pro);**

```
/* Best - First Search */

database
  flight(symbol,symbol,integer)
  visited(symbol)

predicates
  route(symbol,symbol,integer)
  isflight(symbol,symbol,integer)
  displayroute
  purge
  findroute
  add_to_route(symbol)

goal
  assert(flight(newyork, chicago, 1000)),
  assert(flight(chicago, denver, 1000)),
assert(flight(newyork, toronto, 800)),
assert(flight(newyork, denver, 1900)),
assert(flight(toronto, calgary, 1500)),
assert(flight(toronto, los_angeles, 1800)),
assert(flight(toronto, chicago, 500)),
assert(flight(denver, urbana, 1000)),
assert(flight(denver,houston, 1500)),
assert(flight(houston, los_angeles, 1500)),
assert(flight(denver, los_angeles,1000)),
findroute,nl,
write("Lagi?"),
readln(Q),
Q=n.

clauses
findroute:-
write("Asal: "),
readln(A),
write("Tujuan :"), readln(B),
route(A,B,D),
write("Jaraknya adalah",D),nl,
not(displayroute).

route(A,B,C):-
isflight(A,B,C).
```

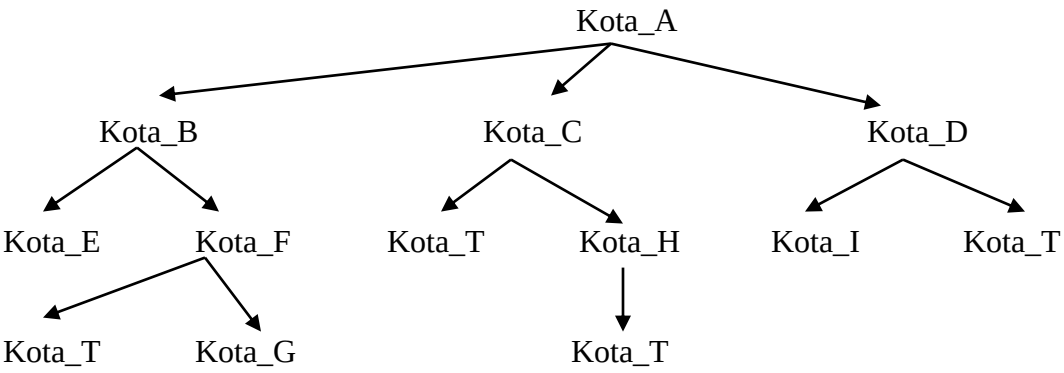
```
route(_,_,D):-
write("Maaf Rute tidak ditemukan"),
nl,
write("Masukkan jarak yang lebih spesifik:"),nl,
D=0, purge.

isflight(T,T2,D):-
flight(T,T2,D),
add_to_route(T).

isflight(T,T2,D):-
flight(T,X,D2),
X <> T2,
add_to_route(T),
isflight(X,T2,D3),
D=D2+D3.

isflight(T,_,D):-
write("pelacakan gagal !", T),
nl, D=0, fail.

add_to_route(T):-
not(visited(T)),
assert(visited(T)),!.
add_to_route(_).

purge:-
retract(visited(_)),
fail,!.

displayroute:-
write("route adalah : "), nl,
visited(A),
write(A),nl,
fail,!.
```

Setelah program di Run hasilnya sebagai berikut :


**Tugas Praktikum 2 ( Nilai 70)**

Setelah anda menjalankan dan mencoba berinteraksi dengan system, cobalah membuat sebuah program yang dikembangkan dari listing 4.1 di atas dengan mengacu pada pohon pelacakan berikut ini : Bila Kota asal : Kota_A

Tujuan : Kota_T




Gambar 4.2. Pohon pelacakan untuk kasus ke-2 BFS

**Referensi ;**

Winiarti, S, Diktat Kuliah Kecerdasan Buatan, 2010.

**MATERI 5**
**METODE SEARCHING/PELACAKAN**
***DEPTH FIRST SEARCH***

**Waktu :** 1.5 Jam

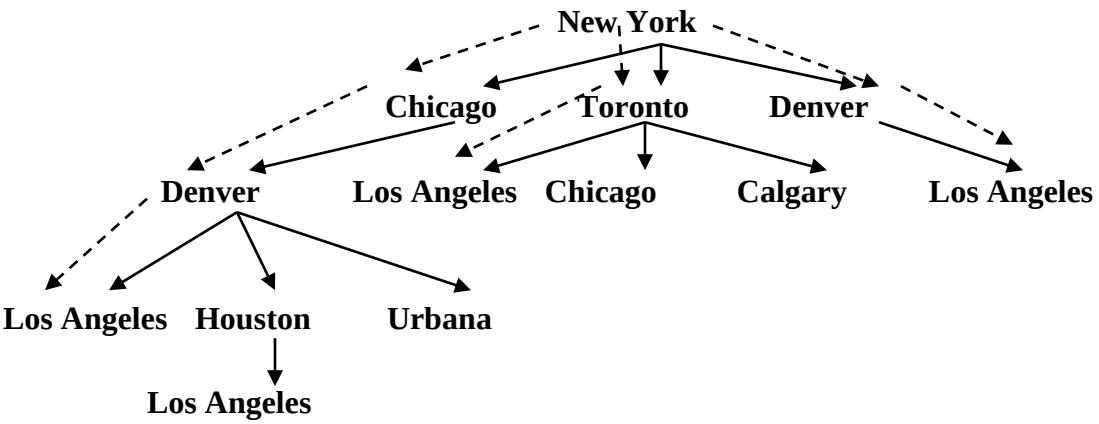
**Kompetensi Dasar :** setelah mengikuti praktikum ini, mahasiswa dapat memahami dan membuat program sederhana untuk metode pelacakan dalam Kecerdasan Buatan.

**Indikator** : Membuat program sederhana dengan bahasa pemrograman prolog untuk pelacakan dengan metode *Depth First Search*.

**Teori Pendukug**

***Depth First Search* (DFS)**

Pencarian yang dilakukan pada semua anaknya sebelum dilakukan pencarian ke node-node yang selevel. Pencarian dimulai dari node akar ke level yang lebig tinggi. Proses ini diulangi terus hingga ditemukan solusinya. Contoh pelacakan dengan metode DFS.




Gambar 5.1. contoh Pelacakan DFS

Pada praktikum kali ini, kita akan mencoba untuk membuat program dengan prolog untuk pelacakan DFS, untuk melihat alur pelacakannya. Untuk mempermudah praktikum ini, silahkan Anda buka kembali Program file **BFS.Pro** yang telah dibuat pada Praktikum 4. Untuk pelacakan dengan DFS maka ada beberapa procedur yang perlu ditambah kan pada **isflight**. Cobalah untuk melihat hasil program DFS.pro pada listing 5.1.

```
/* Depth - First Search */

database
  flight(symbol,symbol,integer)
  visited(symbol)

predicates
  route(symbol,symbol,integer)
  isflight(symbol,symbol,integer)
  displayroute
  purge
  findroute
  add_to_route(symbol)

goal
  assert(flight(newyork, chicago, 1000)),
  assert(flight(chicago, denver, 1000)),
  assert(flight(newyork, toronto, 800)),
  assert(flight(newyork, denver, 1900)),
  assert(flight(toronto, calgary, 1500)),
  assert(flight(toronto, los_angeles, 1800)),
  assert(flight(toronto, chicago, 500)),
  assert(flight(denver, urbana, 1000)),
  assert(flight(denver,houston, 1500)),
  assert(flight(houston, los_angeles, 1500)),
  assert(flight(denver, los_angeles,1000)),
findroute,nl,
  write("Lagi?"),
  readln(Q),
  Q=n.

clauses
  findroute:-
  write("Asal: "),
  readln(A),
  write("Tujuan :"), readln(B),
  route(A,B,D),
  write("Jaraknya adalah",D),nl,
  not(displayroute).

  route(A,B,C):-
  isflight(A,B,C).

  route(_,_,D):-
  write("Maaf Rute tidak ditemukan"),
  nl,
  write("Masukkan jarak yang lebih spesifik:"),nl,
D=0, purge.
```

```
isflight(T,T2,D):-
flight(T,T2,D),
add_to_route(T).

isflight(T,T2,D):-
flight(T,X,D2),
flight(X,T2,D3),
add_to_route(T),
add_to_route(X),
D=D2+D3.

isflight(T,T2,D):-
flight(T,X,D2),
X<>T2,
add_to_route(T),
isflight(X,T2,D3),
D=D2+D3.
isflight(T,_,D):-
write("pelacakan gagal !", T),
nl, D=0, fail.

add_to_route(T):-
  not(visited(T)),
  assert(visited(T)),!.
 add_to_route(_).

 purge:-
   retract(visited(_)),
   fail,!.

displayroute:-
  write(" rutenya : "), nl,
  visited(A),
  write(A),nl,
  fail,!.
```

Setelah Program di Run, hasilnya sebagai berikut :


Gambar 5.2. Tampilan Prolog

**Tugas Praktikum :**

1. Bukalah Program BFS.pro, kemudian lakukan modifikasi untuk membuat pelacakan DFS dengan menambah procedure pada coding tersebut. Setelah disimpan coba jalankan. Isikan kota asal : newyork, kota tujuan : los_angeles. (ingat penulisan nama kota hatus sama dengan yang ada di predicates). Bagaimana hasilnya…? (**Nilai 20)**
2. Untuk kasus kedua sama dengan membuat program untuk kasus BFS pada praktikum 4 ( gambar 4.1). catatlah solusinya. **(Nilai 80)**

**Referensi :**

Winiarti, S, Diktat Kuliah Kecerdasan Buatan, 2010.

**MATERI 6**
**METODE SEARCHING/PELACAKAN**
***HILL CLIMBING***

---

**Waktu : 1.5 Jam**

**Kompetensi Dasar :** setelah mengikuti praktikum ini, mahasiswa dapat memahami dan membuat program sederhana untuk metode pelacakan dalam Kecerdasan Buatan.

**Indikator :** Membuat program sederhana dengan bahasa pemrograman prolog untuk pelacakan dengan metode *Hill Climbing*.

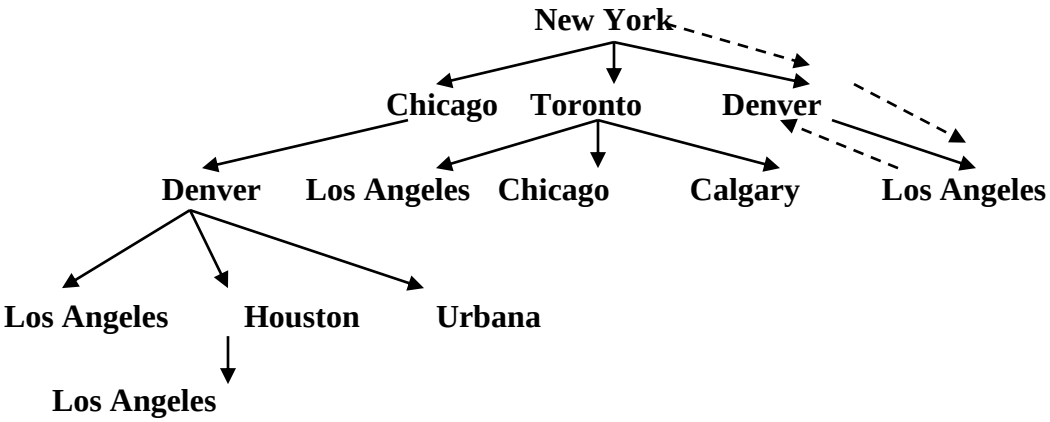
---

**Teori Pendukug**

**Hill Climbing**

Metode ini merupakan jenis pelacakan heuristic, karena dalam pencarian solusinya, ia selalu mempertimbangkan node yang memiliki nilai terbaik atau dengan model bertahap.

- Merupakan metode pelacakan yang mengkombinasikan pelacakan *Generate and Test* dengan *Backtracking*.
- Untuk langkah awal pelacakan dengan memilih node yang memiliki nilai terbaik/terbesar (fungsi heuristik). Contoh pelacakan dengan metode *Hill Climbing*.




Gambar 6.1. contoh Pelacakan *Hill Climbing*

Pada gambar 6.1 terlihat bahwa system meberikan hasil optimal dari newyork → denver → los_angeles. Hal ini karena jarak newyork ke denver memiliki nilai terbaik/terbesar, yakni 1900 bila dibandingkan newyork ke Toronto yang memiliki nilai = 800, dan newyork Chicago bernilai = 1000.

Untk membuat pelacakan dengan *Hill Climbing* , maka prosedur yang ditambahkan pada bagian **isflight** dengan menambah prosedur **findlargest.** Berikut listingnya:

```
/* Hill Climbing Search */

database
  flight(symbol,symbol,integer)
  visited(symbol)

predicates
  route(symbol,symbol,integer)
  isflight(symbol,symbol,integer)
  displayroute
  purge
  findroute
  add_to_route(symbol)
  findlargest(symbol,symbol)

goal
  assert(flight(newyork, chicago, 1000)),
  assert(flight(chicago, denver, 1000)),
  assert(flight(newyork, toronto, 800)),
  assert(flight(newyork, denver, 1900)),
  assert(flight(toronto, calgary, 1500)),
  assert(flight(toronto, los_angeles, 1800)),
  assert(flight(toronto, chicago, 500)),
  assert(flight(denver, urbana, 1000)),
  assert(flight(denver,houston, 1500)),
  assert(flight(houston, los_angeles, 1500)),
  assert(flight(denver, los_angeles,1000)),
findroute,nl,
write("Lagi?"),not(purge),
readln(Q),
Q=n.

clauses
 findroute:-
 write("Asal: "),
 readln(A),
 write("Tujuan :"), readln(B),
 route(A,B,D),
 write("Jaraknya adalah",D),nl,
 not(displayroute).
route(A,B,C):-
isflight(A,B,C).

route(_,_,D):-
write("Maaf Rute tidak ditemukan"),
nl,
```

```
write("Masukkan jarak yang lebih spesifik:"),nl,
D=0, purge.

isflight(T,T2,D):-
flight(T,T2,D),
add_to_route(T).

isflight(T,T2,D):-
  findlargest(T,X),
  add_to_route(T),
  flight(T,X,D2),
  isflight(X,T2,D3),
  D=D2+D3.

isflight(T,T2,D):-
flight(T,X,D2),
X<>T2,
add_to_route(T),
isflight(X,T2,D3),
D=D2+D3.

findlargest(A,B):-
  flight(A,X,D),
  flight(A,Y,D2),
  X<>Y,
  D2>D,
  B=Y.

add_to_route(T):-
  not(visited(T)),
  assert(visited(T)),!.
 add_to_route(_).

 purge:-
   retract(visited(_)),
   fail,!.

displayroute:-
  write(" rutenya : "), nl,
  visited(A),
  write(A),nl,
  fail,!.
```

Hasilnya : bila di Run sebaia berikut.


Gambar 6.2. Tampilan prolog setlah program di run

**Tugas Praktikum ;**

1. Bukalah Program BFS.pro, kemudian lakukan modifikasi untuk membuat pelacakan *Hill Climbing* dengan menambah procedure pada coding tersebut. Setelah disimpan coba jalankan. Isikan kota asal : newyork, kota tujuan : los_angeles. (ingat penulisan nama kota hatus sama dengan yang ada di predicates). Bagaimana hasilnya…? **(Nilai 20)**
2. Untuk kasus kedua sama dengan membuat program untuk kasus BFS pada praktikum 4 ( gambar 4.1) atau kasus DFS pada praktikum 5 (listing 5.1). Catatlah solusinya. Ubah ke dalam *Hill Climbing*. Bagaimanakah solusinya…? **(Nilai 80).**

**Referensi ;**

Winiarti, S, Diktat Kuliah Kecerdasan Buatan, 2010.

# MATERI 7
# PENGENALAN APLIKASI CERDAS DENGAN MATLAB
## *PENGENALAN APLIKASI IMAGE PROCESSING*

**Alokasi Waktu** **: 1.5 Jam**

**Kompetensi Dasar** **:** setelah mengikuti praktikum ini, mahasiswa dapat ;

1. Mengetahui tools yang dipakai untuk menyelesaikan permasalahan yang berhubungan pengolahan citra
2. Mempunyai pengetahuan dasar tentang GUI MATLAB sebagai alat bantu dalam menyelesaikan permasalahan pengolahan citra

**Indikator** **:** Membuat dan mendesain GUI di Matlab secara sederhana.

### A. TEORI PENDUKUNG

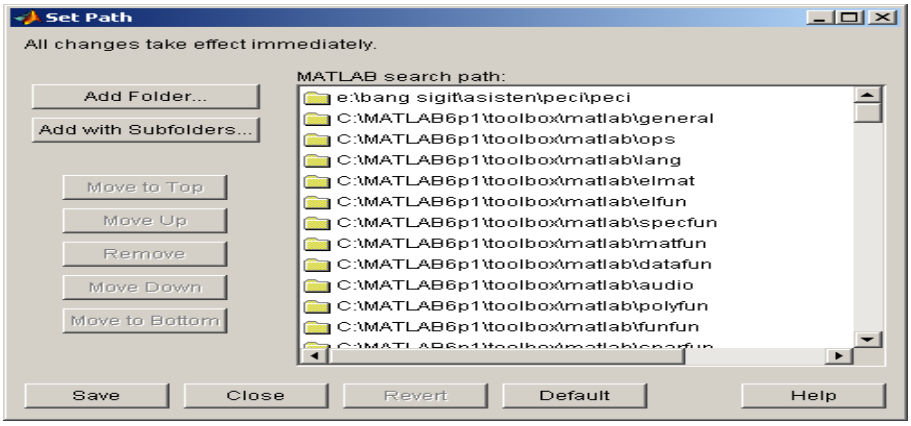
Matlab merupakan salah satu tools dalam pengolahan citra yang banyak digunakan.

### B. LANGKAH PRAKTIKUM

1. Mengadakan Pretest <alokasi waktu :15 menit> : Soal terpisah
2. Tahapan Praktikum
   a. Pengenalan perintah-perintah dalam Matlab
   b. Praktikan mencoba contoh –contoh pemakaian matlab sebagai berikut:

**Langkah penggunaan matlab:**

1. Buka Matlab dengan cara double clik didekstop icon Matlab
2. Memilih direktori kerja.
   - Buatlah sebuah direktori dengan identitas anda di D:
   - Kemudian klik pada menu **File**, pilih **Set Path** maka akan muncul window seperti ini :



Gambar 7.1. tampilan Matlab 7.0

- Tekan tombol **Add Folder**
- Pada window **Browse For Folder,** arahkan direktorinya ke direktori tempat anda membuatnya. Kemudian klik **OK**.
- Lalu tekan **Save**. Maka akan muncul *Full Path* dari direktori yang telah ada set.
- Kemudian pada window utama Maltlab, pada toolbar terdapat kolom **Curent Directory,** arahkan ke direktori yang telah kita buat tadi.
- Setelah masuk maka kita telah berada didirektori kerja kita, karena default Matlab adalah **C:\MATLAB6\work.**

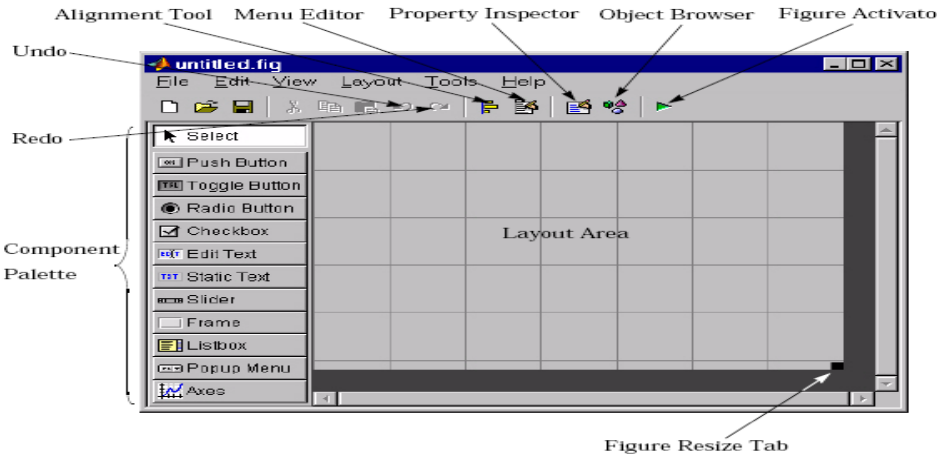
### 3. Mendesain GUI

Desain GUI dalam Matlab bisa dibuka dengan cara :

Dari **Command Window** dapat langsung diketikkan kata **guide atau**

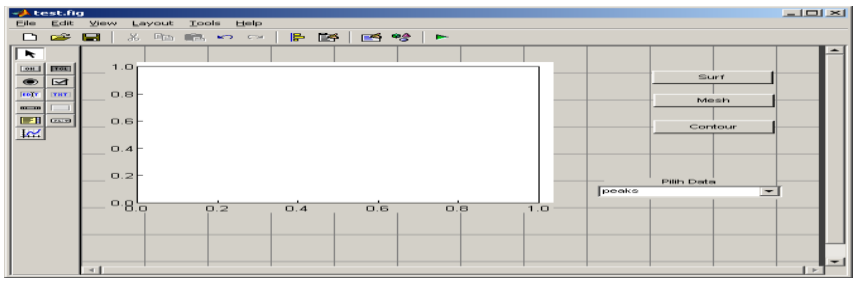
- Pilih dari menu **File** kemudian **New** dan pilih **GUI.**

Maka akan muncul window seperti dibawah ini (dengan penambahan keterangan) :



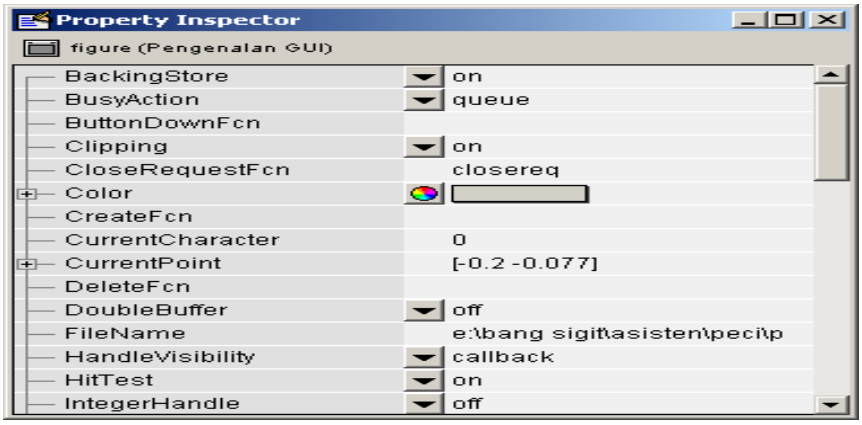

Gambar 7.2. Tampilan menu Editor matlab

Sekarang buatlah sebuah desain seperti di bawah ini :



Gambar 7.3. Tampilan menu Editor matlab untuk memulai desain

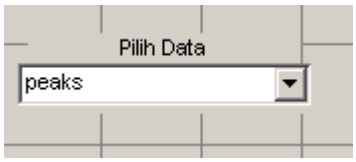
Untuk pengaturan propertisnya (pilih Property Inspector), sehingga muncul gambar seperti berikut :




Gambar 7.3. Tampilan Properti matlab

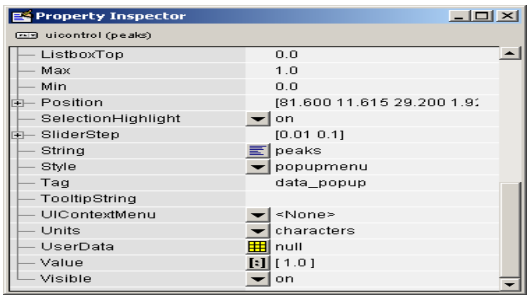
Langkah berikutnya adalah :

- Pada **Property Inspector**, carilah propertis dengan tulisan **String**, kemudian klik lalu isikan dengan **Surf. P**ropertis **Tag**, isikan dengan **surf_pushbutton.** Kemudian gulung propertis ke atas, cari propertis ***Callback***, lalu isikan dengan %**automatic.**
- Untuk tombol kedua selanjutnya isikan sebagai berikut :
  Property **String** : **Mesh**. Properties **Tag** : **mesh_pushbutton**, **Callback** : %**automatic**
- Untuk Tombol **Contour,** Pilih Property **String** : **Contour**, **Tag**: **contour_pushbutton**, Callback : **%automatic**
- Untuk tomol objek *Popup Menu* :

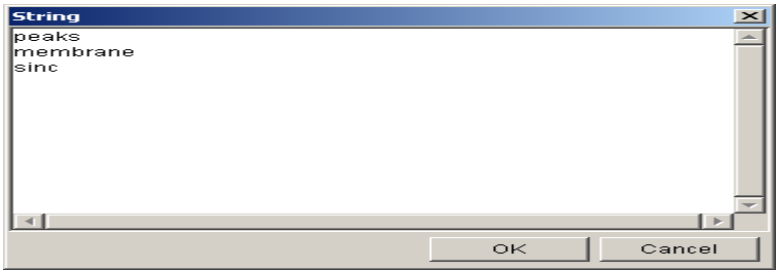



Tambahkan sebuah text diatasnya dengan string **Pilih Data**

Pada objek Popup terdapat pengaturan propertis yang berbeda, diantaranya pada string kita akan mengisi sebanyak tiga data string yaitu : **peaks**, **membrane** dan **sinc.** Dengan cara :




Pada property string kliklah tombol yang berada disebelah kanannya, hingga muncul sebuah form seperti di bawah ini :




Kemudian ketiklah data string dalam form diatas dengan data : **peaks, membrane** dan **sinc.** Kemudian tekan OK.

Pada properti **tag** isikan dengan **data_popup,** dan **Callback** dengan **%auotmatic.** Untuk grafiknya pakailah pada objek dengan nama **Axes** dan tidak perlu mengubah properti yang ada. Aturlah besarnya objek tersebut sesuai selera anda. Kemudian simpan dengan nama file yang anda kehendaki. Maka otomatis akan terbentuk sebuah file yang berekstensi **.m** beserta dengan script yang akan digunakan untuk mengaktifkan dan menjalankan interface tersebut. Misal disimpan dengan nama **test.m.** Untuk mengkoding ikuti aturan berikut :

```
function simple_gui_OpeningFcn(hObject, eventdata, handles, varargin)

handles.peaks=peaks(35);
handles.membrane=membrane;
[x,y] = meshgrid(-8:.5:8);
r = sqrt(x.^2+y.^2) + eps;
sinc = sin(r)./r;
handles.sinc = sinc;
handles.current_data = handles.peaks;
surf(handles.current_data)
```

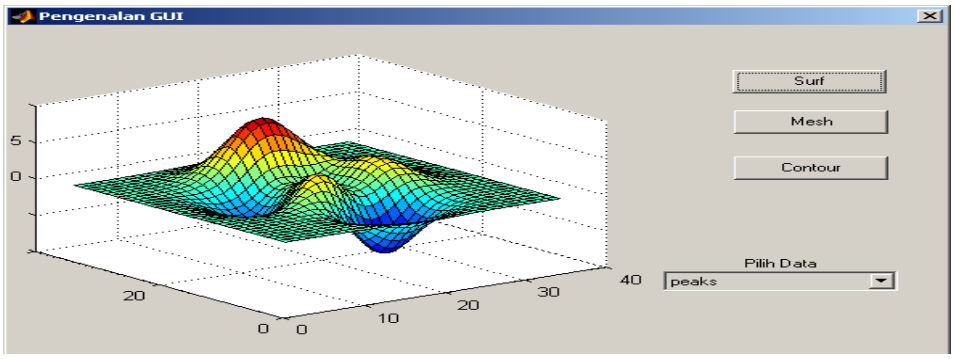
Kemudian lengkapi script yang lain dengan coding dibawah ini :

```
function surf_pushbutton_Callback(hObject, eventdata, handles)
surf(handles.current_data);
function mesh_pushbutton_Callback(hObject, eventdata, handles)
mesh(handles.current_data);
function surf_pushbutton_Callback(hObject, eventdata, handles)
contour(handles.current_data);


function plot_popup_Callback(hObject, eventdata, handles)
val = get(hObject,'Value');
```

```
str = get(hObject, 'String');
switch str{val};
case 'peaks'
   handles.current_data = handles.peaks;
case 'membrane'
   handles.current_data = handles.membrane;
case 'sinc'
   handles.current_data = handles.sinc;
end
guidata(hObject,handles)
```

Jika sudah selesai simpan hasil kerja anda dan coba anda jalankan interface anda dengan cara menklik tanda segitiga berwarna hijau atau dari menu **Tools -> Activate Figure** atau langsung tekan kombinasi **Ctrl + T.**

Jika ada yang salah atau error, silahkan tanya asisten atau coba anda cari penyebab kesalahannya. Jika suskes coba anda jalankan semua tombol dan fungsi yang telah kita buat tadi, contoh hasilnya adalah lihat dibawah ini .




Gambar 7.6 tampilan hasil keluaran dari Matlab

## C. EVALUASI PRAKTIKUM

Kerjakan langkah-langkah di dalam petunjuk praktikum untuk mendapatkan gambar 1.6.

## D. DAFTAR PUSTAKA


[1]. Gonzales, Woods, 1992, *Digital Image Processing*, A-Wesley Publishing, New York

[2]. Mathworks, 2003, *Image Processing Toolbox For Use With Matlab*, The Mathworks Inc, MA.

[3]. www.mathworks.com

**MATERI 8**
**PENGENALAN APLIKASI JARINGAN SYARAF TIRUAN (JST)**

**Waktu : 1,5 Jam**

**Kompetensi Dasar :** Mendesain topologi JST, melkukan konfigurasi neuron, serta melakukan training dan test dataset

**Indikator :**

1. Mengetahui tools yang dipakai untuk menyelesaikan permasalahan yang berhubungan Jaringan Syarat Tiruan
2. Mempunyai pengetahuan dasar tentang topologi JST, konfigurasi neuron, dan melakukan training dan test dataset

## A. TEORI PENDUKUNG

Jaringan syaraf adalah merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut [1]. Karakteristik jaringan syaraf tiruan ditentukan oleh [2]:

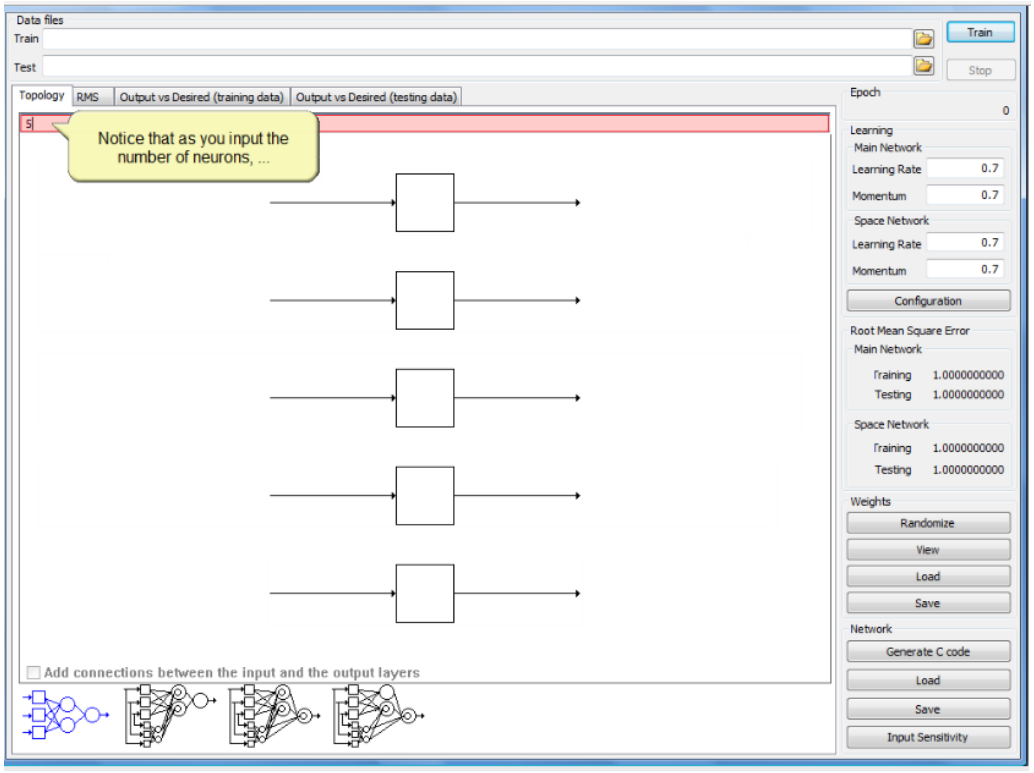
1. Pola hubungan antar-neuron (disebut dengan arsitektur/topologi jaringan)
2. Metode penentuan bobot-bobot sambungan (disebut dengan pelatihan atau proses belajar jaringan)
3. Fungsi aktivasi

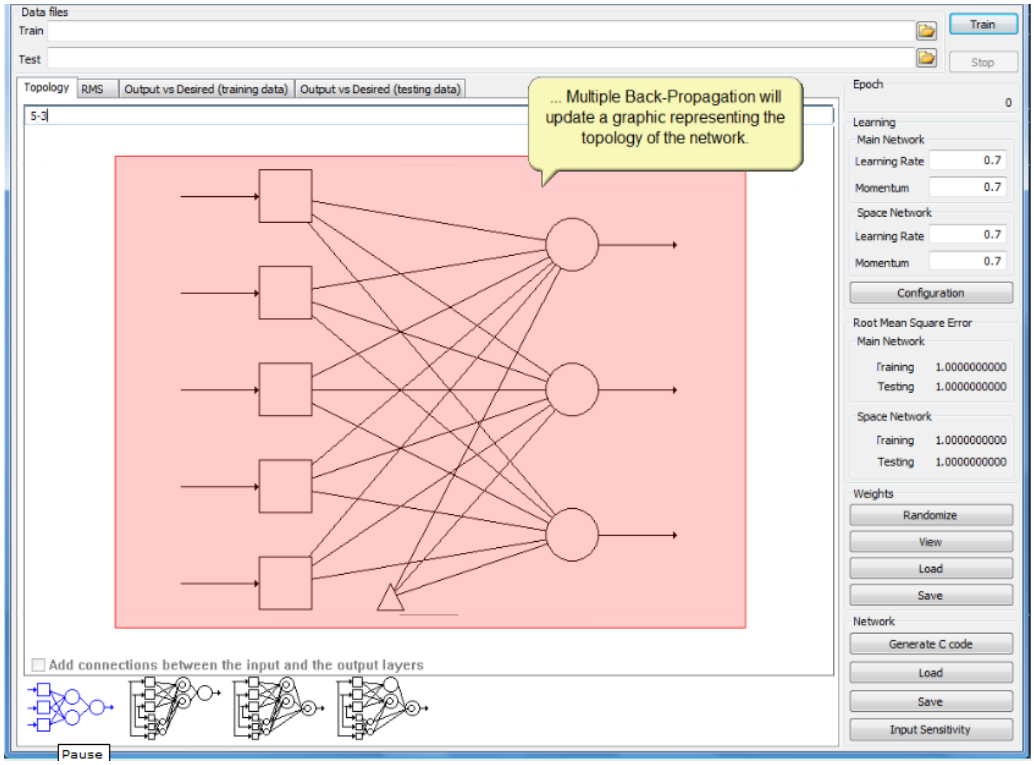
Model-model jaringan syaraf tiruan terdiri dari Hebb, Perceptron, Adaline/Madaline, Backpropagation, Hopfield [2]. Pada praktikum ini akan menggunakan aplikasi software Multiple Back-Propagation (MBP) yang didevelop oleh Noel Lopez (Polytecnic of Guarda, Portugal). Software ini gratis yang dirilis di bawah GPL v3 licence untuk keperluan pelatihan tentang JST dengan model Backpropagation dan menggunakan algoritma Multiple Backpropagation [3].

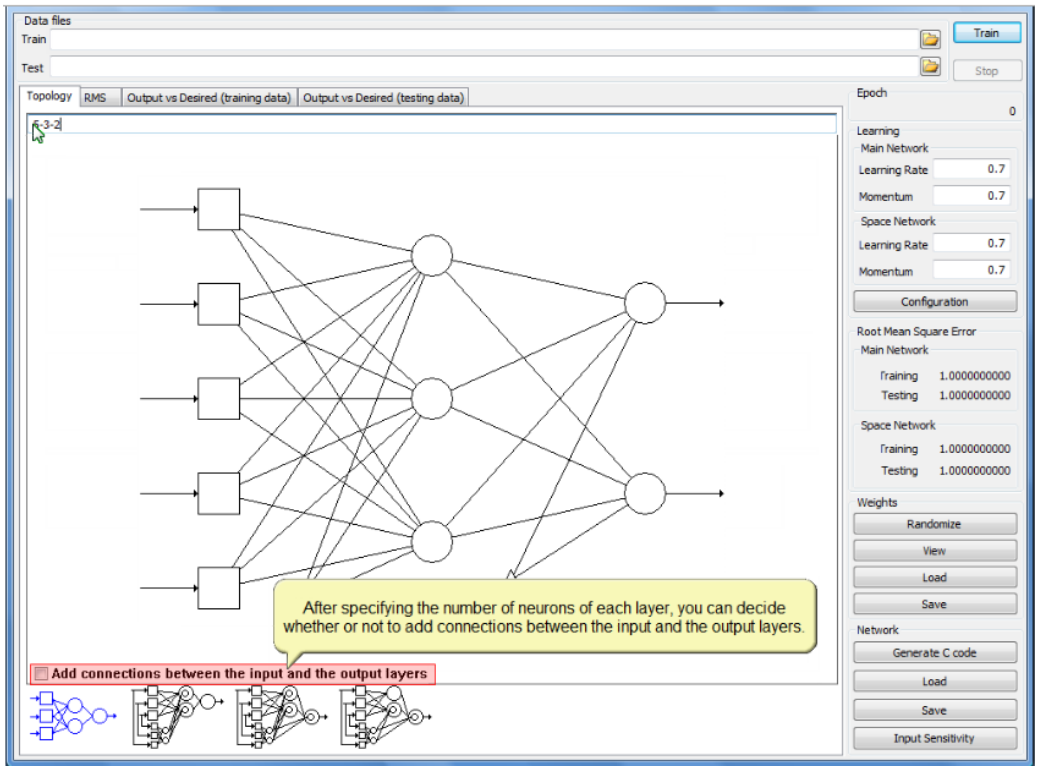
## B. LANGKAH PRAKTIKUM

1. Mengadakan Pretest <alokasi waktu :15 menit> : Soal terpisah
2. Tahapan Praktikum
   a. Pengenalan software MBP
   b. Praktikan mencoba contoh –contoh pemakaian MBP sebagai berikut:

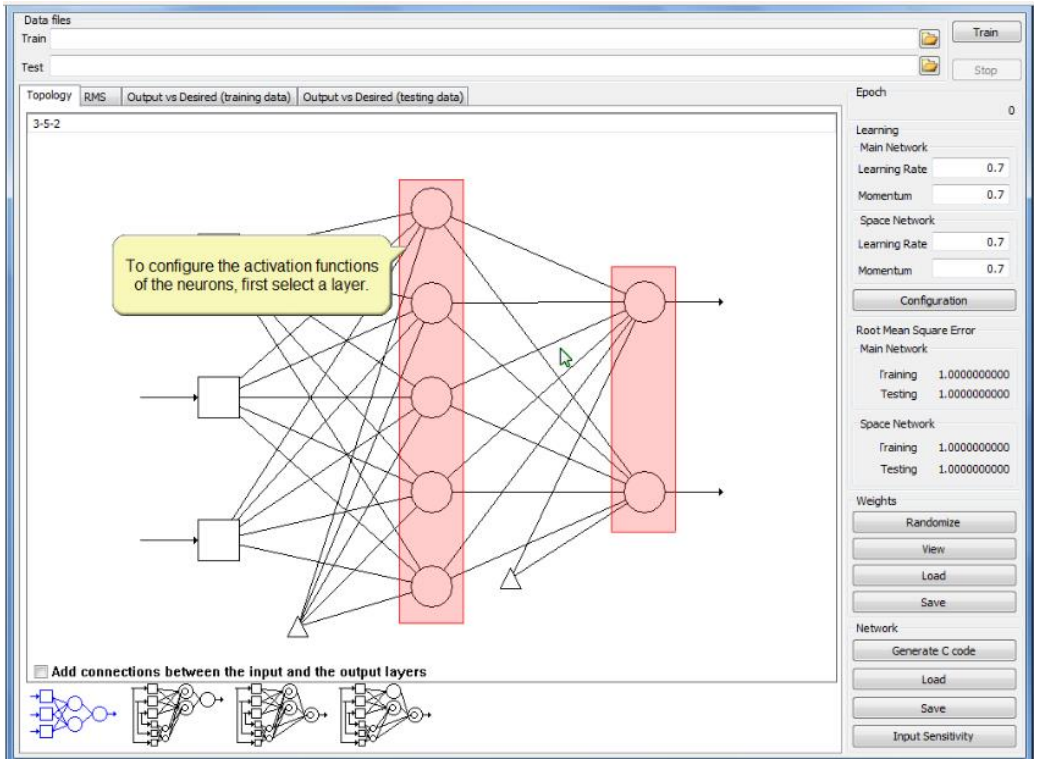
## 1) Desain topologi neural network

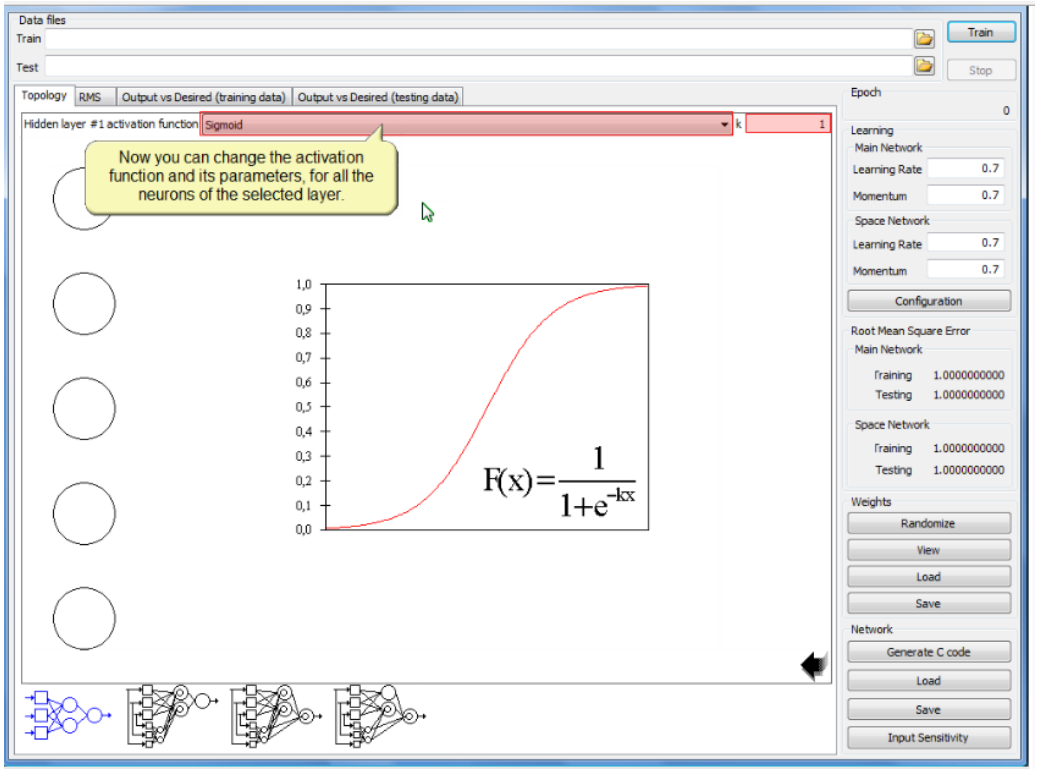


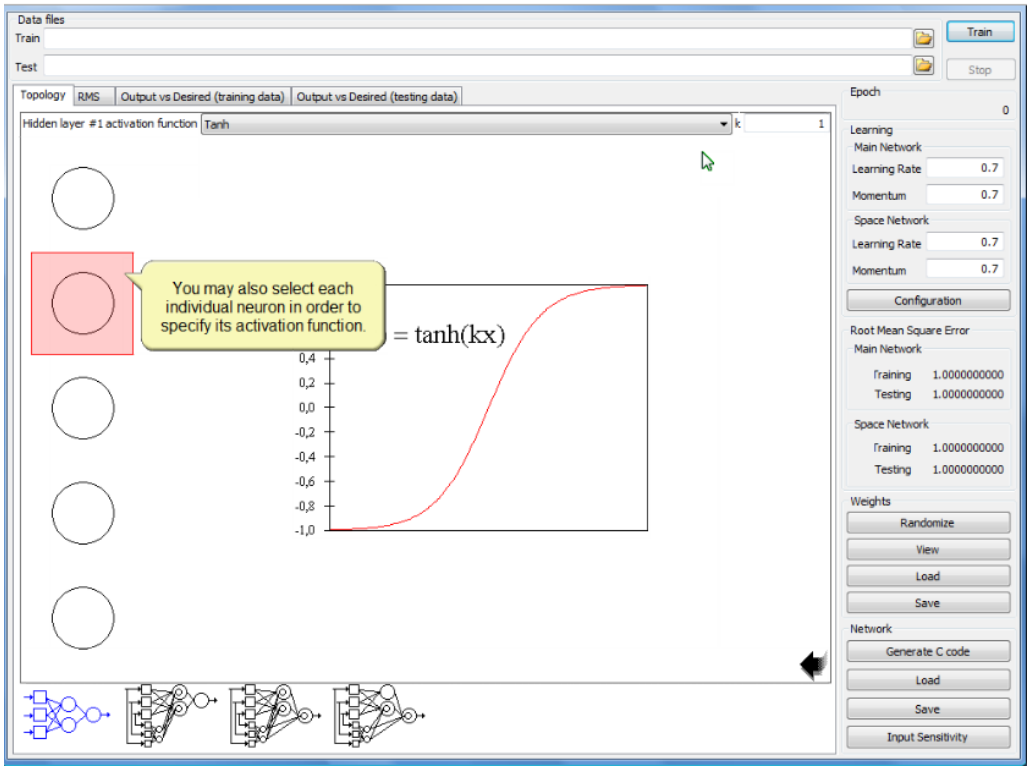





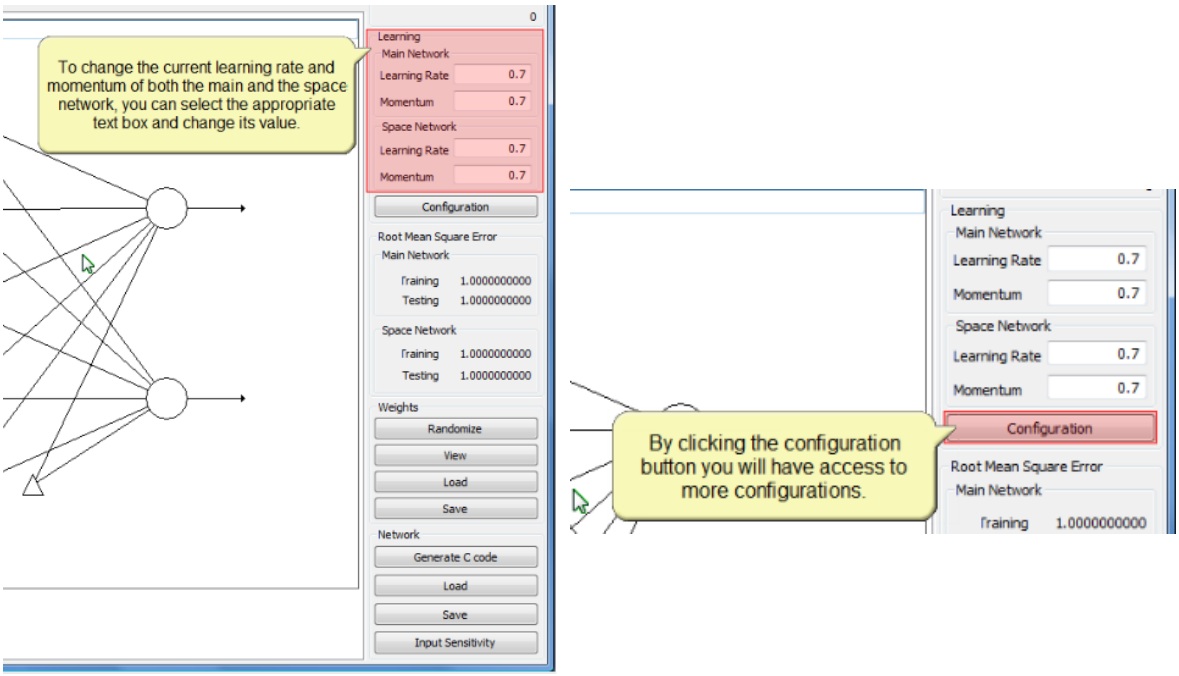

## 2) Konfigurasi fungsi aktivasi neurons

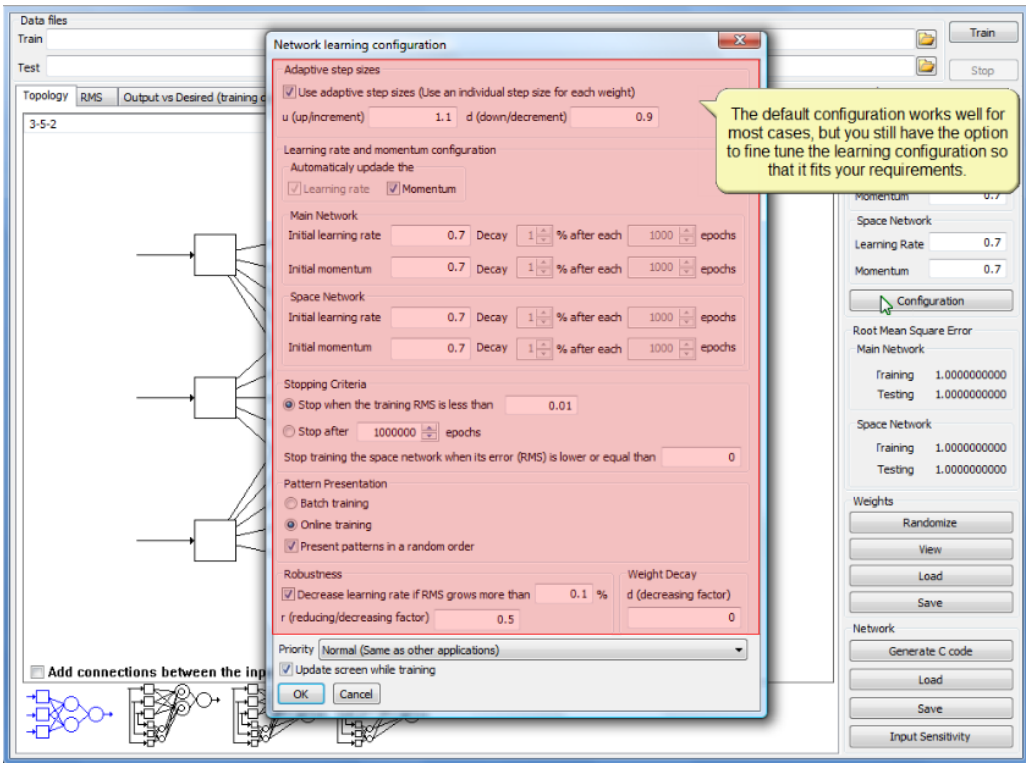



Data files
Train
Test
Train
Stop
Topology
RMS
Output vs Desired (training data)
Output vs Desired (testing data)
Hidden layer #1 activation function
Sigmoid
k
1
Now you can change the activation function and its parameters, for all the neurons of the selected layer.
1,0
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0,0
F(x)=1/(1+e^-kx)
Epoch
0
Learning
Main Network
Learning Rate
0.7
Momentum
0.7
Space Network
Learning Rate
0.7
Momentum
0.7
Configuration
Root Mean Square Error
Main Network
Training
1.0000000000
Testing
1.0000000000
Space Network
Training
1.0000000000
Testing
1.0000000000
Weights
Randomize
View
Load
Save
Network
Generate C code
Load
Save
Input Sensitivity


Data files
Train
Test
Train
Stop
Topology
RMS
Output vs Desired (training data)
Output vs Desired (testing data)
Hidden layer #1 activation function
Tanh
k
1
You may also select each individual neuron in order to specify its activation function.
= tanh(kx)
0,4
0,2
0,0
-0,2
-0,4
-0,6
-0,8
-1,0
Epoch
0
Learning
Main Network
Learning Rate
0.7
Momentum
0.7
Space Network
Learning Rate
0.7
Momentum
0.7
Configuration
Root Mean Square Error
Main Network
Training
1.0000000000
Testing
1.0000000000
Space Network
Training
1.0000000000
Testing
1.0000000000
Weights
Randomize
View
Load
Save
Network
Generate C code
Load
Save
Input Sensitivity

## 3) Mendefinisikan konfigurasi learning neural network

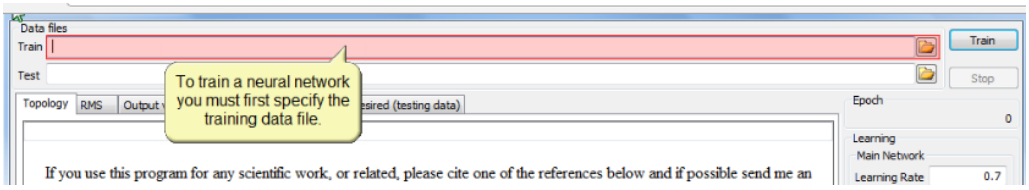


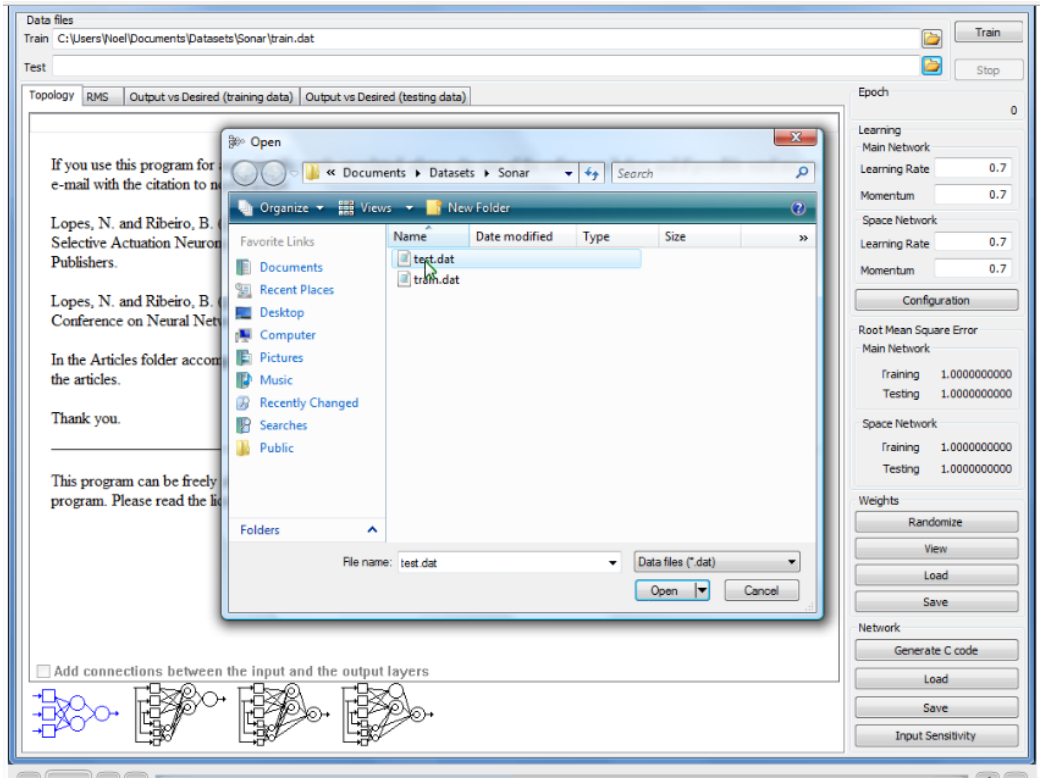
1. Disable (un check) "*Use adaptive step sizes (use an individual step size for each weight*)", sehingga konfigurasi "Larning rate" dan "momentum" aktif (*enable*) agar secara otomatis melakukan *update*
2. Apabila "lerning rate" dan "momentum" dalam keadaan tidak aktif (disable), maka persentase *decay* (error) dapat untuk "*learning rate*" dan "*momentum*" dapat ditentukan secara manual
3. Area **Stopping Criteria** digunakan untuk menentukan kapan proses training berakhir, dengan pilihan kriteria:
   a. *Stop when the training RMS is less than* …. , proses training berhenti ketika root mean square (RMS) berhenti ketika nilainya kurang dari yang diberikan oleh *user*, atau
   b. *Stop after …. Epochs,* proses training berhenti berdasarkan nilai epochs
4. Enabled "*update screen while training*".

## 4) Training Neural Network

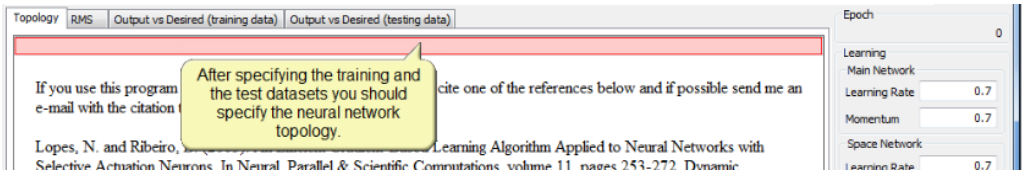
a. Untuk melakukan training nural network, harus ditentukan terlebih dahulu data train-nya

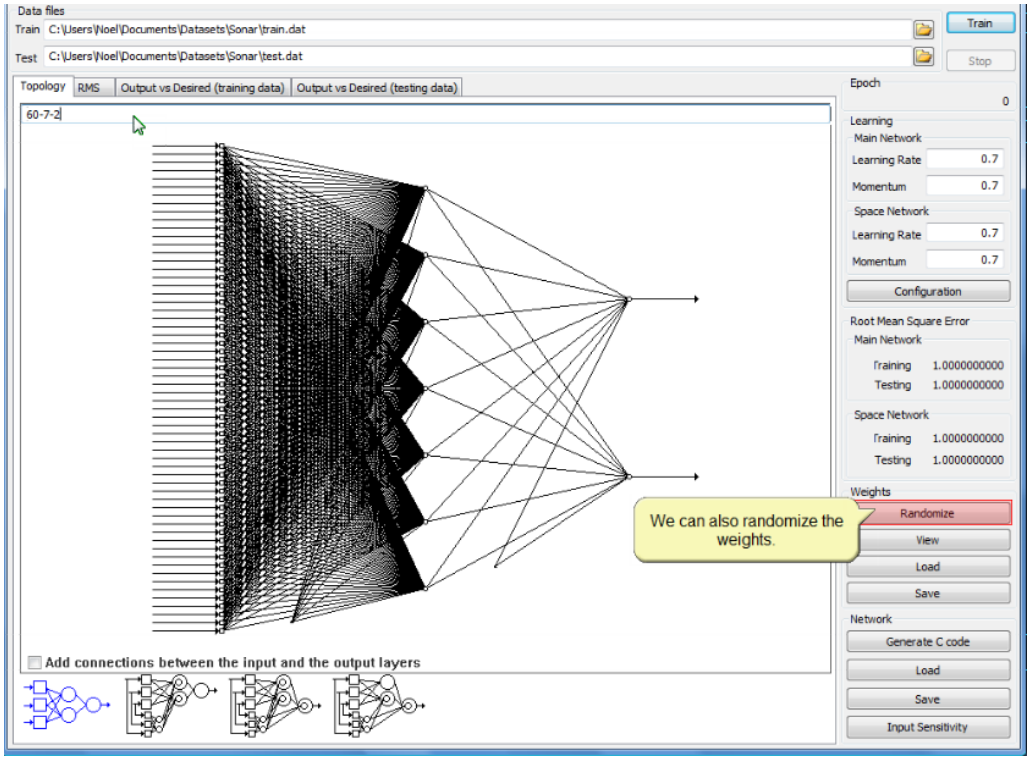



b. Secara default MBP akan meminta file data anda dengan ekstensi *.dat, namun anda dapat menentukan tipe data lainnya, seperti *.txt . Selanjutnya tentukan juga file data testnya.

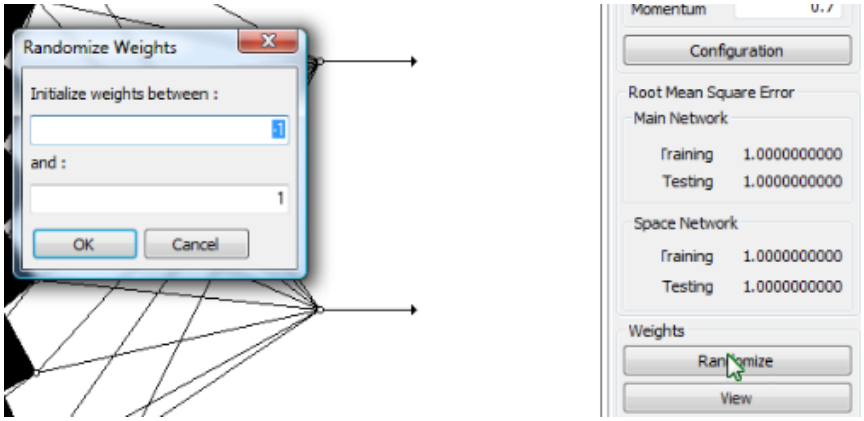


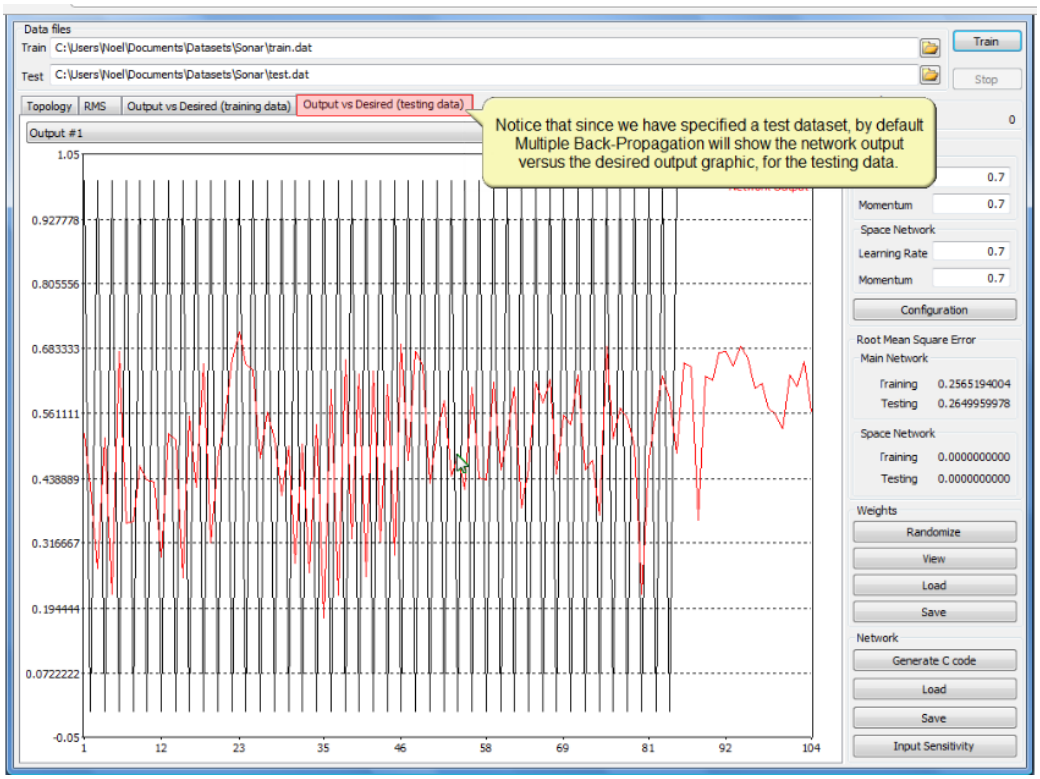

c.Seteleh menentukan dataset train dan test, selanjutnya tentukan topology neural netowork

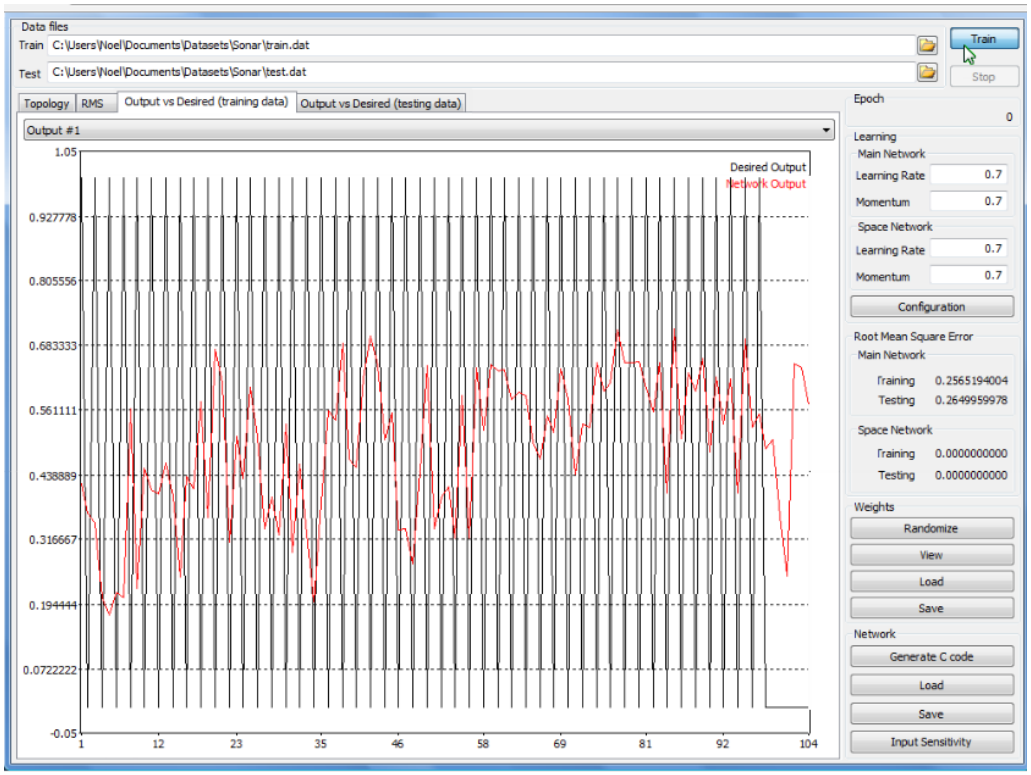


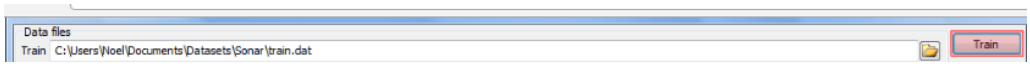

d. Selanjutnya pada area weight, klik **randomize,** maka kita dapat melihat grafik-grafik RMS dan Output vs Desired untuk data train maupun data test

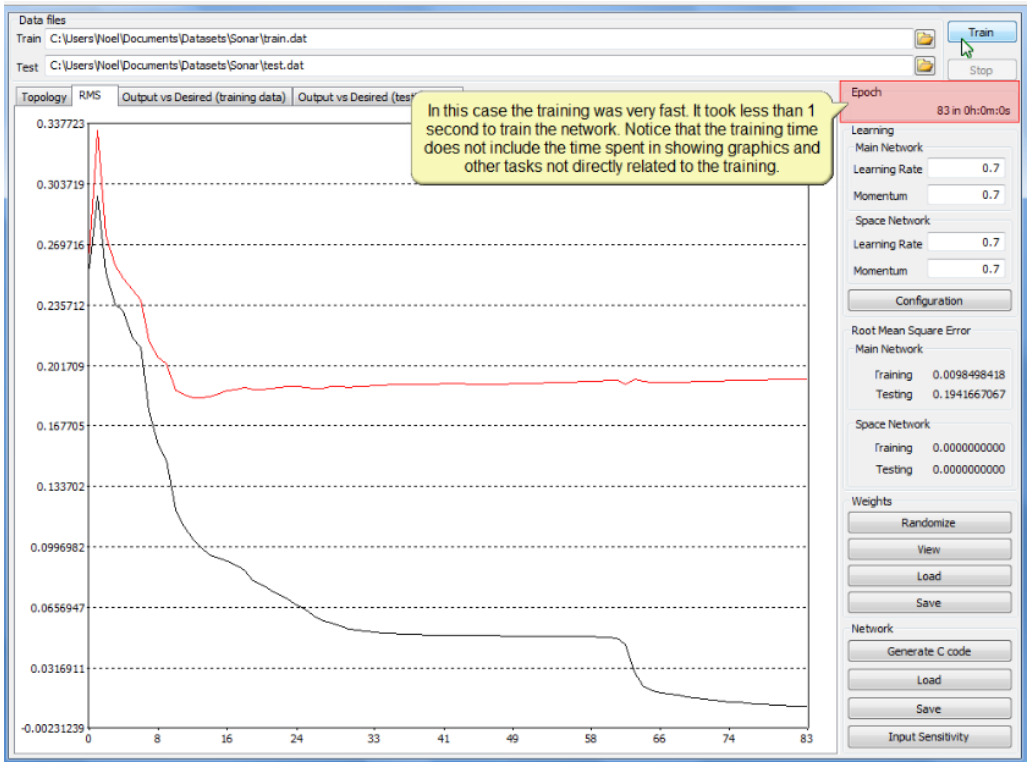



Data files
Train C:\Users\Noel\Documents\Datasets\Sonar\train.dat
Test C:\Users\Noel\Documents\Datasets\Sonar\test.dat
Train
Stop
Topology
RMS
Output vs Desired (training data)
Output vs Desired (testing data)
Notice that since we have specified a test dataset, by default Multiple Back-Propagation will show the network output versus the desired output graphic, for the testing data.
Output #1
1.05
0.927778
0.805556
0.683333
0.561111
0.438889
0.316667
0.194444
0.0722222
-0.05
1
12
23
35
46
58
69
81
92
104
0
0.7
Momentum
0.7
Space Network
Learning Rate
0.7
Momentum
0.7
Configuration
Root Mean Square Error
Main Network
Training
0.2565194004
Testing
0.2649959978
Space Network
Training
0.0000000000
Testing
0.0000000000
Weights
Randomize
View
Load
Save
Network
Generate C code
Load
Save
Input Sensitivity

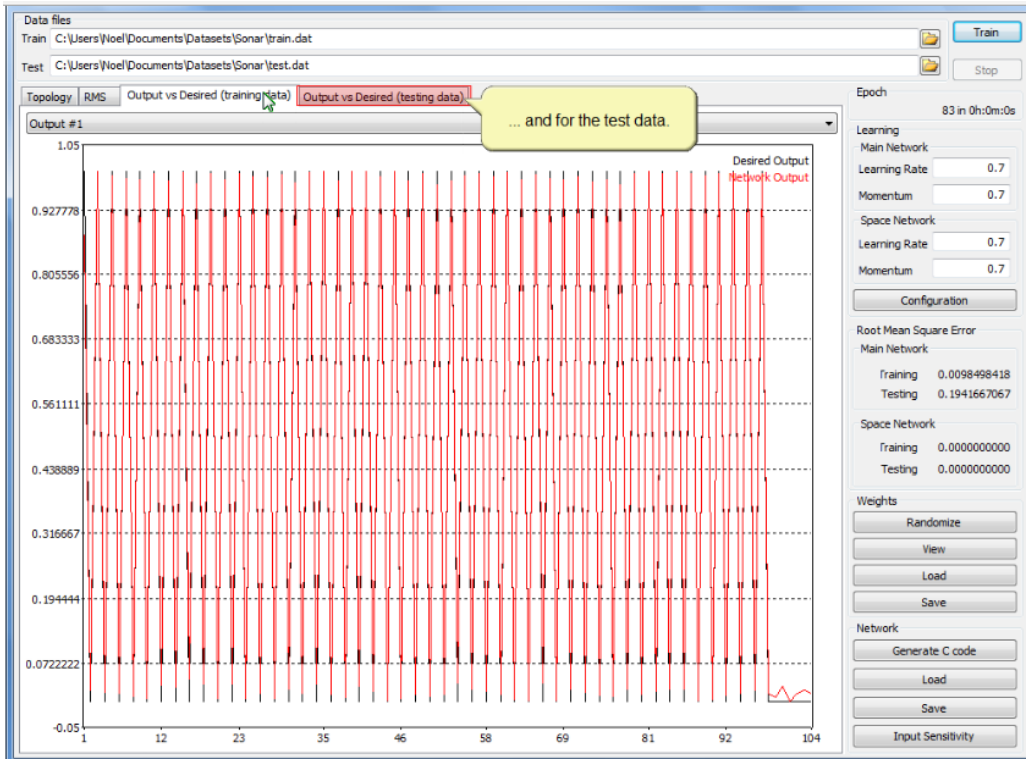

Data files
Train C:\Users\Noel\Documents\Datasets\Sonar\train.dat
Test C:\Users\Noel\Documents\Datasets\Sonar\test.dat
Train
Stop
Topology
RMS
Output vs Desired (training data)
Output vs Desired (testing data)
Output #1
Desired Output
Network Output
1.05
0.927778
0.805556
0.683333
0.561111
0.438889
0.316667
0.194444
0.0722222
-0.05
1
12
23
35
46
58
69
81
92
104
Epoch
0
Learning
Main Network
Learning Rate
0.7
Momentum
0.7
Space Network
Learning Rate
0.7
Momentum
0.7
Configuration
Root Mean Square Error
Main Network
Training
0.2565194004
Testing
0.2649959978
Space Network
Training
0.0000000000
Testing
0.0000000000
Weights
Randomize
View
Load
Save
Network
Generate C code
Load
Save
Input Sensitivity

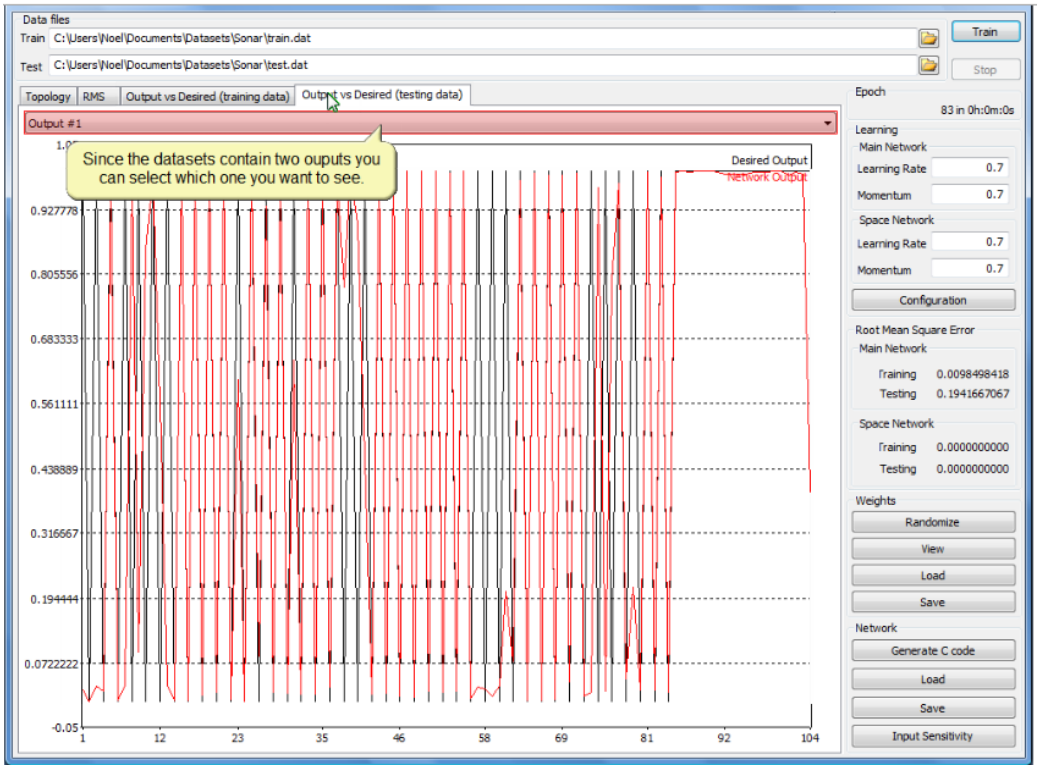
5) Selanjutnya klik tombol **Train**, untuk memulai prosess training



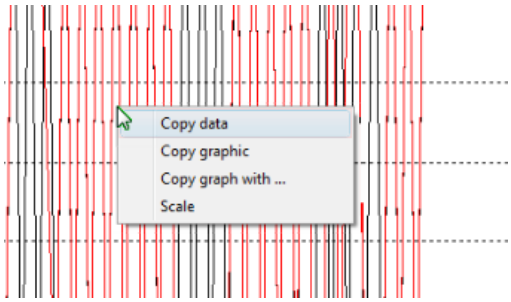




6) Selanjutnya, anda dapat melihat grafik outputnya

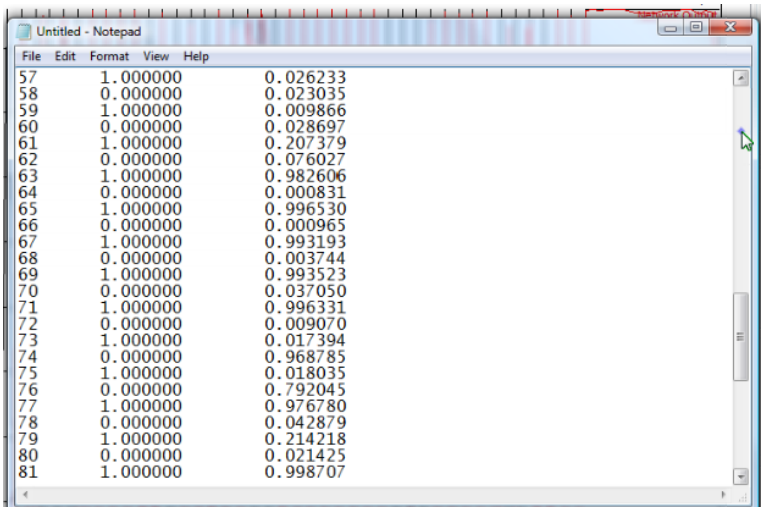





7) Anda dapat meng-copy data atau grafik dengan cara klik kanan pada area grafik yang akan di-copy

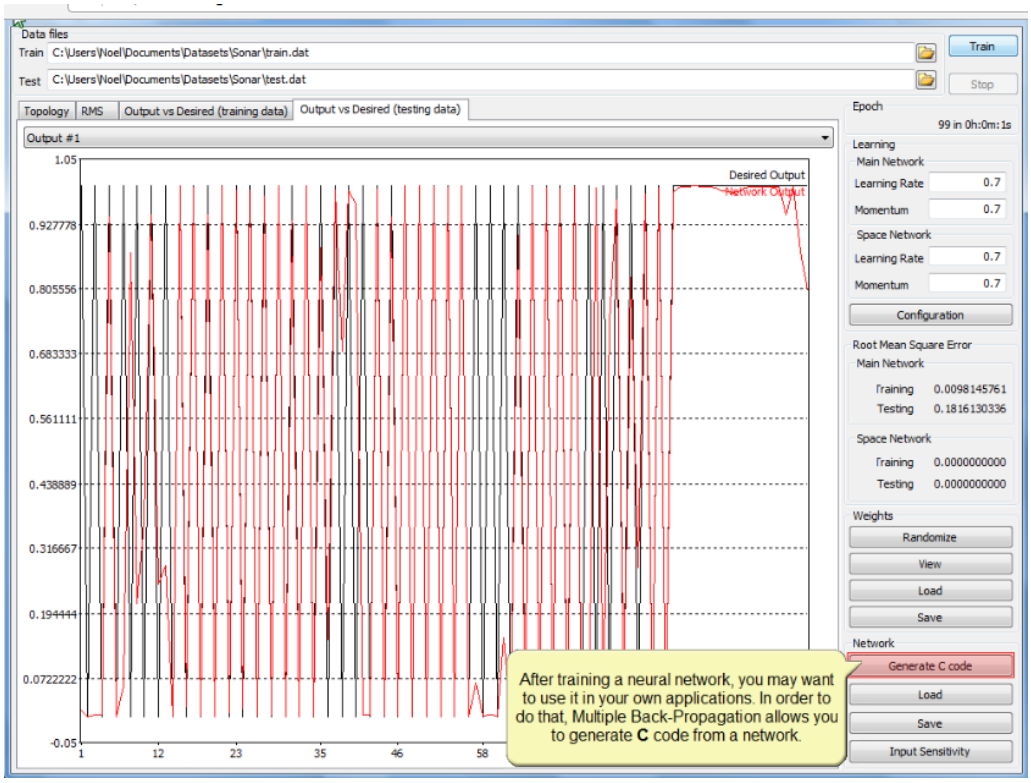



Pilih copy data, dan lakukan paste (sebagai contoh menggunakan notepad)

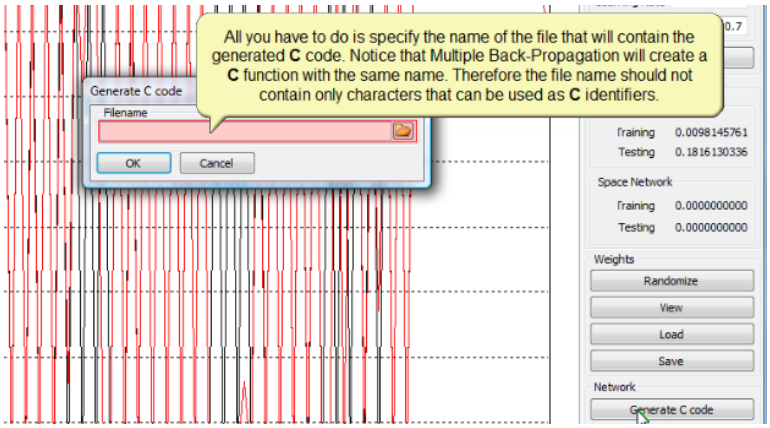


8) Dengan menggunakan MBP anda juga dapat meng-copy source code dalam bahasa C untuk dapat dimanfaatkan untuk keperluan lain dan bahkan konversi ke bahasa pemrograman lain.

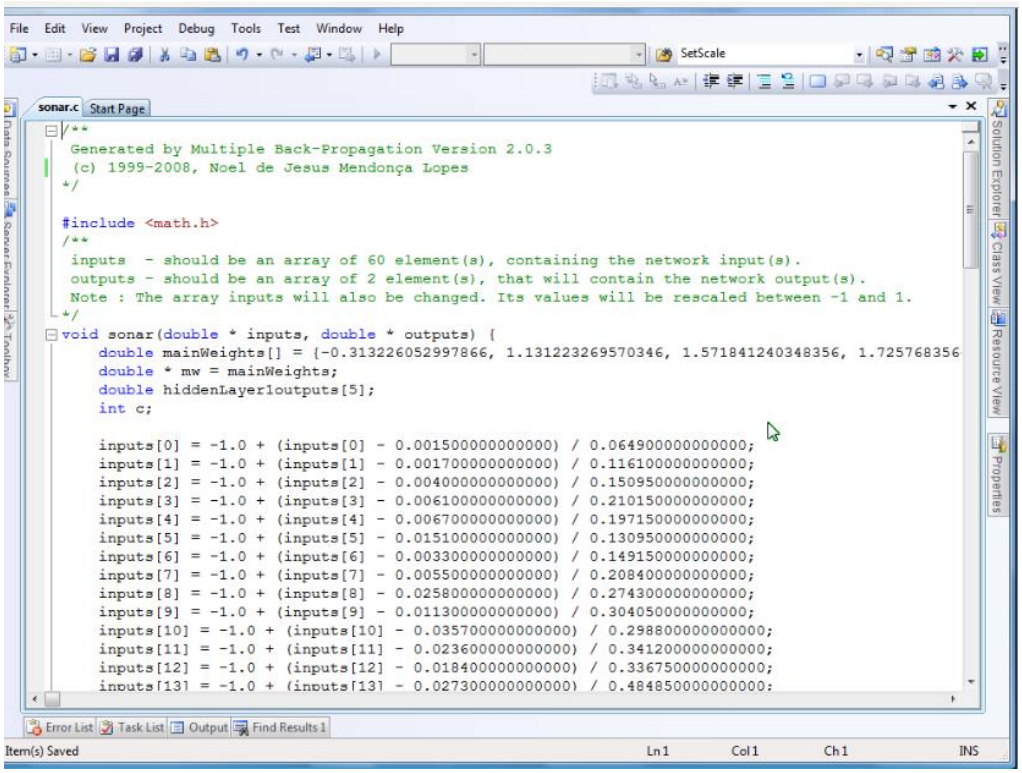
a. Klik **Generate C Code** pada area Network




b. Selanjutnya, tentukan nama file dan tempat penyimpanan source code tersebut




c.Maka contoh source code setelah file tersebut dibuka sebagai berikut




## D. DAFTAR PUSTAKA

[1]. Kusumadewim S. 2003. Artificial Intelligence (Teknik dan Aplikasinya). Graha Ilmu, Yogyakarta.

[2]. Hermawan, A. 2006. Jaringan Syaraf Tiruan. ANDI OFFSET, Yogyakarta.

[3]. http://mbp.sourceforge.net/index.html, tanggal akses 9 September 2015, jam 13.10 WIB.

**MATERI 9**
**MEMBANGUN SISTEM PAKAR**
**(ALUR KONSULTASI/PELACAKAN)**

---

**Waktu : 1.5 Jam**

**Kompetensi Dasar :** mahasiswa setelah mengikuti praktikum ini, dapat membuat alur pelacakan berdasarkan algoritma untuk Kasus Sistem pakar yang diberikan.

**Indikator :** Mahasiswa dapat membuat aplikasi Sistem Pakar

---

**Teori Pendukung**

a) Konsep Umum SP
- sistem yang berusaha mengadopsi pengetahuan manusia ke komputer agar komputer tersebut dapat menyelesaikan masalah seperti yang biasa dilakukan para ahli (pakar).
- Sistem pakar adalah suatu sistem yang bisa menyamai atau meniru kemampuan seorang pakar (Giarratano dan Riley, 2004).
- Sistem pakar merupakan suatu model dan prosedur yang berkaitan, dalam suatu daerah tertentu, yang mana tingkat keahliannya dapat dibandingkan dengan keahlian seorang pakar (Ignizio,1991).
- Sistem pakar adalah program komputer yang didesain untuk meniru kemampuan memecahkan masalah dari seorang pakar

b) Komponen SP

Sistem pakar sebagai aplikasi dalam Kecerdasan buatan terdiri dari komponen, user interface, mesin inferensi dan Representasi pengetahuan. Untuk aplikasi SP ketiga komponen ini harus ada.

**Langkah-langkah Praktikum**

Pada praktikum kali ini, praktikan akan dikenalkan dengan aplikasi Kecerdasan buatan system Pakar. Untuk membangunnya akan menggunakan tool ; GUI design untuk membangun prototype, Visio drawing untuk menggambarkan alur pelacakan dan MS-Access untuk membangun basis data. Tahapan yang akan digunakan dalam praktikum ini adalah mencoba untuk membangun sebuah prototype system pakar. Tahapan pengerjaannya sebagai berikut :

1. Membuat Alur pelacakan
2. Membangun basis data
3. Membangun Desain *User Interface*

Pada tugas praktikum ini, dianggap basis aturan dan basis data telah dibangun. Kita langsung membuat Alur Konsultasi atau pelacakan untuk Aplikasi System Pakar untuk mendiagnosa Penyakit Tulang ini.

1. Membuat Pohon Pelacakan

   Caranya :

   a) Bukalah Program Visio 2003
   b) Gambarlah Alur Pelacakan Untuk Konsultasi Sistem pakar pada kasus ini berdasarkan algoritma berikut ini :
      - Awal Program, Pilih menu konsultasi
      - Inputkan gejala yang dirasakan
      - Cek apakah ada gejala lagi atau yang lain…?
      - Jika Ya, inputkan gejala lain, jika tidak program Stop.
      - Tampilkan hasil diagnose jika tidak ada gejala lagi.
      - Simpan hasil diagnose
      - Eksekusi selesai
   c) Sesuaikan komponen yang ada pada property dengan alur pelacakan tersebut.

2. Membuat Basis Data

   Untuk membuat basis data dapat menggunakan tools MS-Access.

   a. Membuat Konseptual Data

      Untuk kasus Sistem pakar untuk Diagnosa Penyakit Tulang ini, diberikan Desain Konseptual datanya sebagai berikut :

      pada tabel berikut ini :

Tabel 9.1**.** *Desain Konseptual Awal*

| Tipe Entitas | Atribut |
|---|---|
| User | no(*), `username`, `password`, nama, jeniskel, tgllahir, pekerjaan, alamat |
| Penyakit | kode_pnykit(*), nama_pnykit, definisi, CF |
| Gejala | kode_gjl(*), nama_gjl, mb, md |
| Penyebab | kode_pnybab(*), nama_pnybab |
| Saran | kode_saran(*), nama_saran |
| Terapi | kode_terapi(*), nama_terapi |

Keterangan : (*) : Primary Key

   b. Membuat Struktur Tabel

      Berdasarkan kosneptual data pada table 9.1, buatlah struktur tabelnya dengan menggunakan tools MS-Access. Caranya sebagai berikut (contoh membuat table penyakit):

      a. Buka MS-Acess → Blank Database → creat database→ isikan sistempakar → OK.
      b. Buat Nama Tabel : Klik kanan pada table1 → design View → isikan nama table : penyakit
      c. Isikan atribut sesuai nama atribut yang ada pada desain konseptual data pada table 9.1.
      d. Simpan dengan menekan tombol silang → simpan.

e. Untuk memberi kode Primary Key pada suatu atribut tanda atribut yang akan diberi PK dengan cara klik kanan pada atribut tersebut → tekan menu Design → primary Key (symbol kunci).

f. Begitu seterusnya untuk mendesain table-tabel yang lainnya.

**Referensi :**

Winiarti. S, Diktat Kuliah Kecerdasan Buatan, 2010.

**MATERI 10**
**MEMBANGUN SISTEM PAKAR (MEMBUAT USER INTERFACE)**

---

**Waktu** : 1.5 Jam

**Kompetensi Dasar** : mahasiswa setelah mengikuti praktikum ini, dapat membuat alur pelacakan berdasarkan algoritma untuk Kasus Sistem pakar yang diberikan.

**Indikator** : Mahasiswa dapat mendesain prototype system pakar dengan aplikasi GUI .

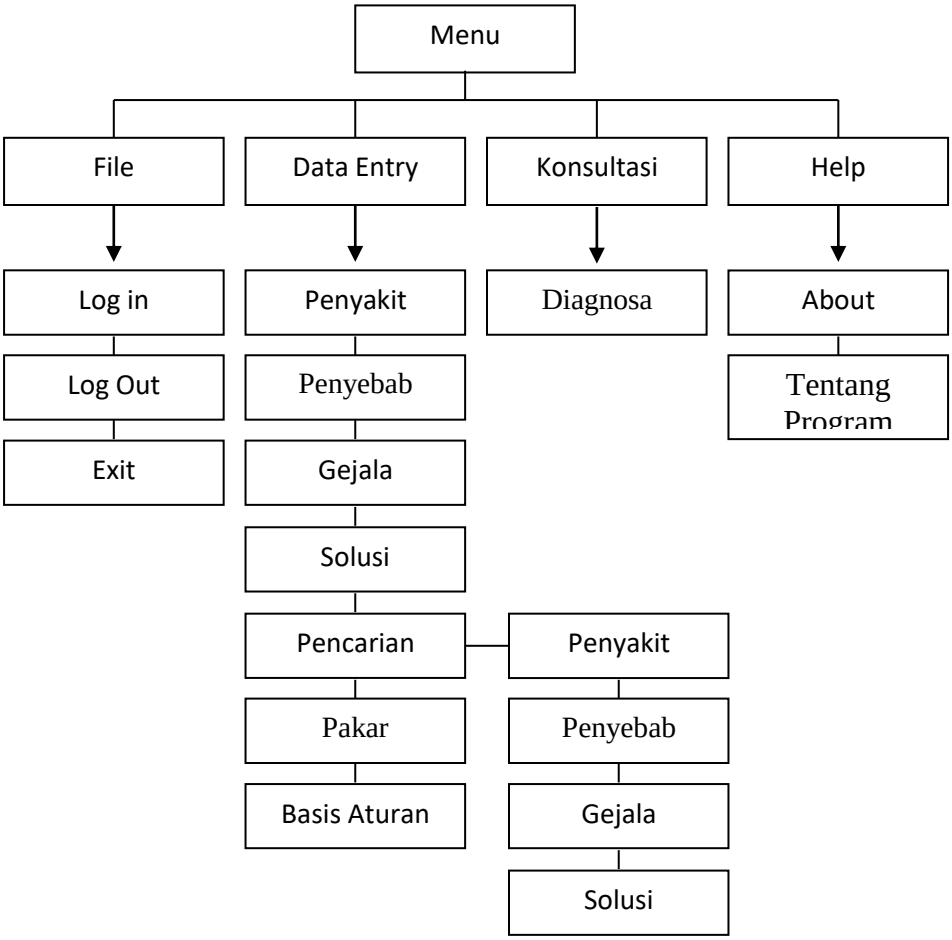
---

**Teori Pendukung**

Pada praktikum kali ini, praktikan akan dikenalkan dengan aplikasi Kecerdasan buatan system Pakar. Untuk membangunnya akan menggunakan tool ; GUI design untuk membangun prototype. Namun karena aplikasi yang digunakan masih versi trial, maka baru diinstal setelah menggu ke 10 dari praktikum ini.

**Langkah Praktikum**

1. Untuk memulai desain antar muka system, kita mengacu pada Rancangan Menu Program di bawah ini.
2. Pastikan tools : GUI sudah terinstall. Jika belum silahkan menginstall atau menghubungi asisten parktikum.
3. Buuatlah desain user interfacenya setelah menjalankan aplikasi GUI.
4. Ikutilah petunjuk cara mengoperasikan aplikasi GUI sesuai petunjuk yang ada dalam Buku Praktikum ini hingga desain berakhir.

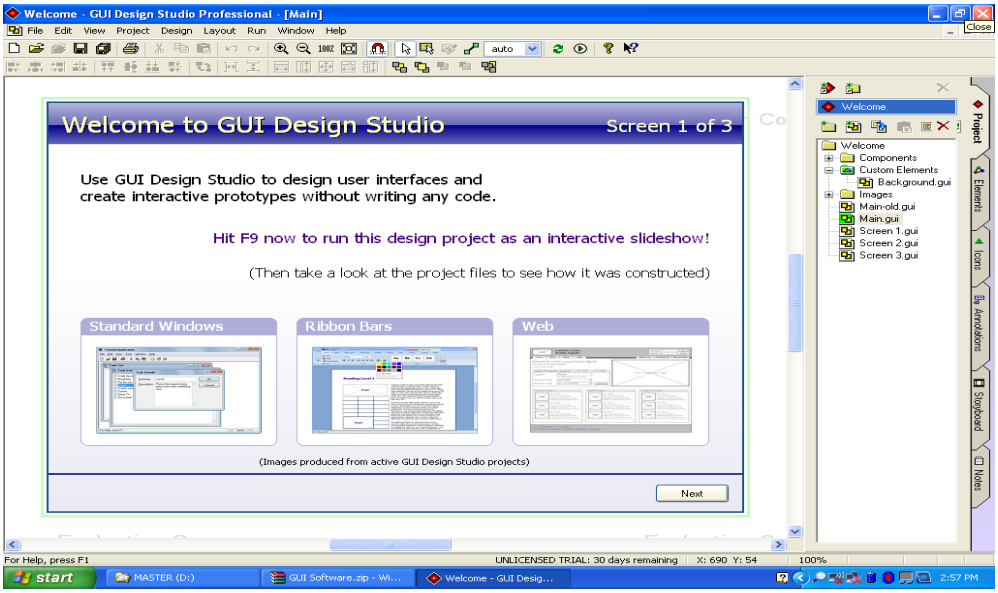



Gambar 10.1. Rancangan Menu Program

**Langkah Praktikum :**

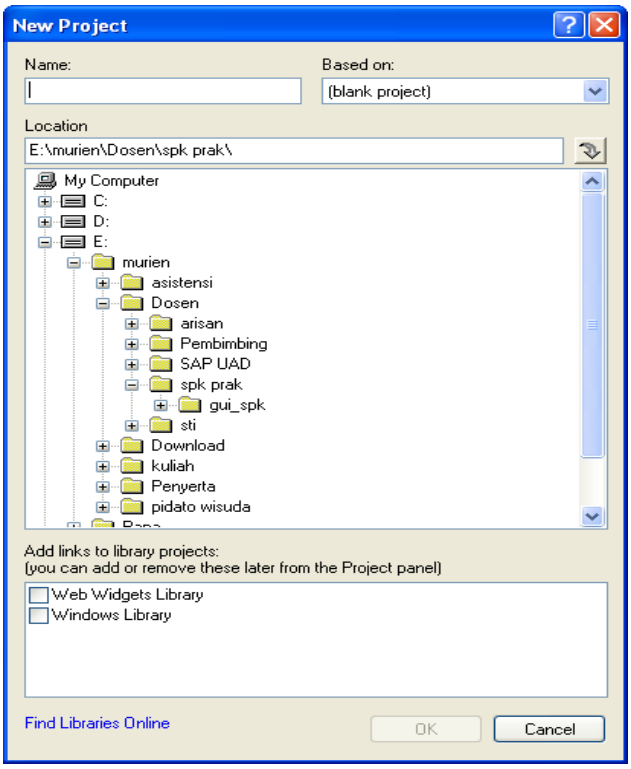
Langkah menggunakan *GUI Design Studio*

a. Buka *GUI Design Studio* maka tampilan form utamanya dapat dilihat dibawah ini :



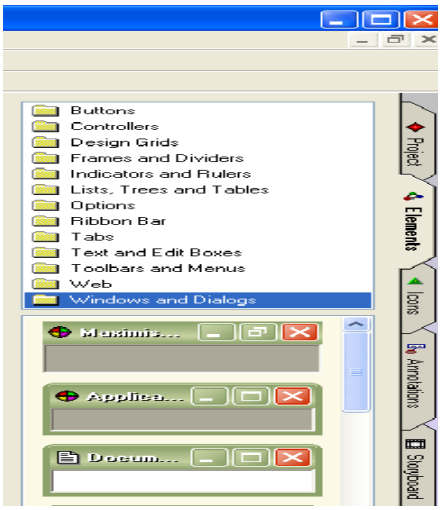

Gambar 10.2. Tampilan Form Utama GUI Design Studio

b. Setelah form utama terbuka, pilih *New Project* dan berikan nama project anda serta jangan lupa memilih lokasi drive mana anda akan menyimpannya. Bila ingin menambahkan link untuk library diberikan centang. Seperti gambar dibawah ini:



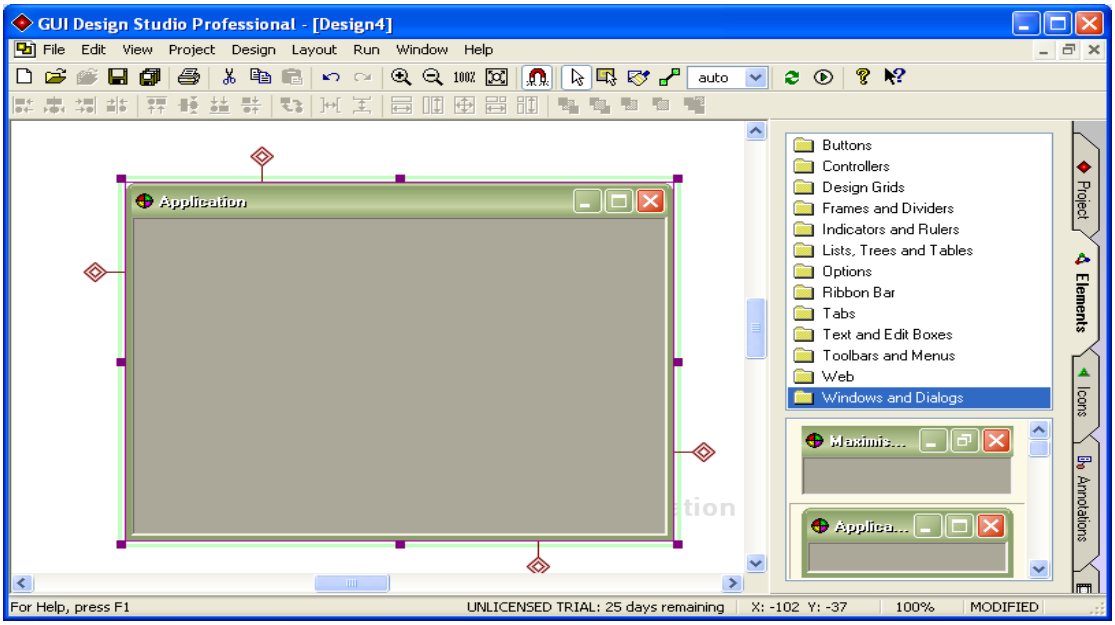

Gambar 10.3. *Form* untuk memilih tempat penyimpanan *New Project*

c. Setelah tersimpan dengan project baru maka anda siap mendesain rancangan yang anda inginkan. Anda dapat menggunakan berbagai macam komponen ada disebelah kiri layar. Ada *Project*, *Element*, *Icon* dan lain-lain. Seperti gambar dibawah:




Gambar 10.4. Komponen untuk membangun *interface*

d. Untuk memulai desain rancangan, pertama silakan buat dulu *form* desain, dengan cara klik *New Design* pada project atau dengan Crtl+N. Setelah keluar jendelan dialog baru silakan anda menambahkan komponen. *Drag* dan *drop* komponen yang sudah dipilih yang diinginkan sesuai dengan tema rancangan. Seperti gambar dibawah ini:



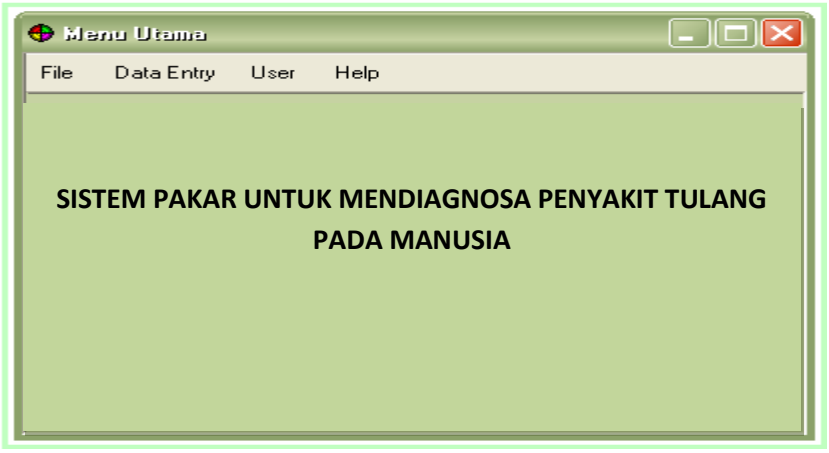

Gambar 10.5.Tampilan lembar kerja tempat membuat desain *interface*

e. Untuk pengaturan setiap komponen atau elemen bisa dilakukan dengan cara *double click* komponen yang akan dimodifikasi. Dan untuk menggesr elemen bisa menggunakan tombol panah atas-bawah dan kiri-kanan pada *keyboard*, untuk

mengecilkan dan membersarkan ukuran elemen bisa menggunakan kombinasi tombol Shift dan tombol 4 arah yang di sebutkan tadi.

**Desain *Interface***

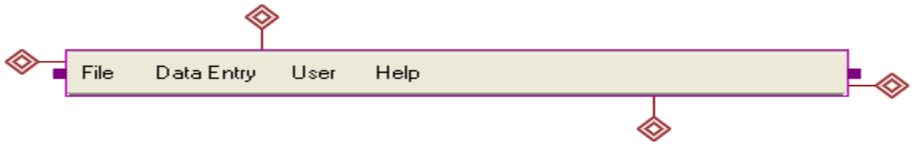
a. Menu Utama




Gambar 10.6. Rancangan menu utama

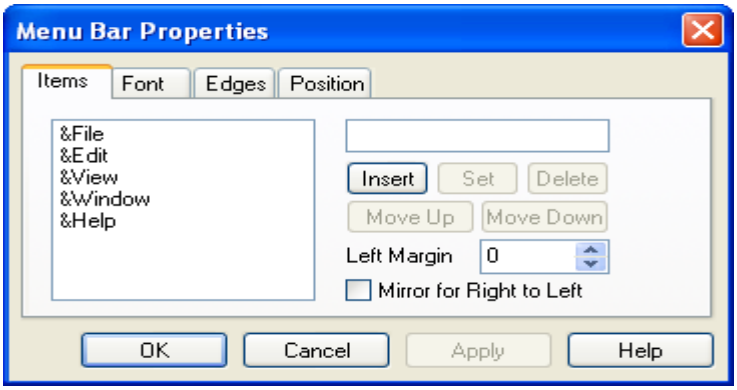
Pada menu utama terdapat beberapa menu, yaitu menu file, data *entry*, *user* dan *help*. Menu file memiliki sub menu *login*, ganti *password* dan *exit*. Data *entry* memiliki sub menu penyakit, penyebab, solusi, gejala dan basis aturan. *User* memiliki sub menu data *user* dan diagnosa, serta *help* memiliki sub menu *about*, petunjuk penggunaan dan programer.

Untuk membuat task bar seperti di bawah ini :



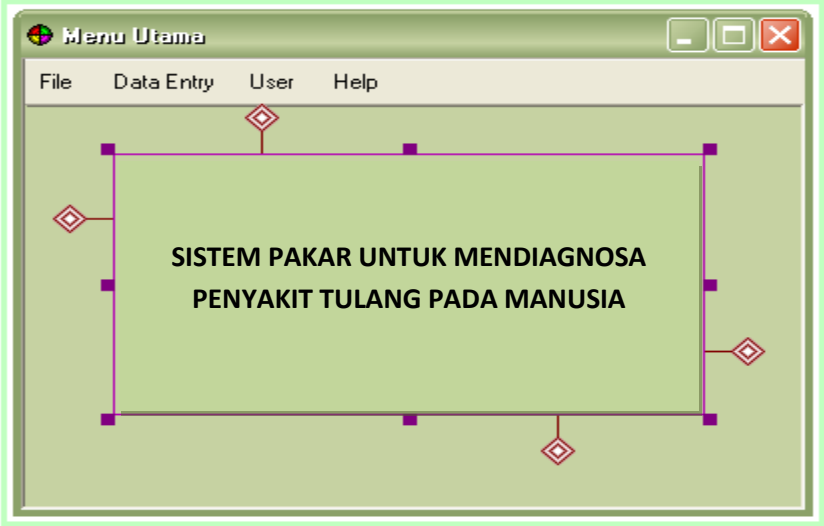

Gambar 10.7. Image Task Bar

- Pilih Elements → Toolbars dan Menus
- Tariklah image Menu Bar, bila ingin melakukan perubahan double klik pada image Menu Bar, kemudian lakukan setting pada Menu Bar properties seperti pada gambar dibawah ini.




Gambar 10.8. Tampilan Menu Bar Properties

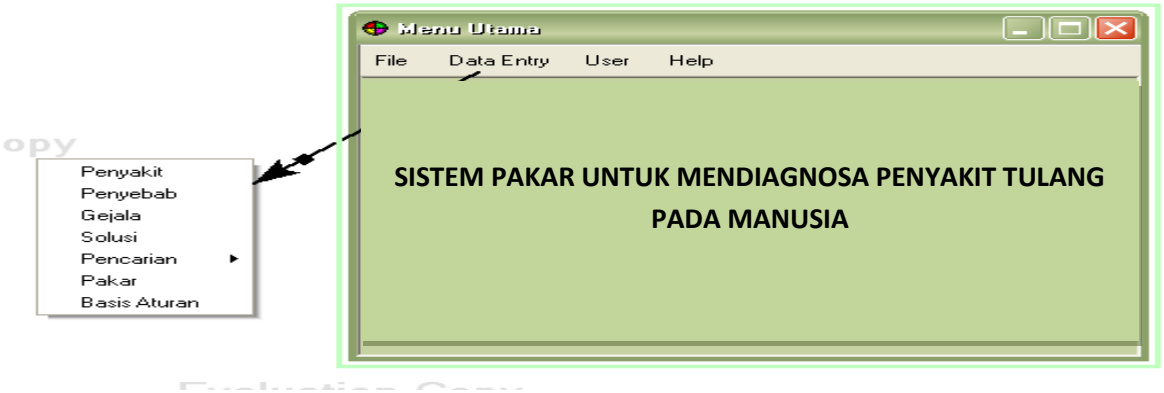
Untuk pemberian judul pada menu utama :




Gambar 10.9. Rancangan Judul Pada Menu Utama

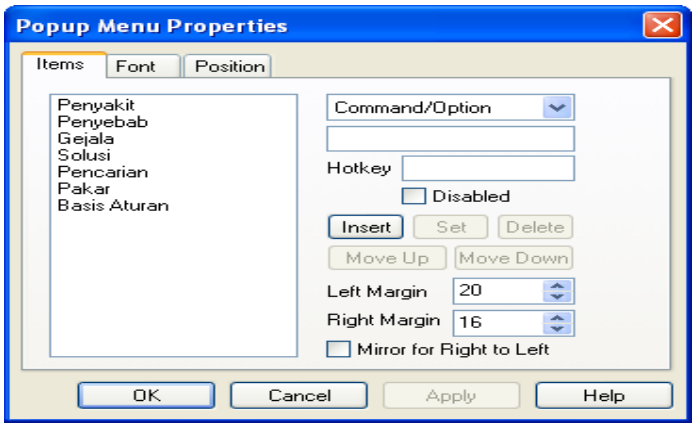
- Pilih Elements → Text and Edit Boxes
- Kemudian lakukan setting pada Text Properties

Untuk mengkoneksikan antara interface satu dengan interface lainnya dapat dilakukan link dari Menu Data Entry ke Popup Menu nya yaitu dengan memilih icon .



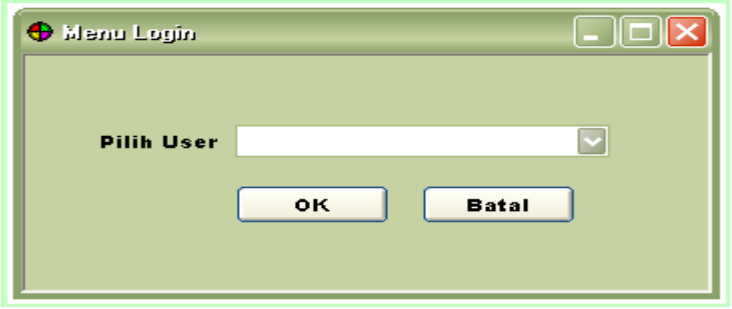

Gambar 10.10. Rancangan Menu Utama dengan Popup Menu

Untuk pembuatan Popup Menu dapat dilakukan dengan cara :
- Pilih Elements → Toolbars dan Menu
- Tariklah image Popup, bila ingin melakukan perubahan double klik pada image popup menu kemudian lakukan setting pada popup menu properties.




Gambar 10.11. Tampilan Popup Menu Properties

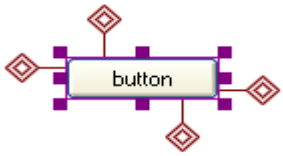
b. Menu Login




Gambar 10.12. Rancangan menu login

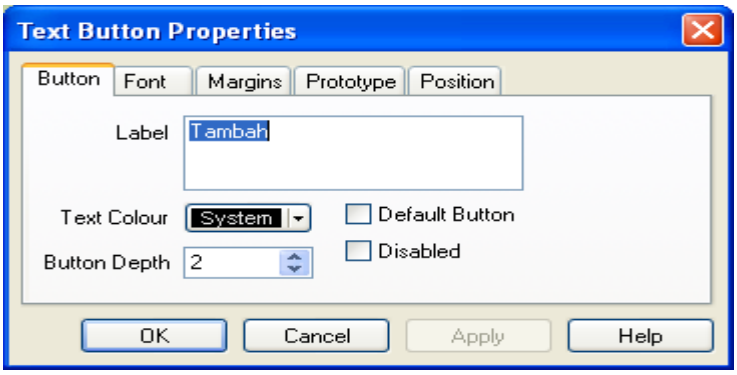
Untuk membuat tombol button :

- Pilih Elements → Buttons
- Kemudian pilih tombol button yang diinginkan, misalkan pada tombol button ini




Gambar 10.13. Image Button

- Bila ingin melakukan perubahan double klik pada tombol button kemudian lakukan penyetingan pada Text Button Properties seperti di bawah ini :
-




Gambar 10.14. Tampilan Text Button Properties

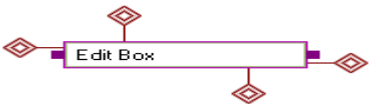
c. Menu Data Penyakit


Gambar 10.15. Rancangan menu data penyakit

Menu ini dipakai untuk *input* data penyakit yang akan disimpan dalam sistem. Memungkinkan untuk menambahkan jenis penyakit baru, ataupun untuk menghapus salah satu jenis penyakit.

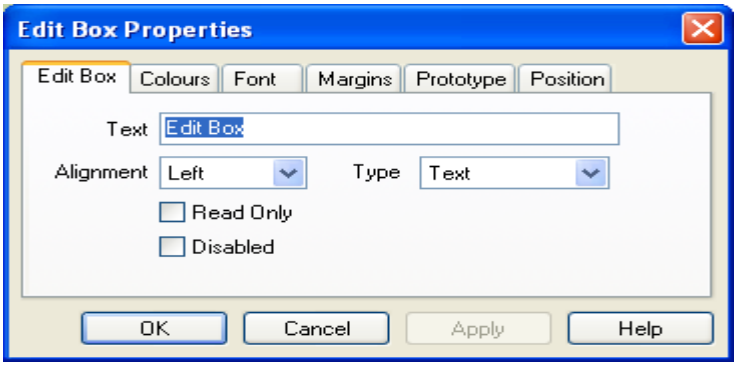
Untuk membuat edit text :

- Pilih Elements → Text and Edit Boxes
- Pilih image



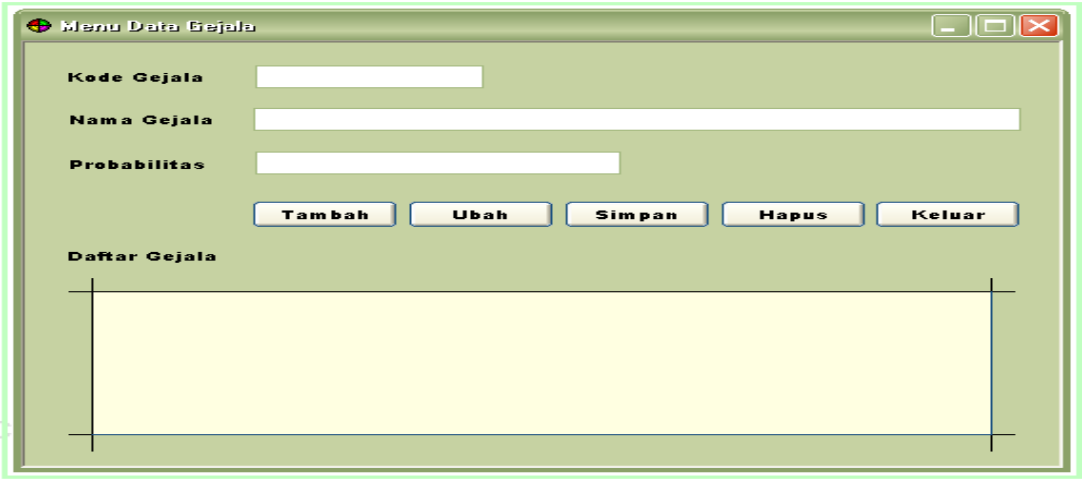

Gambar 10.16. Image Edit Box

- Kemudian lakukan seeting pada Edit Box Proporties seperti gambar di bawah ini dengan cara double klik.




Gambar 10.17. Tampilan Edit Box Properties

d. Menu Data Gejala




Gambar 10.18. Rancangan menu data gejala

Menu ini dipakai untuk *input* data gejala dan probabilitas masing-masing penyakit. Memungkinkan untuk menambahkan gejala baru, ataupun untuk menghapus salah satu gejala.

e. Menu Data Penyebab


Gambar 10.19. Rancangan menu data penyebab

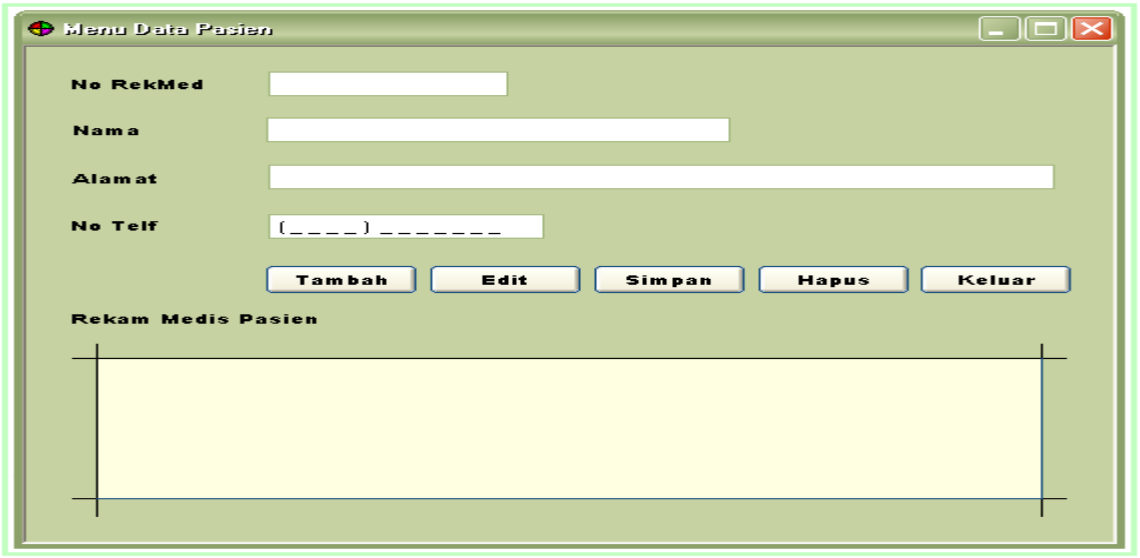
Menu ini dipakai untuk *input* data penyebab penyakit. Memungkinkan untuk menambahkan penyebab baru, ataupun untuk menghapus salah satu penyebab penyakit.

f. Menu Data Solusi


Gambar 10.20. Rancangan menu data solusi

Menu ini dipakai untuk *input* data solusi. Memungkinkan untuk menambahkan solusi baru, ataupun untuk menghapus salah satu solusi dari masing-masing penyakit.

g. Menu Data Pasien




Gambar 10.21. Rancangan menu data pasien

Menu ini dipakai untuk *input* data pasien. Memungkinkan untuk menambahkan pasien baru yang akan menggunakan sistem ini untuk berkonsultasi, ataupun untuk menghapus salah satu pasien.

h. Menu Aturan


Gambar 10.22. Rancangan menu aturan

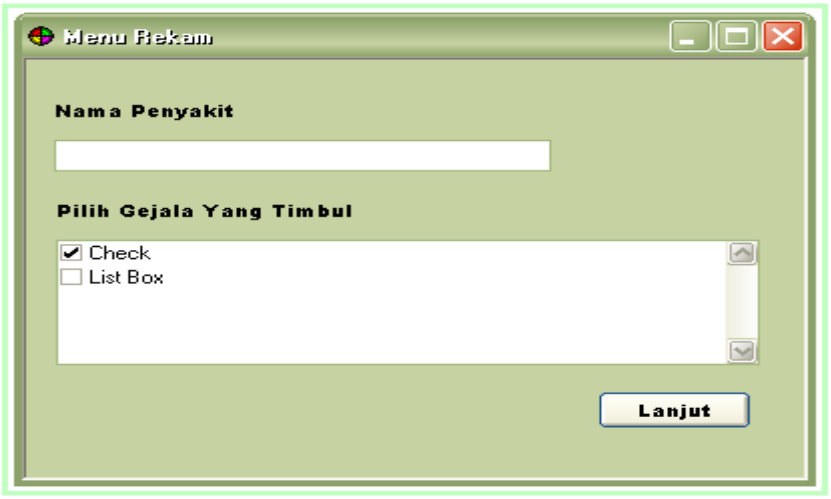
Menu ini digunakan untuk menampilkan kaidah aturan yang dipakai untuk melakukan penelusuran terhadap penyakit dan solusi sesuai penyakit tersebut berdasarkan gejala.

i. Menu Diagnosa


Gambar 10.23. Rancangan menu konsultasi

Menu ini digunakan untuk *user* yang ingin melakukan konsultasi. Nantinya *user* akan memilih gejala yang dirasakan.

j. Form Rekam




Gambar 10.24. Rancangan menu rekam

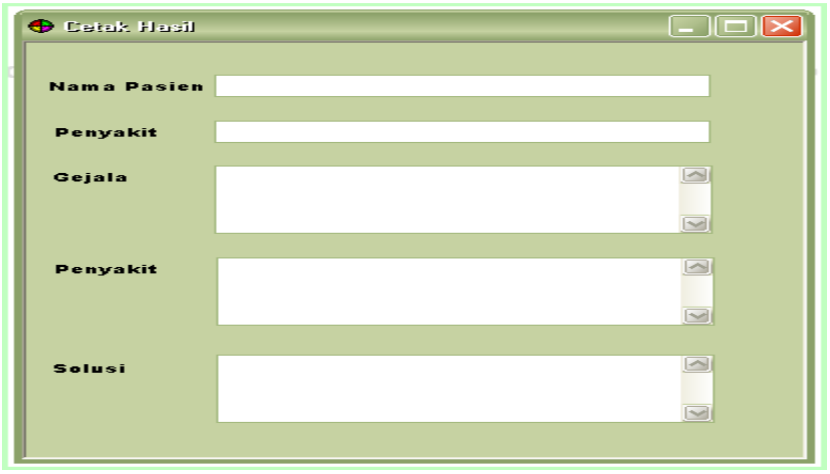
Menu ini lanjutan dari menu konsultasi. Jika pada menu konsultasi *user* memilih untuk rekam, maka gejala yang telah dipilih pada menu konsultasi akan ditampilkan pada form ini.

k. Rancangan Menu Lanjut


Gambar 10.25. Rancangan menu lanjut

Menu ini lanjutan dari form rekam. Pada form ini, sistem akan menampilkan solusi penyakit yang sudah tersimpan dalam *database* yang gejalanya mirip dengan gejala baru yang dimasukkan oleh *user*.

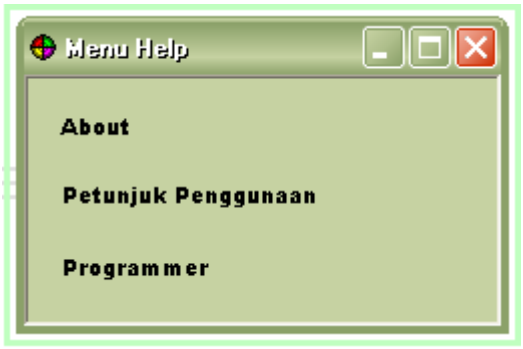
l. Cetak Hasil




Gambar 10.26. Rancangan menu cetak hasil

Menu ini lanjutan dari form lanjut. Jika pada form lanjut *user* memilih untuk cetak, maka hasil konsultasi tersebut dapat dicetak.

m. Menu Help




Gambar 10.27. Rancangan menu help

Menu ini akan menampilkan pilihan *about* yang berisi tentang program, petunjuk penggunaan program dan identitas programmer.

Referensi :

Winiarti.S, Buku Praktikum Kuliah Sistem Informasi, 2010.