PERANGKAT LUNAK UNTUK MENENTUKAN CHORD DAN BASS-LINE PADA SUATU LAGU DENGAN PENGENALAN POLA (JARINGAN SARAF)

Oleh :

HENRY NOVIANUS PALIT

NRP. 2692 100 005

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

1996

PERANGKAT LUNAK UNTUK MENENTUKAN CHORD DAN BASS-LINE PADA SUATU LAGU DENGAN PENGENALAN POLA (JARINGAN SARAF)

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
S u r a b a y a

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
DR. Ir. HANDAYANI TJANDRASA, M.Sc.	Ir. ESTHER HANAYA, M.Sc.
NIP. 130 532 048	NIP. 130 816 212

S U R A B A Y A
Nopember, 1996

Dipersembahkan untuk Papi, Mama,
Kakak, dan Herry

ABSTRAK

Aplikasi komputer sudah banyak dipakai dalam bidang musik. Namun, belum ada aplikasi komputer yang dapat menentukan chord pada suatu lagu.

Pada tugas akhir ini, dibuat aplikasi komputer untuk menentukan chord suatu lagu. Metode yang dipakai adalah dengan pengenalan pola yang menggunakan jaringan saraf ART (*Adaptive Resonance Theory*). Prinsip kerjanya, pola-pola chord seorang musisi dimasukkan dalam jaringan saraf, lalu pola-pola tersebut digunakan untuk menentukan chord suatu lagu. Aplikasi komputer hasil tugas akhir ini menggunakan file MIDI (*Musical Instrument Digital Interface*) sebagai media input dan output lagu.

Setiap musisi mempunyai pemilihan chord-chordnya sendiri. Aplikasi komputer ini fleksibel untuk siapapun pemakainya, karena *user* dapat memasukkan pola-pola chordnya sendiri. Aplikasi komputer ini menggunakan editor not balok, sehingga *user* dapat melihat apakah chord-chord hasil prosesnya sesuai dengan keinginan, dan dapat pula menggantinya. Selain itu, *user* dapat langsung mendengar lagu hasil prosesnya.

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa, karena atas berkat dan kasihNya, tugas akhir ini dapat Penulis selesaikan. Tugas akhir ini merupakan salah satu syarat yang harus dipenuhi oleh Penulis selaku mahasiswa Teknik Informatika FTI-ITS, untuk dapat lulus dan memperoleh gelar.

Penyelesaian tugas akhir ini tak lepas dari bantuan banyak pihak. Untuk itu, Penulis ucapkan terima kasih yang tulus kepada :

- DR. Ir. Handayani Tjandrasa, M.Sc., selaku Ketua Jurusan Teknik Informatika FTI-ITS dan sekaligus dosen pembimbing, yang mengenalkan jaringan saraf kepadaku.
- Ir. Esther Hanaya, M.Sc., selaku dosen pembimbing yang membimbing dan membantuku dalam menyelesaikan tugas akhir ini.
- Ir. F.X. Arunanto, dosen wali yang selalu membimbing dan mendukung studiku.
- Ibu Linggasari, yang mengajar dan membantuku menyelami dunia musik.
- Dosen-dosen Teknik Informatika FTI-ITS, yang telah membimbing dan mengajarku selama kuliah.
- Limin, Agus, dan Eddy, sahabat-sahabat yang selalu membantu, mendukung, dan mendorongku.

- Rekan-rekanku se-angkatan C-98, yang telah membagi suka dan duka bersamaku selama lebih dari 4 tahun.
- Seluruh Civitas Akademika Teknik Informatika FTI-ITS, yang turut berperan dalam keberhasilanku.
- Pihak-pihak lain yang tak dapat kusebutkan satu per satu.

Harapan Penulis, semoga tugas akhir ini dapat bermanfaat bagi para Pembaca. Sekian dan terima kasih.

Surabaya, September 1996.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Sejalan dengan perkembangan teknologi komputer, aplikasi-aplikasi komputer pun semakin meluas ke semua bidang kehidupan manusia. Pada bidang seni musik, komputer banyak digunakan untuk pengolahan suara, pembuatan suara-suara buatan (*synthesizer*), serta untuk pembuatan suatu lagu. Selain itu, aplikasi komputer dapat pula digunakan sebagai penghubung antara keyboard (organ) yang satu dengan yang lain, dan sekaligus sebagai sarana penyimpan data lagu. Salah satu format yang sering digunakan untuk komunikasi antara komputer dengan keyboard (organ) ini adalah MIDI (*Musical Instrument Digital Interface*).

Adanya format komunikasi melalui MIDI ini menimbulkan ide pada Penulis, untuk membuat suatu aplikasi komputer yang dapat membantu pemain musik pemula dalam bermain musik. Aplikasi tersebut berupa program bantu untuk menentukan chord-chord pada suatu lagu. Chord-chord pada suatu lagu cukup sulit untuk ditentukan, bukan saja bagi pemain musik pemula, bahkan bagi pemain musik yang cukup terampil sekalipun. Dengan aplikasi ini, diharapkan pemain musik pemula dapat menguasai suatu lagu

dengan cepat, tanpa perlu mengalami kesulitan dalam menentukan chord-chordnya.

1.2. Permasalahan

Dengan dilandasi pemikiran di atas, maka permasalahan yang timbul adalah bagaimana cara membuat suatu aplikasi komputer yang dapat digunakan untuk menentukan chord-chord pada suatu lagu. Untuk membuat aplikasi ini, diperlukan seorang musisi yang pengetahuannya dijadikan acuan dalam menentukan chord-chord tersebut. Jadi, pengetahuan dari musisi tersebut harus disimpan dalam aplikasi ini, dan selanjutnya akan digunakan untuk menentukan chord-chord suatu lagu. Masalahnya, bagaimana cara memasukkan pengetahuan dari musisi tersebut ke dalam aplikasi ini ?

1.3. Judul dan Tujuan

Judul dari tugas akhir ini adalah "PERANGKAT LUNAK UNTUK MENENTUKAN CHORD DAN BASS-LINE PADA SUATU LAGU DENGAN PENGENALAN POLA (JARINGAN SARAF)".

Tujuan dari tugas akhir ini adalah membuat suatu perangkat lunak (aplikasi) yang dapat membantu seorang pemain musik dalam menentukan chord-chord suatu lagu. Seperti dijelaskan pada Permasalahan di atas, aplikasi ini harus dapat menyimpan pengetahuan dari musisi yang dijadikan acuan.

Untuk itu, salah satu alternatif yang dapat digunakan adalah dengan menggunakan metode Pengenalan Pola.

1.4. Pengertian dan Batasan

Chord adalah tiga nada atau lebih yang dibunyikan bersama-sama, sehingga membentuk harmoni. Chord digunakan untuk memperindah lagu. Ada banyak sekali jenis-jenis chord yang kita kenal. Chord untuk jenis musik yang satu akan berbeda dengan chord untuk jenis musik yang lain. Misalnya, chord untuk musik klasik akan berbeda dengan chord untuk musik jazz.

Bass adalah nada rendah yang ditambahkan pada lagu sebagai harmoni. Chord dan bass saling menunjang dalam menyusun suatu lagu. Nada bass ini biasanya disesuaikan dengan chordnya. Namun, kadangkala nada bass ini tidak sesuai dengan chord-nya, melainkan disusun supaya membentuk suatu urutan tertentu. Urutan bass inilah yang disebut sebagai *bass-line*.

Pengenalan Pola (*Pattern Recognition*) adalah suatu metode yang mendasari prosesnya berdasarkan pola-pola tertentu. Chord-chord pada musik pada dasarnya membentuk pola-pola (aturan-aturan) yang tertentu. Pola-pola dari musisi yang satu dapat berbeda dengan musisi yang lain. Jadi, seorang musisi dapat menyimpan pengetahuannya dalam aplikasi ini, dengan melalui pembuatan pola chord-chord. Nantinya, pola-pola yang disimpan tersebut akan digunakan untuk menentukan chord-chord suatu lagu. Metode

pengenalan pola sendiri dapat dibagi menjadi 3 jenis berdasarkan cara yang digunakan. Pertama, dengan cara statistik, dimana pola-pola tersebut dikelompok-kelompokkan dengan metode statistik. Kedua, dengan cara sintaksis, menggunakan grammar-grammar tertentu. Dan ketiga, dengan jaringan saraf (*neural network*). Cara terakhir inilah yang dipilih, karena kemampuannya untuk menyimpan pola-pola yang dimasukkan *user*.

Pada pembuatan aplikasi ini, diberikan batasan pada jenis musik yang dapat diolah. Jenis musik yang dapat diproses aplikasi ini adalah jenis musik pop, yang chord-chordnya lebih sederhana dibandingkan dengan jenis musik lain. Tanda birama lagu yang dapat diproses aplikasi ini, juga disesuaikan dengan jenis musik pop, antara lain birama 2/4, 3/4, 4/4, dan 6/8.

Lagu yang akan diproses oleh aplikasi ini, harus dimasukkan dalam file dengan format MIDI. File MIDI sendiri ada 3 jenis format, yaitu format 0, format 1, dan format 2. File MIDI yang dapat diproses aplikasi ini adalah file MIDI dengan format 1. Lagu dalam file MIDI tersebut berupa lagu solo (*lead voice*). Nantinya, aplikasi ini akan menambahkan chord dan bass-line untuk memperindah lagu tersebut.

1.5. Metodologi Penelitian

Metodologi penelitian yang digunakan dalam mengerjakan tugas akhir ini adalah sebagai berikut :

- Mencari dan membaca referensi buku maupun jurnal mengenai pengenalan pola.
- Mencari dan membaca referensi buku maupun jurnal mengenai jaringan saraf.
- Menentukan topologi jaringan saraf yang paling sesuai untuk diaplikasikan pada tugas akhir ini. Sebagai hasilnya, topologi jaringan saraf ART (*Adaptive Resonance Theory*) yang dipilih.
- Merancang dan membuat perangkat lunak untuk pelatihan jaringan saraf.
- Menguji perangkat lunak tersebut dan melakukan perubahan-perubahan pada parameter input maupun metode pembobotan dari jaringan saraf, agar hasilnya sesuai dengan yang diinginkan.
- Merancang dan membuat perangkat lunak untuk interaksi dengan user.
- Menguji perangkat lunak kedua.
- Menyusun buku laporan tugas akhir.

1.6. Sistematika Pembahasan

Sistematika pembahasan pada buku laporan tugas akhir ini adalah sebagai berikut :

- Bab I Pendahuluan

  Menjelaskan mengenai latar belakang, permasalahan, tujuan, pengertian dan batasan, serta metodologi penelitian dari tugas akhir ini.

- Bab II Dasar Teori Musik

  Menjelaskan mengenai dasar-dasar teori musik yang dipakai untuk mendukung tugas akhir ini.

- Bab III Format File MIDI

  Menjelaskan mengenai format file MIDI, yang digunakan sebagai media input dan output pada perangkat lunak dari tugas akhir ini.

- Bab IV Pengenalan Pola dengan Jaringan Saraf ART

  Menjelaskan pengertian dari pengenalan pola dan jaringan saraf. Menjelaskan secara luas mengenai jaringan saraf ART : arsitektur, operasi, algoritma, serta proses pelatihannya.

- Bab V Perancangan dan Pembuatan Perangkat Lunak

  Menjelaskan mengenai struktur data maupun algoritma dari kedua perangkat lunak yang dibuat pada tugas akhir ini.

- Bab VI Hasil Uji Coba dan Evaluasi Perangkat Lunak

  Melakukan pengujian terhadap kedua perangkat lunak dari tugas akhir ini, dan mengevaluasi hasil pengujian tersebut.

- Bab VII Penutup

  Berisi kesimpulan dari tugas akhir ini dan saran-saran Penulis untuk Pembaca.

BAB II
DASAR TEORI MUSIK

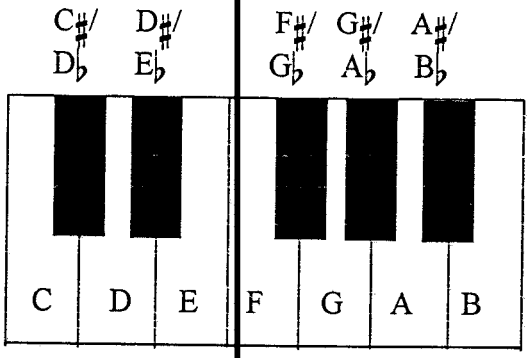
Musik merupakan rangkaian nada-nada yang teratur dan enak didengar. Musik bukan sekedar bunyi semata, namun juga melibatkan teori-teori tertentu di dalamnya, baik teori dalam menulis maupun untuk memainkannya. Berikut ini akan dijelaskan dasar-dasar teori musik.

2.1. Tangga Nada

Nada-nada pada musik diberi simbol dengan 7 huruf alphabet, dari A sampai G. Nada-nada ini disusun membentuk urutan yang disebut tangga nada. Nada-nada ini akan diulang-ulang untuk membentuk nada-nada dengan level lebih tinggi dan lebih rendah. Satu *oktaf* adalah jarak antara dua nada yang yang sama bunyinya, hanya levelnya berbeda satu, ke atas ataupun ke bawah. Contohnya adalah jarak dari nada A ke nada A berikutnya yang lebih tinggi.

Suara antara orang yang satu dengan orang yang lain berbeda-beda. Ada yang suaranya tinggi, dan ada yang rendah. Demikian juga dengan suara alat-alat musik. Untuk itu, kita mengenal nada dasar (nada acuan), yaitu nada yang tangga nadanya dijadikan acuan dalam lagu. Nada yang paling sederhana untuk dijadikan nada dasar adalah nada C. Bila kita menyusun tangga nada

dengan nada dasar C, maka urutan nada yang kita peroleh adalah C-D-E-F-G-A-B. Urutan nada ini kita lafalkan sebagai *do*, *re*, *mi*, *fa*, *sol*, *la*, *si*. Antara nada yang satu dengan nada yang lain, terdapat aturan jarak tertentu. Untuk lebih jelasnya, perhatikan gambar tuts piano yang terdapat di bawah ini.



Gambar 2.1 Gambar Tuts Piano

Di samping ketujuh nada utama tadi, ada 5 nada tambahan lain yang berwarna hitam. Jarak antara nada C-D, D-E, F-G, G-A, dan A-B adalah 1 laras. Sedangkan jarak antara nada E-F dan B-C adalah 1/2 laras. Terlihat bahwa antara E-F dan B-C tidak ada tuts yang berwarna hitam, karena antara kedua nadanya tidak dapat dibagi lagi.

Antara dua nada yang berjarak 1 laras, pasti terdapat satu nada tambahan. Nada-nada tambahan tersebut dapat dinyatakan dengan menambahkan notasi *kres* (#) atau *mol* (b). Tanda kres berarti kita menaikkan nada ½ laras, sedangkan tanda mol berarti kita menurunkan nada ½ laras. Dengan demikian kelima nada tambahan tersebut dapat dituliskan seperti pada

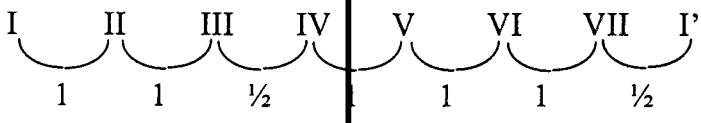
gambar 2.1. Aturan untuk penyebutannya, bila kita menggunakan notasi kres, maka kita beri akhiran is, bila kita menggunakan notasi mol, maka kita beri akhiran es atau s. Jadi penyebutannya C# / Db (Cis / Des), D# / Eb (Dis / Es), F# / Gb (Fis / Ges), G# / Ab (Gis / As), dan A# / Bb (Ais / Bes). Sedangkan cara melafalkannya masing-masing *di, ri, fi, sye, li* (bila menggunakan notasi kres) atau *ro, mo, syu, lo, so* (bila menggunakan notasi mol).

Disamping notasi alphabet, nada-nada tersebut dapat kita tuliskan dengan notasi huruf Romawi (I - VII). Notasi I digunakan untuk menandai nada pertama (nada dasar), notasi II untuk nada kedua, dan seterusnya. Jadi, notasi dengan huruf Romawi ini akan selalu tetap untuk sembarang nada dasar.

Dalam musik, kita mengenal 2 macam tangga nada, yaitu tangga nada ***Mayor*** dan tangga nada ***Minor***. Masing-masing tangga nada tersebut mempunyai aturan jarak yang berbeda antara nada yang satu dengan nada yang lain. Berikut ini aturan jarak dalam kedua tangga nada tersebut.

2.1.1 Tangga Nada Mayor

Tangga Nada Mayor mempunyai aturan jarak sebagai berikut :

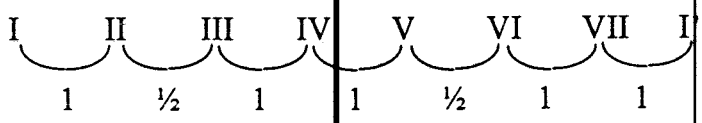


Gambar 2.2. Tangga Nada Mayor

Contoh tangga nada Mayor : C Mayor (C-D-E-F-G-A-B), F Mayor (F-G-A-Bb-C-D-E), G Mayor (G-A-B-C-D-E-F#), dan lain-lain.

2.1.2. Tangga Nada Minor

Tangga Nada Minor mempunyai aturan jarak sebagai berikut :



Gambar 2.3. Tangga Nada Minor

Contoh tangga nada Minor : A Minor (A-B-C-D-E-F-G), B Minor (B-C#-D-E-F#-G-A), G Minor (G-A-Bb-C-D-Eb-F), dan lain-lain.

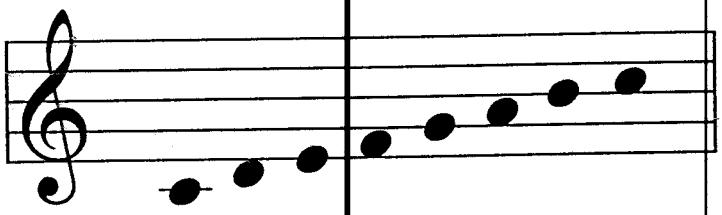
2.2. Notasi Balok

Pada suatu lagu, setiap nada yang dibunyikan, akan mempunyai lama bunyi yang tertentu. Untuk menggambarkan nada tersebut dengan lama bunyi tertentu, kita gunakan *not*. Ada 2 cara untuk menuliskan not-not ini, yaitu dengan notasi angka dan notasi balok. Notasi angka menggunakan angka 1 - 7 sebagai pengganti nada-nada C-D-E-F-G-A-B. Pada tugas akhir ini, Penulis menggunakan notasi balok, jadi notasi balok inilah yang akan dijelaskan dalam tugas akhir ini.

Berikut ini elemen-elemen dalam notasi balok :

2.2.1. Garis Paranada (Staff)

Not-not untuk notasi balok harus diletakkan pada *garis paranada*, yaitu berupa lima garis horisontal sejajar.



Gambar 2.4 Notasi Balok

Not-not yang diletakkan pada garis paranada ada 2 jenis, yaitu not yang terletak pada garis (*lines*) dan not yang terletak antara dua garis (*spaces*). Nada-nada dari suatu tangga nada akan diletakkan secara urut dan berselang-seling antara kedua jenis not tersebut.

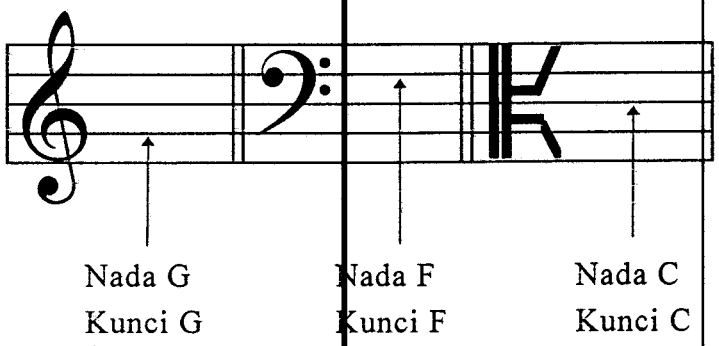
2.2.2. Tanda Kunci (Clef)

Pada notasi balok, dikenal pula tanda kunci (tanda di paling kiri). Tanda kunci menentukan posisi dari nada-nadanya. Ada 3 macam tanda kunci yang dikenal dalam musik :

- Kunci **G** (*treble*). Tanda kunci ini paling banyak digunakan pada notasi balok. Bila tanda kunci ini digunakan, maka nada pada posisi garis kedua dari bawah adalah nada G.

- Kunci **F** (*bass*). Tanda kunci ini biasanya digunakan untuk penulisan bass pada notasi balok, sering pula digunakan untuk penulisan chord. Bila tanda kunci ini digunakan, maka nada pada posisi garis kedua dari atas adalah nada F.
- Kunci **C**. Tanda kunci ini jarang sekali digunakan. Bila kita gunakan tanda kunci ini, maka nada pada posisi garis ketiga adalah nada C.

Contoh penggunaan tanda kunci ini, dapat dilihat pada gambar 2.4. di atas. Pada notasi balok gambar tersebut, digunakan tanda kunci G. Karena nada pada garis kedua dari bawah adalah nada G, maka urutan nada-nada pada gambar tersebut adalah C-D-E-F-G-A-B-C'.

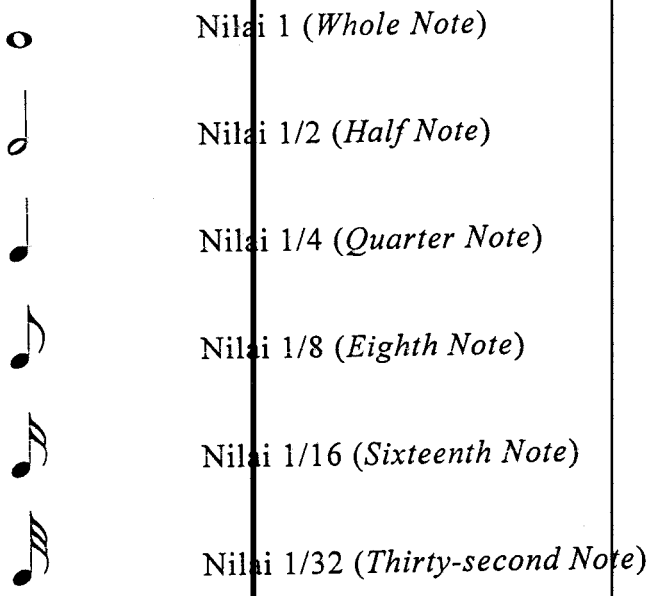


Gambar 2.5. Macam-macam Tanda Kunci

2.2.3. Not dan Tanda Istirahat [1)]

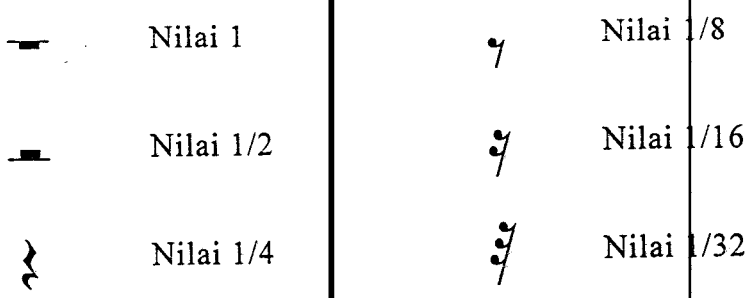
Not-not pada notasi balok mempunyai bentuk-bentuk tertentu untuk menentukan lama bunyi dari nadanya. Berikut macam-macam bentuk not beserta nilainya :

[1)] The Associated Board of The Royal Schools of Music ; **"Rudiments and Theory of Music"**.



Gambar 2.6. Macam-macam Not dan Nilainya

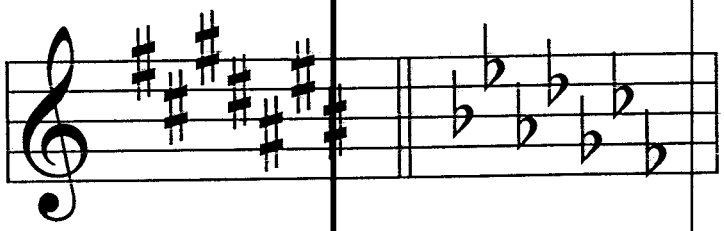
Pada sebuah lagu, kadangkala ada suatu saat dimana tidak ada nada yang dibunyikan. Untuk itu, kita perlu memberikan tanda istirahat. Seperti not-not di atas, tanda istirahat juga ada beberapa macam dengan nilai yang tertentu.



Gambar 2.7. Macam-macam Tanda Istirahat dan Nilainya

2.2.4. Kres (Sharp) dan Mol (Flat)

Untuk menyatakan nada dasar suatu lagu pada notasi balok, kita letakkan tanda kres atau mol pada bagian kirinya.

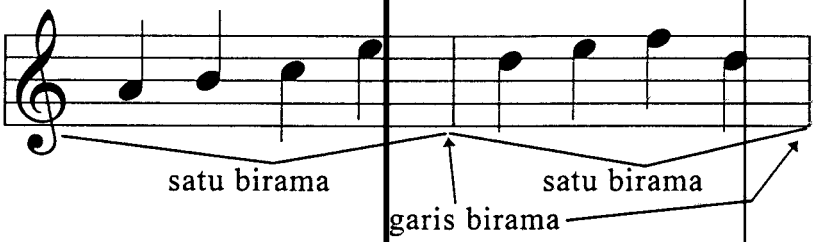


Gambar 2.8. Penggambaran Kres dan Mol pada Notasi Balok

Bila pada sisi kiri notasi balok tersebut tidak ada tanda Kres dan Mol, berarti nada dasar yang digunakan adalah nada C. Urutan nada dasar selanjutnya untuk penggunaan tanda Kres adalah G (1#), D (2#), A (3#), E (4#), B (5#), Fis (6#), Cis (7#). Urutan nada dasar untuk penggunaan tanda Mol adalah F (1b), Bes (2b), Es (3b), As (4b), Des (5b), Ges (6b), Ces (7b). Urutan penggambaran Kres dan Mol harus sesuai dengan gambar di atas.

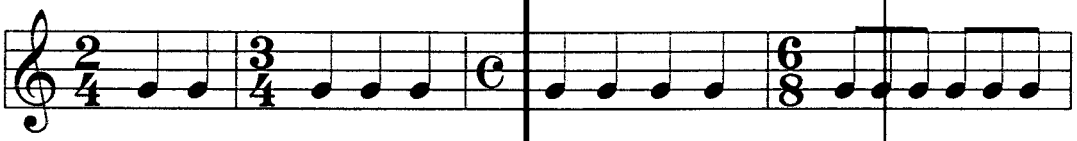
2.2.5. Tanda Birama

Tanda birama suatu lagu akan menentukan jumlah ketukan (*beat*) tiap birama. Satu *birama* adalah daerah yang dibatasi oleh 2 garis birama. Dalam suatu lagu, tiap biramanya harus mempunyai jumlah ketukan yang sama, kecuali bila terjadi perubahan tanda birama di tengah lagu.



Gambar 2.9. Birama dan Garis Birama

Tanda birama dituliskan berupa bilangan pecahan. Pembilangnya menentukan jumlah ketukan tiap birama, sedangkan penyebutnya menyatakan jenis not untuk tiap ketukan. Misalnya, tanda birama 4/4, berarti jumlah ketukan tiap birama adalah 4, dan jenis not untuk tiap ketukan adalah not 1/4 (lihat kembali gambar 2.6. mengenai jenis not). Berikut ini beberapa contoh penggunaan tanda birama.



Gambar 2.10. Contoh-Contoh Tanda Birama

Tanda birama 4/4 sering disimbolkan sebagai **C** (*Common Time*).

2.3. Chord dan Bass

Chord adalah tiga nada atau lebih yang dibunyikan secara bersama-sama, sehingga menghasilkan harmoni. Chord sering digunakan untuk memperindah lagu. Ada banyak sekali jenis chord yang ada, apalagi chord-

chord untuk musik jazz. Pada tugas akhir ini akan dijelaskan beberapa chord yang cukup sering digunakan pada musik pop.

- Chord Major, terdiri dari nada I, III dan V. Misalnya, chord C (C-E-G), chord A (A-C#-E).
- Chord Minor, terdiri dari nada I, nada III yang di-Mol (diturunkan ½ laras), dan nada V. Misalnya, chord Dm (D-F-A), chord Gm (G-Bb-D).
- Chord Dominant 7^{th}, terdiri dari nada I, III, V, dan nada VII yang di-Mol (diturunkan ½ laras). Misalnya, chord E7 (E-G#-B-D), chord B7 (B-D#-F#-A).
- Chord Augmented, terdiri dari nada I, III, dan nada V yang di-Kres (dinaikkan ½ laras). Misalnya, chord C^{+} (C-E-G#), chord F^{+} (F-A-C#).
- Chord Sixth, terdiri dari nada I, III, V, dan VI. Misalnya, chord G6 (G-B-D-E), chord Bb6(Bb-D-F-G).
- Chord Major 7^{th}, terdiri dari nada I, III, V, dan VII. Misalnya, chord AM7 (A-C#-E-G#), chord AbM7 (Ab-C-Eb-G).
- Chord Minor 7^{th}, terdiri dari nada I, nada III yang di-Mol (diturunkan ½ laras), V, dan nada VII yang di-Mol (diturunkan ½ laras). Misalnya, chord Dm7 (D-F-A-C), chord Em7 (E-G-B-D).
- Chord Minor $7^{th\ -5}$, terdiri dari nada I, nada III yang di-Mol (diturunkan ½ laras), nada V yang di-Mol (diturunkan ½ laras), dan nada VII yang di-Mol

(diturunkan ½ laras). Misal, chord $Bm7^{-5}$ (B-D-F-A), chord $F\#m7^{-5}$ (F#-A-C-E).

- Chord Diminished 7^{th}, terdiri dari nada I, nada III yang di-Mol (diturunkan ½ laras), nada V yang di-Mol (diturunkan ½ laras), dan nada VI. Misalnya, chord Cdim (C-Eb-Gb-A), chord Ddim (D-F-Ab-B).
- Chord Suspended 4^{th}, terdiri dari nada I, IV, dan V. Misalnya, chord Gsus4 (G-C-D), chord Esus4 (E-A-B).
- Chord Dominant 7^{th} suspended 4^{th}, terdiri dari nada I, IV, V, dan nada VII yang di-Mol (diturunkan ½ laras). Misalnya, chord Eb7sus4 (Eb-Ab-Bb-Db), chord A7sus4 (A-D-E-G).

Kita dapat pula menuliskan chord tanpa menggunakan nadanya, melainkan menggunakan posisinya pada nada dasar. Misalnya, chord I berarti chord yang sesuai dengan nada dasar, chord V7 berarti chord dominant 7^{th} dari nada kelima, dan lainnya.

Bass adalah suara rendah yang ditambahkan pada lagu. Bass juga ditujukan untuk memperindah lagu. Antara chord dan bass, keduanya sulit untuk dipisahkan. Nada bass biasanya sesuai dengan chordnya. Misalnya, chord C akan diikuti pula dengan bass C. Namun, kadangkala kita mengganti posisi bass pada posisi lainnya. Bila kita menggunakan nada bass berdasarkan nada III (Mayor) atau nada III yang di-Mol (Minor) dari chord tersebut, maka chord itu disebut chord balikan pertama (inversi I). Bila kita menggunakan

nada bass berdasarkan nada V (Mayor ataupun Minor) dari chord tersebut, maka chord itu disebut balikan kedua (inversi II). Biasanya tanda inversi ini diletakkan pada posisi atas, misalnya, $V7^{II}$ (Chord V7 inversi kedua).

BAB III
FORMAT FILE MIDI [2)]

MIDI (*Musical Instrument Digital Interface*) adalah salah satu format komunikasi yang paling sering digunakan untuk menghubungkan antara keyboard (organ) dengan komputer, ataupun antara keyboard (organ) dengan keyboard lainnya. Spesifikasi MIDI terdiri dari perangkat keras (*hardware*) dan perangkat lunak (*software*). Perangkat keras untuk MIDI berupa kabel untuk hubungan secara serial. Sedangkan perangkat lunak untuk MIDI berupa program protokol untuk mengatur komunikasi melalui kabel serial tersebut. Mengenai format komunikasi dari MIDI tidak akan dijelaskan pada tugas akhir ini. Dalam tugas akhir ini, akan lebih ditekankan pada format file MIDI, yaitu file yang menyimpan informasi suatu lagu.

3.1. Format Data

Data-data pada file MIDI tersimpan dalam format 8 bit (*byte*). Bit-0 merupakan bit yang paling rendah (*least significant bit*) dari 1 byte data, sedangkan bit-7 merupakan bit yang paling tinggi (*most significant bit*) dari 1 byte data.

[2)] Rowland, Neil Jr. ; **"MIDI.HLP"**.

3.2. Variable-Length Quantity

Dalam file MIDI, ada suatu aturan tertentu untuk menyimpan nilai kuantitas, yang disebut *variable-length quantity*. Pada format *variable-length quantity* ini, angka kuantitasnya disimpan dalam bentuk 7 bit per byte. Bit-7 dari data-datanya digunakan sebagai *flag*. Tujuan dari format ini adalah untuk meminimalkan ukuran file.

Bila kuantitasnya antara 0-127, maka hanya satu byte yang dibutuhkan untuk menyimpan informasi tersebut dimana bit-7 diset '0'. Bila kuantitasnya lebih dari 127, maka dibutuhkan lebih dari satu byte, dimana bit-7 dari data-data byte tersebut diset '1', kecuali byte terakhir yang diset '0'. Perhatikan contoh-contoh data kuantitas dalam heksadesimal pada tabel berikut :

KUANTITAS	VARIABLE-LENGTH QUANTITY
00000000	00
00000040	40
0000007F	7F
00000080	81 00
00002000	C0 00
00003FFF	FF 7F
00004000	81 80 00
00100000	C0 80 00
001FFFFF	FF FF 7F

KUANTITAS	VARIABLE-LENGTH QUANTITY
00200000	81 80 80 00
08000000	C0 80 80 00
0FFFFFF	FF FF FF 7F

Table 3.1. Contoh Penggunaan Variable-Length Quantity

Karena jumlah byte maksimal untuk menyimpan *variable-length quantity* adalah 4 byte, maka jumlah kuantitas maksimal yang dapat disimpan adalah 0FFFFFFF dalam heksadesimal.

3.3. Chunk

File MIDI terdiri dari blok-blok yang disebut *chunk*. Setiap chunk akan berisi informasi tipe chunk, jumlah data dalam chunk, dan diikuti data-data dari chunk tersebut. Tipe chunk disimpan dalam 4 byte data. Ada 2 tipe chunk dalam file MIDI, yaitu *header chunk* dan *track chunk*. Setiap file MIDI selalu terdiri dari satu header chunk dan diikuti oleh satu atau lebih track chunk. Jumlah data dalam chunk disimpan dalam 4 byte (32 bit) dengan urutan byte terbesar terlebih dulu.

3.3.1 Header Chunk

<Header Chunk> = <chunk-type><length><format><ntrks>
<division>

Header chunk selalu mengawali file MIDI. Pada chunk ini disimpan informasi pokok mengenai data dalam file. Header chunk ditandai dengan 4 byte tipe, **"MThd"** (<*chunk-type*>). Jumlah data dalam header chunk (<*length*>) adalah 6 byte (tipe chunknya tidak ikut dihitung), jadi selalu bernilai 00000006 (4 byte) dalam heksadesimal. Enam byte data tersebut terdiri dari 3 informasi 16 bit, dengan urutan byte terbesar terlebih dulu.

Data 2 byte pertama (<*format*>) menyimpan informasi mengenai format file MIDI. Seperti dijelaskan pada bagian awal, ada 3 format file MIDI yang ada, yaitu format 0, 1, dan 2. Format 0 berarti file MIDI tersebut terdiri dari satu track yang multi-channel. Format 1 berarti file MIDI tersebut terdiri dari satu atau lebih track data sekuensial yang harus dijalankan secara serentak (simultan). Format 2 berarti file MIDI tersebut terdiri dari satu atau lebih track yang tidak saling bergantung (*independent*) dan dijalankan secara sekuensial.

Data 2 byte berikutnya (<*nrks*>) menyimpan informasi mengenai jumlah track yang ada. Pada file MIDI dengan format 0, data ini akan selalu bernilai 1.

Data 2 byte terakhir (<*division*>) menyimpan informasi mengenai jumlah delta. Ada 2 format dari <*division*> ini, yaitu bentuk waktu metrik

(*metrical time*) dan bentuk waktu yang berdasarkan kode waktu (*time-code-based time*).

- Bila bit-15 dari <*division*> bernilai '0', berarti formatnya dalam waktu metrik. Bit-0 hingga bit-14 menyatakan jumlah ketukan waktu (*ticks*) untuk satu not 1/4. Misalnya, bila nilainya 120 desimal, maka not 1/4 terdiri dari 120 ticks, not 1/8 terdiri dari 60 ticks, dan seterusnya.
- Bila bit-15 dari <*division*> bernilai '1', berarti formatnya dalam waktu yang berdasarkan kode waktu. Format ini ditujukan untuk SMPTE (*Society for Motion Picture and Television Engineers*) dan kode waktu MIDI. Bit-8 sampai bit-14 berisi salah satu dari 4 nilai berikut, -24, -25, -29, atau -30, yang merupakan 4 macam standar SMPTE dan kode waktu MIDI. Nilai negatif tersebut disimpan dalam bentuk *two's complement*. Sedangkan bit-0 sampai bit-7 berisi jumlah ticks untuk tiap frame. Nilai yang mungkin adalah 4 (resolusi untuk kode waktu MIDI), 8, 10, 80 (resolusi bit), atau 100. Misalnya, bila file tersebut disimpan dalam resolusi bit dengan kode waktu 30 frame, maka nilai <*division*> adalah E250 heksadesimal.

3.3.2. Track Chunk

<Track Chunk> = <chunk-type><length><MTrk event>+
<MTrk event> = <delta-time><event>
<event> = <MIDI event>|<sysex event>|<meta event>

Track chunk mempunyai kode tipe **"MTrk"** (<*chunk-type*>). Setelah tipe chunk, data selanjutnya adalah <*length*> yang menyimpan jumlah data dalam chunk (<*chunk-type*> tidak ikut dihitung). Data <*length*> disimpan dalam 4 byte.

Setiap track chunk pada dasarnya berisi rangkaian data-data *event*. Tanda positif (+) berarti satu atau lebih. Event ini terlebih dulu didahului oleh waktu delta (<*delta-time*>). Waktu delta ini disimpan dalam format *variable-length quantity*. Waktu delta ini menyatakan jumlah ticks yang mendahului event tersebut. Ada 3 macam event, yaitu MIDI event, *syxtem exclusive* event, dan meta event. Mengenai event-event ini akan dijelaskan lebih lanjut.

3.4. Channel

Untuk mengeluarkan suara-suara musik dengan menggunakan *sound card/sound blaster*, Windows menyediakan 16 channel yang dapat digunakan oleh user.

- Channel 0 sampai 8 : track perluasan untuk melodi (*extended melodic track*).
- Channel 9 : track perluasan untuk perkusi (*extended percussion track*).
- Channel 10 dan 11 : tidak digunakan.

- Channel 12 sampai 14 : track dasar untuk melodi (*base-level melodic track*).
- Channel 15 : track dasar untuk perkusi (*base-level percussion track*).

3.5. Event

3.5.1. MIDI Event

MIDI event ini berupa perintah-perintah yang dikirimkan ke channel suara yang ada. Macam perintahnya :

- 80h - 8Fh : Mematikan bunyi pada channel (0 - 15).
- 90h - 9Fh : Menyalakan bunyi pada channel (0 - 15).

  Kedua perintah tersebut akan diikuti dengan nada yang akan dibunyikan (00h - 7Fh) dan keras-lemah nada tersebut (00h - 7Fh). Nada 00h sama dengan nada C_0 (nada C level 0), sedangkan nada 7Fh sama dengan nada G_{10} (nada G level 10).
- A0h - AFh : Penekanan terhadap nada (*key pressure*).

  Diikuti oleh nada yang mengalami penekanan, serta nilai bobot penekanan (00h - 7Fh).
- B0h - BFh : Kontrol untuk channel tersebut.

  Diikuti oleh kode kontrol (00h - 7Fh) dan nilai dari kontrol tersebut (00h - 7Fh).

- C0h - CFh : Jenis suara untuk channel tersebut.

  Diikuti oleh jenis suara yang dipilih (00h - 7Fh).

- D0h - DFh : Penekanan channel (*channel pressure*).

  Diikuti oleh nilai bobot penekanannya (00h - 7Fh).

- E0h - EFh : Pergeseran nada (*pitch wheel*).

  Diikuti oleh 2 byte nilai pergeserannya (0000h - 7F7Fh).

Pada MIDI event ini dikenal istilah *running status*, yaitu memberikan perintah yang sama tanpa perlu mengulangi perintah tersebut. Misalnya, membunyikan nada C_5 dan nada E_5 bersamaan pada channel 1, instruksinya :

00 91 3C 64 00 3E 64

Keterangan :

00 ⟶ jumlah ticks sebelum event nada C_5.
91 ⟶ perintah untuk membunyikan nada pada channel 1.
3C ⟶ membunyikan nada C_5.
64 ⟶ keras-lemah nada C_5 tersebut.
00 ⟶ jumlah ticks sebelum event nada E_5.
3E ⟶ membunyikan nada E_5.
64 ⟶ keras-lemah nada E_5 tersebut.

Perhatikan bahwa instruksi 91h hanya digunakan satu kali saja. *Running status* ini akan terus berlangsung selama tidak ada instruksi lain yang menggantikannya.

3.5.2. System Exclusive Event

Digunakan untuk mengirimkan suatu data yang khusus, baik berupa satu unit atau berupa paket. Event ini digunakan untuk mengirimkan data untuk tujuan tertentu (bergantung pada peralatan hardware dan softwarenya), dengan kata lain digunakan sebagai *"escape"*. Event ini mempunyai sintaks :

F0 *<length><bytes to be transmitted>*
F7 *<length><bytes to be transmitted>*

Data *<length>* disimpan dalam format *variable-length quantity*. Sintaks pertama digunakan untuk pengiriman paket pertama kali. Untuk pengiriman paket-paket berikutnya digunakan sintaks kedua. Paket-paket data yang dikirimkan perlu diakhiri dengan F7, sehingga program yang menganalisa data tersebut dapat mengetahui bahwa akhir dari paket data telah tercapai.

3.5.3. Meta Event

Meta event adalah event-event yang menyimpan informasi track. Berikut beberapa contoh meta event :

- FF 01 *<len> <text>* Pesan teks untuk apapun (*Text Event*).
- FF 02 *<len> <text>* Menyimpan informasi hak cipta.
- FF 03 *<len> <text>* Menyimpan nama track.

- FF 04 <*len*> <*text*> Menyimpan nama instrumen.
- FF 05 <*len*> <*text*> Menyimpan lirik lagu.
- FF 06 <*len*> <*text*> Memberikan suatu nama tanda.
- FF 07 <*len*> <*text*> Memberikan suatu deskripsi kejadian.
- FF 21 00 Awal track.
- FF 2F 00 Akhir track.
- FF 51 03 *ttttt* Mengeset tempo (dalam mikro detik tiap not 1/4).
- FF 54 05 *hr mn se fr ff* Menyatakan waktu SMPTE, kapan track chunk harus dimulai.
- FF 58 04 *nn dd cc bb* Menyatakan tanda birama,
  nn adalah pembilangnya,
  dd adalah penyebutnya (pangkat negatif dari 2),
  cc adalah jumlah ticks untuk metronome,
  bb adalah jumlah not 1/32 untuk setiap not 1/4.
- FF 59 02 *sf mi* Menyatakan nada dasar lagu,
  sf = -7 : 7 Mol
  sf = -1 : 1 Mol
  sf = 0 : nada dasar C
  sf = 1 : 1 Kres
  sf = 7 : 7 Kres
  mi = 0 : Tangga nada mayor
  mi = 1 : Tangga nada minor.

BAB IV
PENGENALAN POLA DENGAN JARINGAN SARAF ART (ADAPTIVE RESONANCE THEORY)

Pada bab berikut ini akan disinggung sedikit pengertian dari Pengenalan Pola dan Jaringan Saraf. Kemudian akan dijelaskan secara detail mengenai arsitektur, operasi, algoritma, maupun pelatihan dari jaringan saraf ART. Jaringan saraf ART ini merupakan jaringan saraf yang dipilih dalam pembuatan tugas akhir ini.

4.1. Pengenalan Pola

Pengenalan pola (*pattern recognition*) adalah suatu metode yang mendasari prosesnya berdasarkan pola-pola tertentu. Tujuan utama dari pengenalan pola ini adalah klasifikasi : bila kita diberi suatu input, dapatkah kita menganalisa input tersebut dengan menentukan kategori yang penting dari data-data di dalamnya. Untuk lebih mengerti apa maksud dari pengenalan pola ini, misalkan kita mengambil suatu masalah yang mudah bagi sebagian besar orang, yaitu membaca. Teks-teks yang kita baca merupakan pola-pola yang tersusun atas huruf. Sistem visual kita harus melakukan pengenalan pola terhadap huruf-huruf tersebut, dan mengklasifikasikannya. Tugas tersebut

kelihatan mudah, namun bila kita mengalihkan tugas tersebut pada komputer, akan terlihat betapa kompleksnya masalah tersebut. Biasanya kita menyelesaikan masalah pengenalan pola ini dengan melakukan pencocokan (*matching*), dimana huruf tersebut ditempatkan pada frame ukuran tertentu dan dibandingkan dengan pola semua huruf yang disimpan. Apa yang terjadi bila huruf tersebut dipertebal ? Bila kita tidak mempunyai pola kedua untuk huruf tebal, maka proses klasifikasinya bisa gagal.

Bila masalah teks tulisan tangan yang harus dikenali polanya, maka hampir tidak mungkin kita menyimpan semua pola gambar kurva yang sangat bervariasi. Dengan demikian, kita harus menggunakan cara yang lain. Pemrosesan teks tersebut merupakan satu contoh masalah pengenalan pola. Masalah pengenalan pola yang lain misalnya pemrosesan gambar, pemrosesan suara, identifikasi suatu benda, dan lain-lain.

Pada prinsipnya, pengenalan pola terdiri dari dua tahap. Pertama, tahap pengambilan ciri-ciri/karakteristik (*feature extraction*). Kedua, tahap klasifikasi. Ciri-ciri yang kita ambil merupakan suatu ukuran yang diambil dari input yang akan diklasifikasikan, biasanya berupa karakteristik yang terdefinisikan. Misalnya, bila kita hendak membedakan huruf 'E' dengan huruf 'F', kita perlu membandingkan jumlah garis vertikal dan horisontal dari karakter tersebut. Bagian klasifikasi sendiri diberi daftar ciri-ciri yang diukur dari input. Tugasnya mencocokkan ciri-ciri input tersebut dengan kategori

ciri-ciri yang sudah ada. Bagian klasifikasi ini harus memutuskan kategori mana yang paling dekat dengan input tersebut.

4.2. Jaringan Saraf

4.2.1. Jaringan Saraf Manusia

Saraf pada manusia sangat kompleks dan saling berhubungan, membentuk suatu jaringan. Setiap sel saraf (*neuron*) mempunyai input yang disebut *dendrite* dan output yang disebut *axon*. *Dendrite* dari suatu sel saraf akan terhubung ke *axon* dari sel saraf yang lain melalui hubungan (*junction*) yang disebut *synapse*. Hubungan *synapse* ini dapat mengubah efektivitas dari sinyal yang dikirimkan. Ada *synapse* yang merupakan penghubung yang baik, dan dapat melewatkan sinyal dengan baik. Ada pula *synapse* yang jelek, dan hanya melewatkan sedikit sinyal saja. Sinyal-sinyal dari sel-sel saraf input akan dikirimkan melalui *synapse-synapse* yang ada, dan akan diterima badan (*soma*) sel saraf melalui *dendrite-dendrite*-nya.

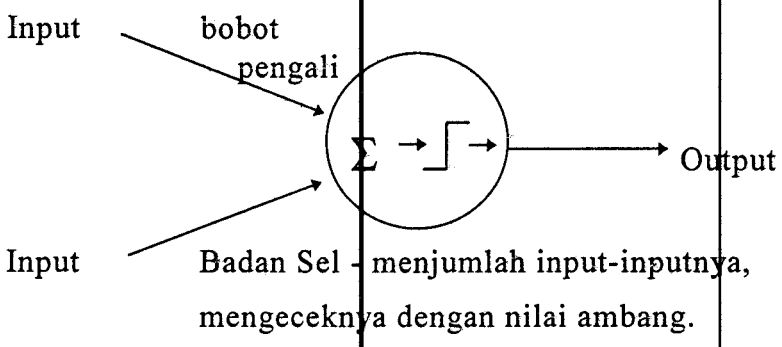
Setiap sel saraf mempunyai nilai ambang tertentu (*threshold*). Bila total input-input yang diterima sel saraf tersebut melebihi nilai ambang ini, sel saraf akan memberikan sinyal outputnya (*firing*). Sebaliknya, bila total input-inputnya lebih kecil atau sama dengan nilai ambang tesebut, sel saraf tidak memberikan sinyal output ke sel saraf berikutnya.

4.2.2. Jaringan Saraf Buatan

Jaringan saraf buatan (*neural network*) pada komputer disusun dengan mengambil karakteristik dari jaringan saraf manusia. Beberapa karakteristik penting yang dapat diambil dari jaringan saraf manusia adalah :

- Output dari sel saraf (*neuron*) dapat aktif (*on*) ataupun tidak aktif (*off*).
- Output *neuron* hanya bergantung pada input-inputnya. Suatu nilai tertentu harus dicapai agar *neuron* menembakkan outputnya.

Cara kerja dari *synapse* yang meneruskan sinyal input ke badan sel saraf, dapat dimodelkan dengan memberikan faktor pengali (bobot) pada tiap input dari sel saraf buatan. *Synapse* yang lebih effisien, yang meneruskan input lebih banyak, akan dimodelkan dengan jalur yang mempunyai bobot lebih besar. Sebaliknya *synapse* yang lebih buruk, yang meneruskan input lebih sedikit, akan dimodelkan dengan jalur yang mempunyai bobot lebih kecil. Model dasar dari sel saraf buatan ini dapat dilihat pada gambar 4.1. di bawah ini.



Gambar 4.1. Model Dasar dari Sel Saraf Buatan

Sel saraf tersebut melakukan penjumlahan input-inputnya yang dikalikan dengan bobot masing-masing, lalu membandingkan hasilnya dengan nilai ambang internal, dan memberi output '1' bila nilai ambang ini terlewati. Sebaliknya, bila tidak melewati nilai ambang, maka outputnya tetap '0'.

4.3. Pengenalan Pola dengan Jaringan Saraf ART

4.3.1. Introduksi

Pengenalan pola banyak mendominasi aplikasi-aplikasi dari jaringan saraf. Pengenalan pola sendiri merupakan ilmu yang sangat luas cakupannya dalam komputer. Ada banyak metode yang dapat dilakukan untuk pengenalan pola. Pada bagian Pendahuluan telah disebutkan 3 macam metode[3)] yang dapat dilakukan, yaitu secara statistik, secara sintaksis/struktural, dan terakhir secara jaringan saraf. Pada tugas akhir ini, dipilih cara yang terakhir.

Ada banyak topologi jaringan saraf yang dapat digunakan untuk pengenalan pola, misalnya feedback, Kohonen, ART, dan lain-lain. Untuk tugas akhir ini, digunakan topologi ART (*Adaptive Resonance Theory*). ART ini dipilih, karena kemampuannya untuk mengatasi masalah yang disebut *stability-plasticity dilemma*. Dilema ini terjadi bila jaringan saraf tidak dapat lagi melakukan proses belajar terhadap informasi baru. Misalnya, pada jaringan saraf dengan banyak lapisan (*layer*), penambahan vektor baru terhadap jaringan saraf yang sudah mengalami pelatihan, akan menimbulkan

3) Schalkoff, Robert J. ; **"Pattern Recognition : Statistical, Structural, and Neural Approaches"** ; hal 18-21.

efek samping berupa perusakan nilai bobot yang sudah didapat dari proses belajar sebelumnya. Selain itu, ART juga dapat mempersingkat waktu pelatihan jaringan saraf yang biasanya memakan waktu cukup lama.

4.3.2. Adaptive Resonance Theory (ART) [4)]

ART diperkenalkan pertama kali oleh Dr. Stephen Grossberg dan Gail Carpenter. Karakteristik utama dari ART ini adalah kemampuannya untuk berpindah mode antara *plastic* (saat proses belajar, bobot-bobot internalnya dapat dimodifikasi) dan *stable* (himpunan klasifikasinya tetap), tanpa merusak bobot-bobot yang diperoleh dari proses belajar sebelumnya.

Ada tiga versi dari ART, yaitu ART-1, ART-2, dan ART-3. ART-1 digunakan untuk pola-pola binary. ART-2 digunakan untuk digunakan untuk menyimpan pola-pola analog. Sedangkan ART-3 ditekankan untuk perhitungan masalah-masalah hierarki. Buku-buku referensi yang Penulis baca hanya menjelaskan arsitektur dari ART-1. Oleh karena itu, pada tugas akhir ini akan dijelaskan ART-1 saja.

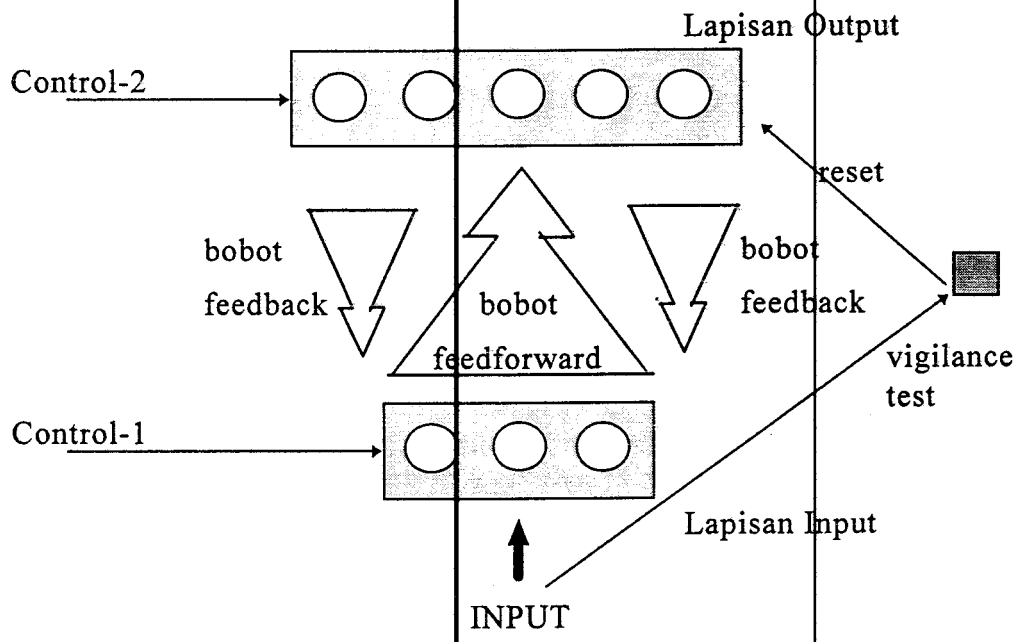
4.3.2.1. Arsitektur ART

ART mempunyai dua buah lapisan. Lapisan pertama adalah lapisan input / pembanding (*comparison layer*), dan lapisan kedua adalah lapisan

---

4) Beale, Russell dan Jackson, Tom ; **"Neural Computing : An Introduction"** ; hal 165-190.
Fu, Limin ; **"Neural Network in Computer Intelligence"** ; hal 101-105.
Schalkoff, Robert J. ; **"Pattern Recognition : Statistical, Structural, and Neural Approaches"** ; hal 276-281.

output / pengidentifikasi (*recognition layer*). Lapisan-lapisan ini saling terhubung, tidak seperti jaringan saraf yang lain, *feedback* dari lapisan output dikirimkan kembali ke lapisan input dan juga ke node-node lain pada lapisan output tersebut (sebagai penghambat / *inhibition*).



Gambar 4.2. Arsitektur ART

Jaringan ART mempunyai vektor bobot *feedforward* dari lapisan input ke lapisan output, dan vektor bobot *feedback* dari lapisan output ke lapisan input. Vektor *feedforward* dan *feedback* masing-masing disimbolkan dengan W dan T.

Pada setiap lapisan terdapat sinyal kontrol yang mengontrol aliran data ke lapisan tersebut pada setiap tahap siklus operasi, yaitu *control-1* dan *control-2*. Control-1 menentukan aliran data untuk lapisan input, dan nilainya

akan mengaktifkan node-node lapisan input antara dua mode : input dan pembandingan. Control-1 bernilai '1' bila input jaringan saraf valid (tidak nol), dan bernilai '0' bila lapisan output aktif. Control-2 berfungsi untuk mengaktifkan dan menonaktifkan node-node lapisan output. Control-2 bernilai '1' bila pola input valid, dan bernilai '0' bila tes nilai ambang gagal (node-node lapisan output akan dinonaktifkan dan level aktivasinya diset nol).

Antara lapisan input dan output terdapat rangkaian *reset*. Rangkaian ini akan melakukan reset terhadap node-node output. Rangkaian ini juga bertanggung jawab untuk membandingkan antara input-input dengan nilai ambang (*vigilance threshold*), dan menentukan apakah perlu dibentuk pola kelas yang baru untuk input tersebut.

4.3.2.2. Operasi ART

Ada beberapa fase belajar dan klasifikasi dalam jaringan ART. Perbedaan yang paling jelas bila dibandingkan dengan jaringan saraf yang lain adalah vektor input dikirimkan bolak-balik antara lapisan-lapisannya dalam suatu siklus proses (*resonated*). Operasi ART dapat dibagi menjadi 4 fase, yaitu fase inisialisasi (*initialisation*), fase identifikasi (*recognition*), fase pembandingan (*comparison*), dan fase pencarian (*searching*).

Berikut ini penjelasan dari masing-masing fase :

☞ Fase Inisialisasi

Pada fase ini, vektor-vektor bobot W dan T harus diinisialisasi. Untuk vektor T semuanya diinisialisasi dengan nilai '1', yang menunjukkan bahwa setiap node output terhubung ke semua node input. Untuk vektor W diinisialisasi dengan nilai real yang ditentukan oleh :

$$w_i = \frac{1}{1+n}$$

dimana n adalah jumlah node input.

Nilai ambang (*vigilance threshold*) juga diset pada jangkauan $0 < \rho < 1$.

☞ Fase Identifikasi

Pada fase ini, vektor input dimasukkan dalam jaringan melalui lapisan input dan nilainya dicocokkan dengan klasifikasi-klasifikasi yang ada pada tiap node lapisan output. Node-node pada lapisan input mempunyai tiga buah input, yaitu : komponen dari vektor input x_i, sinyal *feedback* dari lapisan output, dan sinyal control-1. Aliran data yang melalui lapisan input ini akan dikontrol oleh aturan "dua dari tiga" (*two-thirds*), yaitu bila dua dari tiga input node tersebut aktif, maka output node tersebut adalah '1', sebaliknya outputnya adalah '0'.

Vektor input akan dibandingkan dengan pola yang tersimpan pada tiap node output, dan dicari pola yang paling cocok. Pencarian pola yang paling

cocok ini dilakukan dengan melakukan perkalian dot antara vektor input dengan vektor bobot node, dengan demikian vektor bobot yang paling cocok dengan vektor input akan memberikan hasil terbesar. Beberapa node pada lapisan output kemungkinan akan memberikan aktivasi yang cukup tinggi terhadap suatu vektor input, tetapi dengan adanya penghalang (*inhibition*) antar node output, maka hanya node output terbesar saja yang akan menjadi pemenang. Node output yang menjadi pemenang akan mengirimkan pola yang tersimpan di dalamnya ke node-node lapisan input.

☞ Fase Pembandingan

Dua vektor akan berada pada lapisan input pada fase pembandingan. Vektor pertama merupakan vektor input, sedangkan vektor kedua adalah vektor pola yang dikirimkan oleh node output pemenang. Input ketiga dari lapisan input adalah sinyal control-1, yang pada fase ini bernilai '0' (karena lapisan output aktif). Aturan "dua dari tiga" akan digunakan untuk menghitung output dari tiap node. Vektor input dan vektor pola dari node output akan di-AND-kan untuk menghasilkan output vektor yang baru. Vektor ini disebut vektor pembandingan. Vektor ini akan dikirimkan ke rangkaian *reset* bersama-sama dengan vektor input.

Rangkaian *reset* bertanggung jawab untuk melakukan pengecekan kesamaan antara vektor input dan vektor pembandingan, dengan nilai ambang (*vigilance threshold*) sebagai acuannya. Pengetesan dilakukan

dengan membandingkan jumlah output '1' pada kedua vektor. Hal ini dapat dilakukan dengan melakukan perkalian dot antara kedua vektor, kemudian hasilnya dibandingkan dengan penjumlahan elemen-elemen (bit-bit) vektor input. Nilai rasio ini kemudian dibandingkan dengan nilai ambangnya.

$$S = \frac{\sum t_{ij} x_i}{\sum x_i}$$

tes : Apakah $S > \rho$

Bila S lebih besar dari nilai ambang, ρ, maka klasifikasi tersebut selesai dan keanggotaan kelas tersebut ditunjukkan oleh node yang aktif pada lapisan output. Sebaliknya, bila S lebih kecil atau sama dengan ρ, berarti pola yang benar belum ditemukan, dan jaringan saraf akan melalui fase pencarian.

☞ Fase Pencarian

Selama fase ini, jaringan saraf berusaha mencari vektor lain yang lebih cocok pada lapisan output untuk vektor input saat itu. Pertama, node output yang aktif akan dinonaktifkan (*disabled*) dan outputnya diset '0'. Dengan demikian, node output ini tidak dapat lagi menjadi node pemenang untuk pola vektor input saat itu. Nilai sinyal control-2 akan diset '0'. Input

vektor diaplikasikan kembali pada lapisan output untuk mencari node pemenangnya. Jaringan saraf memasuki fase pembandingan kembali, dimana pola yang baru dites dengan nilai ambang. Proses ini akan diulang-ulang terus, sambil menonaktifkan node-node output, sampai node output yang memberikan nilai rasio lebih besar dari nilai ambang didapatkan. Bila tidak ada lagi node output yang ditemukan, jaringan saraf akan memutuskan bahwa vektor input tersebut merupakan kelas yang tidak dikenal, dan mengalokasikannya pada node output yang kosong.

4.3.2.3. Algoritma ART

Secara keseluruhan, algoritma dari ART dapat dirumuskan sebagai berikut :

1. Inisialisasi.

$$t_{ij}(0) = 1$$

$$w_{ij}(0) = \frac{1}{1 + N}$$

$$0 \le i \le N - 1 \qquad 0 \le j \le M - 1$$

Set ρ, dimana $0 \le \rho \le 1$

$t_{ij}(t)$ dan $w_{ij}(t)$ adalah bobot-bobot untuk hubungan *top-down* dan *bottom-up* antara node input i dan output j pada waktu t. ρ adalah nilai ambang

(*vigilance threshold*) yang menunjukkan seberapa cocok suatu input dengan pola yang disimpan. Ada M node output dan N node input.

2. Masukkan input baru.

3. Hitung kecocokan.

$$\mu_j = \sum_{i=0}^{N-1} w_{ij}(t)\, x_i$$

$$0 \le j \le M - 1$$

μ_j adalah output dari node j, sedangkan x_i adalah elemen ke-i dari input, yang dapat bernilai '0' atau '1'.

4. Pilih pola yang paling cocok.

$$\mu_{j^*} = \max_j[\mu_j]$$

5. Pengetesan.

$$\| X \| = \sum_{i=0}^{N-1} x_i$$

$$\| T \,.\, X \| = \sum_{i=0}^{N-1} t_{ij^*}(t)\, x_i$$

$$\text{Apakah } \frac{\| T \,.\, X \|}{\| X \|} > \rho$$

YA	menuju langkah 7
TIDAK	menuju langkah 6

6. Non-aktifkan node output pemenang.

   Set output dari node output menjadi '0', menuju langkah 3.

7. Rubah bobot dari node output pemenang.

$$t_{ij*}(t+1) = t_{ij*}(t)\, x_i$$

$$w_{ij*}(t+1) = \frac{t_{ij*}(t)\, x_i}{0.5 + \sum_{i=0}^{N-1} t_{ij*}(t)\, x_i}$$

8. Ulangi.

   Aktifkan kembali node-node output yang sebelumnya dinonaktifkan, lalu kembali ke langkah 2.

4.3.2.4. Pelatihan ART

Siklus pelatihan dari ART mempunyai filosofi belajar yang berbeda dengan jaringan saraf lain. Algoritma belajar dari ART dioptimasi agar dapat mengalami pelatihan lagi pada sembarang waktu, untuk menambahkan data yang baru. Inilah solusi praktis dari masalah *stability-plasticity* yang sudah dijelaskan sebelumnya, dan jaringan ART mungkin satu-satunya jaringan saraf yang dapat melakukan proses belajar pada kondisi lingkungan yang berubah-ubah.

Ada dua macam proses belajar dari ART, yaitu proses belajar yang cepat (*fast learning*) dan proses belajar yang lambat (*slow learning*). Disebut proses belajar yang cepat, karena bobot-bobot pada *feedforward* dapat mencapai nilai optimum dengan beberapa siklus belajar saja. Sedangkan pada proses belajar yang lambat, bobot-bobot diadaptasikan perlahan-lahan melalui siklus belajar yang banyak. Keuntungan dari proses belajar yang lambat, prosesnya lebih difokuskan untuk mencari ciri-ciri yang menonjol dari pola input, yang menentukan klasifikasinya.

ART sangat sensitif terhadap perubahan-perubahan pada parameternya selama pelatihan. Parameter yang paling kritis adalah nilai ambangnya (*vigilance threshold*), yang dapat mengubah performansi secara drastis. Parameter lain yang penting, bobot-bobot untuk *feedforward* harus diinisialisasi dengan nilai yang kecil, bila tidak, vektor yang tidak diinisialisasi dengan nilai kecil tersebut akan mendominasi proses pelatihan.

Parameter *vigilance* mengontrol resolusi dari proses klasifikasi. Nilai ambang kecil (< 0,4) akan menghasilkan proses klasifikasi dengan resolusi rendah, sehingga hanya sedikit kelas yang terbentuk. Sebaliknya, nilai ambang yang besar (mendekati 1) akan menghasilkan resolusi klasifikasi yang tinggi, yang berarti sedikit perbedaan saja antara pola-pola input akan mengakibatkan terbentuknya kelas baru. Suatu cara yang cukup optimal adalah dengan mengubah-ubah nilai *vigilance* selama proses pelatihan. Mula-mula nilai

inisialnya diset rendah, sehingga kelas-kelasnya terbentuk dengan cepat dalam bentuk kasar. Kemudian nilai ini dinaikkan selama pelatihan supaya klasifikasinya optimal.

BAB V
PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

Untuk tugas akhir ini, dibuat dua buah perangkat lunak. Perangkat lunak pertama digunakan untuk proses pelatihan untuk mencari bobot-bobot dari jaringan saraf ART, dan sekaligus menyimpan pola-pola chord yang dimasukkan oleh musisi. Sedangkan perangkat lunak kedua merupakan program interaksi dengan user.

Sebelum membahas perancangan dan pembuatan kedua perangkat lunak tersebut, ada beberapa hal yang perlu dijelaskan :

- Kedua perangkat lunak tersebut menggunakan bahasa pemrograman BorlandC++ yang berorientasi objek (OOP). Oleh karena itu, pembahasan program akan menggunakan bahasa tersebut dalam penjelasannya.
- Dalam cuplikan program pada pembahasan perangkat lunak ini, ada beberapa bagian yang bukan merupakan instruksi program, melainkan merupakan komentar. Komentar ditandai dengan // (*double-slash*).
- Dalam pembahasan nanti, ada fungsi dan variable yang tidak dijelaskan secara detail. Fungsi dan variable tersebut ditandai dengan garis bawah. Pada fungsi dan variable tersebut akan diberi sedikit komentar sebagai penjelasan.

5.1. Perangkat Lunak Pelatihan Jaringan Saraf

Sebagaimana sudah dijelaskan pada bab-bab sebelumnya, untuk penentuan chord-chord dari suatu lagu, digunakan metode pengenalan pola dengan menggunakan jaringan saraf. Jaringan saraf yang digunakan untuk pengenalan pola tersebut harus mengalami proses belajar melalui pelatihan. Untuk itulah, perangkat lunak ini dibuat.

Jaringan saraf yang digunakan mempunyai topologi ART (*Adaptive Resonance Theory*). Seperti sudah dijelaskan pada Bab IV sebelumnya, jaringan saraf ART tersebut hanya dapat menyimpan pola-pola binary. Dengan demikian, jaringan saraf ART yang asli tersebut perlu mengalami modifikasi, sesuai dengan kebutuhan yang ada.

Beberapa elemen maupun proses yang dimodifikasi antara lain :

- Node input dari jaringan saraf tersebut harus dapat menerima input selain binary. Selain itu, masing-masing node input mempunyai karakteristik input yang berbeda. Berbeda dengan jaringan saraf ART yang sudah dijelaskan sebelumnya, jaringan saraf untuk penentuan chord ini mempunyai parameter yang tidak sejenis. Parameter-parameter tersebut antara lain : jenis tanda birama, jenis tangga nada, jenis chord, serta jenis-jenis not/nada.

- Antara vektor input dengan pola yang disimpan dalam node output, mempunyai elemen-elemen yang berbeda. Misalnya, pada vektor input kita berikan nada-nada lagu yang hendak ditentukan chordnya. Outputnya tentu saja bukan berupa nada-nada lagi, melainkan berupa chord-chord dari pola yang tersimpan dalam node output.
- Karena inputnya bukan binary, maka operasi yang dilakukan dalam tiap-tiap node bukan lagi operasi perkalian vektor secara dot. Sebagai alternatifnya, digunakan operasi pencocokan (*matching*) dengan memberi tingkat-tingkat bobot tertentu untuk setiap tingkat kecocokannya. Misalnya, untuk node input jenis tangga nada, bila input tangga nada tersebut sama dengan tangga nada yang disimpan dalam pola di node output, maka bobot tingkat kecocokannya adalah 1, sebaliknya bobot tingkat kecocokannya adalah 0,5. Untuk parameter-parameter input yang lain, bobot tingkat kecocokannya akan berbeda.
- Untuk kesederhanaan, sinyal-sinyal control-1 dan control-2 pada jaringan ART ditiadakan. Sebagai gantinya, algoritma programlah yang akan mengatur perubahan fase-fasenya.
- Nilai ambang (*vigilance threshold*) yang digunakan adalah 1. Ini berarti, setiap input yang baru akan dimasukkan sebagai pola baru. Hal ini dilakukan karena tujuan program ini adalah menampung semua pola-pola dari musisi sebanyak-banyaknya.

- Perubahan yang paling penting adalah perubahan dari *unsupervised learning* menjadi *supervised learning*. Perubahan ini dilakukan karena pola-pola yang dimasukkan dalam jaringan ART, sepenuhnya dikontrol oleh musisi.

Dengan modifikasi-modifikasi tersebut, barulah jaringan ART yang baru tersebut dapat digunakan untuk pengenalan pola-pola chord dari musisi.

Perangkat lunak untuk pelatihan jaringan saraf ini dibuat dengan menggunakan bahasa pemrograman BorlandC++ 3.1 for DOS, sedangkan tampilannya menggunakan fasilitas Turbo Vision dari BorlandC++ 3.1 tersebut. Adapun struktur data dan algoritma yang digunakan untuk mendukung perangkat lunak ini, akan dijelaskan berikut ini.

5.1.1. Struktur Data

Struktur data yang digunakan dalam perangkat lunak ini antara lain struktur data untuk definisi elemen musik (digunakan sebagai komponen input dan komponen pola yang disimpan, misalnya : jenis tanda birama, jenis not/nada, dan lain-lain), struktur data untuk sel saraf (*neuron*), serta struktur data untuk jaringan sarafnya.

5.1.1.1. Definisi Elemen Musik

Berikut ini definisi tipe-tipe elemen musik yang digunakan :

- ***enum birama_type***
  {_2_BEAT, _3_BEAT, _4_BEAT, _6_BEAT};

  Mendefinisikan jenis tanda birama.

- ***enum scale_type***
  {M_SCALE, m_SCALE};

  Mendefinisikan jenis tangga nada.

- ***enum note_type***
  {NONE=-1, C, CIS_DES, D, DIS_ES, E, F, FIS_GES, G, GIS_AS, A, AIS_BES, B};

  Mendefinisikan jenis nada.

- ***enum prog_type***
  {MAJOR, MINOR, DOM7, AUG, SIXTH, MAJ7, MIN7, MIN7_5, DIM7, SUS4, _7SUS4};

  Mendefinisikan jenis progresi dari chord.

- ***struct chord_type***

```
{
    note_type note;                          // Nada
    prog_type prog;                          // Progresi chord

    chord_type();                            // Constructors
    chord_type(note_type in_note, prog_type in_prog);
    int notes();                             // Nada-nada chord
    int scale();                             // Tangga nada chord
    friend int operator == (chord_type& chord1, chord_type& chord2);
    friend int operator <= (chord_type& chord1, chord_type& chord2);
};
```

Mendefinisikan jenis chord. Chord terdiri dari nada (***note***) dan progresi chordnya (***prog***). Berikut penjelasan *member-functions* struct tersebut :

- Struct ini mempunyai dua constructor yang tujuannya untuk memberi nilai pada variable ***note*** dan ***prog***. Constructor pertama menginisialisasi ***note*** = ***NONE*** dan ***prog*** = ***MAJOR***. Sedangkan constructor kedua memberi nilai ***note*** dan ***prog*** sesuai parameter-parameter inputnya.
- Fungsi ***int notes()*** bertujuan untuk mengambil nada-nada penyusun chord tersebut. Misalnya, chord C7, nada-nada penyusunnya adalah C-E-G-Bb. Outputnya berupa bit-bit yang menunjukkan posisi nada-nadanya.
- Fungsi ***int scale()*** bertujuan untuk mengambil nada-nada dari tangga nada chord tersebut (tangga nada mayor atau minor, sesuai progresi chordnya). Outputnya juga berupa bit-bit yang menunjukkan posisi nada-nadanya.
- Fungsi ***int operator*** == akan mengeluarkan output '1' bila kedua chord yang dibandingkan sama, sebaliknya memberikan output '0'.
- Fungsi ***int operator*** <= akan mengeluarkan output '1' bila nada-nada penyusun chord pertama merupakan penyusun chord kedua, sebaliknya output '0'.

5.1.1.2. Sel Saraf (Neuron)

Sel saraf ini terdiri dari 2 jenis, yaitu sel saraf output dan sel saraf input. Berikut struktur data masing-masing sel saraf.

☞ Sel Saraf Output

Sel-sel saraf output mempunyai karakteristik yang sama, berbeda dengan sel-sel saraf input yang dibedakan berdasarkan input yang diterimanya. Pada sel saraf output ini disimpan pola-pola yang dimasukkan oleh musisi. Berikut struktur data dari pola yang disimpan.

```
struct pattern
{
    birama_type birama;                 // Jenis tanda birama
    scale_type scale;                   // Jenis tangga nada
    chord_type prev_chord;              // Chord sebelumnya
    chord_vector chords;                // Pola chord-chord
    chord_type next_chord;              // Chord sesudahnya

    pattern();                          // Constructors
    pattern(birama_type in_birama, scale_type in_scale, chord_type in_prev,
        chord_vector& in_chords, chord_type in_next);
    pattern& operator = (pattern& other);
    int operator == (pattern& other);
};
```

chord_vector merupakan suatu class yang menyimpan array dari chord, dimana pola-pola chord satu birama disimpan. Jumlah chord yang disimpan dalam ***chord_vector*** bergantung pada jenis tanda biramanya.

TANDA BIRAMA	JUMLAH CHORD
_2_BEAT	2 chord
_3_BEAT	3 chord
_4_BEAT	4 chord
_6_BEAT	2 chord

Tabel 5.1. Jumlah Pola Chord yang Disimpan

Berikut penjelasan *member-functions* struct ***pattern*** :

- Struct tersebut mempunyai dua buah constructor. Constructor pertama untuk menginisialisasi data-data dengan nilai default (***birama*** = ***_4_BEAT, scale = M_SCALE, prev_chord = next_chord = chord_type (NONE, MAJOR)***, constructor ***chord_vector*** melakukan inisialisasi pointer array dinamis dari ***chords*** dengan ***NULL***), sedangkan constructor kedua untuk menginisialisasi data-data sesuai parameter-parameter inputnya.
- Fungsi ***pattern& operator = (pattern& other)*** digunakan untuk merubah nilai data-datanya sesuai dengan data-data dari ***other***. Outputnya berupa *reference* ke class tersebut.
- Fungsi ***int operator == (pattern& other)*** akan memberikan nilai '1' bila nilai data-datanya sama dengan nilai data-data dari ***other***.

Struktur data dari sel saraf output adalah sebagai berikut :

```
class out_neuron
{
    float* weight;              // Bobot-bobot untuk memroses
                                // vektor input
    char output;                // Output neuron
    float activation;           // Nilai aktivasi neuron
    pattern data_mem;           // Pola yang disimpan

    void init_birama();         // Inisialisasi bobot-bobot
    void init_scale();          // untuk pola baru
    void init_prev_chord();
    void init_chords();
    void init_next_chord();
```

```
public :
    static int many_in;          // Jumlah sel saraf input
    out_neuron();                // Constructor
    ~out_neuron();               // Destructor
    int create();                // Alokasi memori
    void destroy();              // Menghapus alokasi memori
    int is_empty();              // Mengecek apakah sel kosong
    void set(pattern& data);     // Input pola baru
    char& firing();              // Nilai output
    float& act();                // Nilai activation
    birama_type birama();        // Tanda birama
    scale_type scale();          // Tangga nada
    chord_type prev_chord();     // Chord sebelumnya
    chord_vector& chords();      // Pola chord-chord
    chord_type next_chord();     // Chord sesudahnya
    int compare(pattern& in_pattern);
                                 // Membandingkan dengan pola lain
    void inc_weight(int idx, float inc);
                                 // Meningkatkan bobotnya
    float& operator [] (int idx); // Data bobotnya
};
```

Nilai *weight* digunakan untuk menyimpan bobot-bobot untuk proses dari sel-sel saraf input. Nilai *activation* digunakan untuk menyimpan total proses dari semua sel saraf input. Sedangkan *output* merupakan flag untuk menandai node pemenang (1), node bukan pemenang (0), atau node yang dinonaktifkan (-1).

Berikut penjelasan *member-functions* dari class ***out_neuron*** :

- Constructor class tersebut menginisialisasi nilai ***weight*** = ***NULL***, ***output*** = ***0***, dan ***activation*** = ***0***.

- Destructor class tersebut akan menghapus alokasi memori dari ***weight*** dengan cara memanggil fungsi ***destroy()***.
- Fungsi ***int create()*** digunakan untuk mengalokasikan memori yang digunakan oleh ***weight***. Besarnya alokasi memori ini bergantung pada nilai ***many_in*** (jumlah sel saraf input), oleh karena itu nilai tersebut harus ditentukan terlebih dulu. Fungsi ini akan memberi output '1' bila alokasi memori tersebut berhasil.
- Fungsi ***void destroy()*** digunakan untuk menghapus alokasi memori dari ***weight***.
- Fungsi ***int is_empty()*** akan memberi nilai '1' bila sel saraf tersebut kosong, belum terisi pola.
- Fungsi ***void set(pattern& data)*** digunakan untuk mengisi nilai-nilai ***data*** ke ***data_mem*** (input pola baru). Perhatikan, fungsi ***is_empty()*** harus dicek terlebih dulu sebelum memasukkan pola baru ini. Fungsi ini akan melakukan inisialisasi bobot-bobot untuk pola baru tersebut.
- Fungsi-fungsi ***void init_birama()***, ***void init_scale()***, ***void init_prev_chord()***, ***void init_chords()***, serta ***void init_next_chord()*** digunakan untuk menginisialisasi bobot-bobot untuk pola baru. Bobot-bobot ini akan digunakan untuk proses dari sel-sel saraf input, karena itu jumlahnya sesuai dengan jumlah sel saraf input

(pada jaringan saraf ini digunakan 12 sel saraf input). Fungsi-fungsi inilah yang dipanggil oleh fungsi *set* sebelumnya. Untuk inisialisasi ini, masing-masing bobot mempunyai perbandingan nilai sebagai berikut :

KOMPONEN INPUT	PERBANDINGAN
Tanda Birama	6
Tangga Nada	2
Chord Sebelumnya	4
Nada-Nada 1 birama	1 / 1.5 / 1.75 / 2
Nada Sesudahnya	2

Tabel 5.2. Perbandingan Bobot Komponen Input

Mengenai komponen dari input ini dapat dilihat pada bagian perancangan jaringan saraf dari perangkat lunak ini. Komponen nada-nada satu birama (8 sel saraf input) mempunyai nilai perbandingan yang berbeda-beda, bergantung pada tanda birama dan perubahan chordnya.

TANDA BIRAMA	PERBANDINGAN INPUT NADA							
_2_BEAT	2	0	1	0	2	0	1	0
_3_BEAT	2	1	1	1	2	1	0	0
_4_BEAT	2	1	1	1	2	1	1	1
	1.75	1	1.5	1	1.75	1	1	1
	1.75	1	1	1	1.75	1	1.5	1
	1.5	1	1.5	1	1.5	1	1.5	1
_6_BEAT	2	1	1	2	1	1	0	0

Tabel 5.3. Perbandingan Bobot Komponen Input Nada

Pembobotan input nada untuk tanda birama *_4_BEAT* ada 4 cara. Cara pertama digunakan bila chord1 = chord2 dan chord3 = chord4. Cara kedua digunakan bila chord1 ≠ chord2 dan chord3 = chord4. Cara ketiga digunakan bila chord1 = chord2 dan chord3 ≠ chord4. Dan cara terakhir digunakan bila chord1 ≠ chord2 dan chord3 ≠ chord4. Ingat, jumlah chord yang disimpan adalah 4 (lihat tabel 5.1). Penjelasan mengenai input nada-nada itu sendiri dapat dibaca pada bagian perancangan jaringan saraf.

- Fungsi-fungsi *char& firing()* dan *float& act()* masing-masing digunakan untuk mengakses nilai *output* dan *activation*.
- Fungsi-fungsi *birama_type birama()*, *scale_type scale()*, *chord_type prev_chord()*, *chord_vector& chords()*, dan *chord_type next_chord()* digunakan untuk mengakses data-data pola yang tersimpan dalam *data_mem*.
- Fungsi *int compare(pattern& in_pattern)* digunakan untuk membandingkan antara data-data pola dalam *data_mem* dengan data-data pola dalam *in_pattern*. Bila data-datanya sama, fungsi ini memberikan nilai '1', sebaliknya nilai '0'.
- Fungsi *void inc_weight(int idx, float inc)* digunakan untuk meningkatkan bobot *weight* pada posisi *idx* sebanyak *inc*.

- Fungsi ***float& operator [] (int idx)*** digunakan untuk mengakses bobot ***weight*** pada posisi ***idx***.

☞ Sel Saraf Input

Sel-sel saraf input dapat memiliki karakteristik input yang berbeda-beda. Oleh karena itu, sel saraf input dibagi lagi menurut jenisnya, antara lain sel saraf input untuk tanda birama, sel saraf input untuk tangga nada, sel saraf input untuk chord, dan sel saraf input untuk nada.

```
class in_neuron
{
    protected :
        enum
        {
            IN_NEURON,
            IN_NEURON_SCALE, IN_NEURON_BIRAMA,
            IN_NEURON_CHORD, IN_NEURON_NOTE
        } kind;                          // Jenis neuron input
        float output;                    // Output neuron

    public :
        in_neuron();                     // Constructor
        float get_output();              // Nilai output
        virtual void set(void*, int) = 0;    // Memasukkan input
        virtual void process(out_neuron&, int) = 0;      // Pemrosesan
};
```

Class ini merupakan *base-class* dari sel-sel saraf input. Penjelasan dari *member-functions* class ***in_neuron*** tersebut :

- Constructor class ini menginisialisasi nilai ***kind*** = ***IN_NEURON*** dan ***output*** = ***0***.

- Fungsi ***float get_output()*** bertujuan untuk membaca nilai ***output***.
- Fungsi ***void set(void*, int)*** digunakan untuk memasukkan input dari sel saraf tersebut, bergantung pada jenis sel saraf inputnya. Parameter pertama merupakan data input, sedangkan parameter kedua digunakan sebagai penunjuk posisi pada sel saraf input untuk nada.
- Fungsi ***void process(out_neuron&, int)*** digunakan untuk memroses sel saraf input tersebut dengan sel saraf output tertentu sebagai acuannya. Parameter pertama merupakan sel saraf output acuan, sedangkan parameter kedua digunakan sebagai penunjuk posisi pada sel saraf input untuk nada.

Dari *base-class* tersebut diturunkan sel-sel saraf input yang lain :

1. Sel saraf input untuk tanda birama.

```
class in_neuron_birama : public in_neuron
{
    birama_type birama;              // Tanda birama yang diinputkan

    public :
        in_neuron_birama();          // Constructor
        void set(void* thing, int = 0);          // Input jenis tanda birama
        void process(out_neuron& out, int = 0);  // Pemrosesan
};
```

Penjelasan *member-functions* class tersebut :

- Constructor menginisialisasi ***birama = _4_BEAT*** dan nilai ***kind*** dijadikan ***IN_NEURON_BIRAMA***.

- Fungsi *void set(void* thing, int = 0)* digunakan untuk memasukkan input tanda birama ke variable *birama*.
- Fungsi *void process(out_neuron& out, int = 0)* digunakan untuk memroses sel saraf input tersebut dengan *out* sebagai acuannya. Berikut definisinya :

```
void process(out_neuron& out, int)
{
    output = out[BIRAMA];
    // BIRAMA merupakan konstan posisi bobot tanda birama
    if (birama == out.birama())
        output = output * 1;
    else
        if (birama==_2_BEAT && out.birama()==_4_BEAT)
            output = output * 0.75;
        else
            output = output * 0;
}
```

2. Sel saraf input untuk tangga nada.

```
class in_neuron_scale : public in_neuron
{
    scale_type scale;                   // Tangga nada yang diinputkan

    public :
        in_neuron_scale();              // Constructor
        void set(void* thing, int = 0);         // Input jenis tangga nada
        void process(out_neuron& out, int = 0);     // Pemrosesan
};
```

Penjelasan *member-functions* class tersebut :

- Constructor menginisialisasi *scale = M_SCALE* dan nilai *kind* dijadikan *IN_NEURON_SCALE*.

- Fungsi *void set(void* thing, int = 0)* digunakan untuk memasukkan input tangga nada ke variable *scale*.
- Fungsi *void process(out_neuron& out, int = 0)* digunakan untuk memroses sel saraf input tersebut dengan *out* sebagai acuannya. Berikut definisinya :

```
void process(out_neuron& out, int)
{
    if (scale == out.scale())
        output = out[SCALE] * 1;
    else
        output = out[SCALE] * 0.5;
    // SCALE merupakan konstan posisi bobot dari tangga nada
}
```

3. Sel saraf input untuk chord.

```
class in_neuron_chord : public in_neuron
{
    chord_type chord;                    // Chord yang diinputkan

    public :
        in_neuron_chord();               // Constructor
        void set(void* thing, int = 0);          // Input jenis chord
        void process(out_neuron& out, int = 0);      // Pemrosesan
};
```

Penjelasan *member-functions* class tersebut :

- Constructor menginisialisasi *chord = chord_type(NONE, MAJOR)* dan nilai *kind* dijadikan *IN_NEURON_CHORD*.
- Fungsi *void set(void* thing, int = 0)* digunakan untuk memasukkan input chord ke variable *chord*.

- Fungsi *void process(out_neuron& out, int = 0)* digunakan untuk memroses sel saraf input tersebut dengan *out* sebagai acuannya. Berikut definisinya :

```
void process(out_neuron& out, int)
{
    output = out[PREV_CHORD];
    // PREV_CHORD merupakan konstan posisi bobot dari chord
    // sebelum birama tersebut
    if (chord == out.prev_chord())
        output = output * 1;
    else
        if (chord <= out.prev_chord())
            output = output * 0.75;
        else
            if (out.prev_chord() <= chord)
                output = output * 0.5;
            else
                output = output * 0;
}
```

4. Sel saraf input untuk nada.

```
class in_neuron_note : public in_neuron
{
    static int many_neuron;          // Jumlah sel saraf untuk input nada
    static note_type *notes;         // Nada-nada input

    public :
        in_neuron_note();            // Constructor
        ~in_neuron_note();           // Destructor
        static int make(int in_many);        // Alokasi memori
        void set(void* thing, int pos);      // Input jenis nada
        void process(out_neuron& out, int pos);      // Pemrosesan
};
```

Penjelasan *member-functions* class tersebut :

- Constructor mengubah nilai ***kind*** menjadi *IN_NEURON_NOTE*.
- Destructor akan menghapus alokasi memori untuk input nada-nada (***notes***).
- Fungsi ***int make(int in_many)*** mengalokasikan memori untuk ***notes***. Besar alokasi memori ini ditentukan oleh nilai ***many_neuron***, dimana nilai ***many_neuron*** diperoleh dari ***in_many***. Fungsi ini akan mengembalikan nilai '1' bila alokasi memori berhasil. Perhatikan, fungsi ini hanya dijalankan satu kali saja (***static***), berapapun jumlah sel saraf input untuk nada.
- Fungsi ***void set(void* thing, int pos)*** digunakan untuk memasukkan input nada ke array ***notes*** pada posisi ***pos***.
- Fungsi ***void process(out_neuron& out, int pos)*** digunakan untuk memroses sel saraf input tersebut dengan ***out*** sebagai acuannya dan ***pos*** sebagai posisi input nadanya. Berikut definisinya :

```
void process(out_neuron& out, int pos)
{
    if (notes[pos] == NONE)
    // pos menunjukkan posisi nada dalam array notes
    {
        output = 0;
        return;
    }

    output = out[CHORDS + pos];
    // CHORDS merupakan konstan posisi bobot dari pola chord
```

```
if (pos == many_neuron - 1)
{
    if ((1 << notes[pos]) & out.next_chord().notes())
    // Nada notes[pos] ada dalam out.next_chord().notes()
        output = output * 1;
    else
        if ((1 << notes[pos]) & out.next_chord().scale())
        // Nada notes[pos] dalam out.next_chord().scale()
            output = output * 0.375;
        else
            output = output * 0;
    return;
}

birama_type birama = out.birama();
chord_type ref = ChordOnPos(birama, pos);
                                // Chord pada posisi pos
if ((1 << notes[pos]) & ref.notes())
    output = output * 1;
else
{
    int idx = 1;

    while ( pos + idx < many_neuron &&
        InHalfBirama(birama, pos, pos + idx))
        // pos dan pos+idx dalam ½ birama yang sama
    {
        if (notes[pos + idx] == NONE)
            if (pos + idx < MaxNotes(birama))
            // Jumlah nada maksimal untuk jenis birama itu
                break;
            else
                if (notes[many_neuron - 1] == NONE)
                    break;
                else
                    idx = many_neuron - pos - 1;

        chord_type ref2 = ChordOnPos(birama, pos + idx);
                                // Chord pada posisi pos+idx
```

```
            if ((1 << notes[pos + idx]) & ref2.notes())
            {
                if (Delta(notes[pos], notes[pos + idx]) <= 2)
                // Selisih antara nada notes[pos] dengan notes[pos + idx]
                    switch (out.birama())
                    {
                        case _2_BEAT :
                            output = output * 0.75 * (3 - idx) / 2.0;
                            break;
                        case _3_BEAT :
                        case _4_BEAT :
                            output = output * 0.75 * (5 - idx) / 4.0;
                            break;
                        case _6_BEAT :
                            output = output * 0.75 * (4 - idx) / 3.0;
                    }
                else
                    if ((1 << notes[pos]) & ref.scale())
                        output = output * 0.5;
                    else
                        output = output * 0.25;

                idx = 0;
                break;
            }
            else
                if (notes[pos + idx] != notes[pos])
                    break;

            idx = idx + 1;
        }

        if (idx != 0)
            if ((1 << notes[pos]) & ref.scale())
                output = output * 0.375;
            else
                output = output * 0;
    }
}
```

Sel saraf input untuk nada mempunyai karakteristik yang khusus, yaitu semua parameter input nadanya dijadikan satu *array* yang statis. Berbeda dengan sel saraf input yang lain, dimana satu sel saraf akan menyimpan satu informasi saja. Hal ini disebabkan karena proses pada sel saraf input untuk nada akan bergantung pada nada-nada yang lain (lihat fungsi ***process*** dari class ***in_neuron_note***). Untuk memudahkan prosesnya, maka input-input nada tersebut digabung menjadi satu array statis.

5.1.1.3. Jaringan Saraf

Struktur data jaringan saraf ini akan menggabungkan sel-sel saraf input dan sel-sel saraf output. Jumlah sel saraf input adalah 12, sesuai dengan parameter-parameter input yang dibutuhkan, masing-masing 1 sel saraf input untuk tanda birama, 1 sel saraf input untuk tangga nada, 1 sel saraf input untuk chord (mengecek chord sebelumnya), dan 9 sel saraf input untuk nada (mengecek nada-nada satu birama yang berjumlah delapan dan satu nada setelah birama). Berikut struktur data yang digunakan sebagai vektor input dan jaringan saraf.

```
struct in_vector
{
    birama_type birama;
    scale_type scale;
    chord_type prev_chord;
    chord_type next_chord;
    note_type notes[NOTES_VECTOR + 1];
```

```
    in_vector();                        // Constructor
};
```

Constructor dari struct tersebut akan menginisialisasi data-data dengan harga default, yaitu *birama = _4_BEAT, scale = M_SCALE, prev_chord = next_chord = chord_type(NONE, MAJOR), notes[0..NOTES_VECTOR] = NONE.*

```
class network
{
    in_neuron_birama birama;
    in_neuron_scale scale;
    in_neuron_chord prev_chord;
    in_neuron_note notes[NOTES_VECTOR + 1];
    int many_out;                       // Jumlah neuron output
    out_neuron** out;                   // Neuron-neuron output

    int difference(int winner, int other);      // Cari beda antara 2 node
    void change_weight(int winner, int diff);   // Rubah bobot suatu node

    public :
        network();                      // Constructor
        ~network();                     // Destructor
        int create(int vect_out);       // Alokasi memori
        void destroy();                 // Menghapus alokasi memori
        void input(in_vector& input_vector); // Input vektor
        out_neuron& output(int idx); // Mengakses sel saraf output
        int many_out_nodes();           // Nilai many_out
        void process();                 // Pemrosesan
        float get_winner(int& winner, float limit = 0);         // Ambil pemenang
        void disable_output(int idx);  // Nonaktifkan sel saraf output
        int generate(in_vector& input_vector, chord_vector& chords);
                                        // Memasukkan pola baru
};
```

NOTES_VECTOR merupakan suatu tetapan yang nilainya 8 (jumlah input nada untuk satu birama). Input dari sel-sel saraf input nada ini berasal dari nada-nada satu birama. Jumlah nada yang diambil dan cara memasukkannya dalam sel-sel saraf tersebut dapat dilihat pada tabel berikut :

TANDA BIRAMA	JUMLAH NADA	INPUT SEL SARAF							
_2_BEAT	4	1	1	2	2	3	3	4	4
_3_BEAT	6	1	2	3	4	5	6	-	-
_4_BEAT	8	1	2	3	4	5	6	7	8
_6_BEAT	6	1	2	3	4	5	6	-	-

Tabel 5.4. Input Nada pada Sel-Sel Saraf untuk Input Nada

Perhatikan, nada yang diambil dan chord yang tersimpan dalam pola jumlahnya berbeda (bandingkan tabel 5.1 dengan tabel 5.4). Cara pencocokan antara nada-nada dengan pola chord dapat dilihat pada tabel berikut :

TANDA BIRAMA	CHORD PADA SEL SARAF							
_2_BEAT	1	1	1	1	2	2	2	2
_3_BEAT	1	1	2	2	3	3	-	-
_4_BEAT	1	1	2	2	3	3	4	4
_6_BEAT	1	1	1	2	2	2	-	-

Tabel 5.5. Pencocokan Pola Chord pada Sel-Sel Saraf untuk Input Nada

Sel-sel saraf output disimpan dalam bentuk array, karena sel-sel saraf ini jumlahnya tidak tentu, berbeda dengan sel saraf input yang sudah tertentu.

- Fungsi ***out_neuron& output(int idx)*** digunakan untuk mengakses sel saraf

Penjelasan *member-functions* class ***network*** di atas :

- Constructor menginisialisasi ***many_out = 0*** dan ***out = NULL***.
- Destructor akan menghapus alokasi memori dari ***out*** dengan memanggil fungsi ***destroy()***.
- Fungsi ***int create(int vect_out)*** mengalokasikan memori untuk sel saraf output (***out***) sebanyak ***many_out***, dimana sebelumnya ***many_out*** diisi dengan nilai ***vect_out***. Fungsi ini akan mengembalikan nilai '1' bila alokasi memori berhasil.
- Fungsi ***void destroy()*** menghapus alokasi memori dari ***out***.
- Fungsi ***void input(in_vector& input_vector)*** digunakan untuk memasukkan input. Fungsi ini akan memasukkan data-data input ke sel-sel saraf input yang bersesuaian.
- Fungsi ***out_neuron& output(int idx)*** digunakan untuk mengakses sel saraf output pada index array ***idx***.
- Fungsi ***int many_out_nodes()*** digunakan untuk mengetahui jumlah sel saraf output. Fungsi akan mengembalikan nilai dari ***many_out***.
- Fungsi ***void process()*** digunakan untuk melakukan proses dengan kondisi input saat itu. Fungsi ini akan melakukan *looping* proses untuk semua sel saraf output. Berikut definisi dari fungsi ini :

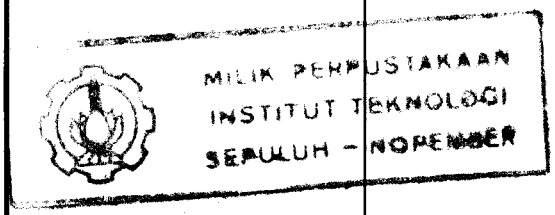
```
void process()
{
    for (int i = 0; i < many_out; ++i)
```

```
    {
        out[i] -> firing() = 0;
        if (out[i] -> is_empty())
            out[i] -> act = 0;
        else
        {
            birama.process(*out[i]);
            scale.process(*out[i]);
            prev_chord.process(*out[i]);
            for (int j = 0; j <= NOTES_VECTOR; ++j)
                notes[j].process(*out[i]);

            out[i] -> act() = birama.get_output();
            out[i] -> act() = out[i] -> act() + scale.get_output();
            out[i] -> act() = out[i] -> act() + prev_chord.get_output();
            for (j = 0; j <= NOTES_VECTOR; ++j)
                out[i] -> act() = out[i] -> act() + notes[j].get_output();
        }
    }
}
```

- Fungsi ***float get_winner(int& winner, float limit = 0)*** digunakan untuk mengambil sel saraf output pemenang. Parameter ***winner*** digunakan untuk mengambil index dari sel saraf output pemenang, dan parameter ***limit*** digunakan untuk memberi nilai batasan nilai aktivasi untuk sel saraf output pemenang tersebut (aktivasi dari sel saraf output tersebut harus lebih besar atau sama dengan ***limit***). Fungsi ini akan mengembalikan nilai aktivasi dari sel saraf output pemenang. Nilai ini nantinya dapat digunakan lagi sebagai nilai ***limit*** untuk mencari sel saraf output pemenang berikutnya.

- Fungsi ***void disable_output(int idx)*** digunakan untuk menonaktifkan sel saraf output pada posisi *idx*. Fungsi ini dijalankan bila kita ingin mengambil sel saraf output pemenang berikutnya. Jadi sel saraf output pemenang sebelumnya harus dinonaktifkan dulu, barulah sel saraf output pemenang berikutnya dapat diperoleh.
- Fungsi ***int difference(int winner, int other)*** digunakan untuk mencari komponen yang berbeda antara dua pola dalam sel saraf output, masing-masing pada posisi ***winner*** dan ***other***. Pencarian komponen yang berbeda dilakukan dengan membandingkan masing-masing komponen yang bersesuaian. Bila ada perbedaan, maka posisi bitnya di-set '1'. Nilai-nilai posisi bit inilah yang dikembalikan oleh fungsi. Perbedaan ini nantinya digunakan untuk meningkatkan bobot dari sel saraf output ***winner***.
- Fungsi ***void change_weight(int winner, int diff)*** melakukan peningkatan bobot pada sel saraf output ***winner***. Komponen-komponen yang ditingkatkan bobotnya, ditentukan oleh nilai posisi bit ***diff*** (diperoleh dari fungsi ***difference*** di atas). Peningkatan bobot dilakukan sebesar selisih antara nilai aktivasi tertinggi dengan nilai aktivasi ***winner***. Bila ***winner*** mempunyai nilai aktivasi tertinggi, maka tidak akan terjadi peningkatan bobot. Nilai peningkatan bobot tersebut akan dibagi-bagikan kepada masing-masing komponen yang ditingkatkan, dengan porsi perbandingan sebagai berikut :



KOMPONEN INPUT	PERBANDINGAN PENINGKATAN BOBOT
Tanda Birama	2
Tangga Nada	1
Chord Sebelumnya	2
Nada-Nada 1 birama	1
Nada Sesudahnya	1

Tabel 5.6. Perbandingan Peningkatan Bobot Komponen Input

Peningkatan bobot pada komponen nada-nada satu birama akan mencakup nada-nada satu chord (lihat tabel 5.5). Misalnya, pada birama ***_4_BEAT***, bila sel saraf input nada posisi ketiga akan ditingkatkan bobotnya, maka posisi keempat pun harus ikut ditingkatkan, karena satu chord dengan nada posisi ketiga (sama-sama dicocokkan dengan chord kedua).

- Fungsi ***int generate(in_vector& input_vector, chord_vector& chords)*** digunakan untuk memasukkan pola baru. Gabungan beberapa elemen dari ***input_vector*** dengan ***chords*** akan menyusun pola yang diinginkan (lihat kembali struct ***pattern*** sebelumnya). Elemen ***input_vector*** yang bukan penyusun pola adalah ***notes***. Berikut definisi fungsi ini :

```
int generate(in_vector& input_vector, chord_vector& chords)
{
    input(input_vector);
    process();

    int winner;
    float limit = get_winner(winner);
```

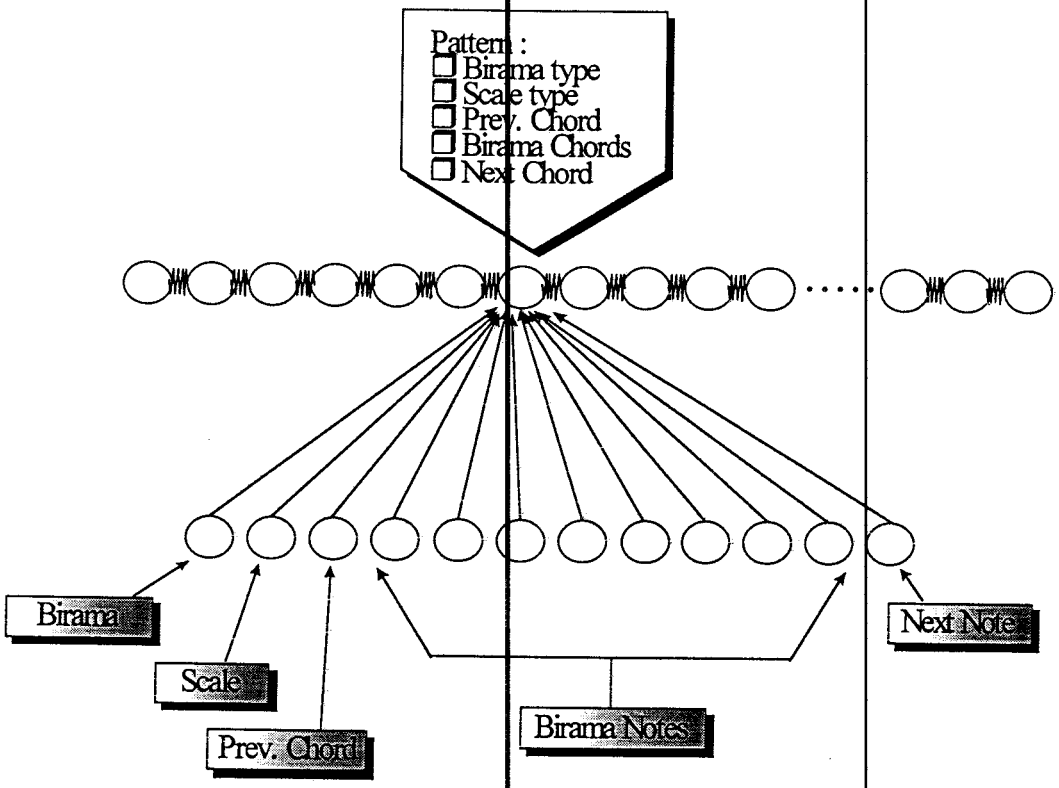
```
while (limit > 0 && out[winner] -> compare(inPattern) == 0)
// inPattern merupakan gabungan elemen input_vector dengan chords
{
    SavePos(winner);                    // Simpan posisi winner
    disable_output(winner);
    limit = get_winner(winner);
}

if (limit > 0)
{
    int win_ = winner;
    while (win_ >= 0)
    {
        SavePos(win_);                  // Simpan posisi win_
        disable_output(win_);
        get_winner(win_, limit);
    }

    int diff = 0;
    for (int i = 0; i < many_out; ++i)
        if (IsSavedPos(i) && i != winner)          // Periksa apakah i tercatat
            diff = diff | difference(winner, i);   // Operasi boolean OR
    if (diff > 0)
        change_weight(winner, diff);

    return winner;
}
else
{
    InputNewPattern(winner, inPattern);
    // inPattern merupakan gabungan elemen input_vector dengan chords
    // Posisi node output pola baru akan ditunjukkan oleh winner
    return winner;
}
}
```

Berikut arsitektur dari jaringan saraf ART pada perangkat lunak ini :



Gambar 5.1. Jaringan Saraf ART pada Perangkat Lunak

5.1.2. Algoritma

Algoritma dari perangkat lunak ini mirip dengan algoritma dari ART, hanya ada beberapa bagian yang dirubah.

1. Inisialisasi bobot dengan nol (0).

   Pada perangkat lunak ini, hanya ada satu jenis bobot, yaitu bobot yang digunakan untuk menghitung pemrosesan vektor input (*feedforward*); sedangkan untuk arah proses dari lapisan output ke lapisan input, bobotnya

diganti dengan pola-pola sel saraf output. Inisialisasi ini akan dilakukan oleh masing-masing sel saraf output (***class out_neuron***).

2. Masukkan vektor input.

   Komponen-komponen dari vektor input dimasukkan ke dalam masing-masing sel saraf input (fungsi ***input*** pada ***class network***).

3. Proses vektor input.

   Pemrosesan ini dilakukan di dalam sel-sel saraf input dengan cara *looping* untuk semua sel saraf output. Jadi sel saraf output pertama diambil sebagai parameter, lalu sel-sel saraf input melakukan proses. Hasil total prosesnya diserahkan kembali ke sel saraf output tersebut untuk disimpan sebagai nilai aktivasi. Kemudian proses dilanjutkan dengan sel saraf output kedua, dan seterusnya. Proses ini dilakukan oleh fungsi ***process*** pada ***class network***. Seperti telah dijelaskan sebelumnya, proses yang dilakukan dalam sel-sel saraf input ini adalah proses pencocokan (*matching*), lihat pada fungsi ***process*** dari masing-masing jenis sel saraf input.

4. Tentukan sel saraf output pemenang.

   Bersamaan dengan dilakukannya proses input pada langkah 3, semua output dari sel saraf output akan diset '0'. Kemudian, untuk menentukan sel saraf output pemenang, kita cari sel saraf dengan nilai aktivasi tertinggi. Output dari sel saraf tersebut kita set '1' (fungsi ***get_winner*** pada ***class network***).

Bila tidak ditemukan lagi sel saraf output yang berisikan pola, berarti vektor input tersebut dijadikan pola yang baru.

5. Membandingkan pola dari sel saraf output pemenang dengan vektor input. Nilai ambang yang digunakan adalah 1, jadi pola tersebut harus benar-benar sama dengan vektor input.

   Bila sama, menuju langkah 7. Bila tidak sama, menuju langkah 6.

6. Nonaktifkan sel saraf output pemenang (output diset '-1' pada fungsi ***disable_output*** dari ***class network***), lalu menuju langkah 4.

   Pemrosesan input hanya dilakukan satu kali saja, jadi tidak perlu melakukan langkah 3 kembali.

7. Pengubahan bobot.

   Pengubahan bobot dilakukan apabila sel saraf output yang cocok tersebut bukan merupakan pemenang pertama. Pengubahan bobot dilakukan dengan mencari komponen yang berbeda antara sel saraf itu dengan sel-sel saraf lain yang menjadi pemenang sebelumnya. Pengubahan bobot hanya terjadi pada komponen-komponen yang berbeda tersebut (fungsi ***change_weight*** pada ***class network***).

5.2. Perangkat Lunak Interaksi dengan User

Perangkat lunak ini merupakan program interaksi yang ditujukan untuk user yang ingin menentukan chord-chord suatu lagu. Input dari perangkat

lunak ini adalah file MIDI dengan format 1 dan berisi satu track data lagu saja (*lead vocal*). Perangkat lunak ini akan memroses lagu tersebut dengan menambahkan chord dan bass-line. Output perangkat lunak ini juga berupa file MIDI yang sudah ditambahkan chord dan bass-line. Perangkat lunak untuk interaksi dengan user ini dibuat dengan menggunakan bahasa pemrograman BorlandC++ 4.5 for Windows, dimana operating system yang dipakai adalah Windows 95.

Untuk memroses lagu tersebut, perangkat lunak ini menggunakan jaringan saraf ART. Bobot-bobot untuk jaringan saraf ini dapat diperoleh dari file hasil perangkat lunak untuk pelatihan di atas. Karena menggunakan jaringan saraf yang sama, maka perangkat lunak ini juga membutuhkan struktur data yang sama dengan perangkat lunak pelatihan. Bedanya, jaringan saraf perangkat lunak ini ditambahkan satu data berikut :

enum {FULL_BIRAMA, HALF_BIRAMA, EVERY_BEAT} changeMode;

Data tersebut digunakan sebagai parameter untuk menentukan pola pemenang yang akan dipilih. Untuk lebih jelasnya, dapat dilihat pada bagian Algoritma dari perangkat lunak ini.

5.2.1. Struktur Data

Disamping struktur data untuk jaringan saraf, perangkat lunak ini juga membutuhkan struktur data untuk menganalisa file MIDI. Struktur data untuk

file MIDI ini digunakan untuk menyimpan event-event yang ada dalam file tersebut. Ada 3 event yang dapat disimpan, yaitu event dari kunci nada dasar, event dari tanda birama, dan event dari nada-nada lagu. Event-event lainnya dari file MIDI tidak disimpan karena tidak diikutsertakan dalam proses.

```
struct TMIDIEvent
{
    enum
        {NO_EVENT, KEY_EVENT, TIMESIGN_EVENT, NOTE_EVENT}
        event;                          // Jenis event
    TPosition position;                 // Posisi event tersebut
    TMIDIEvent *next;                   // Event berikutnya

    TMIDIEvent();                       // Constructor
    ~TMIDIEvent();                      // Destructor
};
```

TPosition merupakan struct yang menyimpan posisi event tersebut. Posisi event tersebut terdiri dari nomor birama dan posisi dalam birama (satuan not 1/16). Satuan not 1/16 dijadikan satuan terkecil dalam perangkat lunak ini. Bila ada event yang posisinya tidak bersatuan not 1/16, maka event tersebut akan diberikan posisi satuan not 1/16 yang terdekat. Berikut penjelasan *member-functions* dari struct di atas :

- Constructor dari struct tersebut akan menginisialisasi ***event*** = ***NO_EVENT*** dan ***next*** = ***NULL***. Constructor dari ***TPosition*** akan menginisialisasi ***position*** pada birama dan satuan 1/16 pertama.
- Destructor dari struct tersebut akan menghapus alokasi memori dari ***next*** bila bukan ***NULL***.

Struct tersebut merupakan *base-class*. Dari struct tersebut diturunkan struct untuk event-event kunci nada dasar, tanda birama, dan nada-nada.

1. Event kunci nada dasar.

```
struct TMIDIKeyEvent : public TMIDIEvent
{
    struct
    {
        int sharp_flat      : 4;        // Jumlah mol/kres
        unsigned reserved   : 3;        // Tidak digunakan
        unsigned maj_min    : 1;        // Lagu mayor atau minor
    } key;

    TMIDIKeyEvent();                    // Constructor
};
```

Constructor struct tersebut akan menginisialisasi ***key.sharp_flat = 0*** dan ***key.maj_min = 0*** (Mayor), serta mengubah *event* menjadi ***KEY_EVENT***.

2. Event tanda birama.

```
struct TMIDITimeSignEvent : public TMIDIEvent
{
    TTimeSign timeSign;                 // Tanda birama

    TMIDITimeSignEvent();               // Constructor
};
```

TTimeSign merupakan struct bit-field yang menyimpan pembilang dan penyebut dari tanda birama. Constructor struct tersebut akan menginisialisasi ***timeSign*** dengan tanda birama 4/4, serta mengubah *event* menjadi ***TIMESIGN_EVENT***.

3. Event nada-nada.

```
struct TMIDINoteEvent : public TMIDIEvent
{
    BYTE note;                          // Kode nada
    unsigned value;                     // Jumlah ticks

    TMIDINoteEvent();                   // Constructor
};
```

Constructor struct tersebut menginisialisasi ***note = 0*** dan ***value = 0***, serta mengubah *event* menjadi *NOTE_EVENT*.

Event-event tersebut masih terpisah-pisah, karena itu perlu disatukan menjadi satu list yang urut dalam class berikut.

```
class TMIDIData
{
    unsigned ticksPerQuarter;           // Jumlah ticks per not 1/4
    TMIDIEvent *listEvent;              // List dari event-event
    TMIDIEvent *pointer;                // Pointer untuk membaca list event-event

    public :
        TMIDIData();                    // Constructor
        ~TMIDIData();                   // Destructor
        void Destroy();                 // Menghapus alokasi memori
        int IsEmpty();                  // Cek bila tidak ada event
        unsigned& Ticks();              // Nilai ticksPerQuarter
        void ToFirstEvent();            // Menuju event paling awal
        TMIDIEvent* GetEvent(int moved = MOVED);        // Ambil event
        void operator << (TMIDIEvent *nextEvent);       // Tambah event
};
```

Penjelasan *member-functions* dari class tersebut :

- Constructor class tersebut menginisialisasi ***ticksPerQuarter = 0*** dan *listEvent = pointer = NULL*.

- Destructor class tersebut akan menghapus alokasi memori dengan memanggil fungsi ***Destroy()***.
- Fungsi ***void Destroy()*** digunakan untuk menghapus alokasi memori dari ***listEvent***.
- Fungsi ***int IsEmpty()*** akan mengembalikan nilai '1' bila ***listEvent*** bernilai ***NULL***, sebaliknya nilai '0' yang akan dikembalikan fungsi.
- Fungsi ***unsigned& Ticks()*** digunakan mengakses ***ticksPerQuarter***.
- Fungsi ***void ToFirstEvent()*** digunakan untuk menuju ke event paling awal. Caranya adalah dengan mengisikan nilai ***listEvent*** ke ***pointer***.
- Fungsi ***TMIDIEvent* GetEvent(int moved = MOVED)*** digunakan untuk mengambil event yang ditunjuk oleh ***pointer***. Parameter ***moved*** digunakan untuk menentukan apakah ***pointer*** akan digeser ke event berikutnya atau tidak.
- Fungsi ***void operator << (TMIDIEvent *nextEvent)*** digunakan untuk menambah event ke dalam daftar (*list*) event. Event yang tersimpan dalam daftar ini akan diurutkan berdasarkan nilai ***position*** (lihat kembali ***struct TMIDIEvent***).

Perangkat lunak ini akan menyimpan data file MIDI dalam class ***TMIDIData***. Data-data dalam class inilah yang digunakan sebagai input dari jaringan saraf.

5.2.2. Algoritma

Adapun algoritma dari perangkat lunak untuk interaksi user ini adalah sebagai berikut :

1. Ambil bobot-bobot untuk jaringan saraf ART.

   Jaringan saraf tersebut tersimpan dalam variable global ***netART***.

2. Input file MIDI.

   Data dalam file MIDI ini langsung disimpan berupa event-event dalam class ***TMIDIData***.

3. Ambil data satu birama.

   Dilakukan dengan mengambil event-event nada yang mempunyai posisi birama yang sama. Event-event kunci nada dasar digunakan sebagai acuan untuk transformasi, sedangkan event-event tanda birama digunakan sebagai acuan jumlah not dalam satu birama.

4. Proses data satu birama.

   Sebelum data satu birama tersebut dikirimkan sebagai vektor input jaringan saraf, terlebih dulu data-data tersebut ditransformasikan ke nada dasar C. Pada proses ini dapat terjadi lebih dari satu pola pemenang pada jaringan saraf, untuk itu pola pemenang dipilih dengan analisa sebagai berikut :

```
int AnalizeNetART(in_vector& data_in)
{
    int winner = -1;                    // Pola pemenang
```

```
int win = 0;
float result = 0.0;
do
{
    result = netART.get_winner(win, result);  // Ambil pola pemenang
    if (win >= 0)
    {
        netART.disable_output(win);           // Pemenang dinonaktifkan
        if (winner < 0)
            winner = win;
        else
        {
            out_neuron& out_winner = netART.output(winner);
            out_neuron& out_win = netART.output(win);
            // Ambil sel-sel saraf output yang hendak dibandingkan
            if (out_winner.chords() == out_win.chords())
            {                                 // Bila pola chordnya sama
                if (out_winner.next_chord().note !=
                    out_win.next_chord().note &&
                    out_win.next_chord().note ==
                    data_in.notes[NOTES_VECTOR])
                    winner = win;
                    // Pola win lebih cocok antara nada input dengan chord
            }
            else
            {
                int num1 = NumChordsChange(out_winner.chords());
                int num2 = NumChordsChange(out_win.chords());
                // Jumlah perubahan chord dalam satu birama
                if (num1 != num2)
                    switch (netART.changeMode)
                    // Mode ini dipilih user melalui menu
                    {
                        case FULL_BIRAMA :
                            if (num2 < num1)
                                winner = win;
                            break;
                        case HALF_BIRAMA :
                            if (num2 == 2)
                                winner = win;
```

```
                                else
                                    if (num2 < num1)
                                        winner = win;
                                break;
                            case EVERY_BEAT :
                                if (num2 > num1)
                                    winner = win;
                        }
                    else
                    {
                        num1 = NumChordsMatch(out_winner.chords(),
                            data_in);
                        num2 = NumChordsMatch(out_win.chords(),
                            data_in);
                        // Jumlah kecocokan antara pola-pola chord
                        // dengan nada-nada input
                        if (num2 > num1)
                            winner = win;
                    }
                }
            }
        }
        while (win >= 0);

        return winner;
    }
```

5. Simpan data hasil proses.

   Data hasil proses jaringan saraf disimpan dalam class *TMIDIData* yang lain, yang khusus menyimpan data-data chord. Sebelumnya, data-data tersebut ditransformasikan dulu ke nada dasar yang seharusnya (karena sebelumnya ditransformasikan ke nada dasar C).

6. Analisa untuk bass-line.

Data hasil analisa untuk bass-line ini juga disimpan dalam class ***TMIDIData*** yang terpisah, yang khusus menyimpan data-data bass.

7. Ulangi proses untuk birama selanjutnya. Menuju langkah 3.

BAB VI
HASIL UJI COBA DAN EVALUASI PERANGKAT LUNAK

Setelah perancangan dan pembuatan perangkat lunak, dilakukan uji coba terhadap perangkat lunak tersebut, untuk mengetahui apakah hasilnya sesuai dengan tujuan dari tugas akhir ini. Berikut ini hasil uji coba dan evaluasi dari kedua perangkat lunak yang dibuat pada tugas akhir ini.

6.1. Perangkat Lunak Pelatihan Jaringan Saraf

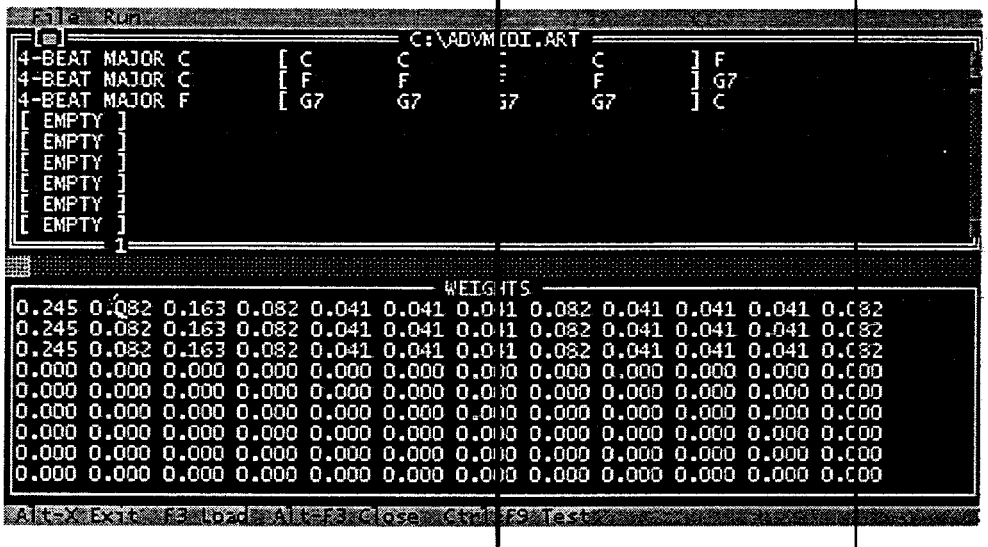
6.1.1. Uji Coba

Dalam pembuatan perangkat lunak ini, telah dilakukan beberapa kali uji coba sebelum didapatkan parameter input dan sistem pembobotan (model dari jaringan saraf) seperti yang dijelaskan pada bagian Perancangan dan Pembuatan Perangkat Lunak. Untuk menguji perangkat lunak ini, diisikan beberapa pola-pola chord ke dalam perangkat lunak ini. Setelah itu jaringan saraf tersebut diuji coba, apakah outputnya sudah sesuai dengan yang diharapkan. Bila outputnya belum sesuai, maka parameter input ataupun sistem pembobotannya diperbaiki. Hasil akhir model jaringan saraf ini cukup optimal, karena dapat digunakan untuk penentuan chord suatu lagu sesuai yang diharapkan.

Setelah model jaringan sarafnya diperoleh, dilakukan pengujian terhadap pola-pola yang dimasukkan dalam jaringan saraf. Dalam perangkat lunak ini, disamping terdapat menu untuk mengisikan pola input baru, tersedia pula menu untuk menguji jaringan saraf tersebut. Model dari menu pengujian ini hampir sama dengan pengisian pola input baru, hanya saja pengujian ini tidak akan mengubah bobot dari jaringan saraf.

Mula-mula, pola-pola input baru dimasukkan, kemudian dilakukan pengujian dengan memberikan input nada-nada satu birama. Bila hasil pengujian ini belum sesuai dengan yang diharapkan, maka kita dapat meningkatkan bobot suatu pola dengan memasukkan pola inputnya kembali. Jaringan saraf akan secara otomatis mengubah bobot dari pola output yang sesuai dengan pola input tersebut.

Tampilan dari perangkat lunak ini dapat dilihat pada gambar berikut.



Gambar 6.1. Tampilan Perangkat Lunak Pelatihan Jaringan Saraf

6.1.2. Evaluasi

Dari pengujian yang dilakukan, didapatkan bahwa hasil-hasil yang diperoleh akan sesuai dengan yang kita harapkan. Jadi jaringan saraf tersebut memang dapat menentukan chord suatu lagu.

Kadangkala ada lebih dari satu pola yang mempunyai total aktivasi tertinggi (pemenangnya lebih dari satu). Hal ini bukanlah merupakan kesalahan, karena memang semua pola tersebut dapat dipakai untuk kondisi tersebut, tergantung pada keinginan dari masing-masing musisi untuk memilihnya. Bila kita menginginkan agar salah satu pola lebih unggul daripada yang lain, maka kita inputkan kembali pola inputnya, dan jaringan saraf tersebut akan meningkatkan bobot pola tersebut, sehingga pola tersebut menjadi lebih unggul.

Dapat pula terjadi, output yang diharapkan tidak sesuai. Kita dapat pula mengubahnya dengan meningkatkan bobot pola output yang diinginkan, seperti cara di atas. Setelah peningkatan bobot, dapat terjadi outputnya masih belum sesuai dengan yang diharapkan. Mengapa demikian ?

Pada bab sebelumnya disebutkan bahwa peningkatan bobot terjadi pada komponen-komponen pola yang berbeda dengan pola pemenang. Misalkan total nilai perbandingan (lihat tabel 5.6) komponen yang berbeda adalah n, maka masing-masing komponen tersebut ditingkatkan sebesar $(1/n)$ * nilai

perbandingan * δ, dimana δ adalah selisih antara nilai aktivasi tertinggi dengan nilai aktivasi pola tersebut. Bila jaringan saraf tidak dapat menemukan komponen yang berbeda sesuai dengan yang dikehendaki, maka jaringan saraf akan meningkatkan bobot seluruh komponennya. Akibatnya, peningkatan bobotnya menjadi kecil (perhatikan, bila *n* besar, maka nilai peningkatan bobot masing-masing komponen menjadi kecil). Peningkatan yang kecil inilah yang menyebabkan pola tersebut tetap tidak bisa menjadi pemenang. Masalah ini dapat diatasi dengan meningkatkan kembali bobot pola tersebut, hingga pola tersebut menjadi pemenang.

6.2. Perangkat Lunak Interaksi dengan User

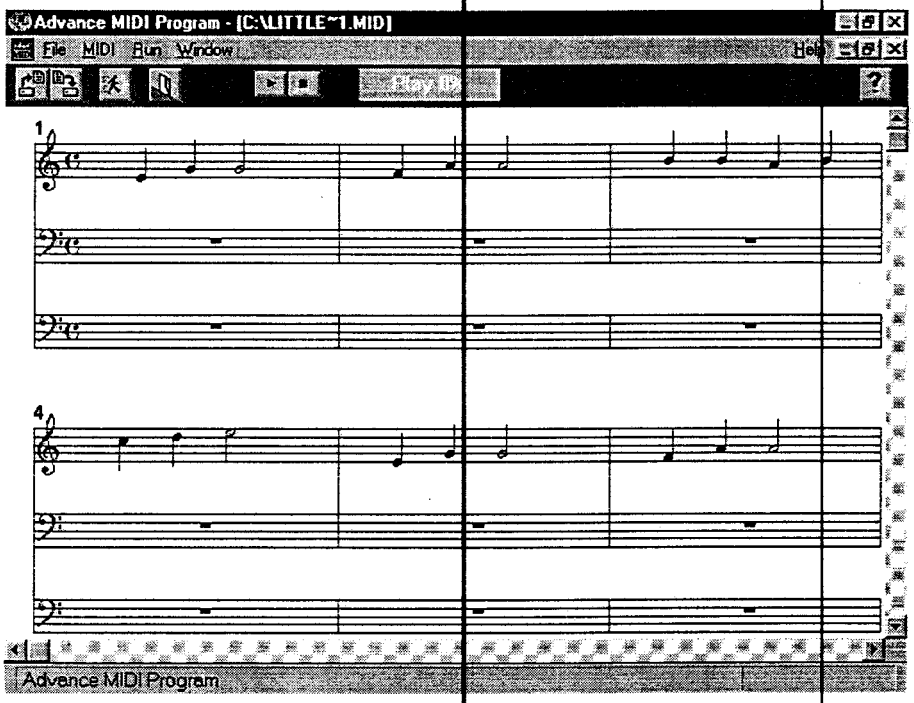
Pengujian dan evaluasi perangkat lunak ini pada dasarnya sama dengan perangkat lunak sebelumnya, karena jaringan saraf yang digunakan sama. Berikut ini akan diberikan contoh lagu yang hendak ditentukan chordnya.

6.2.1. Uji Coba

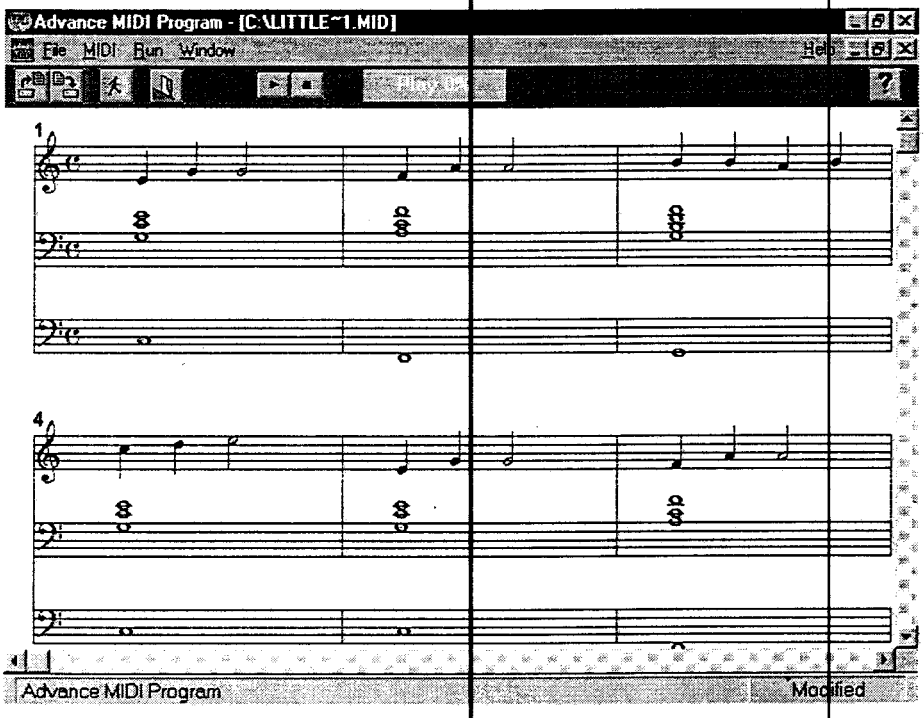
Salah satu urutan (*sequence*) pola chord yang cukup sering adalah I-IV-V7-I. Urutan pola itu dimasukkan jaringan saraf dan hasilnya sebagai berikut :

1. *4-BEAT MAJOR C* *[C C C C] F*
2. *4-BEAT MAJOR C* *[F F F F] G7*
3. *4-BEAT MAJOR F* *[G7 G7 G7 G7] C*

Kemudian, dimasukkan lagu *"Little Brown Jug"* pada perangkat lunak ini.



Gambar 6.2. Tampilan Perangkat Lunak Interaksi dengan User Sebelum Diproses



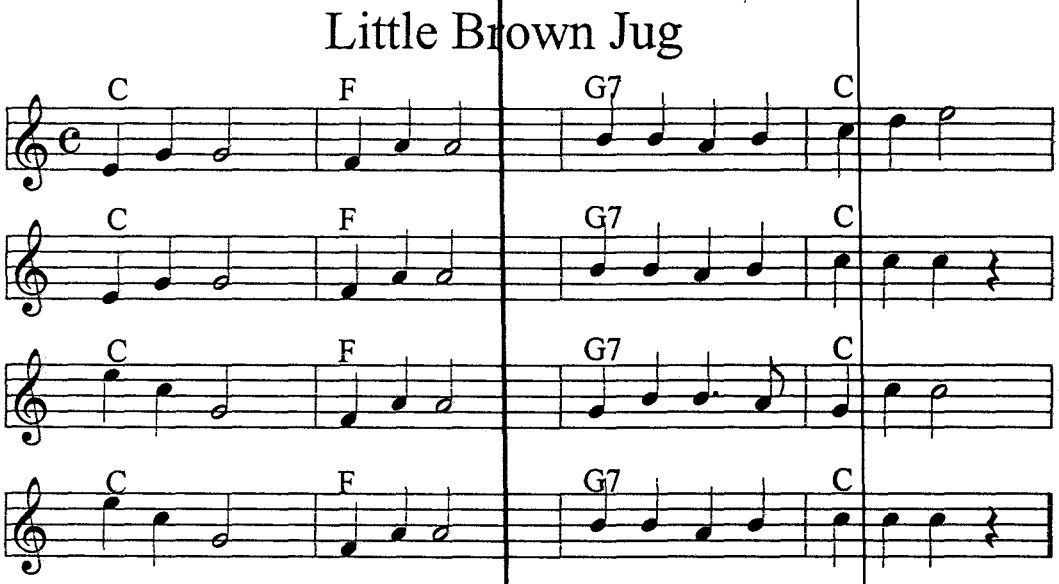
Gambar 6.3. Tampilan Perangkat Lunak Interaksi dengan User Sesudah Diproses

6.2.2. Evaluasi

Setelah ketiga pola di atas dimasukkan ke dalam jaringan saraf, maka jaringan saraf melakukan inisialisasi bobot-bobot sesuai nilai perbandingan bobot pada tabel 5.2 dan 5.3. Karena jenis biramanya ***_4_BEAT***, maka perbandingan bobot jenis birama = 6, jenis tangga nada = 2, jenis chord sebelumnya = 4, jenis nada-nada satu birama = 2 + 1 + 1 + 1 + 2 + 1 + 1 + 1 = 10, dan jenis nada sesudahnya = 2. Total nilai perbandingan adalah 6 + 2 + 4 + 10 + 2 = 24.

- Bobot inisialisasi untuk jenis birama = 6.0 / (24 + 0.5) = 0.245
- Bobot inisialisasi untuk jenis tangga nada = 2.0 / (24 + 0.5) = 0.082
- Bobot inisialisasi untuk jenis chord sebelumnya = 4.0 / (24 + 0.5) = 0.163
- Bobot inisialisasi untuk jenis nada-nada satu birama :
  - Nada-1 = Nada-5 = 2.0 / (24 + 0.5) = 0.082
  - Nada-2 = Nada-3 = Nada-4 = Nada-6 = Nada-7 = Nada-8 = 1.0 / (24 + 0.5) = 0.041
- Bobot inisialisasi untuk jenis nada sesudahnya = 2.0 / (24 + 0.5) = 0.082

Nilai 0.5 ditambahkan pada penyebut dengan tujuan agar total bobot inisialisasinya tidak mencapai nilai 1.0 (hanya mendekati nilai 1.0). Hal ini dimaksudkan agar total bobotnya sedapat mungkin tidak melebihi nilai 1.0, meskipun bobot-bobotnya ditingkatkan nantinya. Hasil inisialisasi bobot-bobotnya dapat dilihat pada gambar 6.1 sebelumnya.



Gambar 6.4. Contoh Lagu Setelah Diproses

Dari proses penentuan chord, diperoleh hasil seperti gambar 6.4 di atas. Terlihat bahwa pola chord tersebut sesuai dengan yang diinginkan, yaitu menggunakan urutan pola I-IV-V7-I.

Nada-nada penyusun masing-masing chord :

- Chord C terdiri dari nada-nada C-E-G.
- Chord F terdiri dari nada-nada F-A-C.
- Chord G7 terdiri dari nada-nada G-B-D-F.

Berikut evaluasi perhitungan dalam penentuan chord-chord dari 4 (empat) birama pertama :

❑ Birama-1

☞ Pola : *4-BEAT MAJOR C [C C C C] F*

Karena jenis birama dan jenis tangga nadanya sama (cocok), maka operasi pencocokannya memberikan nilai 1.0 x bobot. Operasi pencocokan jenis chord sebelumnya memberikan nilai 0.0, karena birama-1 tersebut tidak mempunyai chord sebelumnya. Nada-nada satu birama adalah E-E-G-G-G-G-G-G (jumlah sel saraf inputnya 8, jadi pencacahannya untuk setiap nilai not 1/8). Karena nada E dan G terdapat dalam chord C, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Nada sesudahnya adalah nada F. Nada ini terdapat dalam chord F, sehingga operasi pencocokannya juga memberikan nilai 1.0 x bobot.

Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (0.0 x 0.163) +
(1.0 x 0.082) + (1.0 x 0.041) + (1.0 x 0.041) + (1.0 x 0.041) +
(1.0 x 0.082) + (1.0 x 0.041) + (1.0 x 0.041) + (1.0 x 0.041) +
(1.0 x 0.082) = 0.819

☞ Pola : *4-BEAT MAJOR C [F F F F] G7*

Karena jenis birama dan jenis tangga nadanya cocok, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Operasi pencocokan chord sebelumnya memberikan nilai 0.0. Nada E tidak terdapat dalam chord F, sehingga operasi pencocokan nada-1 dan nada-2 memberikan nilai 0.375 x bobot. Demikian juga nada G tidak terdapat dalam chord F, sehingga operasi pencocokan dari nada-3 dan nada-4 memberikan nilai 0.375 x bobot. Operasi pencocokan nada-5 (nada G) memberikan

nilai 0.75 x 0.25 x bobot, karena nada berikutnya, nada F, terdapat dalam chord G7. Operasi pencocokan nada-6 memberikan nilai 0.75 x 0.5 x bobot. Operasi pencocokan nada-7 memberikan nilai 0.75 x 0.75 x bobot. Operasi pencocokan nada-8 memberikan nilai 0.75 x 1.0 x bobot. Nada sesudahnya adalah nada F, karena nada tersebut terdapat dalam chord G7, maka operasi pencocokannya memberikan nilai 1.0 x bobot.

Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (0.0 x 0.163) +
(0.375 x 0.082) + (0.375 x 0.041) + (0.375 x 0.041) +
(0.375 x 0.041) + (0.75 x 0.25 x 0.082) + (0.75 x 0.5 x 0.041) +
(0.75 x 0.75 x 0.041) + (0.75 x 1.0 x 0.041) +
(1.0 x 0.082) = 0.570

☞ Pola : *4-BEAT MAJOR F* *[G7 G7 G7 G7] C*

Karena jenis birama dan jenis tangga nadanya cocok, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Operasi pencocokan chord sebelumnya memberikan nilai 0.0. Nada E tidak terdapat dalam chord G7, sehingga operasi pencocokan nada-1 dan nada-2 memberikan nilai 0.5 x bobot (nilai 0.5 karena nada berikutnya, nada G, terdapat dalam chord G7). Kemudian nada-3 sampai nada-8, nada G, karena terdapat dalam chord G7, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Nada sesudahnya adalah nada F, karena nada tersebut

tidak terdapat dalam chord C, maka operasi pencocokannya memberikan nilai 0.375 x bobot.

Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (0.0 x 0.163) +
(0.5 x 0.082) + (0.5 x 0.041) + (1.0 x 0.041) + (1.0 x 0.041) +
(1.0 x 0.082) + (1.0 x 0.041) + (1.0 x 0.041) + (1.0 x 0.041) +
(0.375 x 0.082) = 0.706

Dari ketiga pola tersebut, pola pertama memberikan nilai paling tinggi, sehingga chord yang dihasilkan adalah chord **C**.

- Birama-2
  - Pola : *4-BEAT MAJOR C [C C C C] F*

    Karena jenis birama dan jenis tangga nadanya cocok, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Operasi pencocokan jenis chord sebelumnya memberikan nilai 1.0 x bobot, karena chord sebelumnya (chord C) cocok. Nada-nada satu birama adalah F-F-A-A-A-A-A-A. Karena nada F dan A tidak terdapat dalam chord C, maka operasi pencocokannya memberikan nilai 0.375 x bobot. Nada sesudahnya adalah nada B. Nada ini tidak terdapat dalam chord F, sehingga operasi pencocokannya memberikan nilai 0.0 (nada B juga tidak terdapat dalam tangga nada F mayor).

    Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (1.0 x 0.163) +
    (0.375 x 0.082) + (0.375 x 0.041) + (0.375 x 0.041) +
    (0.375 x 0.041) + (0.375 x 0.082) + (0.375 x 0.041) +

(0.375 x 0.041) + (0.375 x 0.041) + (0.0 x 0.082) = 0.644

☞ Pola : *4-BEAT MAJOR C [F F F F] G7*

Karena jenis birama dan jenis tangga nadanya cocok, maka operasi pencocokan memberikan nilai 1.0 x bobot. Operasi pencocokan chord sebelumnya memberikan nilai 1.0 x bobot, karena chord sebelumnya (chord C) cocok. Nada F dan A terdapat dalam chord F, sehingga operasi pencocokan nada-nadanya memberikan nilai 1.0 x bobot. Nada sesudahnya adalah nada B, karena nada tersebut terdapat dalam chord G7, maka operasi pencocokannya memberikan nilai 1.0 x bobot.

Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (1.0 x 0.163) + (1.0 x 0.082) + (1.0 x 0.041) + (1.0 x 0.041) + (1.0 x 0.041) + (1.0 x 0.082) + (1.0 x 0.041) + (1.0 x 0.041) + (1.0 x 0.041) + (1.0 x 0.082) = 0.982

☞ Pola : *4-BEAT MAJOR F [G7 G7 G7 G7] C*

Karena jenis birama dan jenis tangga nadanya cocok, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Operasi pencocokan chord sebelumnya memberikan nilai 0.0, karena chord sebelumnya (chord C) tidak cocok. Nada F terdapat dalam chord G7, sehingga operasi pencocokan nada-1 dan nada-2 memberikan nilai 1.0 x bobot. Kemudian nada-3 sampai nada-8, nada A, karena tidak terdapat dalam chord G7, maka operasi pencocokannya memberikan nilai 0.375 x bobot. Nada sesudahnya adalah nada B, karena nada tersebut tidak

terdapat dalam chord C, maka operasi pencocokannya memberikan nilai 0.375 x bobot.

Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (0.0 x 0.163) +
(1.0 x 0.082) + (1.0 x 0.041) + (0.375 x 0.041) +
(0.375 x 0.041) + (0.375 x 0.082) + (0.375 x 0.041) +
(0.375 x 0.041) + (0.375 x 0.041) + (0.375 x 0.082) = 0.588

Dari ketiga pola tersebut, pola kedua memberikan nilai paling tinggi, sehingga chord yang dihasilkan adalah chord **F**.

- Birama-3
  - Pola : *4-BEAT MAJOR C [C C C C] F*

    Karena jenis birama dan jenis tangga nadanya cocok, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Operasi pencocokan jenis chord sebelumnya memberikan nilai 0.0, karena tidak cocok dengan chord sebelumnya (chord F). Nada-nada satu birama adalah B-B-B-B-A-A-B-B. Karena nada B dan A tidak terdapat dalam chord C, maka operasi pencocokan nada-1 sampai nada-6 memberikan nilai 0.375 x bobot. Operasi pencocokan nada-7 memberikan nilai 0.75 x 0.75 x bobot, karena nada berikutnya, nada C, terdapat dalam chord F. Operasi pencocokan nada-8 memberikan nilai 0.75 x 1.0 x bobot. Nada sesudahnya adalah nada C. Nada ini terdapat dalam chord F, sehingga operasi pencocokannya memberikan nilai 1.0 x bobot.

Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (0.0 x 0.163) +
(0.375 x 0.082) + (0.375 x 0.041) + (0.375 x 0.041) +
(0.375 x 0.041) + (0.375 x 0.082) + (0.375 x 0.041) +
(0.75 x 0.75 x 0.041) + (0.75 x 1.0 x 0.041) +
(1.0 x 0.082) = 0.586

☞ Pola : *4-BEAT MAJOR C [F F F F] G7*

Karena jenis birama dan jenis tangga nadanya cocok, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Operasi pencocokan chord sebelumnya memberikan nilai 0.0, karena tidak cocok dengan chord sebelumnya (chord F). Nada B tidak terdapat dalam chord F, sehingga operasi pencocokan nada-1 memberikan nilai 0.75 x 0.25 x bobot (nada berikutnya, nada A, terdapat dalam chord F). Operasi pencocokan nada-2 memberikan nilai 0.75 x 0.5 x bobot. Operasi pencocokan nada-3 memberikan nilai 0.75 x 0.75 x bobot. Operasi pencocokan nada-4 memberikan nilai 0.75 x 1.0 x bobot. Nada-5 dan nada-6, yaitu nada A, terdapat dalam chord F, sehingga operasi pencocokannya memberikan nilai 1.0 x bobot. Nada-7 dan nada-8, yaitu nada B, tidak terdapat dalam chord F, sehingga operasi pencocokannya memberikan nilai 0.0 (nada B juga tidak terdapat dalam tangga nada F mayor). Nada sesudahnya adalah nada C, karena nada tersebut tidak terdapat dalam chord G7, maka operasi pencocokannya memberikan nilai 0.375 x bobot.

Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (0.0 x 0.163) +
(0.75 x 0.25 x 0.082) + (0.75 x 0.5 x 0.041) +
(0.75 x 0.75 x 0.041) + (0.75 x 1.0 x 0.041) +
(1.0 x 0.082) + (1.0 x 0.041) + (0.0 x 0.041) + (0.0 x 0.041) +
(0.375 x 0.082) = 0.565

☞ Pola : *4-BEAT MAJOR F [G7 G7 G7 G7] C*

Karena jenis birama dan jenis tangga nadanya cocok, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Operasi pencocokan chord sebelumnya memberikan nilai 1.0 x bobot, karena chord sebelumnya (chord F) cocok. Nada B terdapat dalam chord G7, sehingga operasi pencocokan nada-1 sampai nada-4 memberikan nilai 1.0 x bobot. Nada-5, nada A, tidak terdapat dalam chord G7, sehingga operasi pencocokannya memberikan nilai 0.75 x 0.75 x bobot (karena nada berikutnya, nada B, terdapat dalam chord G7). Demikian pula dengan nada-6, operasi pencocokannya memberikan nilai 0.75 x 1.0 x bobot. Sedangkan nada-7 dan nada-8, nada B, terdapat dalam chord G7, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Nada sesudahnya adalah nada C, karena nada tersebut terdapat dalam chord C, maka operasi pencocokannya memberikan nilai 1.0 x bobot.

Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (1.0 x 0.163) +
(1.0 x 0.082) + (1.0 x 0.041) + (1.0 x 0.041) + (1.0 x 0.041) +
(0.75 x 0.75 x 0.082) + (0.75 x 1.0 x 0.041) +
(1.0 x 0.041) + (1.0 x 0.041) + (1.0 x 0.082) = 0.936

Dari ketiga pola tersebut, pola ketiga memberikan nilai paling tinggi, sehingga chord yang dihasilkan adalah chord **G7**.

- Birama-4
  - Pola : *4-BEAT MAJOR C [C C C C] F*

    Karena jenis birama dan jenis tangga nadanya cocok, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Operasi pencocokan jenis chord sebelumnya memberikan nilai 0.0, karena tidak cocok dengan chord sebelumnya (chord G7). Nada-nada satu birama adalah C-C-D-D-E-E-E-E. Karena nada C terdapat dalam chord C, maka operasi pencocokan nada-1 dan nada-2 memberikan nilai 1.0 x bobot. Nada-3, nada D, tidak terdapat dalam chord C, sehingga operasi pencocokannya memberikan nilai 0.75 x 0.75 x bobot (nada sesudahnya, nada E, terdapat dalam chord C). Demikian pula dengan nada-4, operasi pencocokannya memberikan nilai 0.75 x 1.0 x bobot. Operasi pencocokan nada-5 sampai nada-8 memberikan nilai 1.0 x bobot, karena nada E terdapat dalam chord C. Nada sesudahnya adalah nada E. Nada ini tidak terdapat dalam chord F, sehingga operasi pencocokannya memberikan nilai 0.375 x bobot.

    Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (0.0 x 0.163) +
    (1.0 x 0.082) + (1.0 x 0.041) + (0.75 x 0.75 x 0.041) +
    (0.75 x 1.0 x 0.041) + (1.0 x 0.082) + (1.0 x 0.041) +

(1.0 x 0.041) + (1.0 x 0.041) + (0.375 x 0.082) = 0.740

☞ Pola : *4-BEAT MAJOR C [F F F F] G7*

Karena jenis birama dan jenis tangga nadanya cocok, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Operasi pencocokan chord sebelumnya memberikan nilai 0.0, karena tidak cocok dengan chord sebelumnya (chord G7). Nada C terdapat dalam chord F, sehingga operasi pencocokan dari nada-1 dan nada-2 memberikan nilai 1.0 x bobot. Nada-3 dan nada-4, yaitu nada D, tidak terdapat dalam chord F, sehingga operasi pencocokannya memberikan nilai 0.375 x bobot. Operasi pencocokan nada-5 sampai nada-8 (nada E) memberikan nilai 0.375 x bobot, karena nada E tidak terdapat dalam chord F. Nada sesudahnya adalah nada E, karena nada tersebut tidak terdapat dalam chord G7, maka operasi pencocokannya memberikan nilai 0.375 x bobot.

Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (0.0 x 0.163) +
(1.0 x 0.082) + (1.0 x 0.041) + (0.375 x 0.041) +
(0.375 x 0.041) + (0.375 x 0.082) + (0.375 x 0.041) +
(0.375 x 0.041) + (0.375 x 0.041) + (0.375 x 0.082) = 0.588

☞ Pola : *4-BEAT MAJOR F [G7 G7 G7 G7] C*

Karena jenis birama dan jenis tangga nadanya cocok, maka operasi pencocokannya memberikan nilai 1.0 x bobot. Operasi pencocokan chord sebelumnya memberikan nilai 0.0, karena tidak cocok dengan

chord sebelumnya (chord G7). Nada C tidak terdapat dalam chord G7, sehingga operasi pencocokan nada-1 memberikan nilai 0.75 x 0.75 x bobot (nada berikutnya, nada D, terdapat dalam chord G7). Sedangkan operasi pencocokan nada-2 memberikan nilai 0.75 x 1.0 x bobot. Kemudian nada-3 dan nada-4, yaitu nada D, terdapat dalam chord G7, sehingga operasi pencocokannya memberikan nilai 1.0 x bobot. Selanjutnya nada-5, nada E, karena tidak terdapat dalam chord G7, maka operasi pencocokannya memberikan nilai 0.75 x 0.25 x bobot (nada berikutnya, yaitu nada E, terdapat dalam chord C). Operasi pencocokan nada-6 memberikan nilai 0.75 x 0.5 x bobot. Operasi pencocokan nada-7 memberikan nilai 0.75 x 0.75 x bobot. Operasi pencocokan nada-8 memberikan nilai 0.75 x 1.0 x bobot. Nada sesudahnya adalah nada E, karena nada tersebut terdapat dalam chord C, maka operasi pencocokannya memberikan nilai 1.0 x bobot.

Jadi total bobotnya = (1.0 x 0.245) + (1.0 x 0.082) + (0.0 x 0.163) +
(0.75 x 0.75 x 0.082) + (0.75 x 1.0 x 0.041) + (1.0 x 0.041) +
(1.0 x 0.041) + (0.75 x 0.25 x 0.082) + (0.75 x 0.5 x 0.041) +
(0.75 x 0.75 x 0.041) + (0.75 x 1.0 x 0.041) +
(1.0 x 0.082) = 0.652

Dari ketiga pola tersebut, pola pertama memberikan nilai paling tinggi, sehingga chord yang dihasilkan adalah chord **C**.

Dari empat contoh birama awal tersebut, terlihat bahwa penentuan chord banyak ditentukan oleh nada-nada satu birama. Bila dalam satu birama banyak terdapat nada-nada C-E-G, maka chord yang diperoleh adalah chord C. Bila dalam satu birama banyak terdapat nada-nada F-A-C, maka chord yang diperoleh adalah chord F. Bila dalam satu birama banyak terdapat nada-nada G-B-D-F, maka chord yang diperoleh adalah chord G7. Jadi, bila nada-nada penyusun chord tersebut banyak terdapat dalam suatu birama, maka chord tersebut mempunyai kemungkinan besar untuk dipilih sebagai chord pada birama tersebut.

Untuk birama-birama selanjutnya dapat diperoleh hasil yang sama dengan perhitungan di atas, sehingga diperoleh hasil seperti pada gambar 6.4 di atas. Data-data perhitungan selengkapnya, dapat dilihat pada Lampiran C pada akhir tugas akhir ini. Hasil perhitungan pada evaluasi ini agak berbeda dengan hasil perhitungan pada Lampiran C tersebut, karena pada evaluasi ini bobot-bobotnya sudah dibulatkan terlebih dulu menjadi 3 digit di belakang koma.

BAB VII
PENUTUP

Pada bagian penutup ini, akan disimpulkan hasil-hasil yang diperoleh dari tugas akhir ini. Selain itu, Penulis juga memberikan beberapa saran kepada Pembaca yang berminat untuk mengembangkan program aplikasi hasil tugas akhir ini.

7.1. Kesimpulan

Penentuan chord suatu lagu memang agak sulit untuk dilakukan, karena permainan musik memang membutuhkan ketajaman perasaan (*feeling*) dari masing-masing pemain musik. Oleh karena itu, untuk membuat aplikasi yang dapat menentukan chord suatu lagu, diperlukan suatu metode yang dapat menyimpan *feeling* dari pemusik tersebut. Dalam tugas akhir ini, hal tersebut dilakukan dengan menggunakan metode pengenalan pola.

Dari percobaan tugas akhir ini, dapat ditarik kesimpulan-kesimpulan sebagai berikut :

- Metode pengenalan pola dapat digunakan untuk memecahkan masalah penentuan chord suatu lagu. Alasannya, chord-chord suatu lagu pada

dasarnya membentuk pola-pola urutan tertentu. Pola-pola inilah yang disimpan untuk digunakan menentukan chord lagu.

- Untuk penyimpanan pola ini dapat digunakan jaringan saraf, dalam hal ini dengan menggunakan ART (*Adaptive Resonance Theory*).
- Untuk penentuan chord lagu ini, pola-pola yang disimpan berupa pola-pola chord satu birama. Komponen-komponen lain yang disimpan dalam pola ini antara lain adalah tanda birama, jenis tangga nada, serta chord sebelum dan sesudah birama tersebut.
- Program aplikasi penentuan chord lagu ini tidak akan dapat memenuhi keinginan semua musisi. Karena setiap musisi akan memiliki ciri pemilihan chordnya sendiri-sendiri. Untuk itu, musisi tersebut dapat membuat pola chordnya sendiri dengan melakukan pelatihan terhadap jaringan sarafnya.

7.2. Saran

Program ini semata-mata hanyalah alat bantu awal saja, dan tidak dapat dijadikan acuan yang pasti, karena musik sendiri memang kaya dengan improvisasi. Setiap musisi hendaknya berusaha mempertajam perasaannya sendiri dengan banyak mendengarkan lagu dan banyak berlatih tentunya.

Program ini dapat dikembangkan lebih jauh apabila pembaca ada yang berminat. Misalnya saja, dengan menambah jenis-jenis tanda birama yang

dapat diproses, menambah macam-macam chordnya, menambah jenis musik yang dapat diproses, memberikan tambahan rhythm, membuat iringan-iringan (*accompaniment*) yang lebih bervariasi, dan lain-lainnya.

Editor yang digunakan dalam program ini merupakan editor yang "mati", dalam arti user tidak dapat melakukan perubahan (*edit*) terhadapnya. Oleh karena itu, model editor ini pun dapat dikembangkan pula. Misalnya dengan memberikan kemampuan untuk membuat lagu baru ataupun mengubah lagu yang sudah ada.

DAFTAR PUSTAKA

1. The Associated Board of The Royal Schools of Music ; **"RUDIMENTS AND THEORY OF MUSIC"** ; The Associated Board of The Royal Schools of Music ; England ; 1938.
2. Beale, Russell dan Jackson, Tom ; **"NEURAL COMPUTING : AN INTRODUCTION"** ; IOP Publishing Ltd. ; London ; 1990.
3. Fu, Limin;**"NEURAL NETWORK IN COMPUTER INTELLIGENCE"**; McGraw-Hill, Inc. ; International Edition ; Singapore ; 1994.
4. Rao, Valluru B. dan Rao, Hayagriva V. ; **"C++ NEURAL NETWORKS AND FUZZY LOGIC"** ; Management Information Source, Inc. ; First Edition ; New York ; 1993.
5. Ricigliano, Daniel A. ; **"POPULAR & JAZZ HARMONY"** ; Donato Music Publishing Co. ; Revised Edition ; New York ; 1967.
6. Rowland, Neil Jr. ; **"MIDI.HLP" (MIDI Help untuk Windows).**
7. Schalkoff, Robert J. ; **"PATTERN RECOGNITION : STATISTICAL, STRUCTURAL, AND NEURAL APPROACHES"** ; John Wiley & Sons, Inc. ; Canada ; 1992.
8. Walnum, Clayton ; **"OBJECT-ORIENTED PROGRAMMING WITH BORLAND C++ 4"** ; Que Corporation ; USA ; 1994.

9. Welstead, Stephen T. ; **"NEURAL NETWORK AND FUZZY LOGIC APPLICATIONS IN C/C++"** ; John Wiley & Sons, Inc. ; Canada ; 1994.

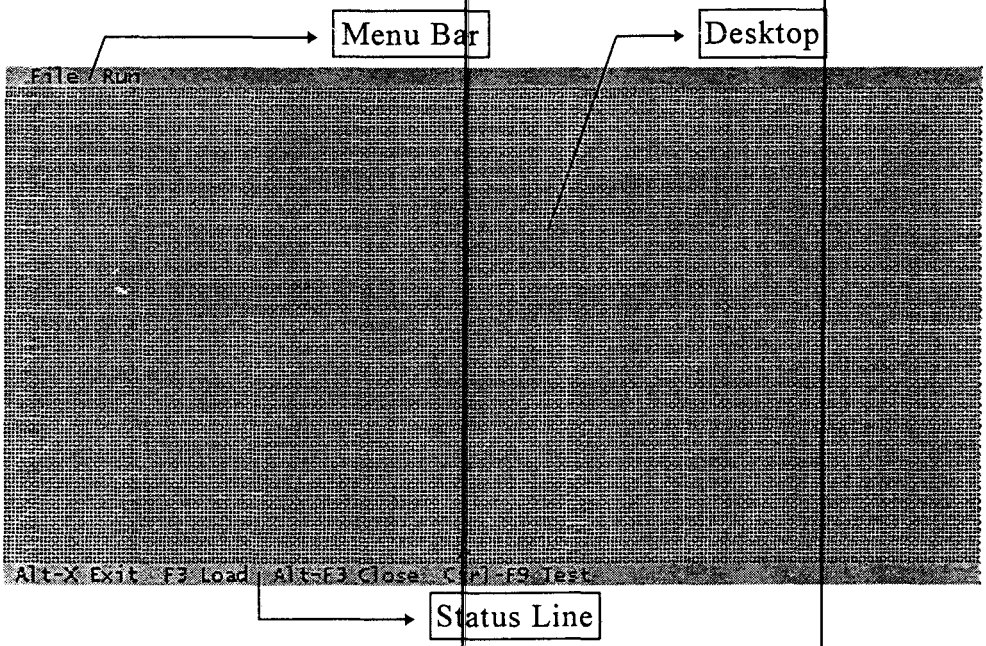
Lampiran A :
Petunjuk Pemakaian Perangkat Lunak Pelatihan Jaringan Saraf

A.1. Spesifikasi Hardware dan Software

Untuk menggunakan perangkat lunak ini, spesifikasi yang dibutuhkan antara lain :

- Komputer PC AT-386 atau yang lebih tinggi.
- RAM minimal 4 MB.
- Sistem Operasi DOS 6.0 atau yang lebih tinggi.

A.2. Tampilan Dasar



Perangkat lunak ini menggunakan fasilitas Turbo Vision dari BorlandC++ 3.1 for DOS. Dengan fasilitas Turbo Vision ini, dengan mudah dapat diisikan menu-menu pada Menu Bar maupun pada Status Line.

A.2.1. Menu pada Menu Bar

- Menu *File*, terdiri dari submenu-submenu :
  - *New*

    Untuk membuat sistem pembobotan jaringan saraf yang baru. Secara default, jumlah sel saraf output (untuk menyimpan pola) yang disediakan adalah sepuluh (10) sel saraf.
  - *Load* (*shortcut* F3)

    Untuk mengambil sistem pembobotan jaringan saraf yang sudah pernah dibuat, dan tersimpan dalam file. *Extension* standar dari file ini adalah ART.
  - *Save* (*shortcut* F2)

    Untuk menyimpan sistem pembobotan jaringan saraf yang ada di memori ke file.
  - *Save as*

    Untuk menyimpan sistem pembobotan jaringan saraf yang ada di memori ke file, dengan nama yang lain.

- *Close* (*shortcut* Alt-F3)

  Untuk menutup window sekaligus menghapus alokasi memori dari pembobotan jaringan saraf. Bila data pembobotan ini belum disimpan, program akan mengkonfirmasikan apakah data tersebut akan disimpan atau tidak.

- *Exit* (*shortcut* Alt-X)

  Untuk keluar dari program aplikasi.

❑ Menu *Run*, terdiri dari submenu-submenu :

- *Test* (*shortcut* Ctrl-F9)

  Untuk melakukan pengetesan terhadap pola-pola yang tersimpan dalam jaringan saraf, tanpa merubah pembobotannya.

- *Process*

  Untuk memasukkan pola-pola baru ke dalam jaringan saraf. Jaringan saraf secara otomatis akan menambahkan pola tersebut ke dalam datanya, bila pola tersebut belum ada. Bila sudah ada, jaringan saraf akan melakukan perubahan pembobotan seperlunya.

- *Increase neurons*

  Untuk menambah jumlah sel saraf output. Bila sel saraf output sudah terisi semua, maka pola baru tidak akan dapat disimpan. Oleh karena itu, sel saraf outputnya perlu ditambah melalui submenu ini.

A.2.2. Menu pada Status Line

- Alt-X *Exit*

  Sama seperti submenu *File* | *Exit*.

- F3 *Load*

  Sama seperti submenu *File* | *Load*.

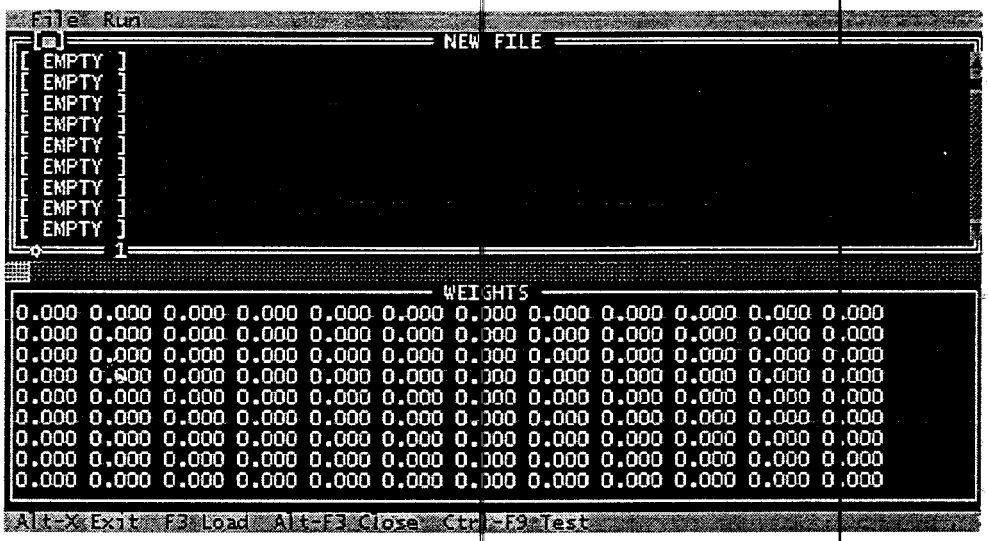
- Alt-F3 *Close*

  Sama seperti submenu *File* | *Close*.

- Ctrl-F9 *Test*

  Sama seperti submenu *Run* | *Test*.

A.3. Tampilan Window



Bila submenu *File* | *New* dipilih oleh *user*, maka akan muncul 2 buah window yang berlainan. Window sebelah atas menunjukkan pola-pola yang

tersimpan dalam jaringan saraf, sedangkan window sebelah bawah menunjukkan nilai-nilai bobot dari masing-masing komponen pola. Data tiap pola dituliskan secara horisontal. Jumlah bobot untuk masing-masing pola adalah 12 nilai, sesuai dengan jumlah parameter input dari jaringan saraf.

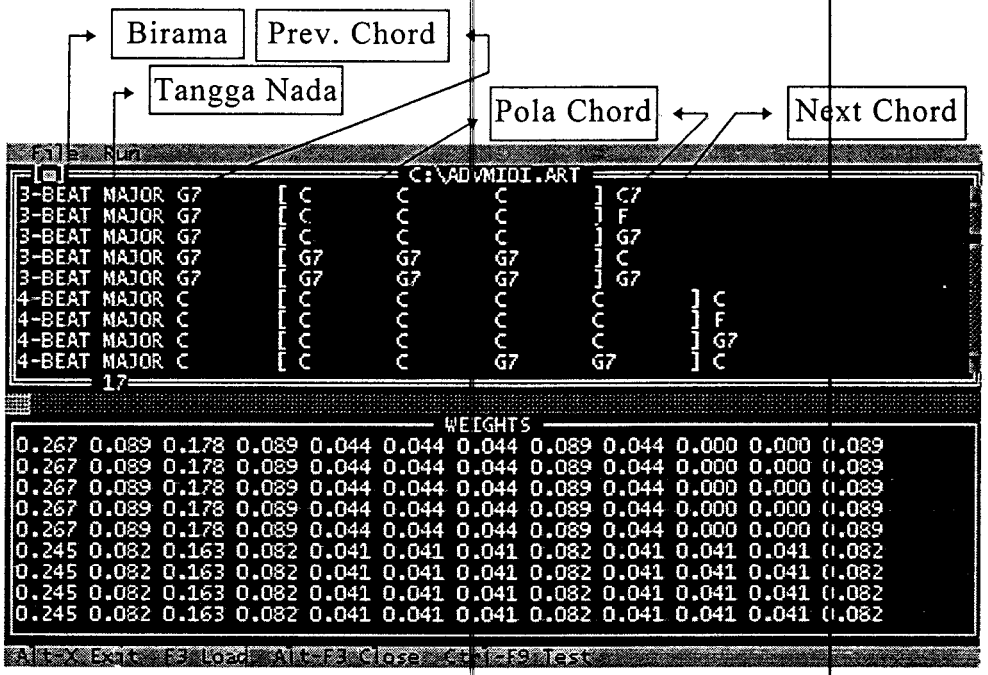
Urutan 12 parameter input tersebut dari kiri ke kanan adalah :

- Jenis Tanda Birama. [1 sel saraf input]
- Jenis Tangga Nada. [1 sel saraf input]
- Chord Sebelum Satu Birama tersebut (*previous chord*). [1 sel saraf input]
- Nada-Nada Input Satu Birama. [8 sel saraf input]
- Nada Sesudah Satu Birama tersebut (*next note*). [1 sel saraf input]

Sedangkan urutan komponen dari pola adalah :

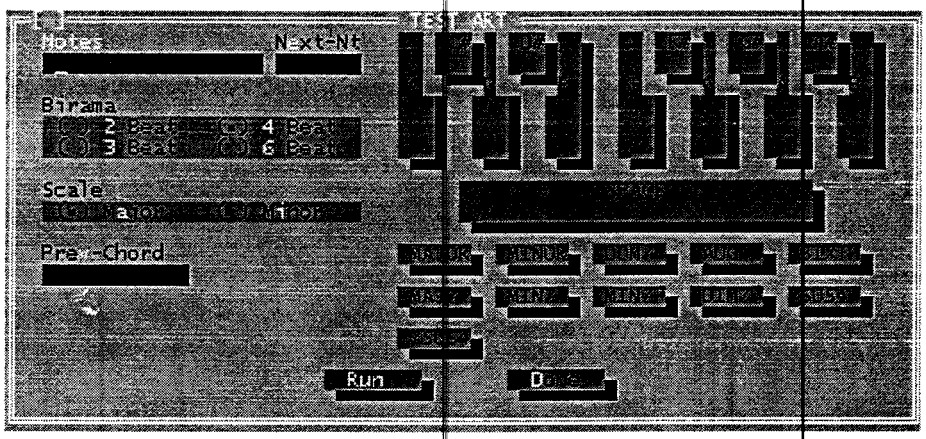
- Jenis Tanda Birama.
- Jenis Tangga Nada.
- Chord Sebelum Satu Birama tersebut (*previous chord*).
- Pola Chord Satu Birama.
- Chord Sesudah Satu Birama tersebut (*next chord*).

Pada gambar di atas, seluruh sel saraf outputnya masih kosong, karena sistem pembobotannya baru dibuat. Berikut ini contoh sistem pembobotan yang sudah terisi.



A.4. Test dan Input Pola

A.4.1. Test Pola



Pengetesan pola dilakukan dengan memilih submenu *Run* | *Test* atau penekanan *shortcut* Ctrl-F9. Akan muncul *dialog-box* untuk mengisikan jenis birama, jenis tangga nada, nada-nada satu birama, dan chord sebelum birama

tersebut. Disediakan pula *button-button* untuk memilih nada dan progresi chord di sisi sebelah kanan.

Nada-nada yang diisikan harus ditulis dengan huruf besar. Macam nada-nadanya adalah : "C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", dan "B". Antara nada yang satu dengan nada yang lain, dipisahkan dengan spasi. Cara lain mengisi nada adalah dengan memilih *button* nada yang diinginkan (harus menggunakan mouse), kemudian diikuti *button* SPACE untuk memisahkan antar nada. Jumlah maksimal nada yang diinputkan adalah 8 buah, sesuai parameter input nada dari jaringan saraf, dan selebihnya akan diacuhkan.

Untuk pengisian chord, ditulis nadanya terlebih dulu, kemudian diikuti progresi chordnya. Nada dan progresi chord ini harus dijadikan satu, tidak boleh ada spasi di antaranya. Antara chord yang satu dengan chord yang lain, dipisahkan dengan spasi. Macam-macam progresi chord :

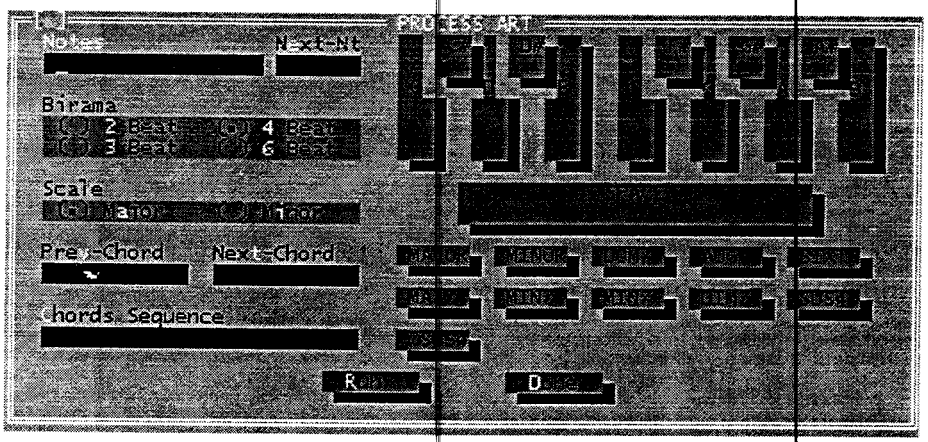
- BLANK (chord major)
- "m" (chord minor)
- "7" (chord dominant 7^{th})
- "+" (chord augmented)
- "6" (chord sixth)
- "M7" (chord major 7^{th})

- "m7" (chord minor 7^{th})
- "m7-5" (chord minor $7^{th\ -5}$)
- "dim7" (chord diminished 7^{th})
- "sus4" (chord suspended 4^{th})
- "7sus4" (chord dominant 7^{th} suspended 4^{th})

Cara lain juga dapat dilakukan dengan memilih *button* nada, kemudian *button* progresi chord, dan diikuti *button* SPACE untuk memisahkan antar chord.

Setelah data input satu birama tersebut selesai, tekan ENTER, atau tekan *button Run* untuk menjalankan pengetesan. Program aplikasi akan menampilkan semua pola-pola pemenang yang cocok dengan data input tersebut.

A.4.2. Input Pola



Penginputan pola dilakukan dengan memilih submenu *Run* | *Process*. Akan muncul *dialog-box* yang mirip dengan pengetesan pola di atas, dengan

beberapa tambahan input. Tambahan input pada *dialog-box* itu antara lain pola chord satu birama, nada sesudah birama tersebut, serta chord sesudah birama tersebut. Cara pengisian nada dan chord tersebut sama dengan cara pengisian pada pengetesan pola.

Setelah data input satu birama dimasukkan, tekan *button Run* untuk memroses pola baru tersebut. Penambahan pola baru akan terlihat pada kedua window tampilan tersebut.

Lampiran B :
Petunjuk Pemakaian Perangkat Lunak Interaksi dengan User

B.1. Spesifikasi Hardware dan Software

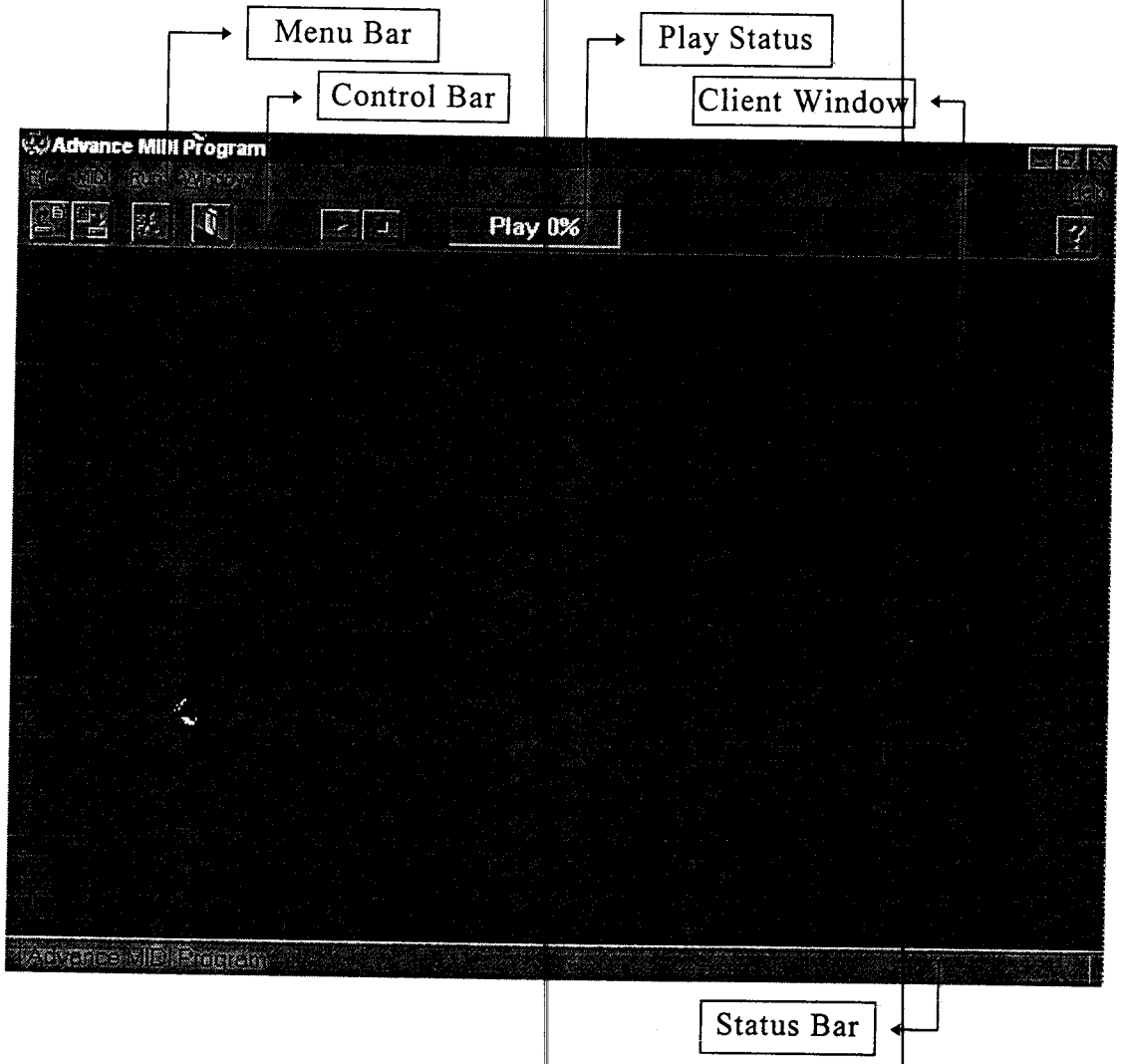
Untuk menggunakan perangkat lunak ini, spesifikasi yang dibutuhkan antara lain :

- ❑ Komputer PC AT-486 atau yang lebih tinggi.
- ❑ RAM minimal 8 MB.
- ❑ Sistem Operasi Windows 95.
- ❑ Sound Blaster dan Media Player dengan MIDI Sequencer.

B.2. Tampilan Dasar

Program aplikasi ini menggunakan sistem operasi Windows 95, sehingga tampilan windownya sesuai dengan spesifikasi Windows 95 tersebut. Program aplikasi ini menggunakan model MDI (*Multiple Document Interface*) untuk membuka file-file MIDI (*extension* MID), dimana setiap file MIDI akan berhubungan dengan satu window anak (*MDI child*).

Berikut ini tampilan awal dari program aplikasi untuk menentukan chord suatu lagu.



Terlihat bahwa kontrol dari program ini terletak pada menu (menu bar) dan *button icon-icon* (control bar).

B.2.1. Menu pada Menu Bar

- Menu *File*, terdiri dari submenu-submenu :
  - *Open* (*shortcut* F3)

Untuk membuka file MIDI.

- *Save* (*shortcut* F2)

  Untuk menyimpan file MIDI dari window anak yang aktif.

- *Save as*

  Untuk menyimpan file MIDI dari window anak yang aktif dengan nama yang lain.

- *Close*

  Untuk menutup window anak yang aktif. Program akan melakukan konfirmasi apabila file belum disimpan.

- *Exit* (*shortcut* Alt-X)

  Keluar dari program aplikasi.

❑ Menu *MIDI*, terdiri dari submenu-submenu :

- *Play* (*shortcut* Ctrl-P)

  Membunyikan file MIDI pada window anak yang aktif.

- *Pause* (*shortcut* Ctrl-A)

  Menghentikan sementara file MIDI yang dimainkan. Permainan file MIDI yang dihentikan, dapat diteruskan kembali dengan memilih submenu *Play* di atas.

- *Stop* (*shortcut* Ctrl-S)

  Menghentikan file MIDI yang dimainkan. Bila memainkan file MIDI kembali, maka akan dimainkan dari awal.

- *Mode*

  Memilih mode permainan : *Repeat* (dimainkan berulang-ulang) atau *No Repeat* (dimainkan sekali saja).

❑ Menu *Run*, terdiri dari submenu-submenu :

- *Parameter*

  Mengganti parameter untuk proses penentuan chord. Parameter ini antara lain : mode dan nilai ambang (*threshold*) untuk *browse*. Mode digunakan oleh program aplikasi untuk menentukan pola pemenang (bila ada lebih dari satu pola pemenang). Mode yang tersedia adalah ***Full Birama***, ***Half Birama***, atau ***Every Beat***. Sedangkan nilai ambang digunakan pada saat mengganti chord suatu birama.

- *Process* (*shortcut* Ctrl-F9)

  Menentukan chord suatu lagu pada file MIDI dari window anak yang aktif.

- *Change Chords*

  Mengubah chord suatu birama lagu pada file MIDI dari window anak yang aktif. Program akan menggunakan nilai ambang (*threshold*) yang dimasukkan *user* untuk menentukan pola-pola yang akan dipilih. Sel saraf output yang nilai aktivasinya lebih besar atau sama dengan nilai ambang, akan dimasukkan dalam *list box*, sehingga user dapat memilih pola chord yang dikehendakinya.

- Menu *Window*, terdiri dari submenu-submenu :
  - *Cascade*

    Mengatur window-window anak agar tersusun saling menumpuk (*overlaping*) dengan rapi.
  - *Tile Vertical*

    Mengatur window-window anak agar tersusun memenuhi *window client* dengan arah vertikal.
  - *Tile Horizontal*

    Mengatur window-window anak agar tersusun memenuhi *client window* dengan arah horisontal.
  - *Arrange Icons*

    Mengatur *icon* dari window-window anak pada *client window*.
  - *Close All*

    Menutup semua window anak.
- Menu *Help*, terdiri dari submenu-submenu :
  - *Contents* (*shortcut* F1)

    Memberikan keterangan bantuan mengenai cara menggunakan program aplikasi ini.
  - *About*

    Menampilkan label dari program aplikasi ini.

B.2.2. Icon pada Control Bar

Icon-icon pada *control bar* secara urut dari kiri ke kanan :

- *Open*

  Sama dengan submenu *File* | *Open*.

- *Save*

  Sama dengan submenu *File* | *Save*.

- *Process*

  Sama dengan submenu *Run* | *Process*.

- *Exit*

  Sama dengan submenu *File* | *Exit*.

- *Play-Pause*

  Untuk memainkan dan menghentikan sementara permainan lagu dari file MIDI pada window anak yang aktif. Merupakan penggabungan dari submenu *MIDI* | *Play* dan submenu *MIDI* | *Pause*.

- *Stop*

  Sama dengan submenu *MIDI* | *Stop*.

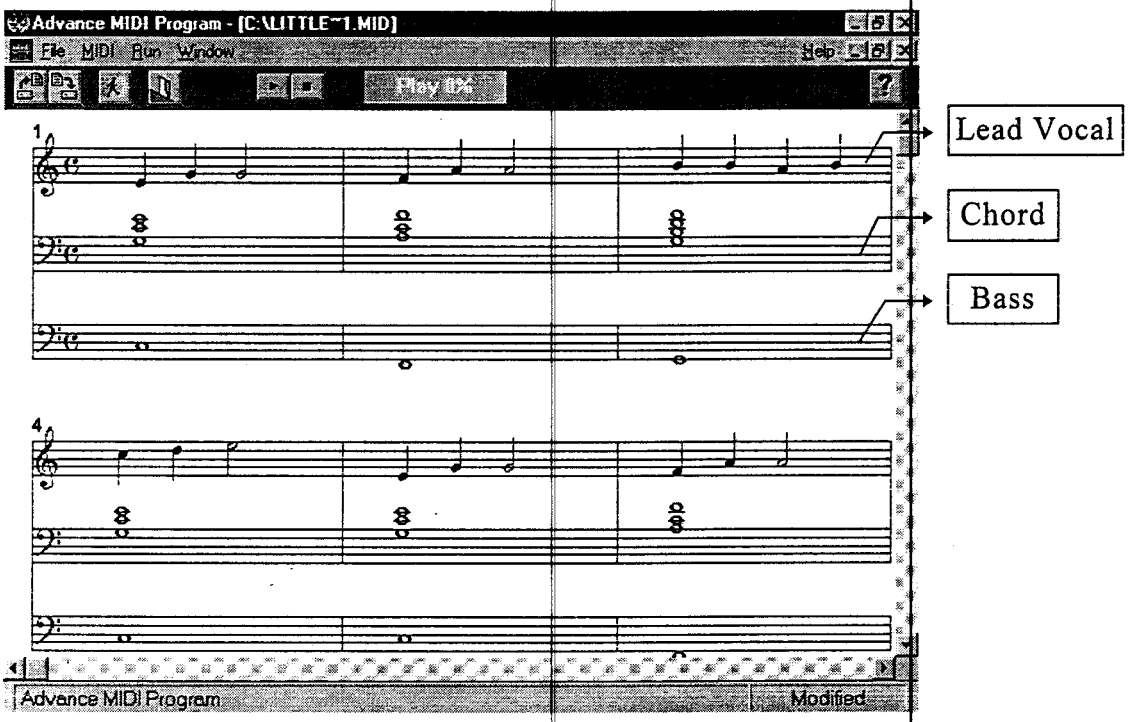
- *Help*

  Sama dengan submenu *Help* | *Contents*.

Selain itu, pada *control bar* juga terdapat *play status* yang menunjukkan berapa persen suatu lagu sudah dimainkan.

B.3. Menjalankan Program

Sebelum menjalankan program aplikasi, terlebih dahulu pastikan bahwa file "ADVMIDI.ART" terletak pada direktori yang sama dengan program ini. File ini merupakan file penyimpan sistem pembobotan dari jaringan saraf. Bila *user* ingin menggunakan sistem pembobotan jaringan sarafnya sendiri, maka harus dibuat melalui perangkat lunak pelatihan jaringan saraf, lalu memberi nama file hasilnya dengan "ADVMIDI.ART".



Input file MIDI yang diberikan harus mempunyai format 1, dan terdiri dari lagu solo (*lead vocal*), tanpa ada nada-nada yang tumpang tindih (*overlaping*). Bila input file MIDI ini tidak sesuai kriteria, program aplikasi

dapat membukanya, namun tidak dapat memrosesnya (window anak tidak menampilkan apa-apa).

Program aplikasi ini juga menyediakan fasilitas untuk mengganti chord suatu birama (submenu *Run* | *Change Chords*). Untuk menggunakan fasilitas ini, terlebih dulu tentukan nilai ambang (*threshold*) melalui submenu *Run* | *Parameter*. Semakin besar nilai ambang ini, maka semakin sedikit pola chord yang dapat dipilih, namun hasilnya akan lebih akurat (sesuai dengan hasil dari jaringan saraf). Sebaliknya, semakin kecil nilai ambang ini, maka semakin banyak pola chord yang dapat dipilih, dan hasilnya kurang akurat. Program aplikasi ini memberi nilai ambang *default* 0.75 saat pertama kali dijalankan.

Lampiran C :
Perhitungan Jaringan Saraf untuk Contoh Uji Coba dan Evaluasi Perangkat Lunak

Perhitungan ini mengacu pada contoh yang terdapat pada Bab VI. Pada contoh tersebut, digunakan lagu *"Little Brown Jug"*, yang terdiri dari 16 birama. Jaringan saraf pada program ini terdiri dari 12 sel saraf input, yaitu :

- S1 ⇨ sel saraf input jenis tanda birama
- S2 ⇨ sel saraf input jenis tangga nada
- S3 ⇨ sel saraf input chord sebelum birama tersebut
- S4 - S11 ⇨ sel-sel saraf input nada-nada satu birama (8 sel saraf)
- S12 ⇨ sel saraf input nada sesudah birama tersebut

Dalam sel-sel saraf output jaringan saraf tersebut, tersimpan 3 pola :

1. *4-BEAT MAJOR C [C C C C] F*
2. *4-BEAT MAJOR C [F F F F] G7*
3. *4-BEAT MAJOR F [G7 G7 G7 G7] C*

Angka-angka berikut ini merupakan hasil perhitungan tiap sel saraf input dari jaringan saraf, yang diperoleh langsung dari uji coba program, dengan catatan, angka-angka di sini dibulatkan menjadi 3 digit di belakang koma. Total output sel-sel saraf input tersebut diletakkan pada kolom terakhir dari setiap tabel. Masing-masing sel saraf input akan melakukan perhitungan untuk setiap pola yang tersimpan di dalam jaringan saraf, dalam hal ini 3 pola.

❑ Birama-1

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.000	0.082	0.041	0.041	0.041	0.082	0.041	0.041	0.041	0.082	0.816
2	0.245	0.082	0.000	0.031	0.015	0.015	0.015	0.015	0.015	0.023	0.031	0.082	0.569
3	0.245	0.082	0.000	0.041	0.020	0.041	0.041	0.082	0.041	0.041	0.041	0.031	0.704

❑ Birama-2

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.163	0.031	0.015	0.015	0.015	0.031	0.015	0.015	0.015	0.000	0.643
2	0.245	0.082	0.163	0.082	0.041	0.041	0.041	0.082	0.041	0.041	0.041	0.082	0.980
3	0.245	0.082	0.000	0.082	0.041	0.015	0.015	0.031	0.015	0.015	0.015	0.031	0.587

❑ Birama-3

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.000	0.031	0.015	0.015	0.015	0.031	0.015	0.023	0.031	0.082	0.584
2	0.245	0.082	0.000	0.015	0.015	0.023	0.031	0.082	0.041	0.000	0.000	0.031	0.564
3	0.245	0.082	0.163	0.082	0.041	0.041	0.041	0.046	0.031	0.041	0.041	0.082	0.934

❑ Birama-4

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.000	0.082	0.041	0.023	0.031	0.082	0.041	0.041	0.041	0.031	0.737
2	0.245	0.082	0.000	0.082	0.041	0.015	0.015	0.031	0.015	0.015	0.015	0.031	0.587
3	0.245	0.082	0.000	0.046	0.031	0.041	0.041	0.015	0.015	0.023	0.031	0.082	0.651

❑ Birama-5

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.163	0.082	0.041	0.041	0.041	0.082	0.041	0.041	0.041	0.082	0.980
2	0.245	0.082	0.163	0.031	0.015	0.015	0.015	0.015	0.015	0.023	0.031	0.082	0.732
3	0.245	0.082	0.000	0.041	0.020	0.041	0.041	0.082	0.041	0.041	0.041	0.031	0.704

❑ Birama-6

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.163	0.031	0.015	0.015	0.015	0.031	0.015	0.015	0.015	0.000	0.643
2	0.245	0.082	0.163	0.082	0.041	0.041	0.041	0.082	0.041	0.041	0.041	0.082	0.980
3	0.245	0.082	0.000	0.082	0.041	0.015	0.015	0.031	0.015	0.015	0.015	0.031	0.587

❑ Birama-7

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.000	0.031	0.015	0.015	0.015	0.031	0.015	0.023	0.031	0.082	0.584
2	0.245	0.082	0.000	0.015	0.015	0.023	0.031	0.082	0.041	0.000	0.000	0.031	0.564
3	0.245	0.082	0.163	0.082	0.041	0.041	0.041	0.046	0.031	0.041	0.041	0.082	0.934

❑ Birama-8

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.000	0.082	0.041	0.041	0.041	0.082	0.041	0.000	0.000	0.031	0.684
2	0.245	0.082	0.000	0.082	0.041	0.041	0.041	0.082	0.041	0.000	0.000	0.031	0.684
3	0.245	0.082	0.000	0.031	0.015	0.015	0.015	0.031	0.015	0.000	0.000	0.082	0.531

❑ Birama-9

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.163	0.082	0.041	0.041	0.041	0.082	0.041	0.041	0.041	0.082	0.980
2	0.245	0.082	0.163	0.041	0.020	0.041	0.041	0.015	0.015	0.023	0.031	0.082	0.798
3	0.245	0.082	0.000	0.031	0.015	0.020	0.020	0.082	0.041	0.041	0.041	0.031	0.648

❑ Birama-10

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.163	0.031	0.015	0.015	0.015	0.031	0.015	0.015	0.015	0.031	0.673
2	0.245	0.082	0.163	0.082	0.041	0.041	0.041	0.082	0.041	0.041	0.041	0.082	0.980
3	0.245	0.082	0.000	0.082	0.041	0.015	0.015	0.015	0.015	0.023	0.031	0.082	0.645

❑ Birama-11

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.000	0.082	0.041	0.015	0.015	0.031	0.015	0.015	0.015	0.031	0.587
2	0.245	0.082	0.000	0.031	0.015	0.000	0.000	0.031	0.023	0.031	0.041	0.082	0.579
3	0.245	0.082	0.163	0.082	0.041	0.041	0.041	0.082	0.041	0.041	0.031	0.082	0.969

❑ Birama-12

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.000	0.082	0.041	0.041	0.041	0.082	0.041	0.041	0.041	0.031	0.765
2	0.245	0.082	0.000	0.041	0.020	0.041	0.041	0.082	0.041	0.041	0.041	0.031	0.704
3	0.245	0.082	0.000	0.082	0.041	0.015	0.015	0.041	0.020	0.020	0.020	0.082	0.663

❑ Birama-13

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.163	0.082	0.041	0.041	0.041	0.082	0.041	0.041	0.041	0.082	0.980
2	0.245	0.082	0.163	0.041	0.020	0.041	0.041	0.015	0.015	0.023	0.031	0.082	0.798
3	0.245	0.082	0.000	0.031	0.015	0.020	0.020	0.082	0.041	0.041	0.041	0.031	0.648

❑ Birama-14

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.163	0.031	0.015	0.015	0.015	0.031	0.015	0.015	0.015	0.000	0.643
2	0.245	0.082	0.163	0.082	0.041	0.041	0.041	0.082	0.041	0.041	0.041	0.082	0.980
3	0.245	0.082	0.000	0.082	0.041	0.015	0.015	0.031	0.015	0.015	0.015	0.031	0.587

❑ Birama-15

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1	0.245	0.082	0.000	0.031	0.015	0.015	0.015	0.031	0.015	0.023	0.031	0.082	0.584
2	0.245	0.082	0.000	0.015	0.015	0.023	0.031	0.082	0.041	0.000	0.000	0.031	0.564
3	0.245	0.082	0.163	0.082	0.041	0.041	0.041	0.046	0.031	0.041	0.041	0.082	0.934

❑ Birama-16

P	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	Tot.
1*	0.245	0.082	0.000	0.082	0.041	0.041	0.041	0.082	0.041	0.000	0.000	0.000	0.653
2	0.245	0.082	0.000	0.082	0.041	0.041	0.041	0.082	0.041	0.000	0.000	0.000	0.653
3	0.245	0.082	0.000	0.031	0.015	0.015	0.015	0.031	0.015	0.000	0.000	0.000	0.449

Pada masing-masing tabel, terdapat satu baris yang diberi tanda. Baris-baris tersebut menunjukkan pola yang dipilih (pola pemenang). Khusus untuk birama-8 dan birama-16, terdapat 2 (dua) pola pemenang, sehingga pola pemenangnya ditentukan berdasarkan nada dasarnya.